Escuela de Algoritmos de Aproximación
Algoritmos glotones y Búsqueda local

Marco Antonio Heredia Velasco

Universidad Autónoma Metropolitana Unidad Azcapotzalco

Estudiaremos

Dos técnicas estandar para diseñar algoritmos y heurísticas:

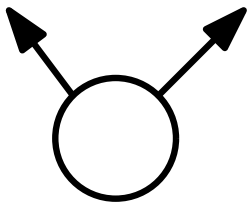
- Algoritmos glotones.
- Algoritmos de búsqueda local.

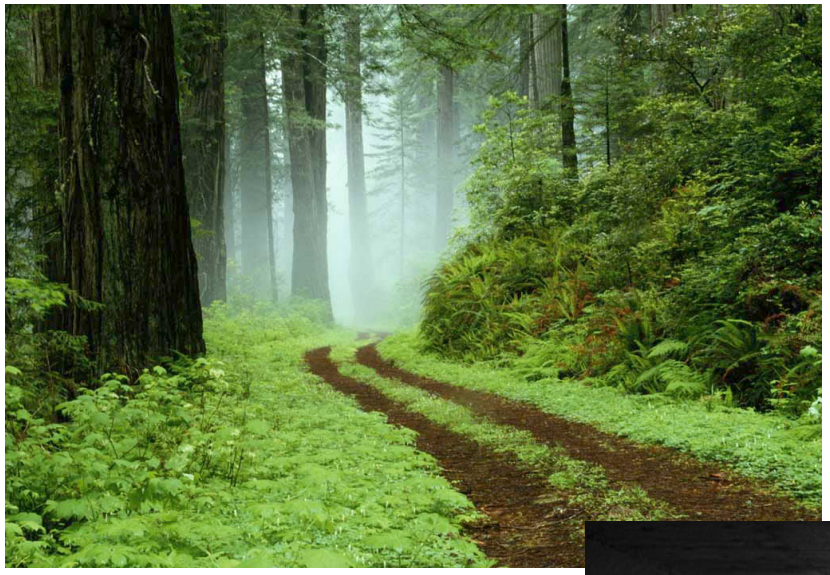
Ambos métodos toman una secuencia de decisiones, y en cada decisión se optimiza una elección local.

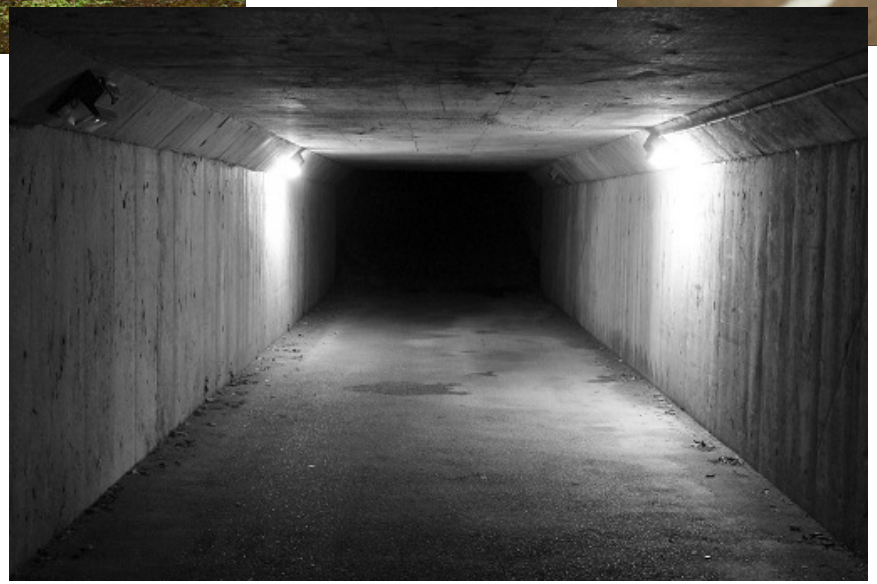
Algoritmo glotón (Greedy)

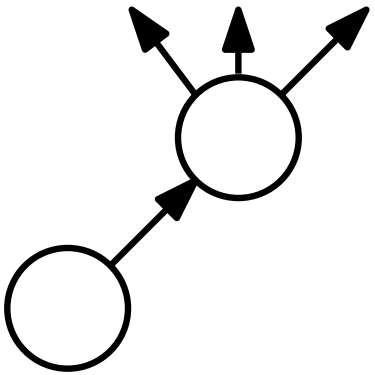
- Voraz, ávido, avaro, devorador ... goloso.
- Cada parte de una solución se construye tomando una decisión, que al momento es la mejor posible.

Muestra

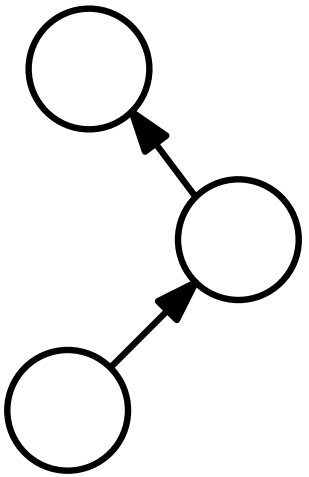









END

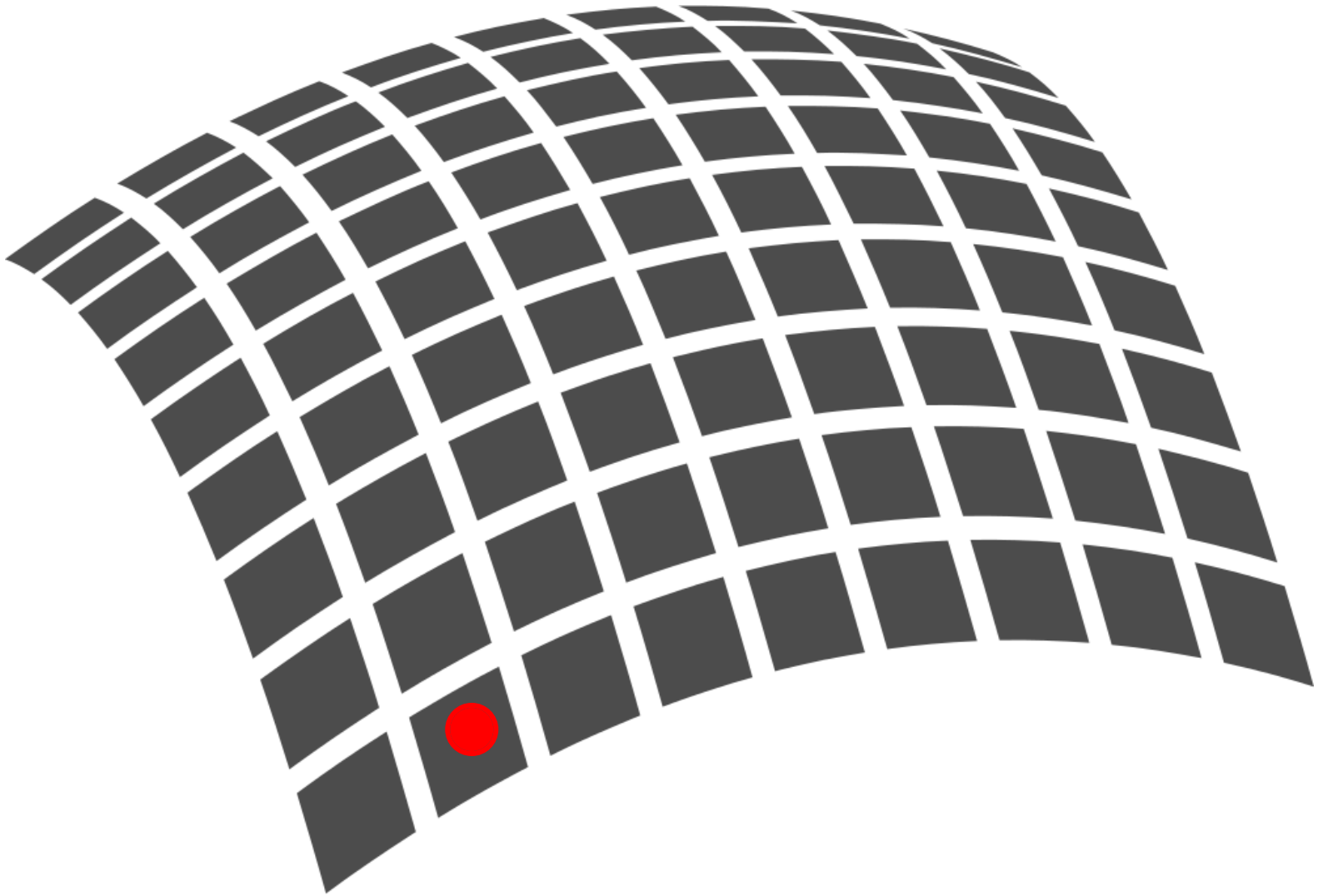


Búsqueda local (local search)

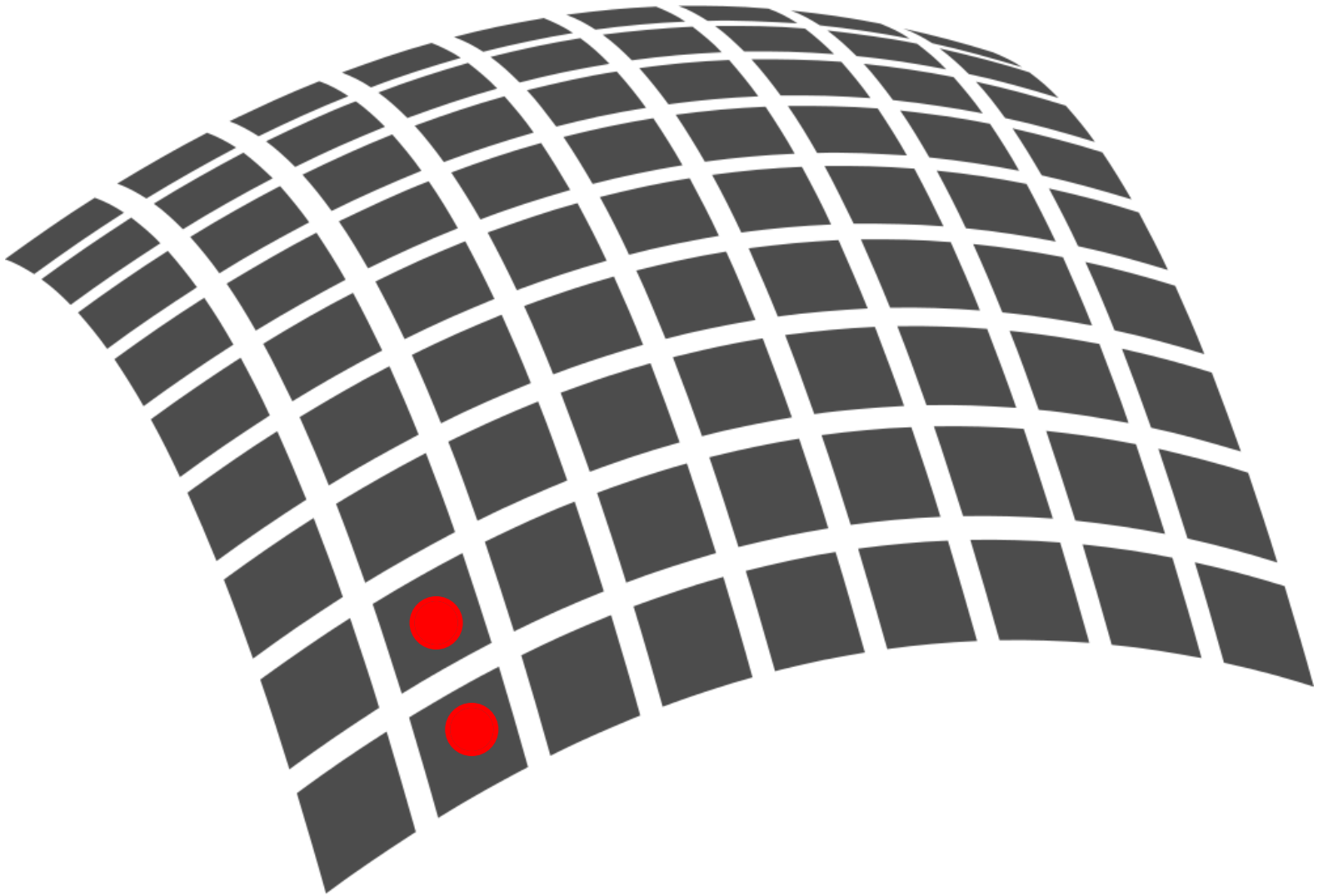
- Se parte de una solución factible.
- Verifica si una pequeña modificación a la solución, obtiene una solución mejor (que mejore la función objetivo).

Búsqueda local (local search)

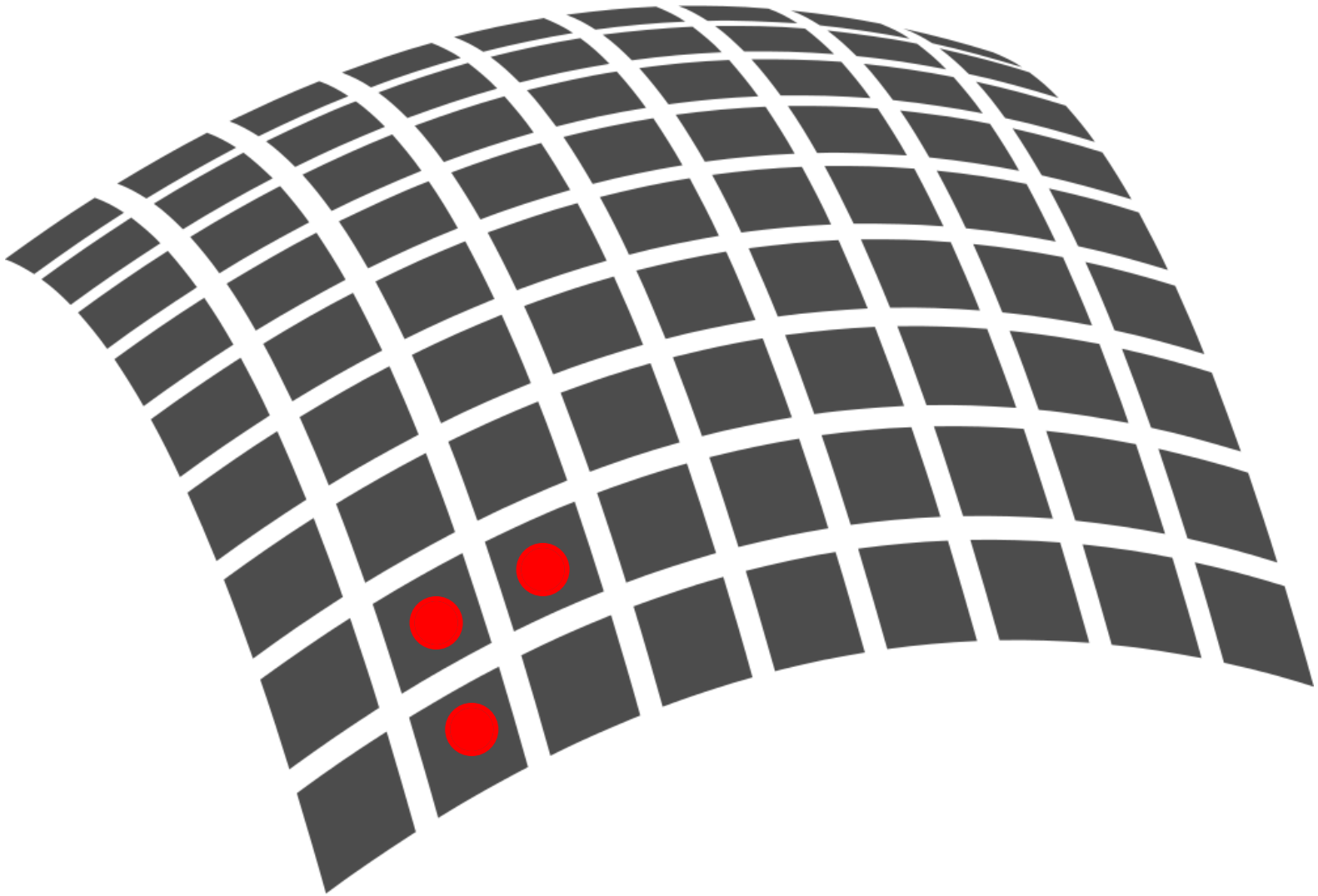
Búsqueda local (local search)



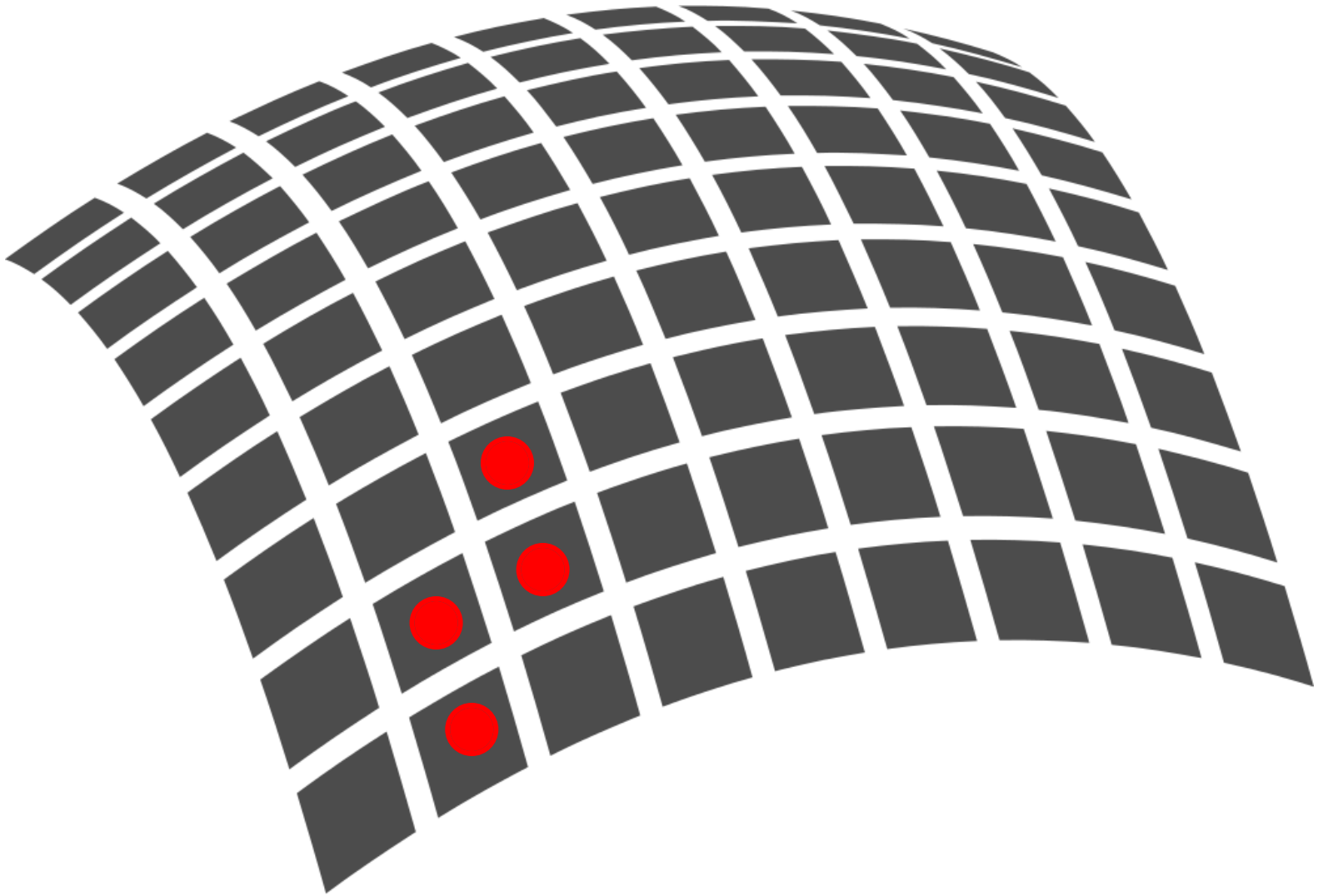
Búsqueda local (local search)



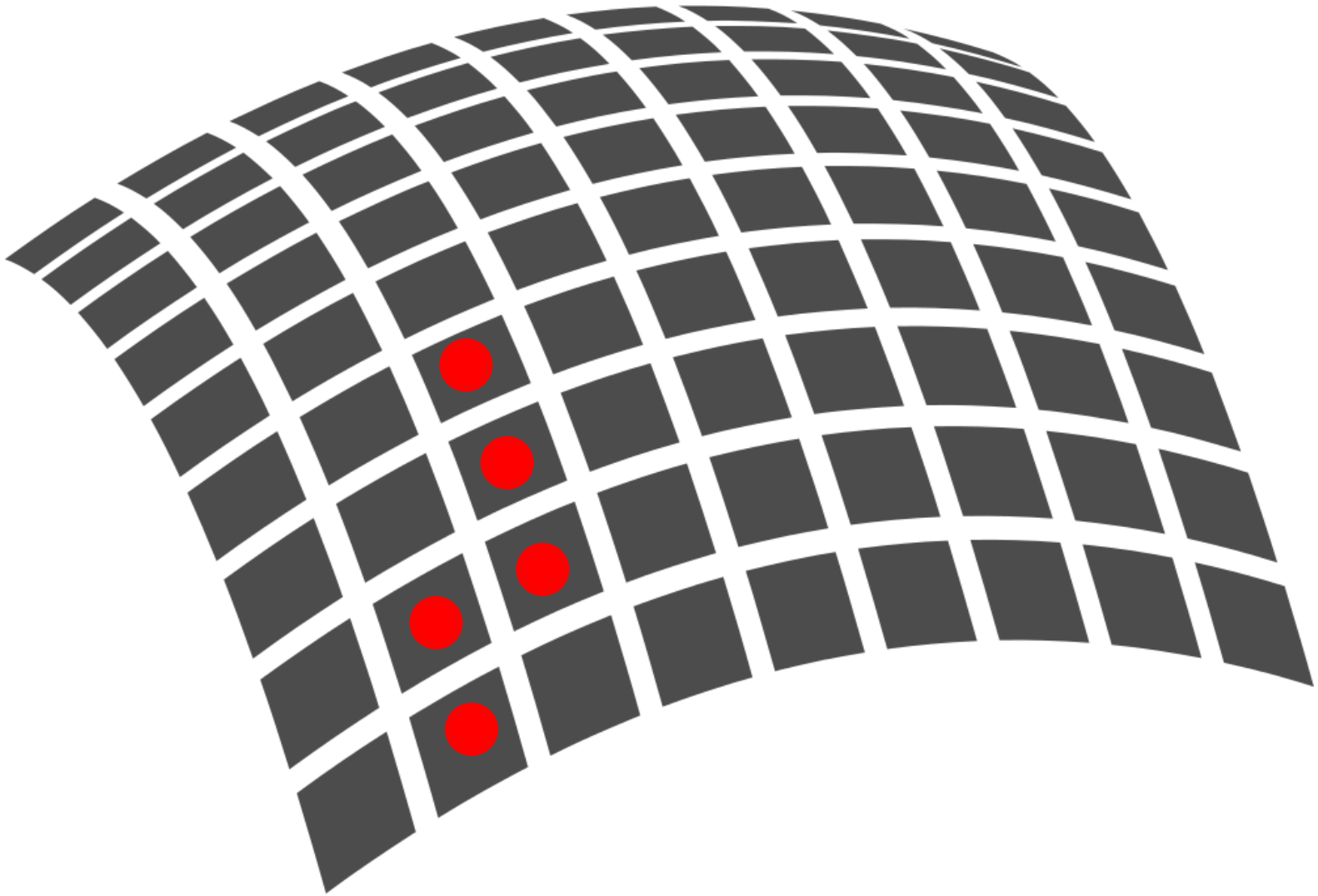
Búsqueda local (local search)



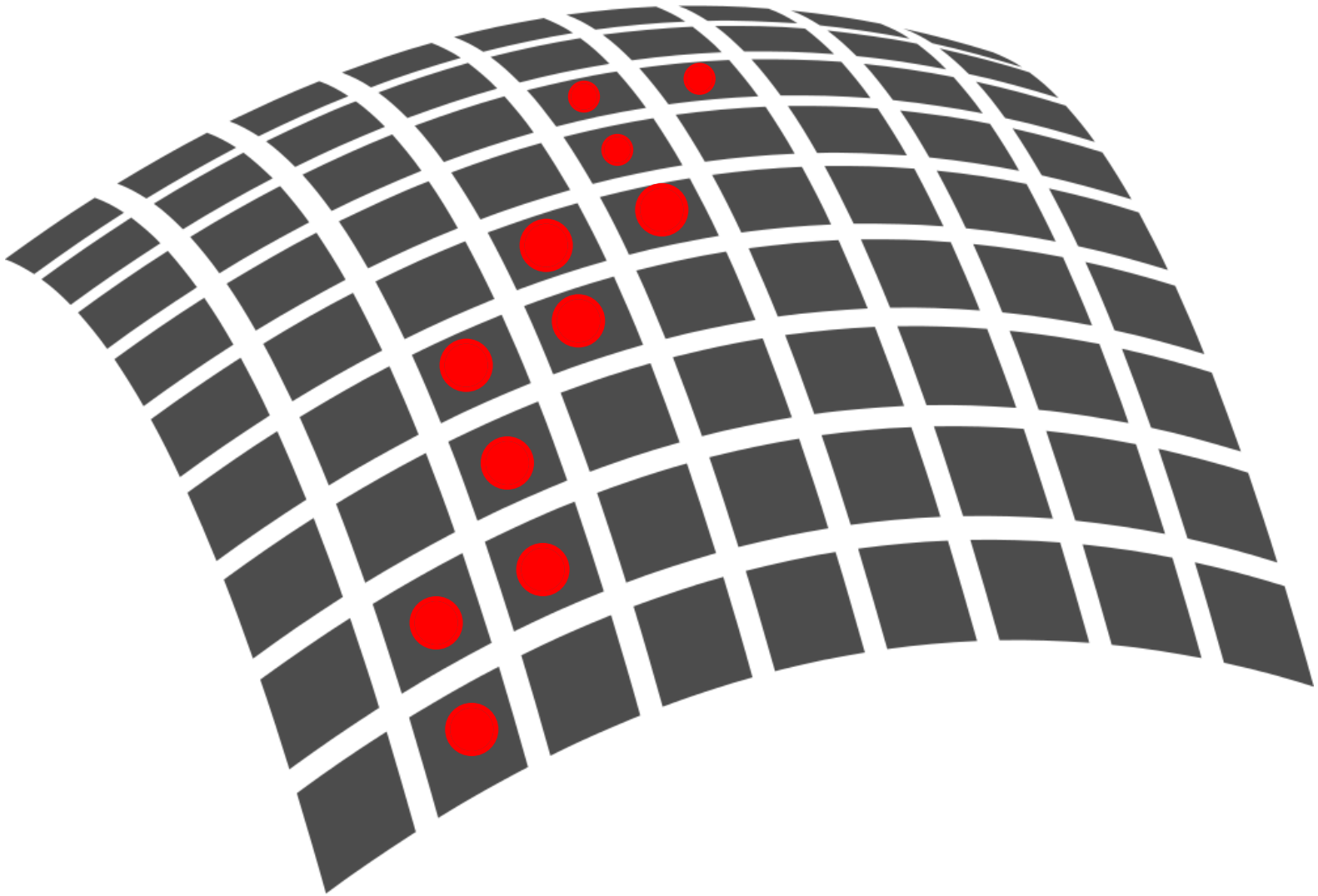
Búsqueda local (local search)



Búsqueda local (local search)



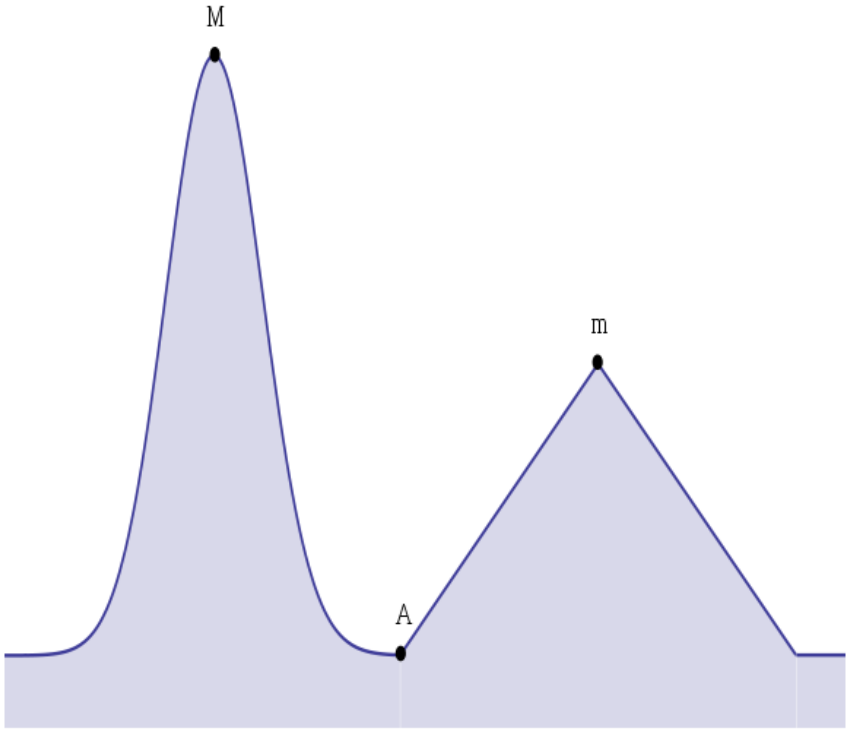
Búsqueda local (local search)

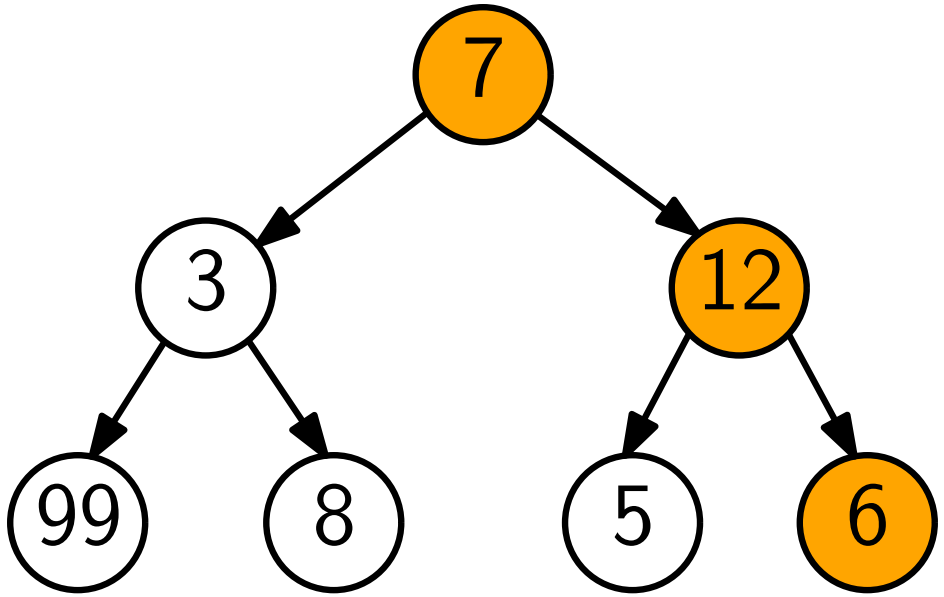


Ambos métodos:

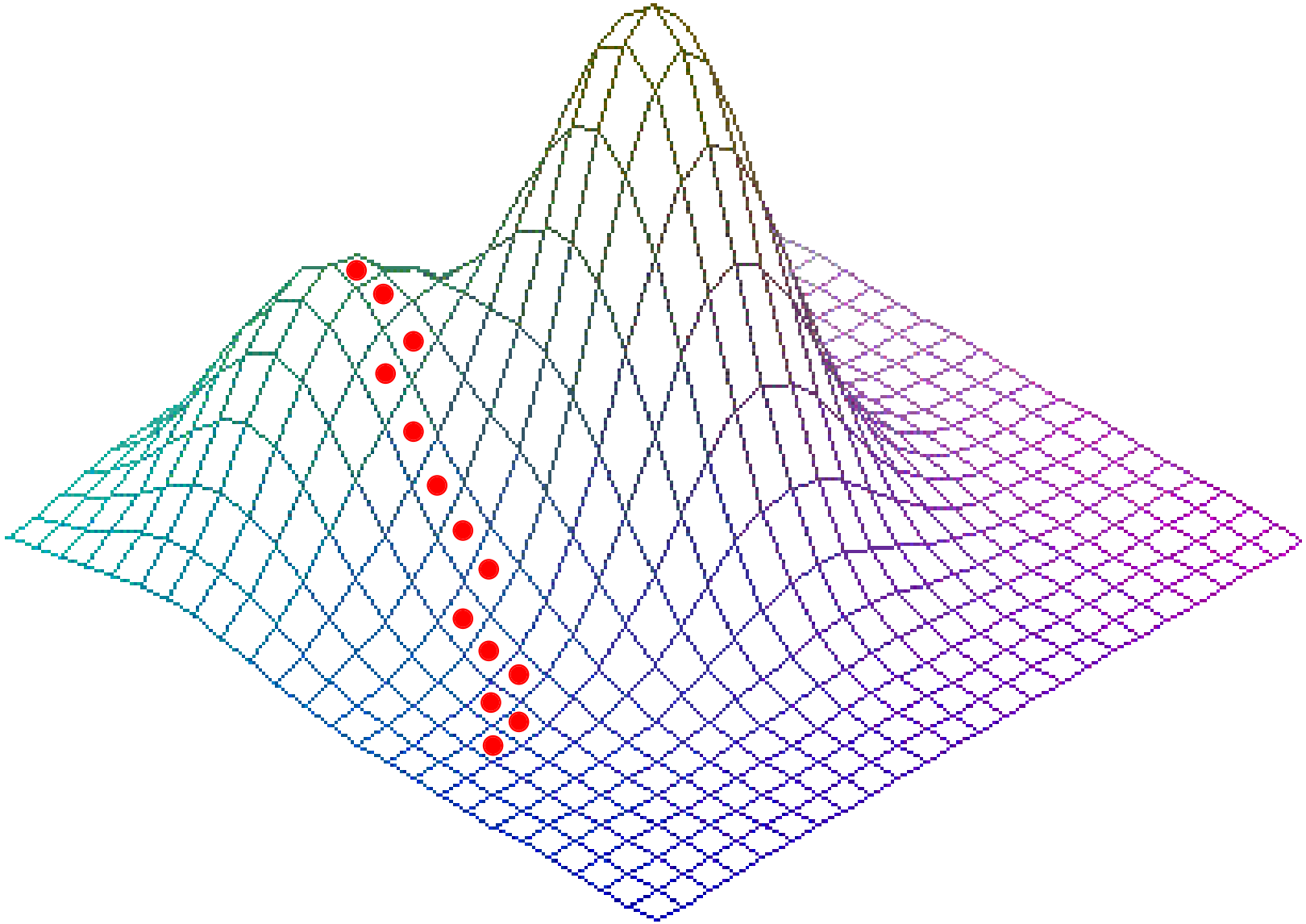
- Opciones populares para heurísticas de problemas NP-completos.
- Fáciles de implementar.
- Naturales: ya se usaban para aproximar antes del concepto de NP-completez.
- Buenos tiempos de procesamiento, pero...

Ambos pueden fallar



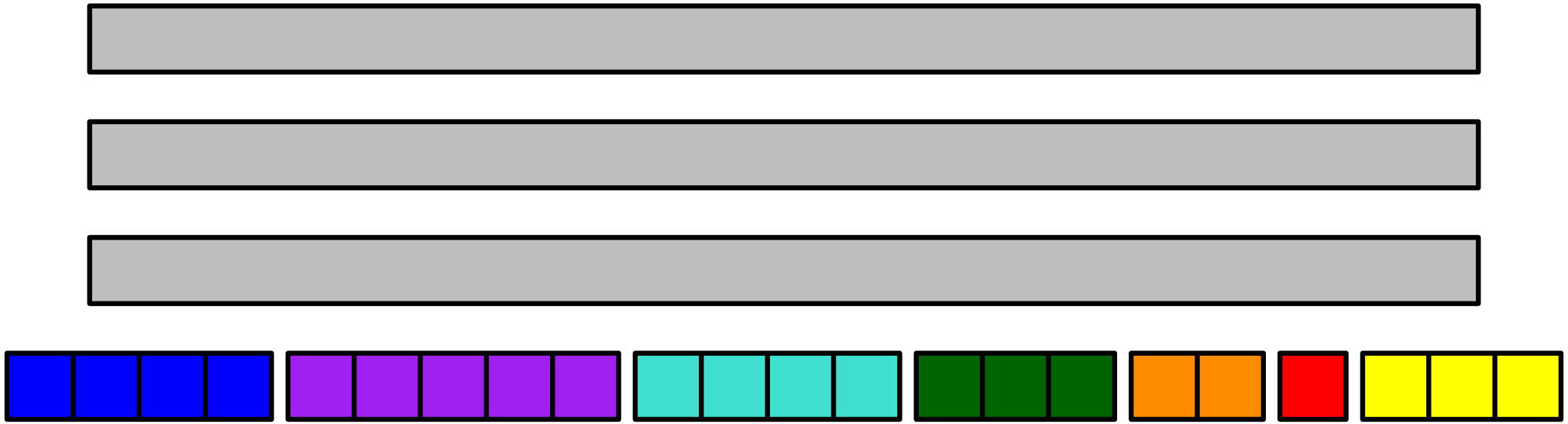


Ambos pueden fallar



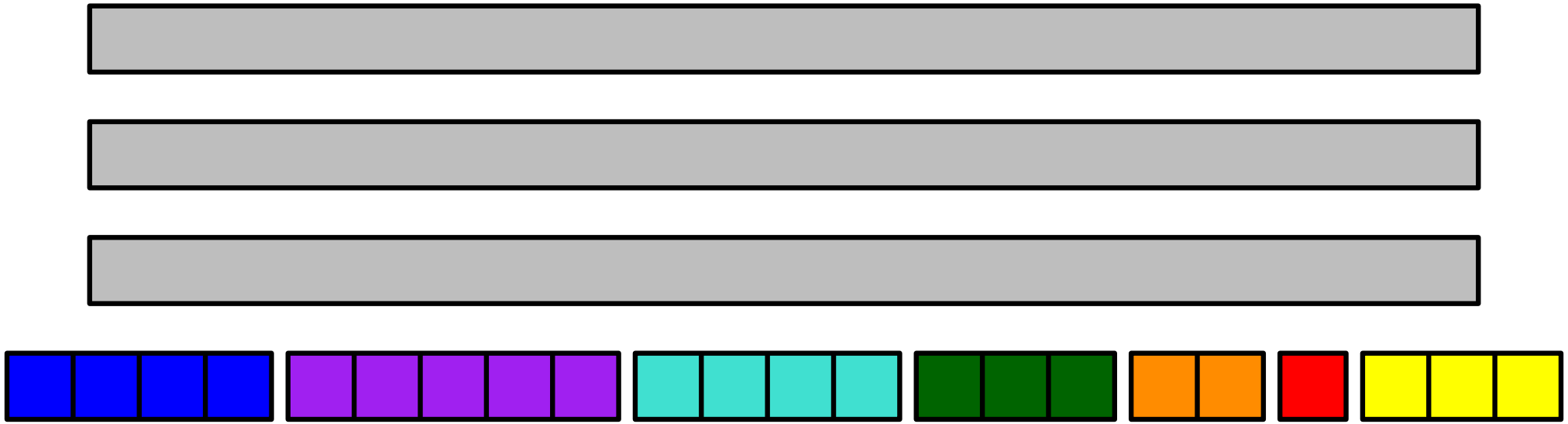
Planeación de tareas en máquinas paralelas idénticas

Planeación en máquinas paralelas idénticas



- n tareas a ejecutar en m máquinas idénticas
- Cada tarea tiene un tiempo de procesamiento p_j.
- Cuando una tarea se ha empezado se debe completar.
- La planificación empieza en el tiempo 0.

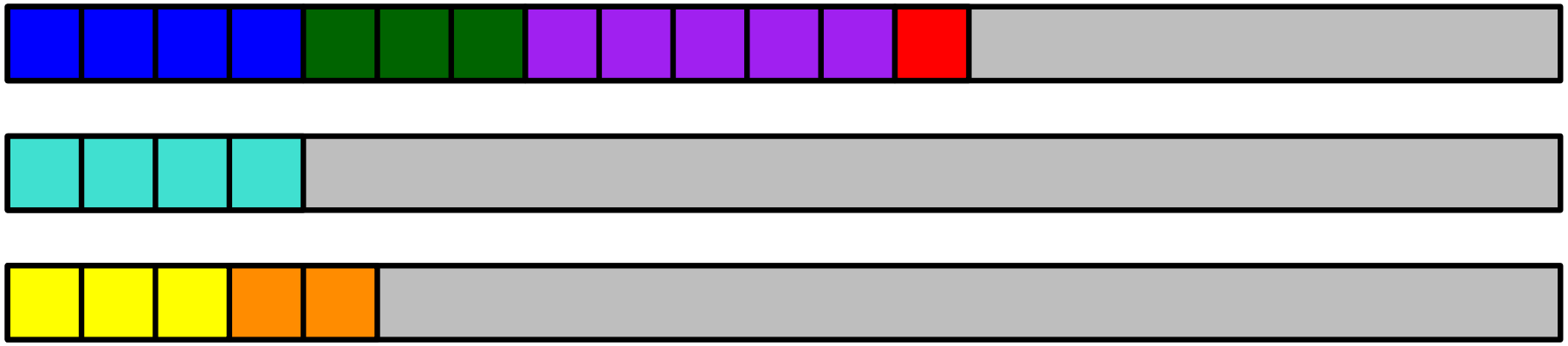
Planeación en máquinas paralelas idénticas



- Tiempo de finalización de la tarea $j = F_j$.
- La meta es planificar todos los trabajos de manera que se minimice el máximo tiempo de finalización:

$$\text{minimiza } F_{max} = \text{máx}_{i=1\ldots n} F_j$$

Algoritmo de búsqueda local



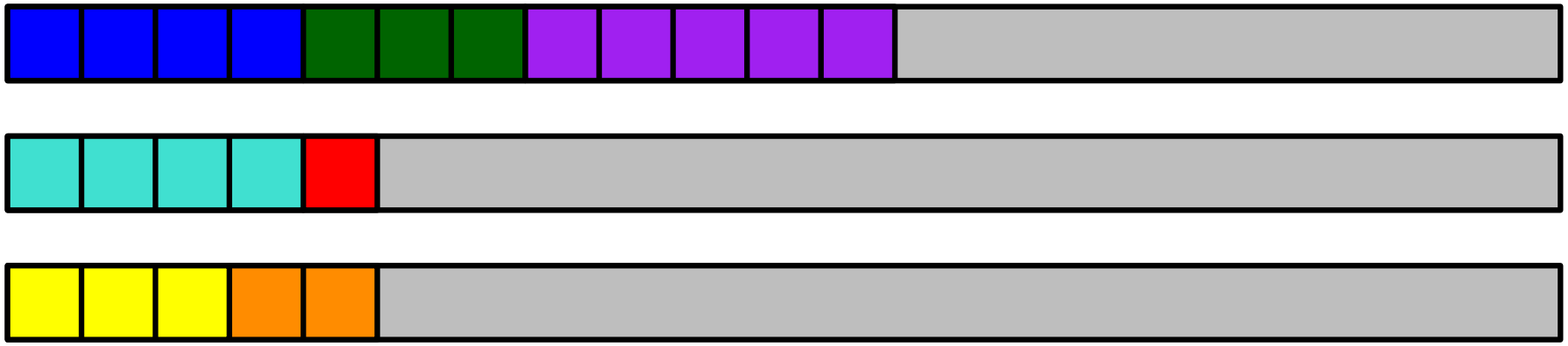
- Empezar con cualquier planificación

- Considera la tarea que termina al último:

    - Si reasignarla a otra máquina hace que se termine antes, entonces reasignala a aquella máquina que haga que se termine primero

- Repite hasta que el trabajo que termine al último ya no pueda reasignarse.

Algoritmo de búsqueda local



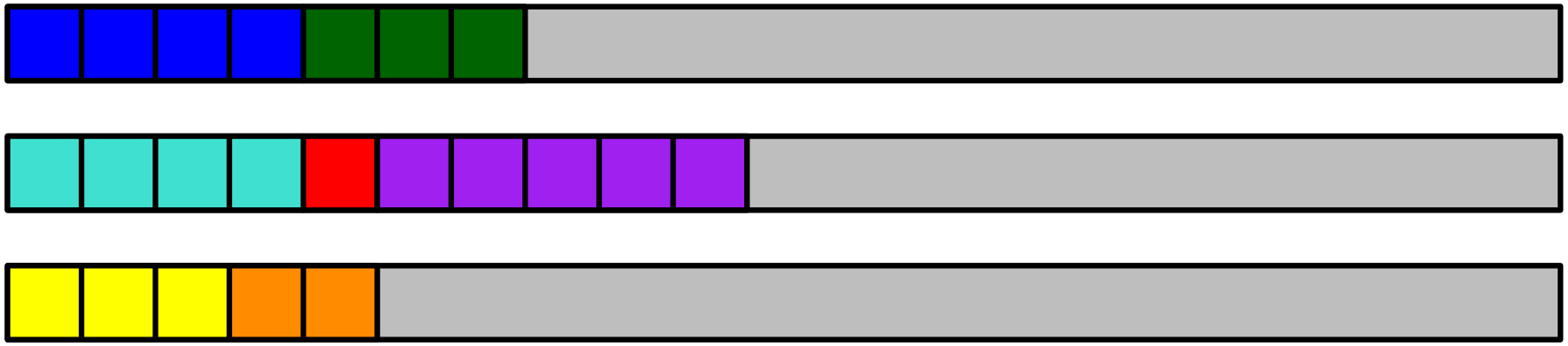
- Empezar con cualquier planificación

- Considera la tarea que termina al último:

    - Si reasignarla a otra máquina hace que se termine antes, entonces reasignala a aquella máquina que haga que se termine primero

- Repite hasta que el trabajo que termine al último ya no pueda reasignarse.

Algoritmo de búsqueda local



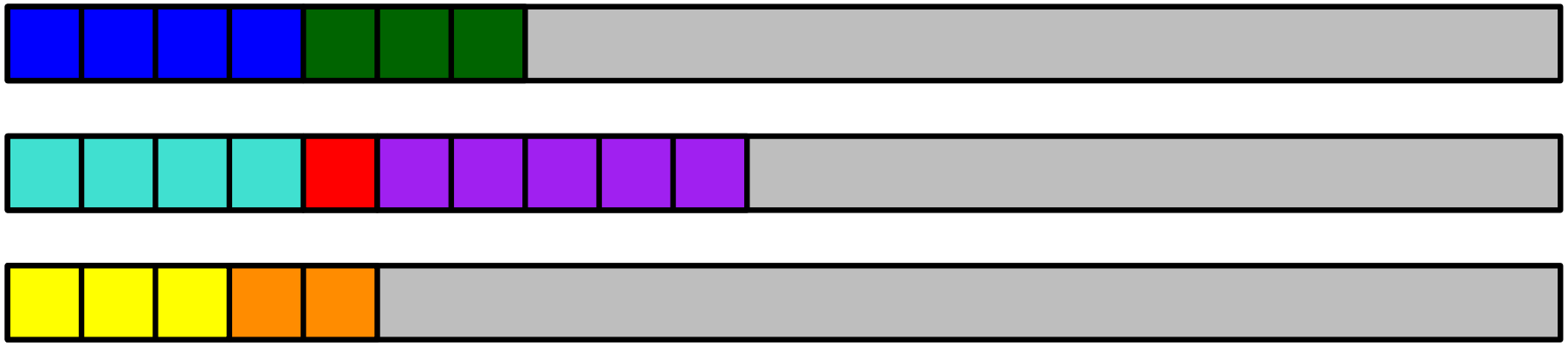
- Empezar con cualquier planificación

- Considera la tarea que termina al último:

    - Si reasignarla a otra máquina hace que se termine antes, entonces reasignala a aquella máquina que haga que se termine primero

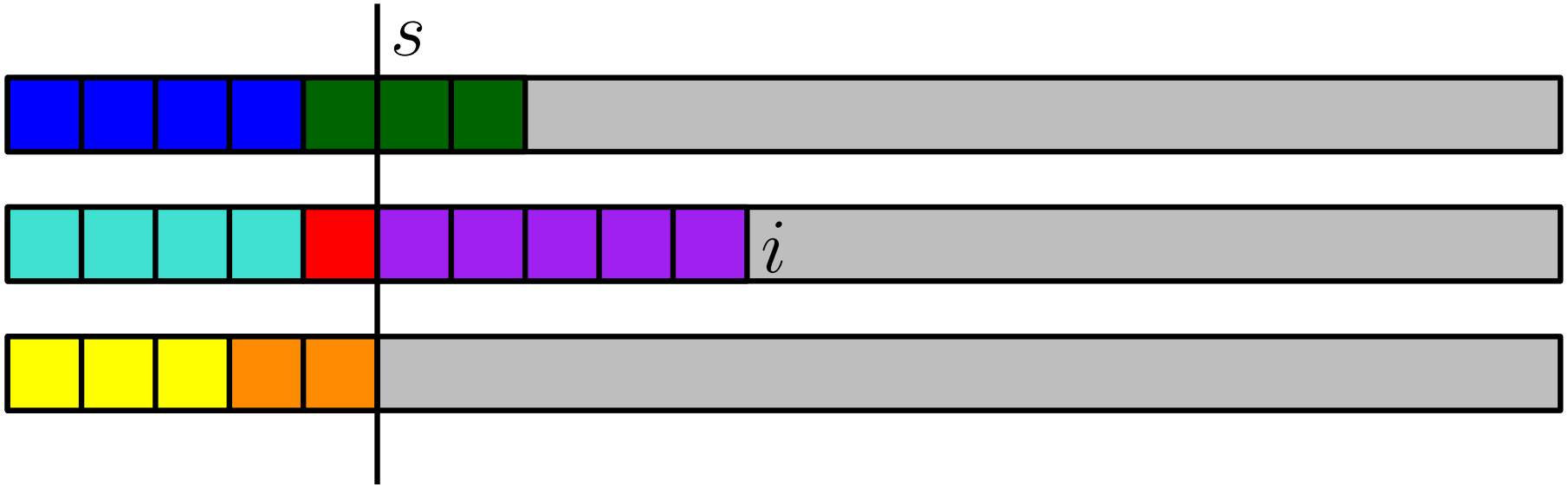
- Repite hasta que el trabajo que termine al último ya no pueda reasignarse.

Algoritmo de búsqueda local



- Este algoritmo de búsqueda es una 2-aproximación.
  - Obtiene una solución válida. ✓
  - Está a un factor 2 del óptimo.
  - Termina en tiempo polinomial.

Factor 2 del óptimo



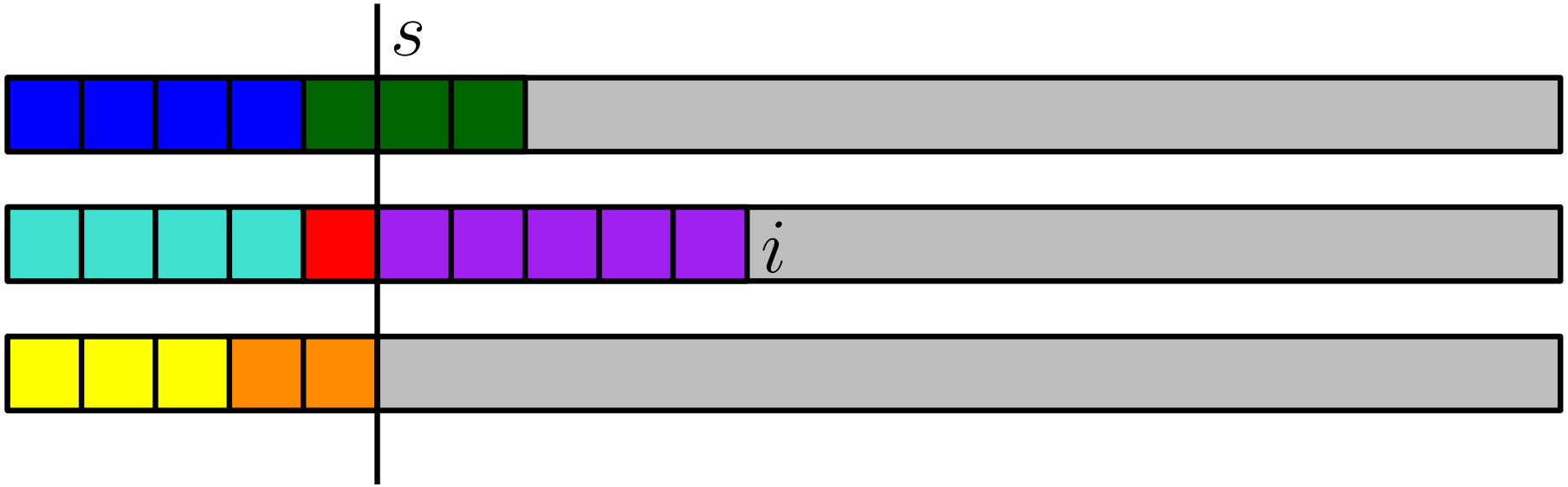
- Llamemos F^* al tiempo óptimo.

- Cada tarea debe de procesarse, entonces $F^* \geq \text{máx}_{i=1\ldots n}\, p_i$

- Hay una máquina que al menos se le asigna el promedio de la carga total: $F^* \geq \sum_{i=1\ldots n} p_i/m$

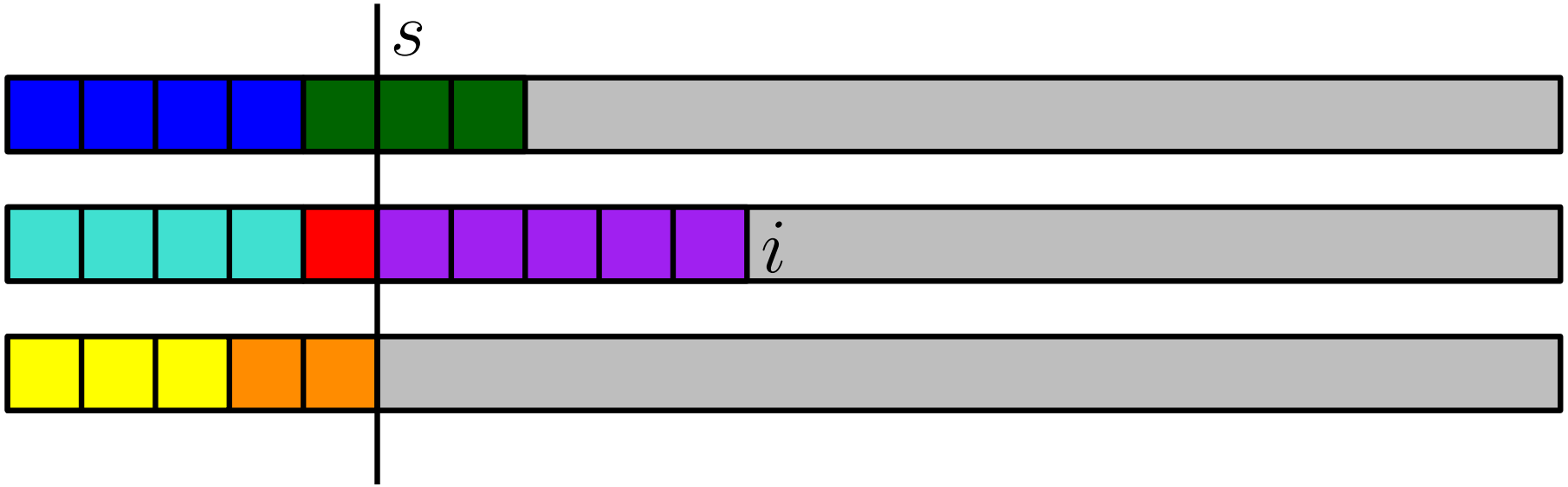
- Todas las demás máquinas deben estar ocupadas hasta que se ejecuta i en el tiempo s. $(s = F_i - p_i)$

Factor 2 del óptimo



- Se parte la planificación en 2 partes, antes de s y después de s.
- Después $\leq F^*$ pues sabemos $F^* \geq \text{máx}_{i=1\ldots n}\, p_i$
- Antes:
  - Todas las máquinas están ocupadas entonces el trabajo total $= m * s$
  - $m * s \leq \sum_{i=1\ldots n} p_i$ entonces $s \leq \sum_{i=1\ldots n} p_i/m \leq F^*$

Factor 2 del óptimo



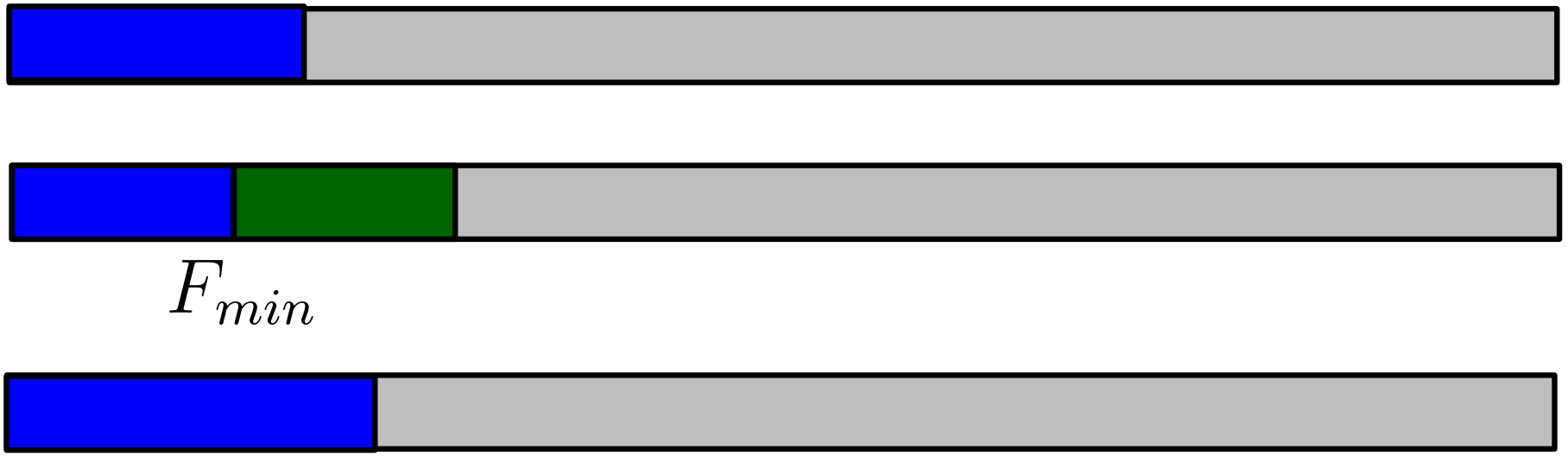
- Tiempo de esta planificación $\leq 2F^*$.

Tiempo polinomial

- El mínimo tiempo de finalización F_{min} nunca decrementa.

Tiempo polinomial

- El mínimo tiempo de finalización F_{min} nunca decrementa.



Tiempo polinomial

- Ninguna tarea es reasignada más de una vez.
- Por contradicción

F_{min}

F'_{min}

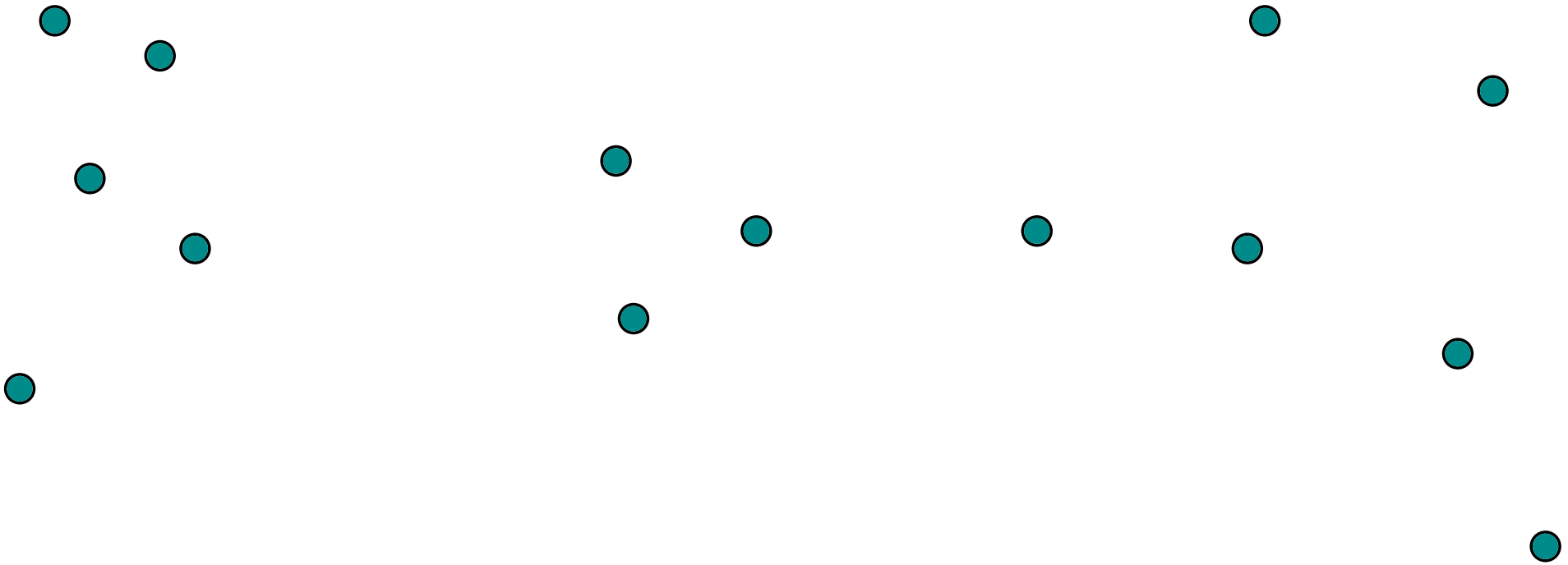
- $F_{min} > F'_{min}$, esto es una contradicción.

El algoritmo termina en a lo más n iteraciones

K-centros

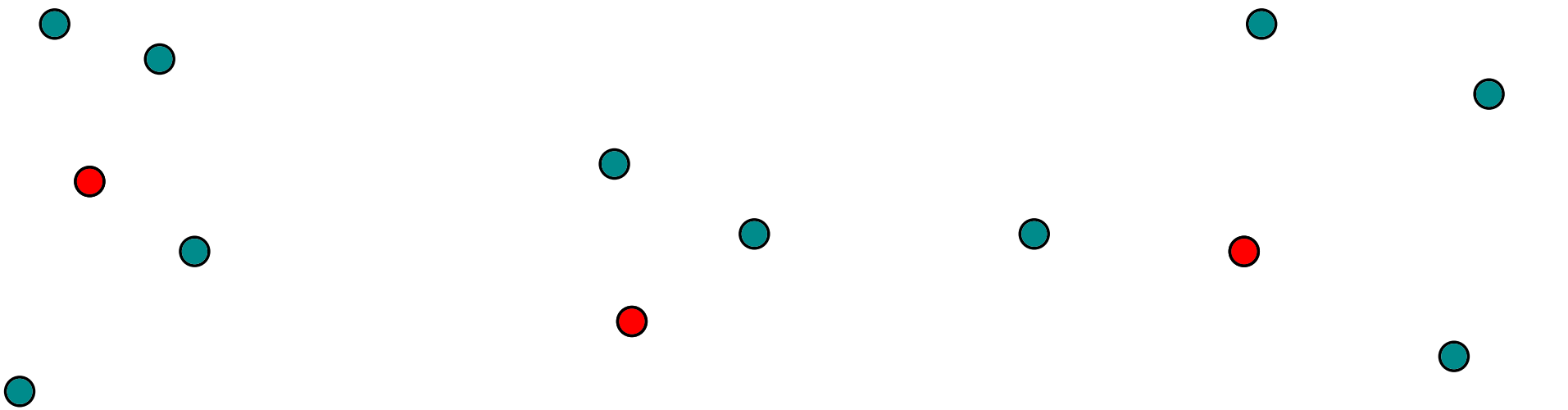
k-centros

Conjunto V de n puntos en el plano y un entero positivo k.



k-centros

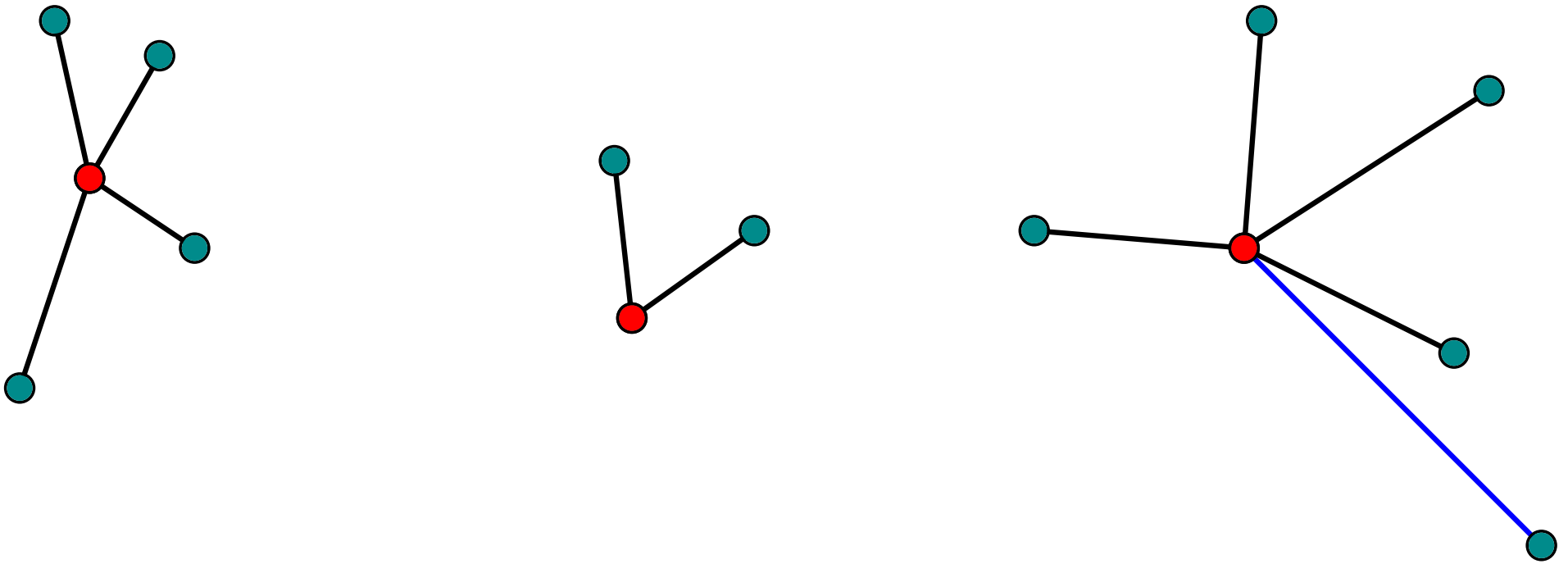
Conjunto V de n puntos en el plano y un entero positivo k.



$k = 3$

k-centros

Conjunto V de n puntos en el plano y un entero positivo k.

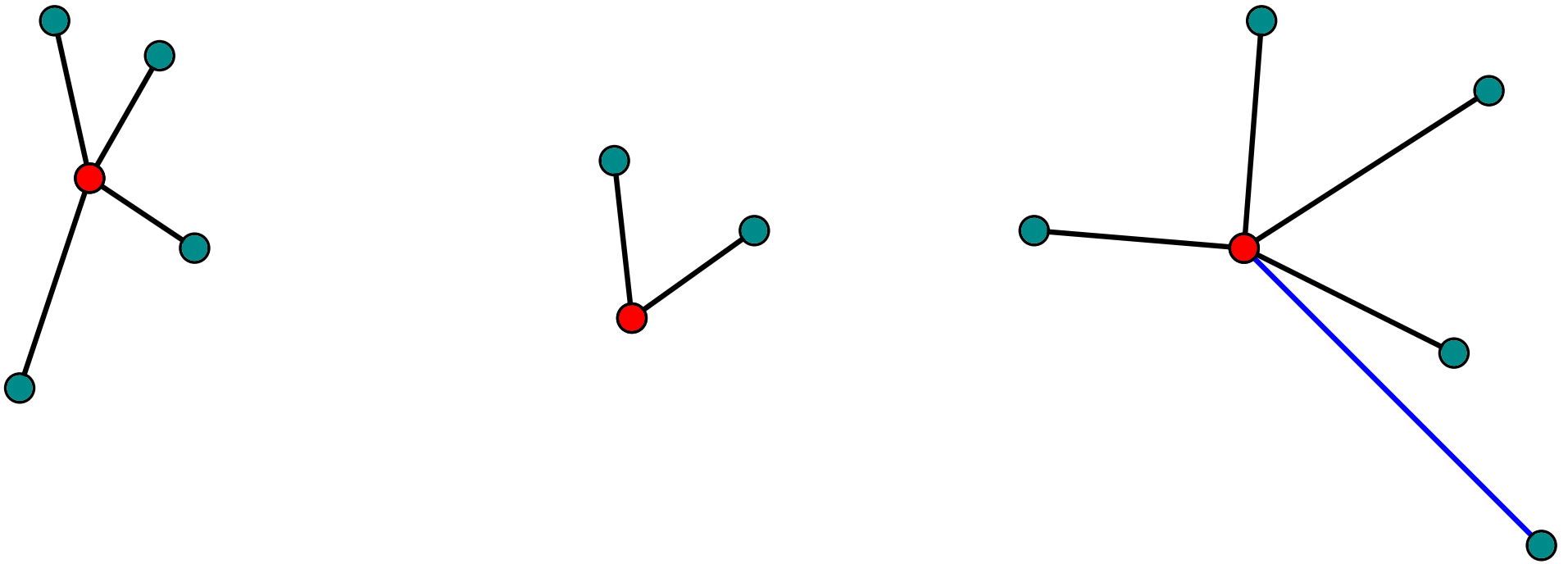


$k = 3$

k-centros

Conjunto V de n puntos en el plano y un entero positivo k.

Seleccionar k centros tal que se minimiza la máxima distancia de un punto a su centro más cercano.

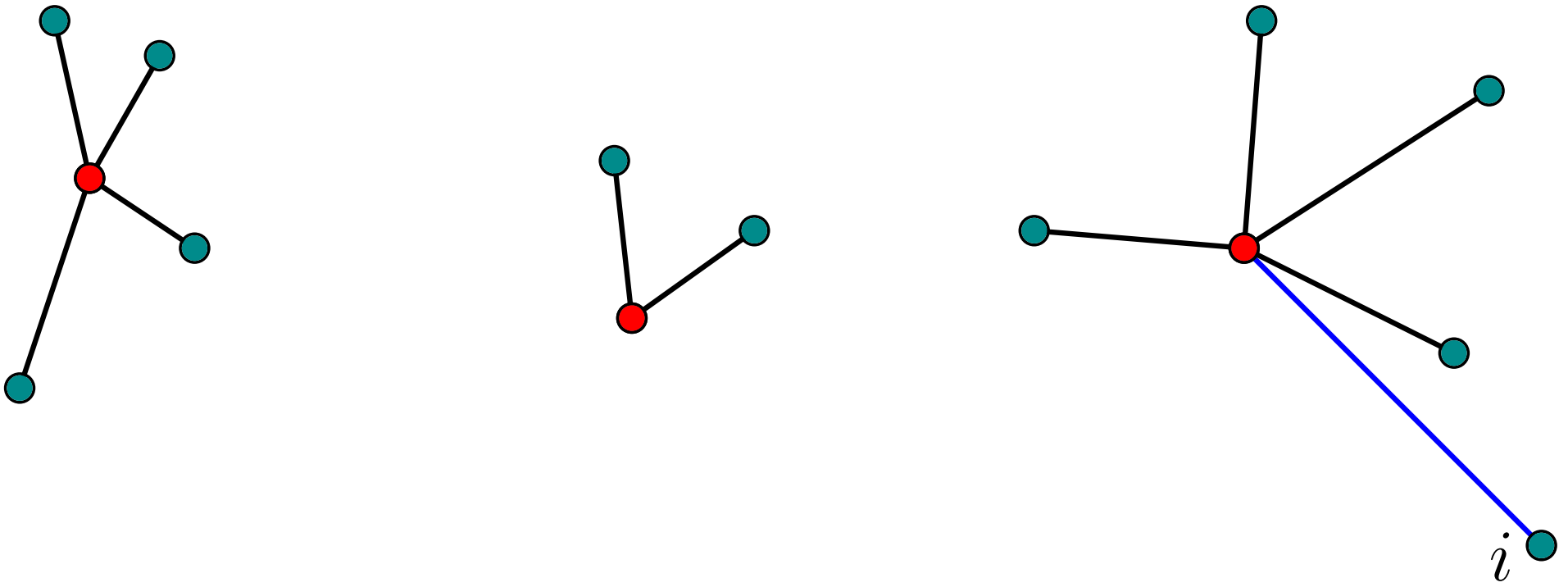


$k = 3$

k-centros

Conjunto V de n puntos en el plano y un entero positivo k.

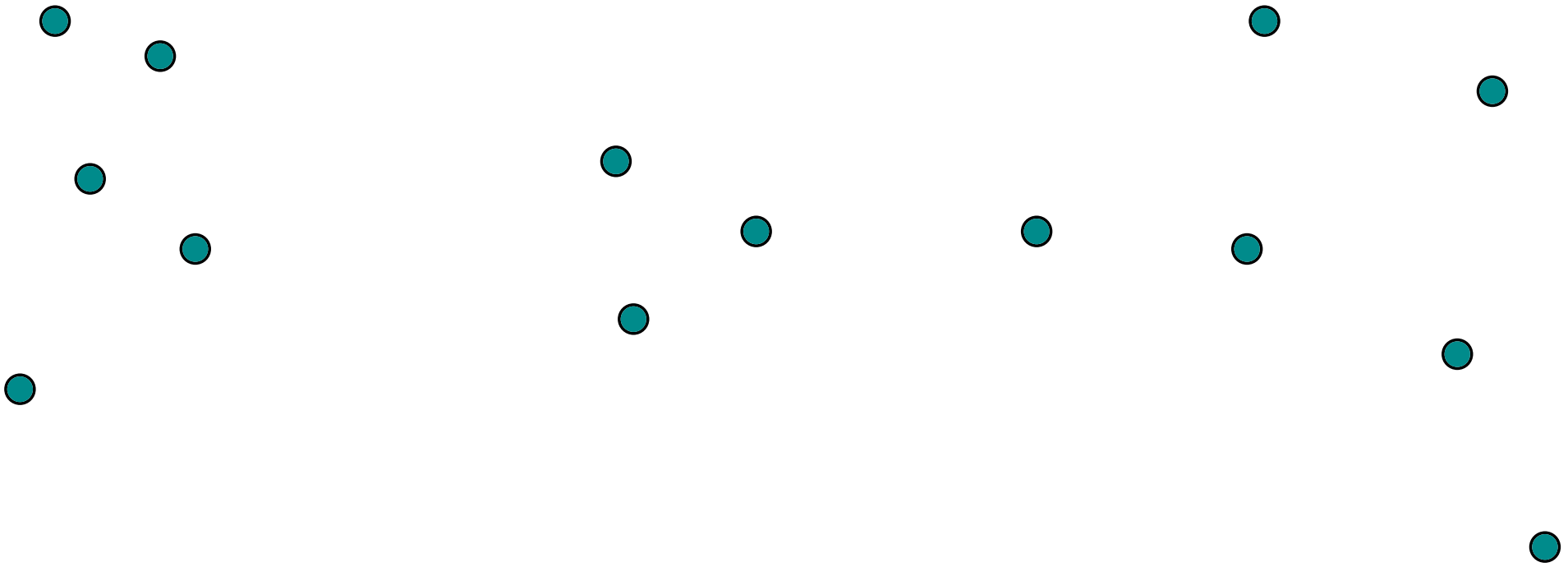
Encontrar $S \subseteq V$ con $|S| = k$ que: minimice $\max_{i \in V} d(i, S)$ donde $d(i, S) := \min_{j \in S} d(i, j)$.



$k = 3$

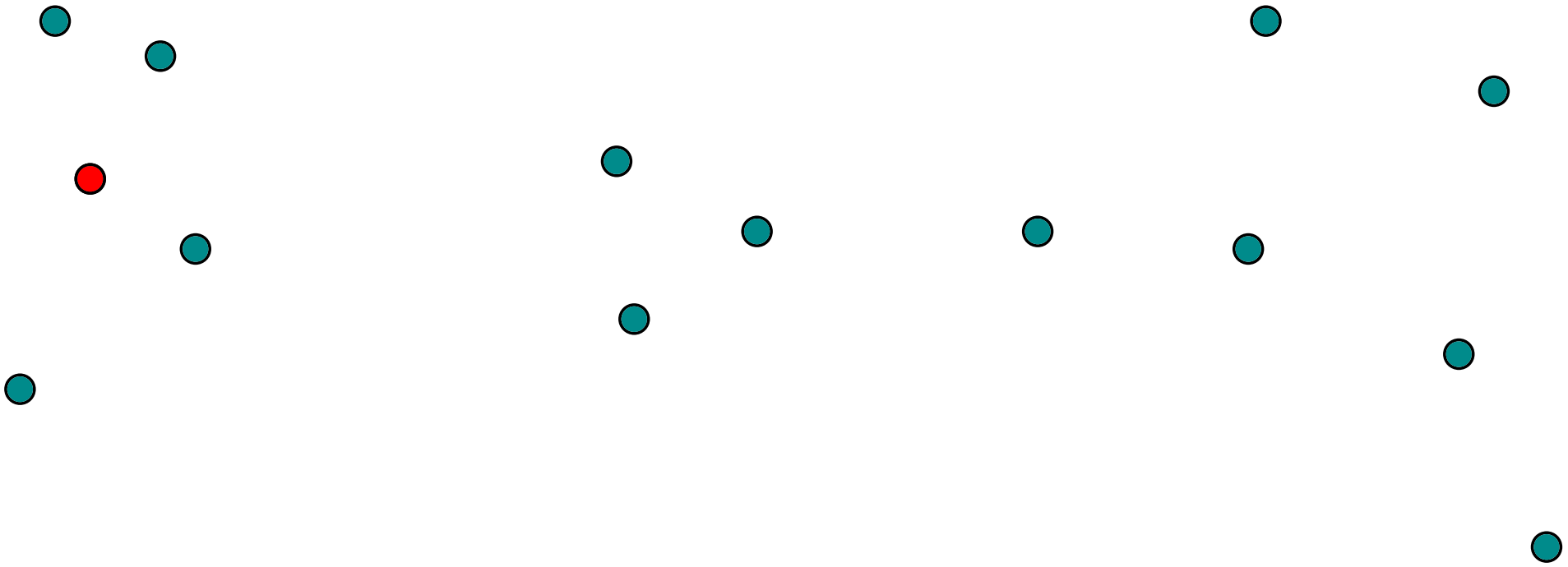
Algoritmo Glotón

- Toma $i \in V$ y $S = \{i\}$.
- Mientras $|S| < k$:
  - Encuentra el punto j lo más alejado posible de los puntos en S (el que maximice $d(j, S)$).
  - añádelo a S.



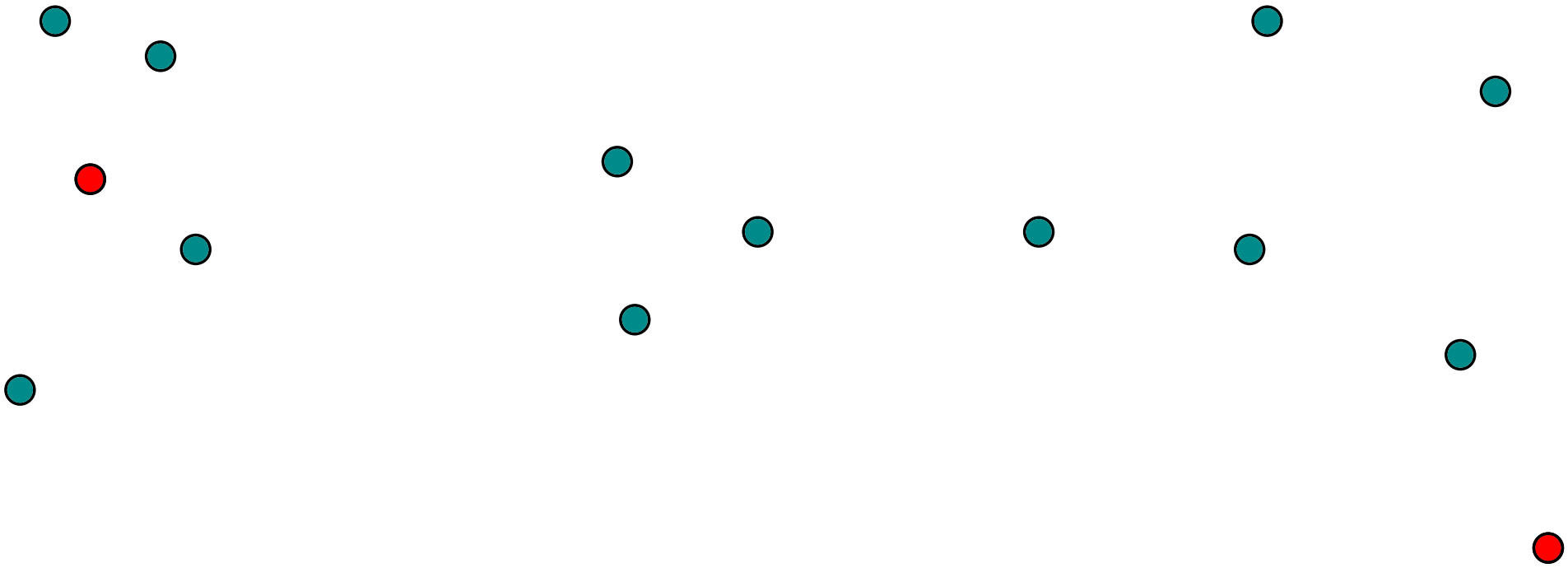
Algoritmo Glotón

- Toma $i \in V$ y $S = \{i\}$.
- Mientras $|S| < k$:
  - Encuentra el punto j lo más alejado posible de los puntos en S (el que maximice $d(j, S)$).
  - añádelo a S.



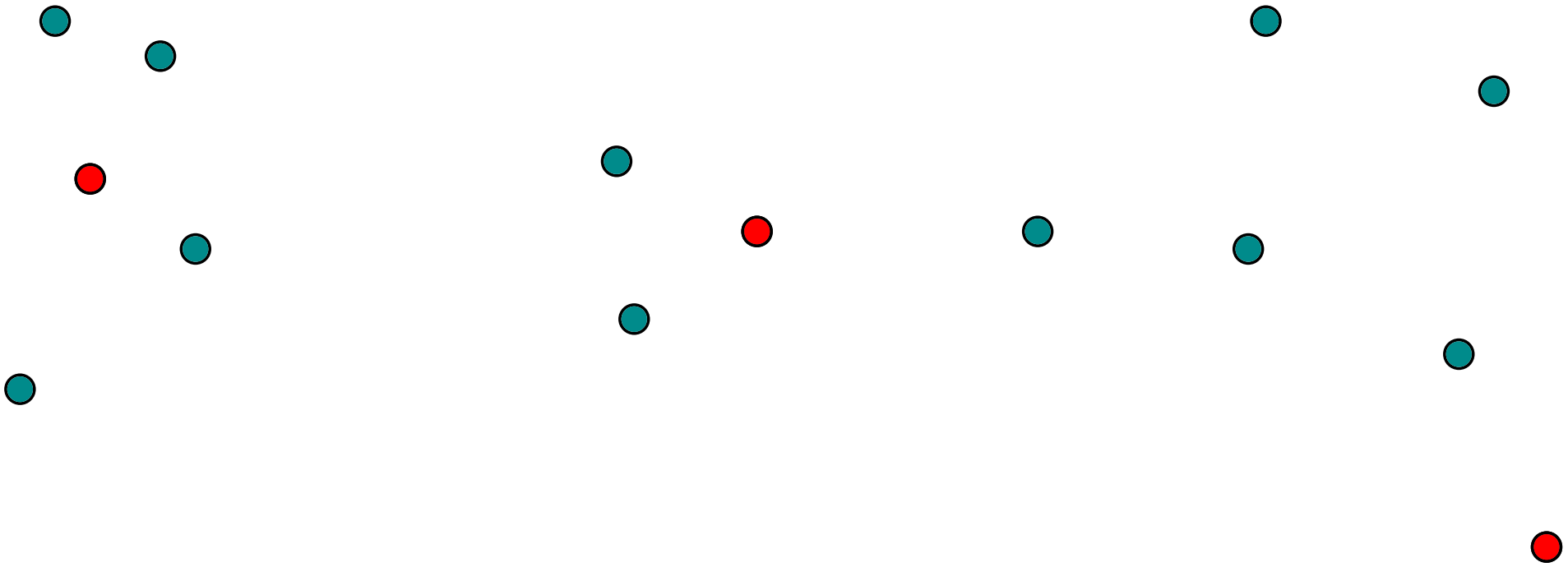
Algoritmo Glotón

- Toma $i \in V$ y $S = \{i\}$.
- Mientras $|S| < k$:
  - Encuentra el punto j lo más alejado posible de los puntos en S (el que maximice $d(j, S)$).
  - añádelo a S.



Algoritmo Glotón

- Toma $i \in V$ y $S = \{i\}$.
- Mientras $|S| < k$:
  - Encuentra el punto j lo más alejado posible de los puntos en S (el que maximice $d(j, S)$).
  - añádelo a S.



Algoritmo Glotón

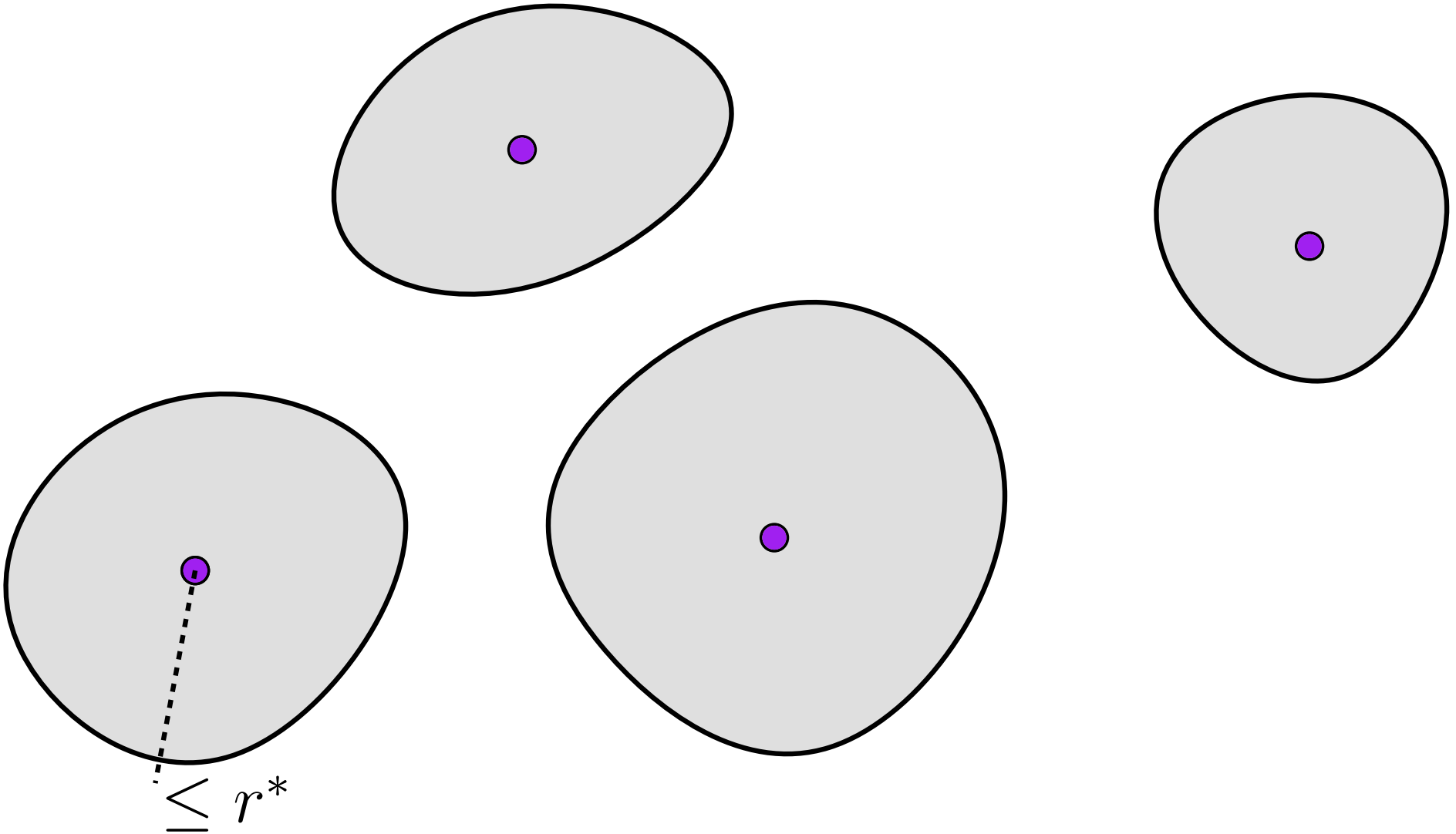
- Este algoritmo glotón es una 2-aproximación.
  - Obtiene una solución válida. ✓
  - Termina en tiempo polinomial. ✓
  - Está a un factor 2 del óptimo.

factor 2 del óptimo

Toma una solución óptima y su valor r^*. ■

factor 2 del óptimo

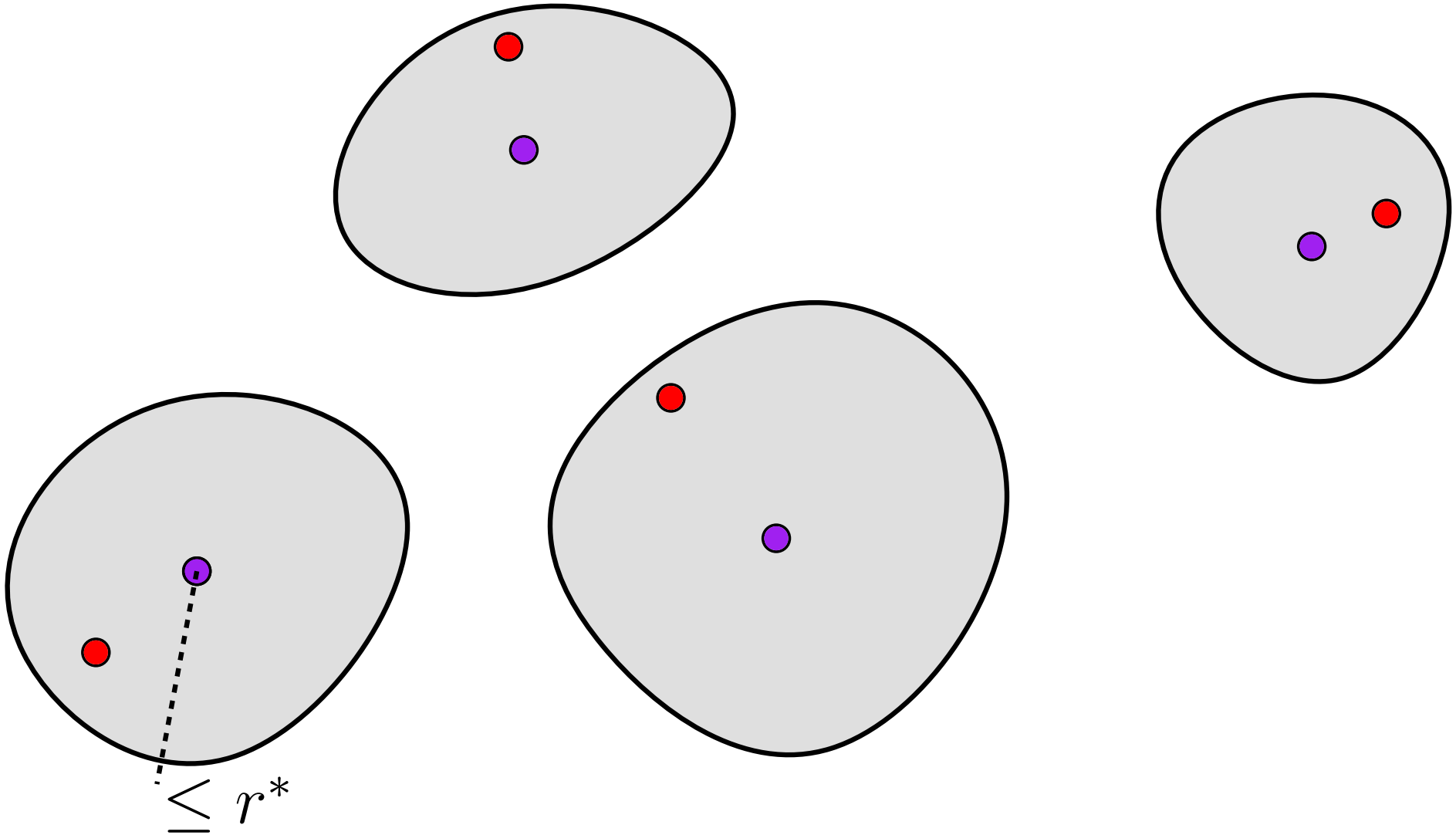
Toma una solución óptima y su valor r^*. ■



factor 2 del óptimo

Toma una solución óptima y su valor r^*.

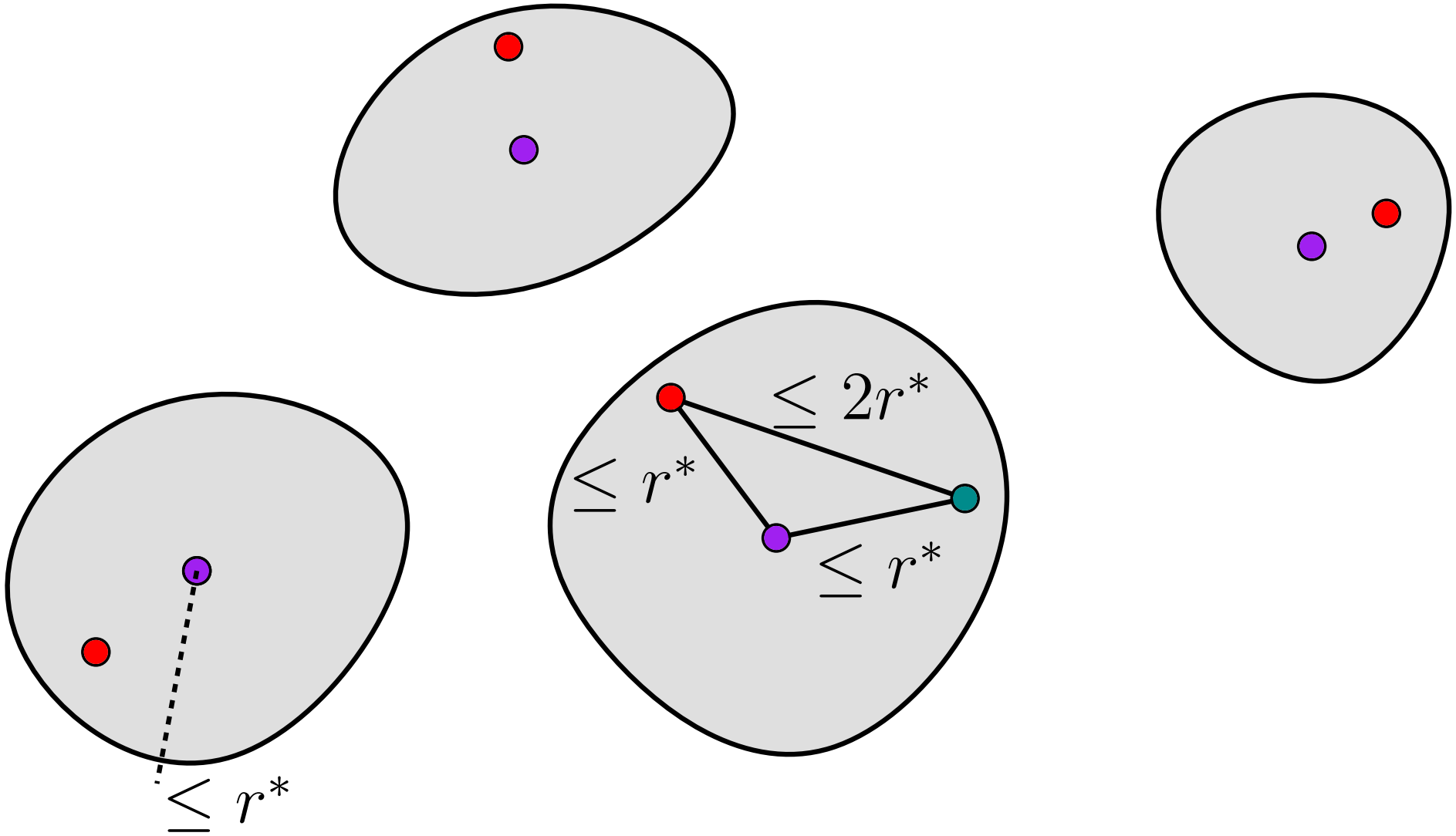
¿Dónde están los centros del glotón?



factor 2 del óptimo

Toma una solución óptima y su valor r^*.

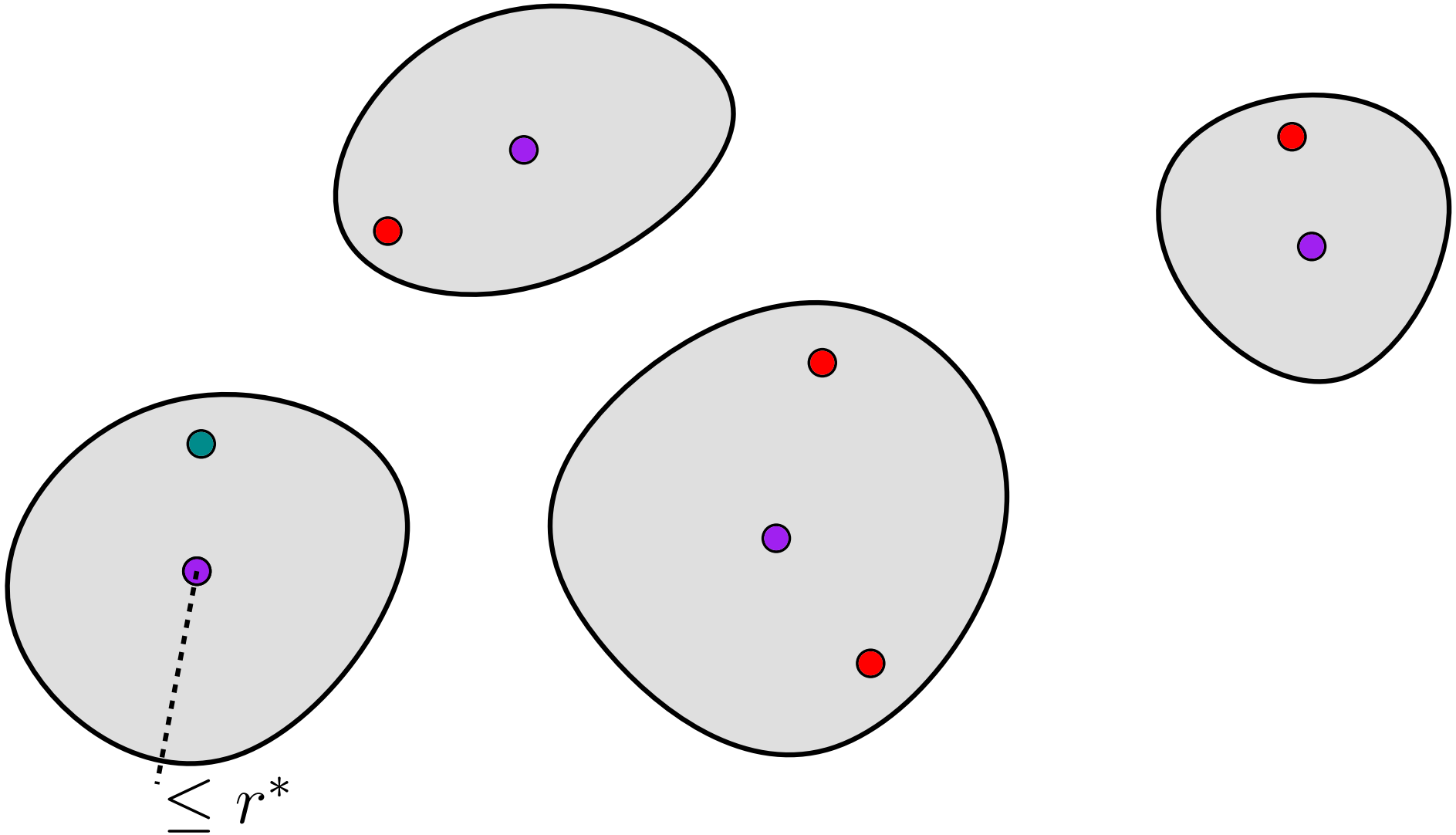
¿Dónde están los centros del glotón?



factor 2 del óptimo

Toma una solución óptima y su valor r^*.

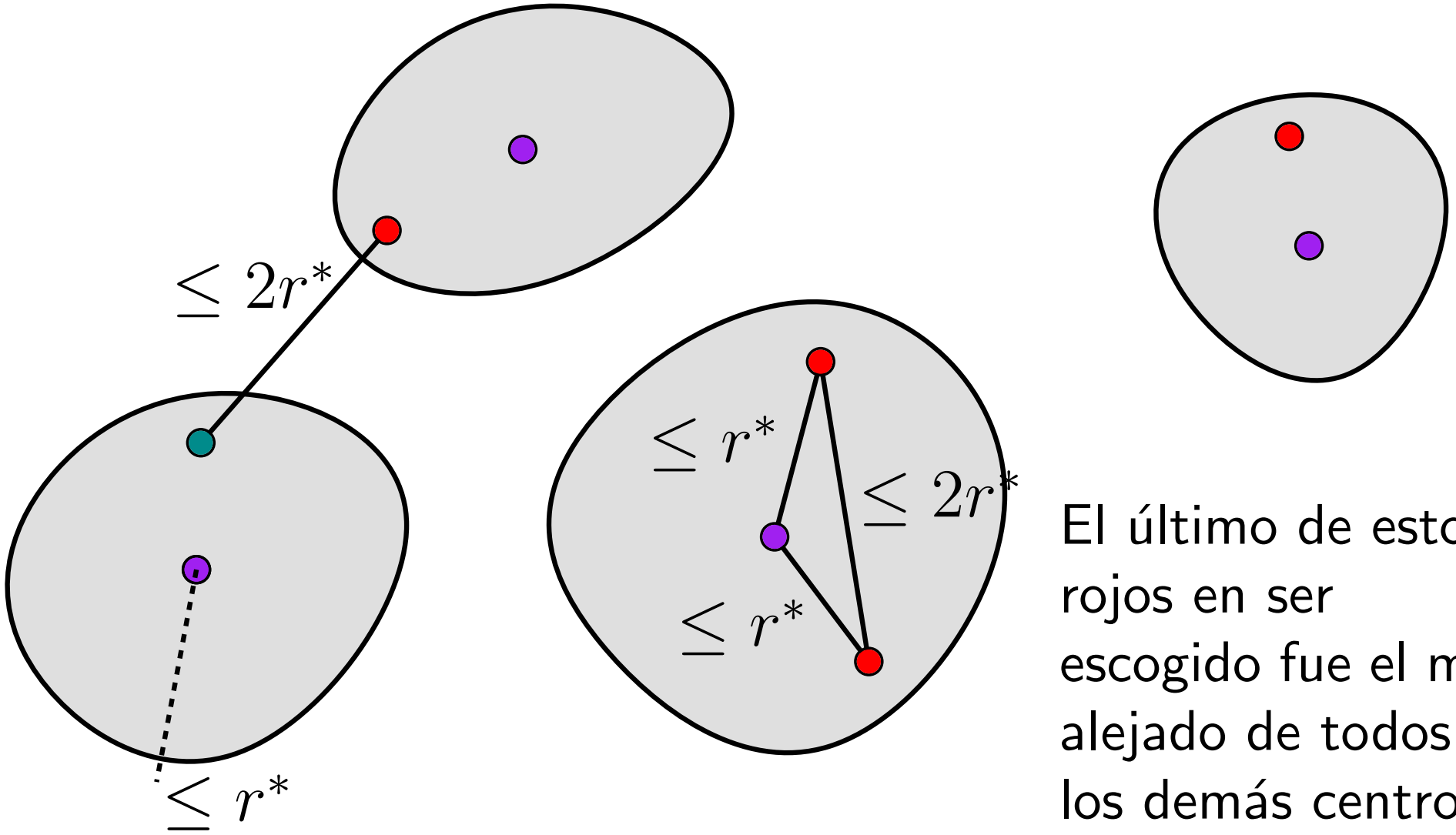
¿Dónde están los centros del glotón?



factor 2 del óptimo

Toma una solución óptima y su valor r^*.

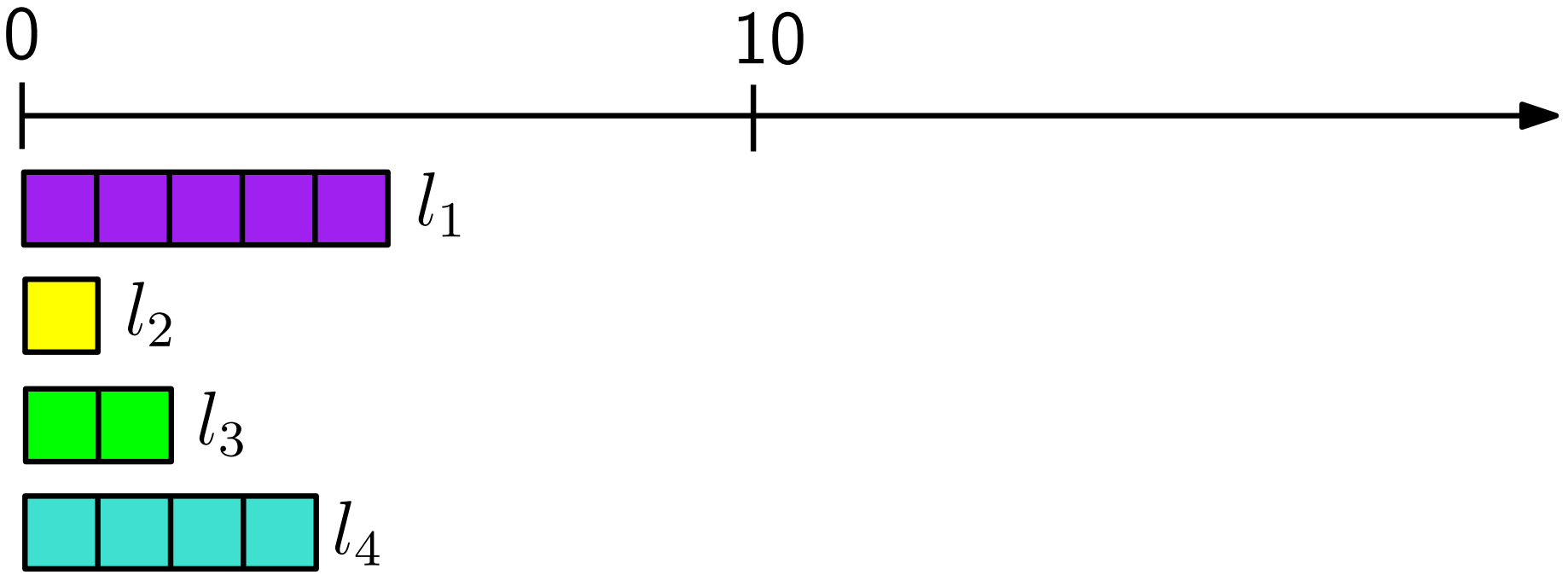
¿Dónde están los centros del glotón?



El último de estos rojos en ser escogido fue el más alejado de todos los demás centros.

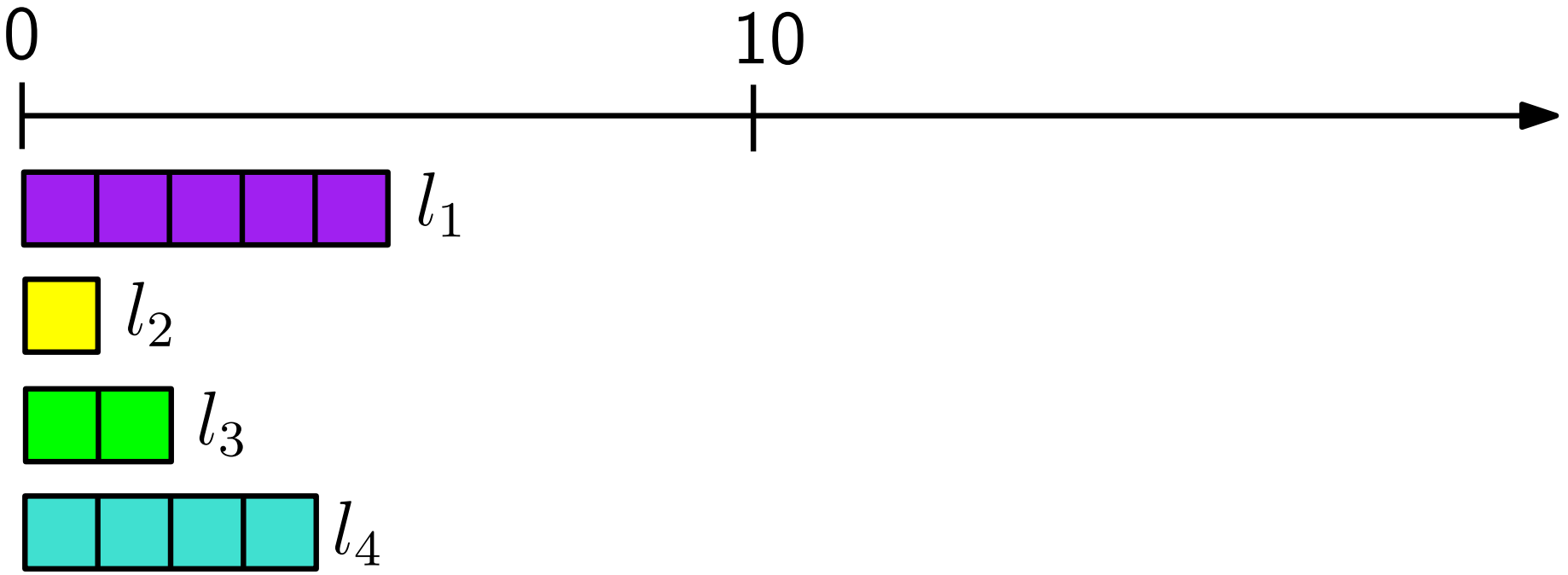
Planeación de Tareas en sólo una máquina con tiempos de vencimiento

Planeación de tareas en una máquina con vencimientos



- n tareas
- Cada tarea tiene un tiempo de procesamiento p_j, fecha de liberación l_j, fecha de vencimiento v_j.
- Cuando una tarea se ha empezado se debe completar.
- La planificación empieza en el tiempo 0.

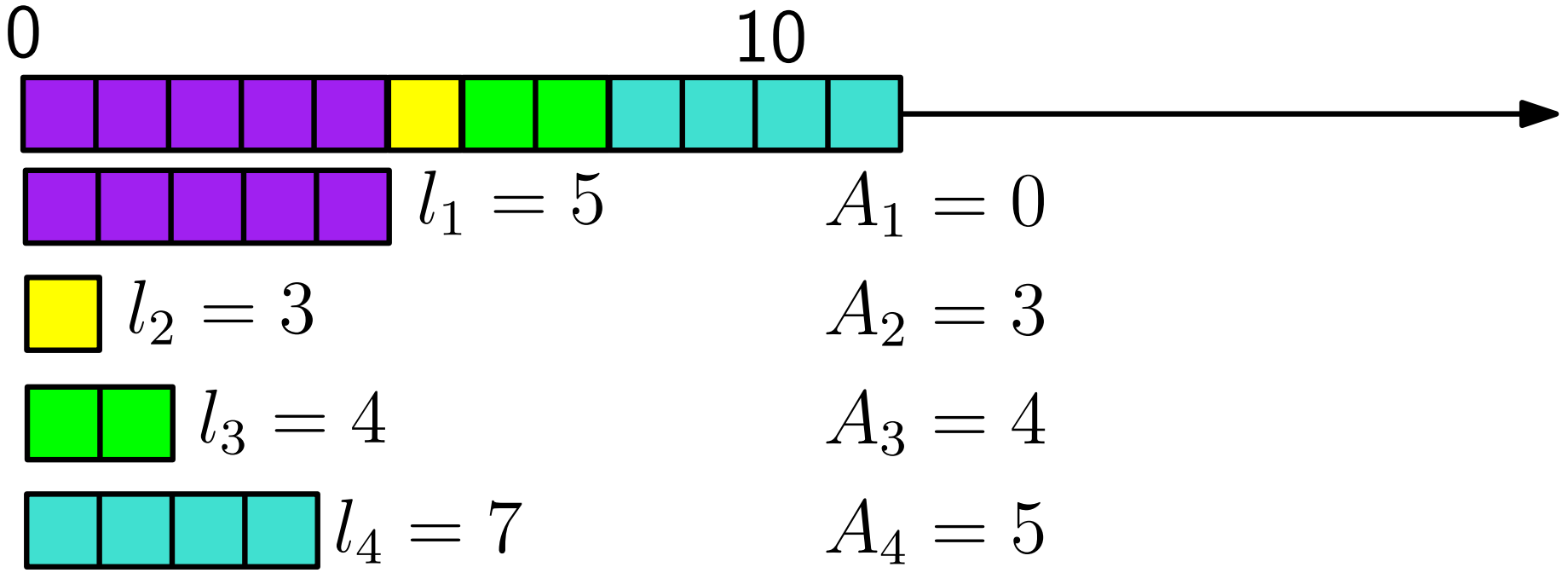
Planeación de tareas en una máquina con vencimientos



- Tiempo de finalización de la tarea $j = F_j$.
- El atraso de la tarea j es $A_j = F_j - v_j$
- La meta es planificar todos los trabajos minimizando el máximo atraso:

$$\text{minimiza } A_{max} = \text{máx}_{i=1\ldots n} A_j$$

Planeación de tareas en una máquina con vencimientos



- Tiempo de finalización de la tarea $j = F_j$.
- El atraso de la tarea j es $A_j = F_j - v_j$
- La meta es planificar todos los trabajos minimizando el máximo atraso:

$$\text{minimiza } A_{max} = \text{máx}_{i=1\ldots n} A_j$$

Planeación de tareas en una máquina con vencimientos

- NP-duro decidir si todas las tareas se pueden completar en su fecha de vencimiento.

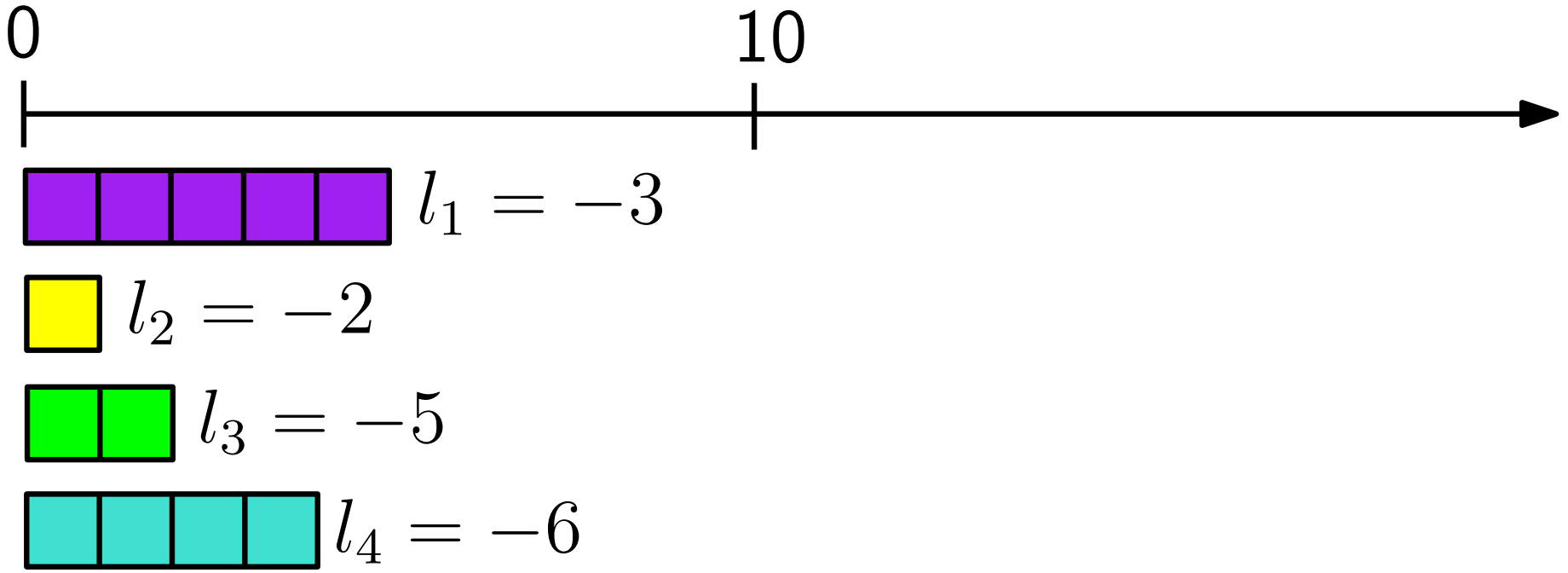
- Problema: supón que el valor óptimo es 0 entonces

    - Una $\alpha-$ aproximación debe encontrar una solución con valor a lo más $\alpha * 0 = 0$.

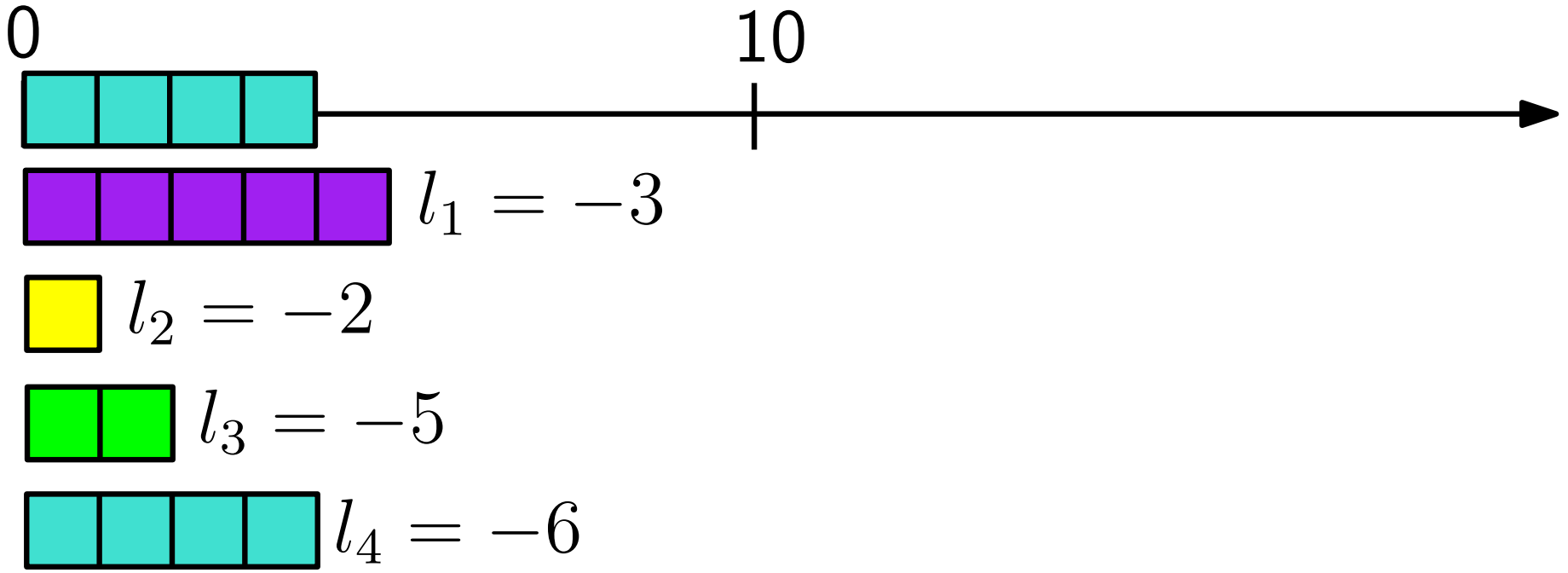
    - No existe dicho algoritmo a menos que P=NP.

    - Solución: Se asume que todos los tiempos de vencimiento son negativos, así el valor óptimo siempre es positivo.

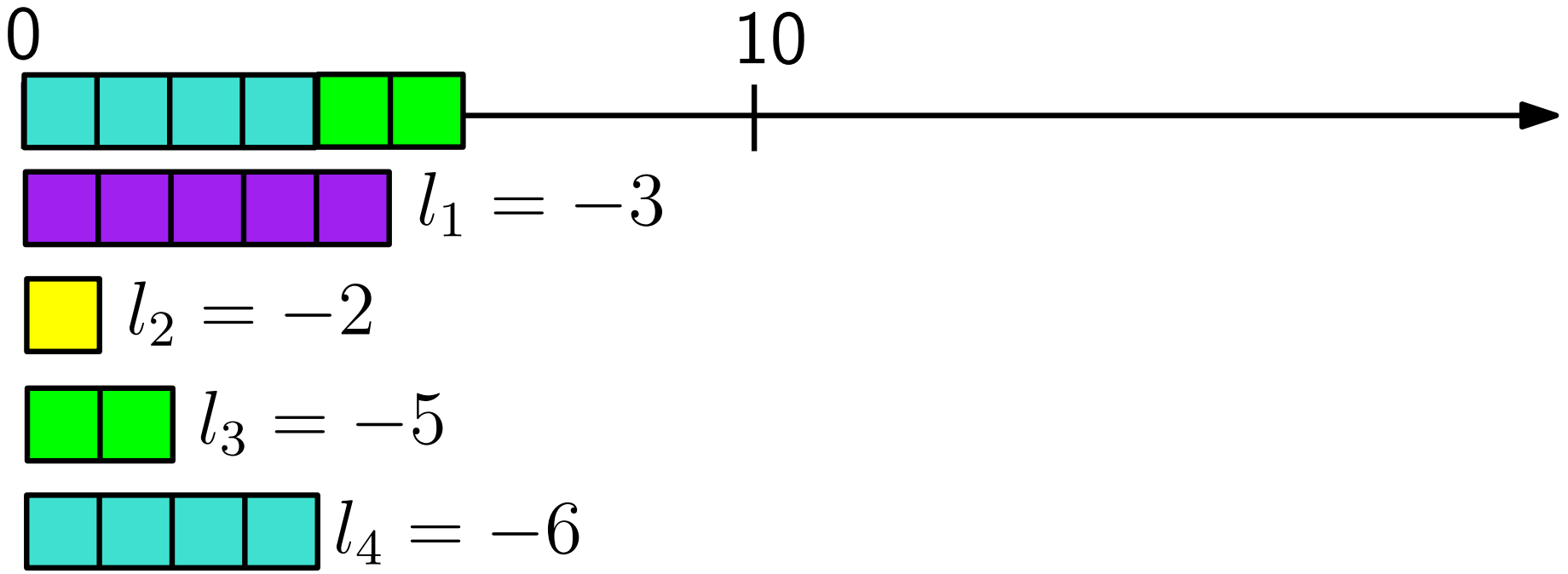
Fecha de vencimiento más temprana



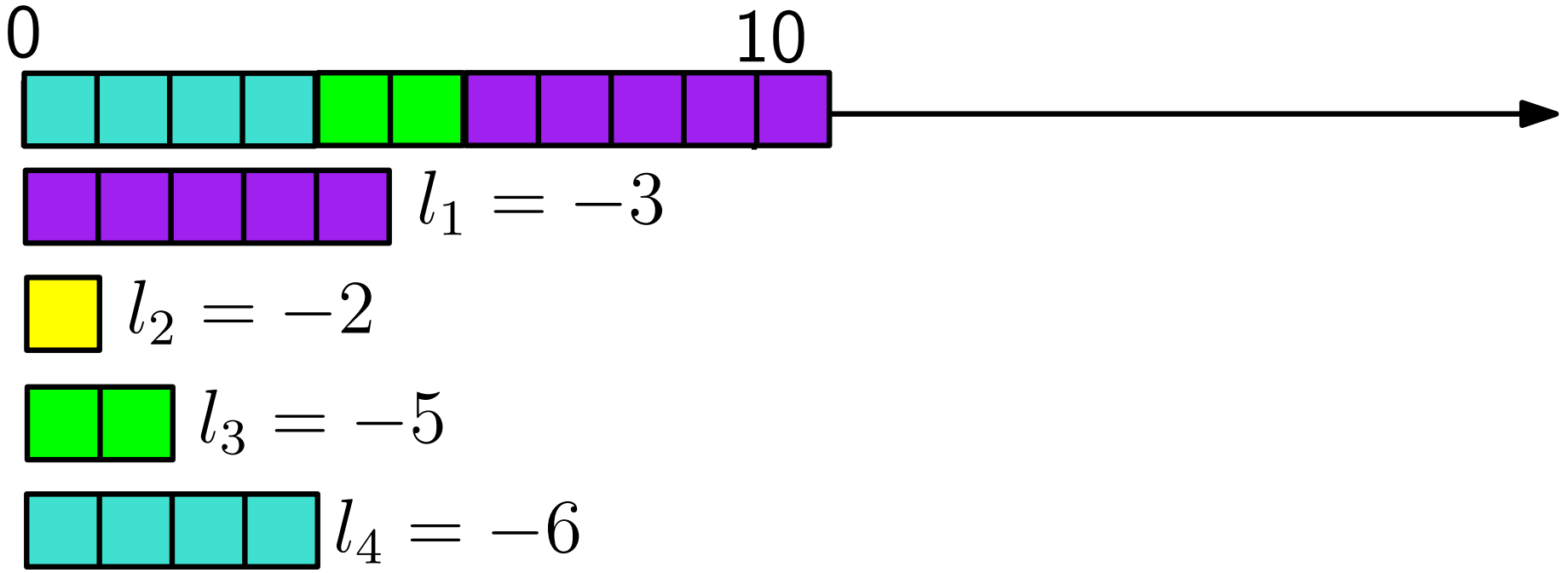
Fecha de vencimiento más temprana



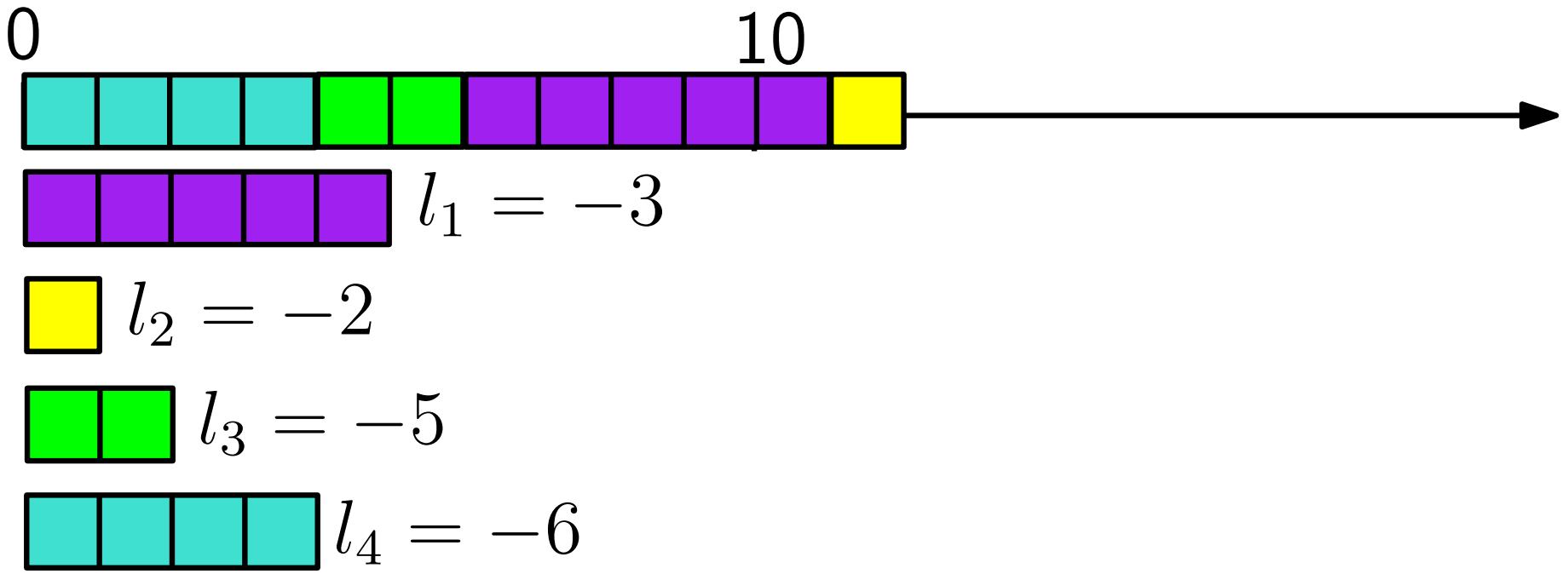
Fecha de vencimiento más temprana



Fecha de vencimiento más temprana



Fecha de vencimiento más temprana



Algoritmo glotón

Fecha de vencimiento más temprana

- Este algoritmo glotón es una 2-aproximación.
  - Obtiene una solución valida. ✓
  - Termina en tiempo polinomial. ✓
  - Está a un factor 2 del óptimo.

Cota inferior

- Sea S un subconjunto de tareas.
    - $l(S) = \text{mín}_{j \in S}\, l_j$
    - $p(S) = \sum_{j \in S} p_j$
    - $v(S) = \text{máx}_{j \in S}\, v_j$
    - Valor óptimo A^*

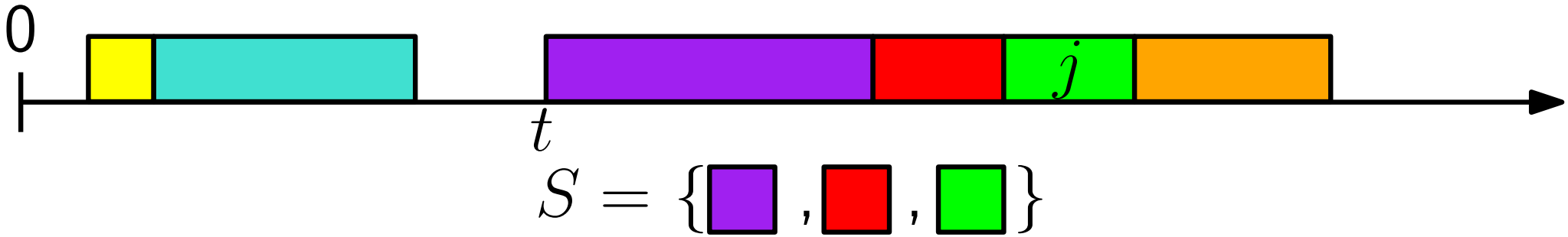
- Proposición. Para cualquier subconjunto S de tareas: $A^* \geq l(S) + p(S) - v(S)$.

Cota inferior

- Proposición. Para cualquier subconjunto S de tareas: $A^* \geq l(S) + p(S) - v(S)$.

- Demostración.
  - Considera la planeación óptima y fíjate en S.
  - Ninguna tarea se puede procesar antes de $l(S)$.
  - El tiempo de procesamiento necesarios es $p(S)$.
  - La última tarea a ser procesada, no se puede completar antes de $l(S) + p(S)$.
  - $v_i \leq v(S)$ entonces el atraso de i es al menos $l(S) + p(S) - v(S)$.
  - $A^* \geq A_i$

Factor de aproximación



- j: trabajo con máximo atraso $(A_{\text{máx}} = A_j = F_j - v_j)$.

- t: La última vez que estuvo desocupada la máquina antes de F_j.

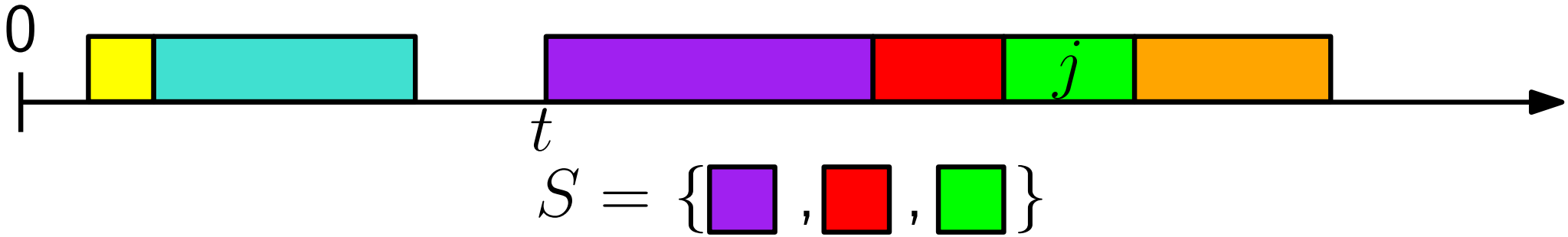
- S: conjunto de trabajos en el intervalo $[t, F_j]$

- Tenemos:

$$l(S) = t \text{ y } p(S) = F_j - t.$$

$$F_j = p(S) + t = p(S) + l(S).$$

Factor de aproximación



- Usando la proposición:

$$A^* \geq l(S) + p(S) - v(S) \geq l(S) + p(S) = F_j.$$

$$A^* \geq l_j + p_j - v_j \geq -v_j.$$

- $A_{\text{máx}} = F_j - v_j \leq 2A^*$.