

Tema 5

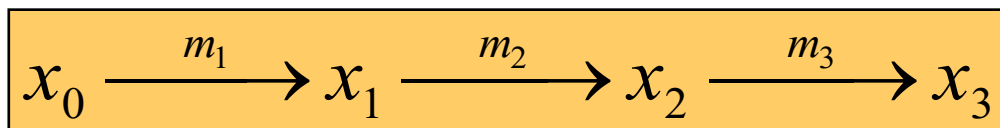
Planificación y Asignación Optimizada de Recursos



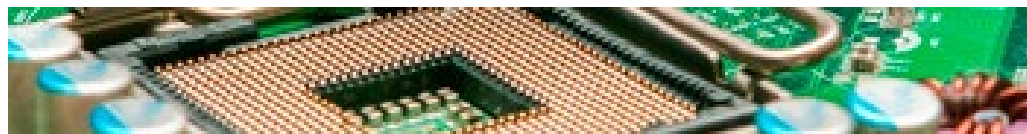
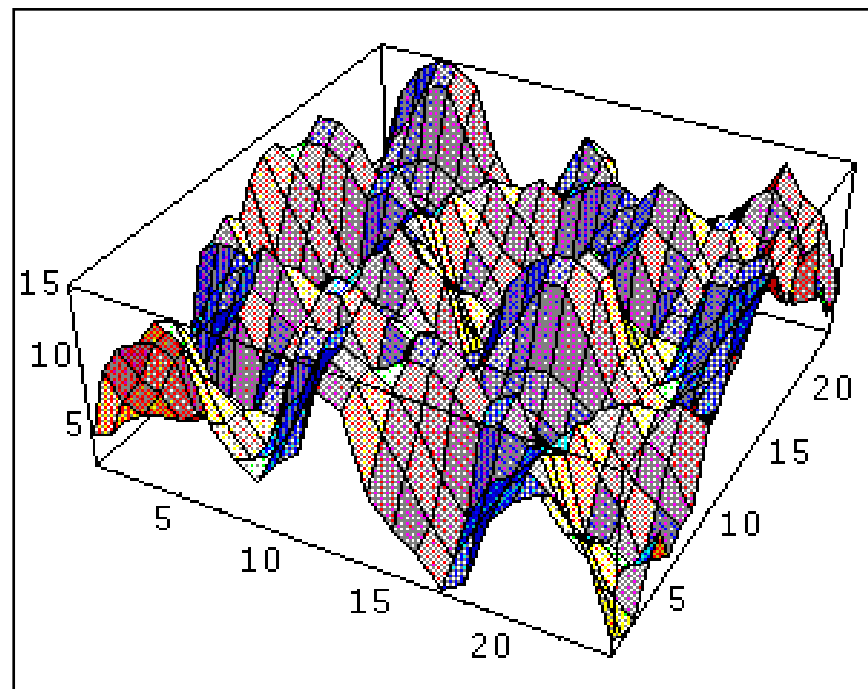
- Búsqueda local
- Metaheurísticas para problemas de secuenciación



- **Definición:** Cada solución x tiene un conjunto de soluciones asociadas $N(x)$, que se denomina **entorno** de x .
- **Definición:** Dada una solución x , cada solución x' de su entorno $N(x)$ puede obtenerse directamente a partir de x mediante una operación llamada **movimiento**.
- El método se basa en explorar el entorno de una solución y seleccionar una nueva solución en él (i.e. realizar el movimiento asociado). Desde la nueva solución se explora su entorno y se repite el proceso.



- **Greedy:** Seleccionar la solución con **mejor evaluación** de la función objetivo, siempre que sea mejor que la actual.
- El algoritmo se detiene cuando la solución no puede ser mejorada.
- A la solución encontrada se le denomina **óptimo local** respecto al entorno definido.
- Miopía del Método



- Componentes de la búsqueda local:
 - Representación de una solución
 - Permutación de n trabajos para el problema de una única máquina
 - Secuencia en cada máquina (job-shop)
 - Tiempos de inicio y de fin de cada tarea..
 - Espacio de búsqueda de soluciones S
 - Todas las posibles permutaciones de trabajos ($n!$)
 - Todas las posibles permutaciones de trabajos en cada máquina $(n!)^m$



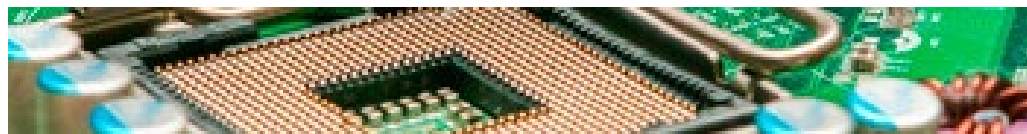
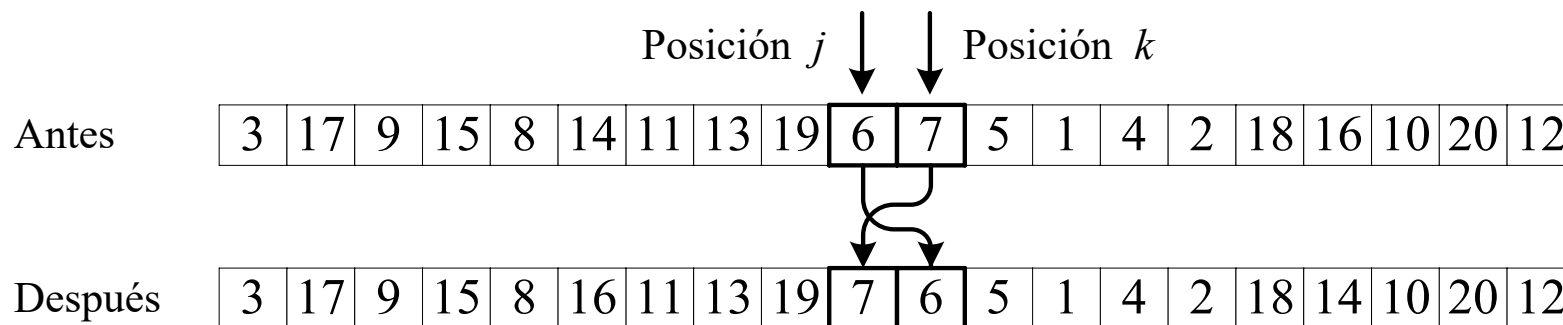
- Relación de vecindad $N \subseteq S \times S$
 - Vecindario de inserción
 - Vecindario de intercambio
 - Vecindario de intercambio adyacente...
 - $N(s)$: vecinos de la solución s
- Función de evaluación $f: S \rightarrow \mathbb{R}$
 - Función objetivo, a partir de una solución obtener un valor (makespan, total weighted tardiness...)
 - $f(s)$: valor de la función objetivo para la solución s



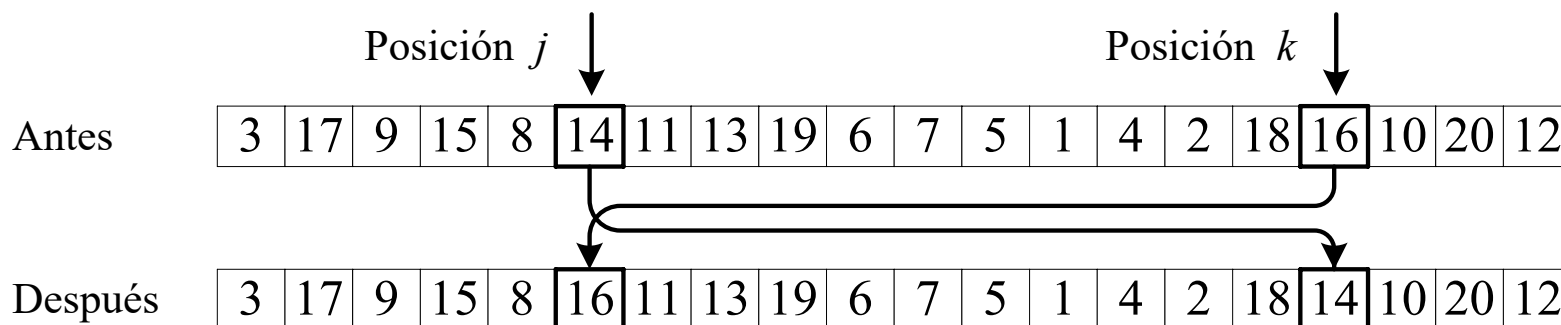
- Realización de la búsqueda
 - ¿Cómo buscar en el vecindario?
 - ¿Buscamos todos los vecinos y nos quedamos con el mejor o nos quedamos con el primero que mejore la función de evaluación?
- Criterio de aceptación
 - ¿Aceptamos solo soluciones que mejoren la función de evaluación?
 - ¿Permitimos aceptar soluciones peores de manera temporal? En ese caso... ¿cuánto peores?



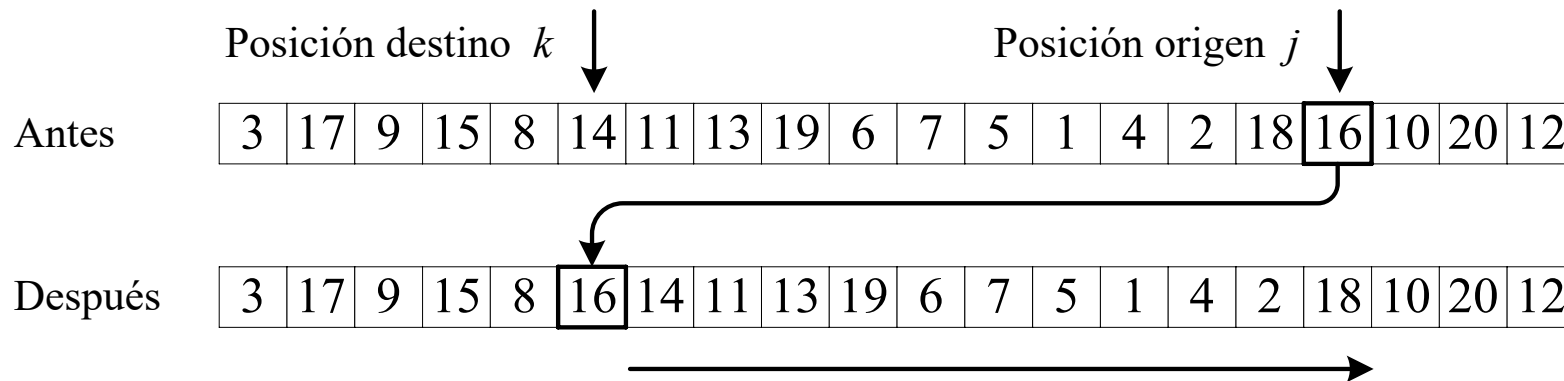
- Tipos de vecindarios N más comunes en scheduling
 - Vecindarios de intercambio adyacente o “swap”



- Vecindarios de intercambio o “interchange/exchange”



■ Vecindarios de inserción o “insert”



■ ¡La definición del vecindario es dependiente del problema!



■ Procedimiento simple de búsqueda local (descent search / Iterative improvement)

Determinar solución inicial x

Mientras mejorado=true

 mejorado=false

 escoger un vecino x' de x

 si $f(x') < f(x)$ entonces

$x = x'$

 mejorado=true



- Ventajas:
 - Fácil de implementar una vez los componentes de la búsqueda local están claros
 - Rápido si generar vecinos y evaluar solución es eficiente
 - General y fácil de entender
- Desventajas
 - **Estancamiento en óptimos locales**
 - Miopía en la búsqueda
 - Puede ser lento para algunos tipos de problemas
 - Búsqueda en círculos



- Sencillos mecanismos para escapar de óptimos locales
 - Reinicializar la búsqueda desde otra solución inicial
 - Aceptar en algún momento de la búsqueda peores soluciones
 - No hay garantías de que ninguno de estos mecanismos permita escapar de los óptimos locales en todas las situaciones y casos



- En la mayoría de los problemas de scheduling los individuos de una población se representan mediante permutaciones de los trabajos

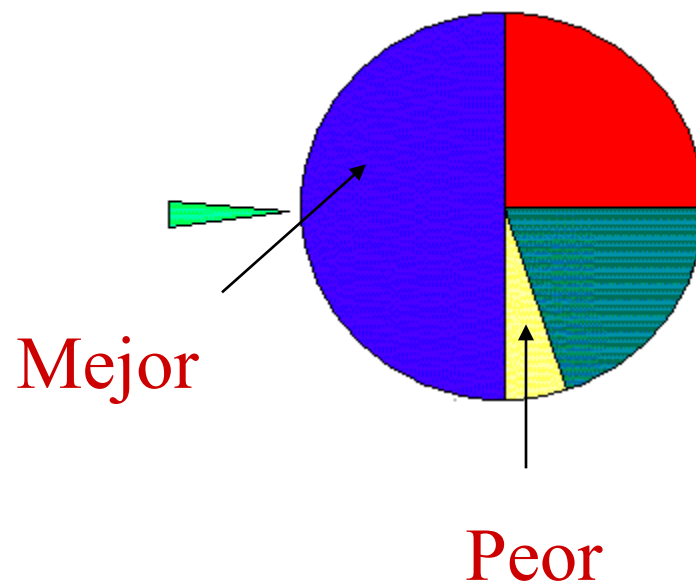
3	17	9	15	8	14	11	13	19	6	7	5	1	4	2	18	16	10	20	12
---	----	---	----	---	----	----	----	----	---	---	---	---	---	---	----	----	----	----	----

- El valor de adecuación de cada individuo suele ser directamente el valor de la función objetivo
- Se puede mapear

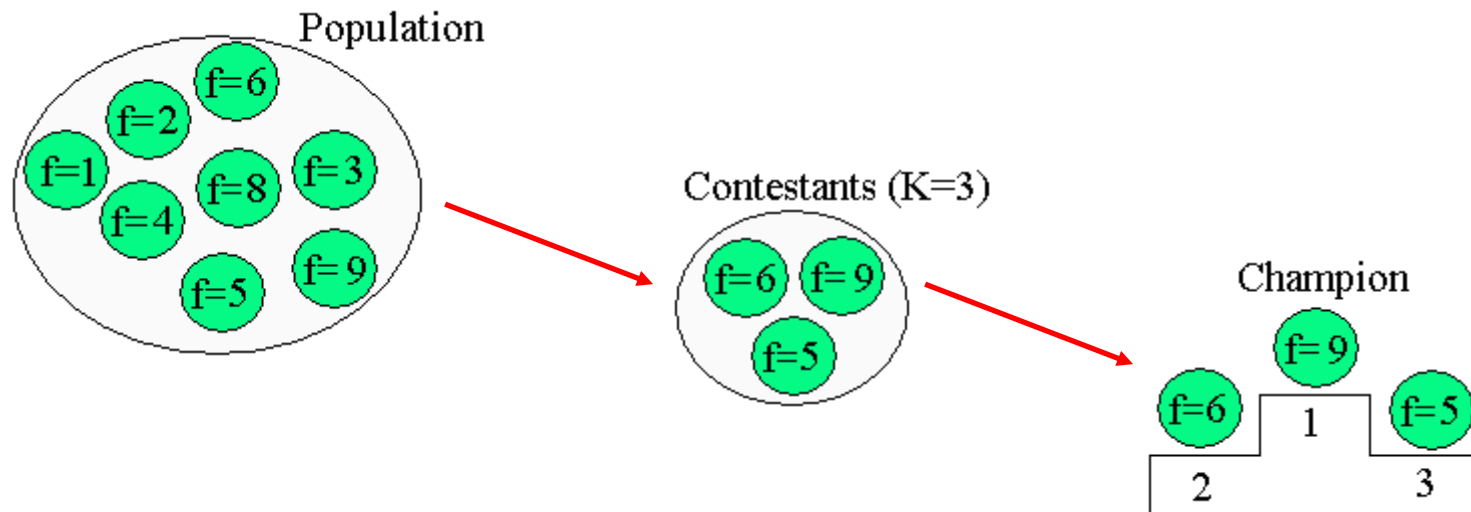


- La selección se suele hacer de manera que aquellos individuos con mayor valor de adecuación tengan mayor probabilidad de ser seleccionados
- **Presión** vs. **Diversidad** en la selección
- Selección simple por **ruleta**:

$$P_{select_i} = \frac{f(i)}{\sum_{j=1}^{tam_pob} f(j)}$$



- Selección por **torneo** :
 - Se extraen k individuos por el método de ruleta y se selecciona el mejor
 - Mayor presión que con la ruleta
 - Normalmente $k=2$



- Selección por **ranking** :
 - Los elementos se ordenan de mejor a peor valor de adecuación
 - La probabilidad de selección es directamente proporcional a la posición en la ordenación



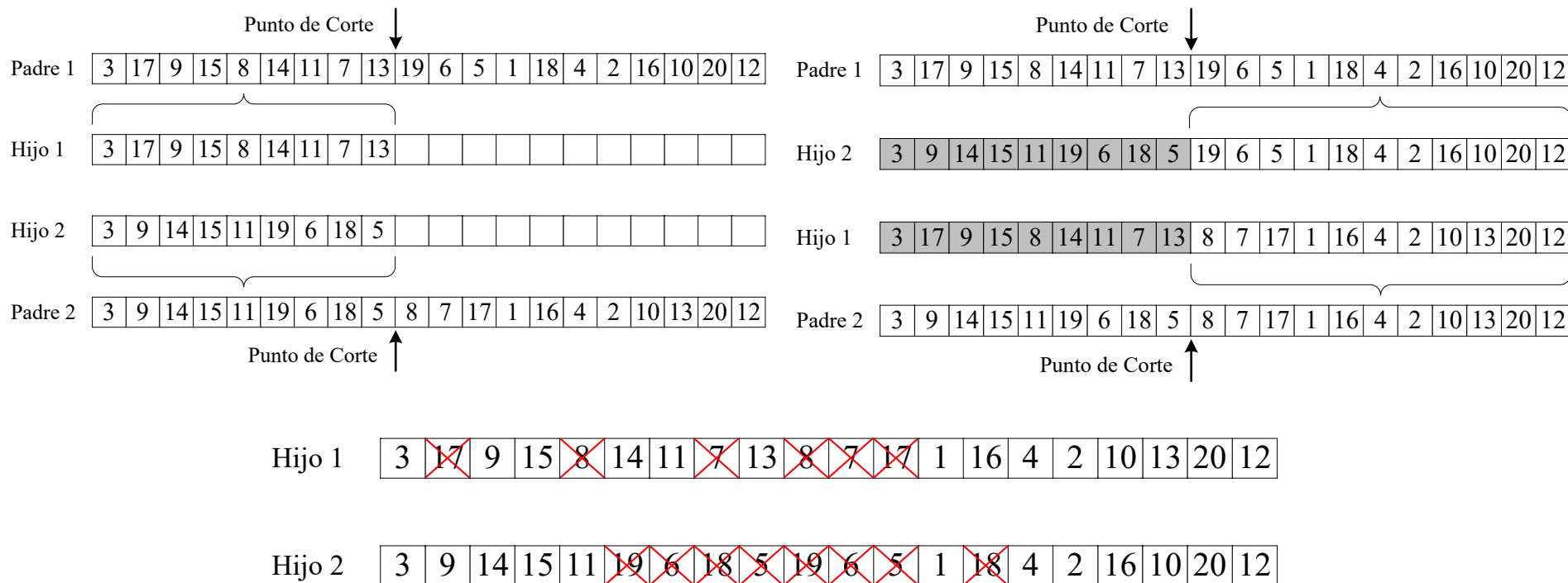
■ Cruce:

- Los individuos seleccionados se emparejan y se cruzan
- La idea es generar nuevas soluciones que contengan las mejores características de los padres
- El cruce es dependiente del problema
- Decenas de cruces distintos propuestos para problemas de scheduling
- Cruce de un punto, de dos puntos, uniforme, PMX, UOX...

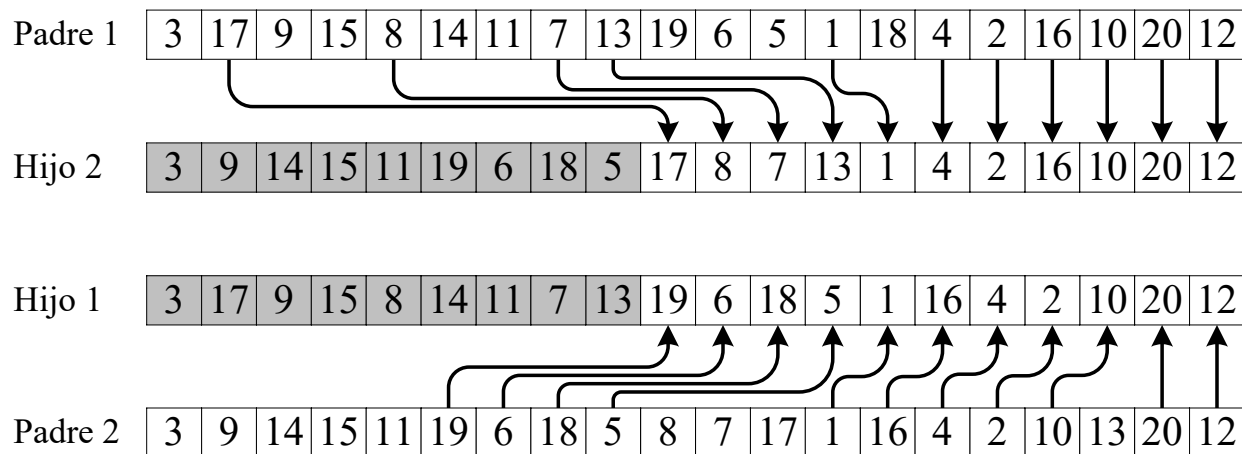


Ejemplo de cruce de un punto

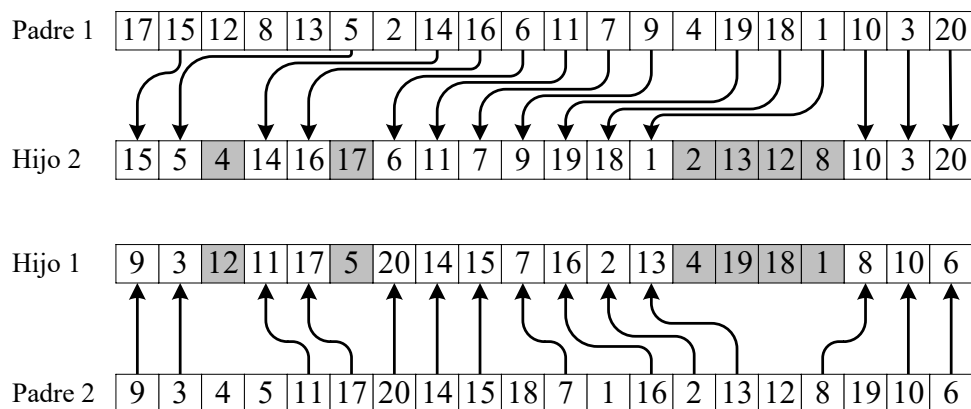
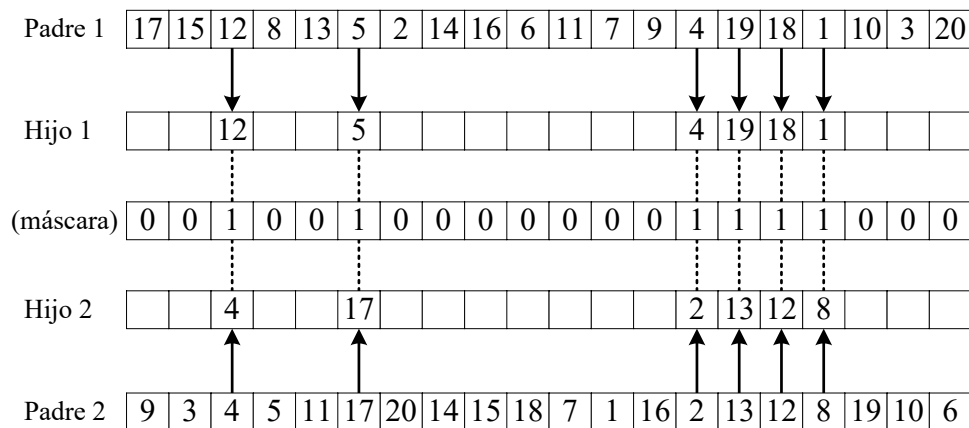
Cruces convencionales no funcionan con permutaciones



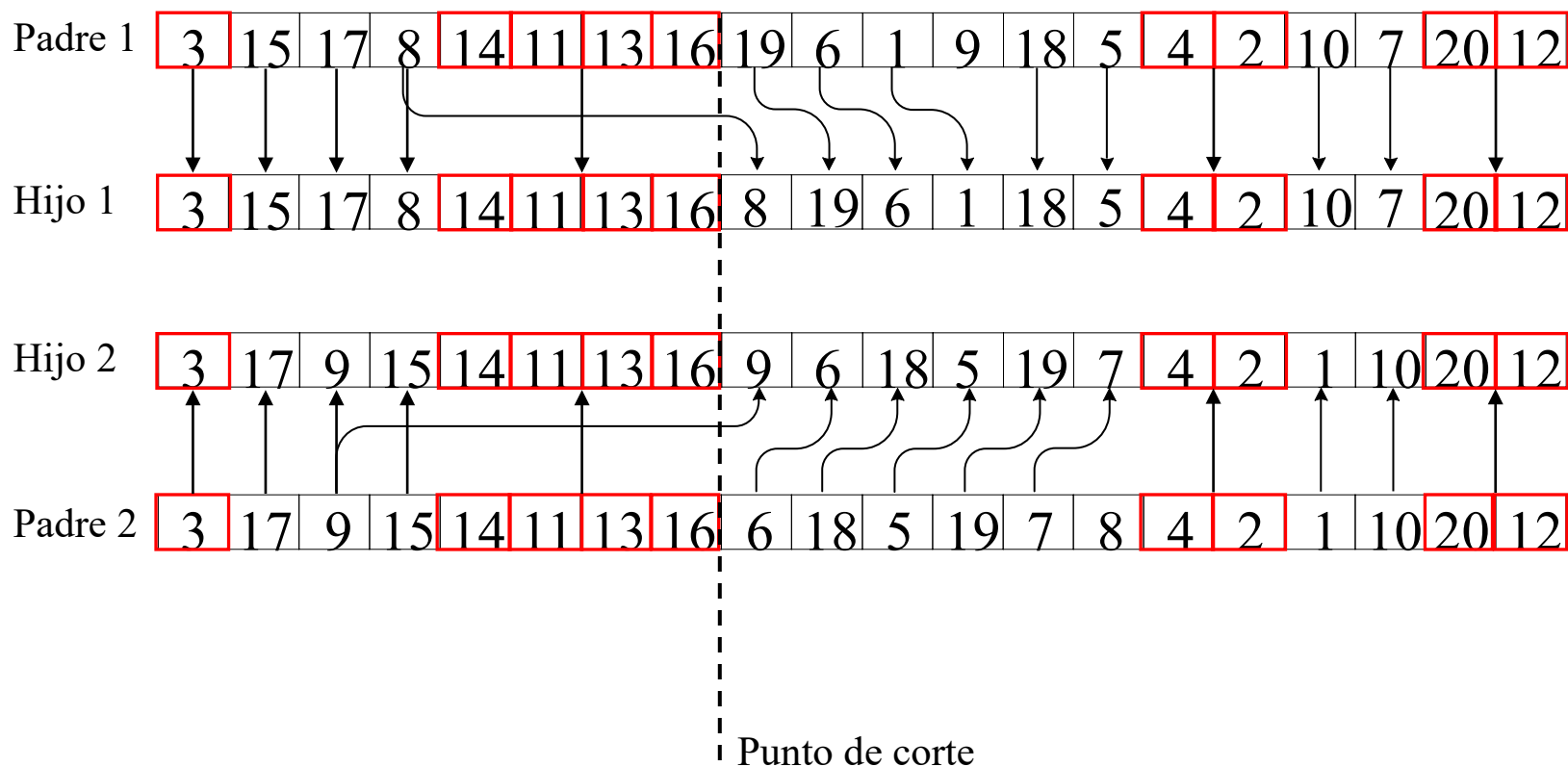
■ Cruce de un punto por orden



■ Cruce uniforme por orden



■ Nuevos tipos de cruce: Similar Job Order Crossover (SJOX):



- Normalmente los nuevos individuos generados tras el cruce se mutan
 - La mutación se hace normalmente aplicando uno o más movimientos a cada individuo dentro de algún tipo de vecindario
- Los nuevos individuos tras la mutación reemplazan normalmente a los padres en la población
 - O reemplazan a los peores individuos
 - Tamaño de población constante vs. no constante...
- Cientos de papers para Gas, Goldberg (1989), Michalewicz (1994)



- Búsqueda Local Iterativa o ILS
- Principio muy simple
- Buenos resultados

Determinar solución inicial x

$x' = \text{busqueda_local}(x)$

$\text{mejor} = x'$

Mientras no *criterio_parada*

$y = \text{perturbar}(\text{mejor})$

$y' = \text{busqueda_local}(y)$

Si *criterio_aceptación*(y')

$\text{mejor} = y'$



- Solución inicial: puede ser aleatoria o la obtenida con alguna heurística eficaz
- Perturbar: cambiar o “mutar” la mejor solución obtenida hasta el momento
 - Perturbación fuerte: búsqueda aleatoria
 - Perturbación débil: dificultad para escapar de óptimos locales
- Criterio aceptación
 - Simple: solo se aceptan mejores soluciones
 - Complejo: parecido a SA



- Ejemplo para el taller de flujo $F|\mu|C_{\max}$:
 - Solución inicial: NEH
 - Búsqueda local: vecindario de inserción
 - Perturbación:
 - Dos movimientos de inserción y uno de intercambio
 - Criterio de aceptación
 - Aceptar solo si la nueva solución es mejor que la anterior
 - Aceptar de acuerdo a SA con una temperatura T constante
- Stützle (1998)



■ Basado en ILS

Determinar solución inicial x

$x' = \text{busqueda_local}(x)$ (opcional)

$\text{mejor} = x'$

Mientras no *criterio_parada*

$y = \text{destruir}(\text{mejor})$

$y' = \text{reconstruir}(y)$

$y'' = \text{busqueda_local}(y')$ (opcional)

Si *criterio_aceptación*(y'')

$\text{mejor} = y''$

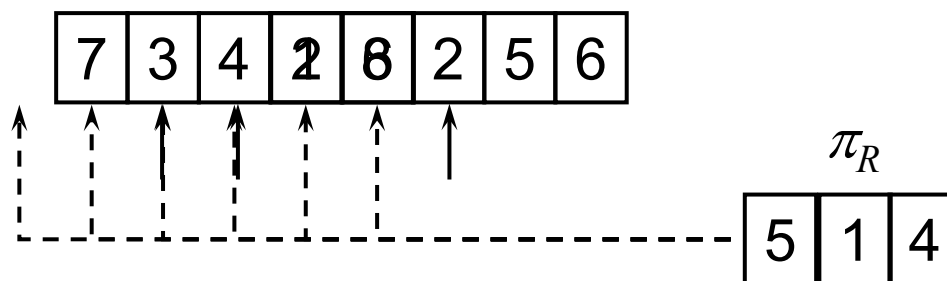


- En vez de perturbar una solución como en ILS o de reconstruirla completamente como en GRASP, en IG se eliminan algunos componentes de la solución para luego reinsertarlos aplicando alguna heurística constructiva
- Búsqueda local y criterio de aceptación muy parecidos a ILS










- Ejemplo para el $F|prmu|C_{\max}$:
 - Solución inicial por NEH
 - Búsqueda local en el vecindario de inserción

Fase
de
destrucción



Método	NEHT	GA_RMA	HGA_RMA	SA_OP	SPIRIT	GA_CHEN	GA_REEV
DRPM	3.35	1.13	0.57	2.37	5.09	4.83	1.61

Método	GA_MIT	ILS	GA_AA	M_MMAS	PACO	IG_RS	IG_RS _{LS}
DRPM	2.42	1.06	2.28	0.88	0.75	0.78	0.44

	SA		GA		IG		OTROS/HÍBRIDOS
	TS		ACO		ILS		



- Algoritmos de Colonias de Hormigas (ACO):
- Algoritmos que simulan el comportamiento de las hormigas en su búsqueda de alimento
- Dorigo et al. (1996)
- Población de soluciones u **hormigas**
- Cada hormiga es una solución que se construye con alguna heurística y de acuerdo a las **feromonas** depositadas por otras hormigas
- Si la solución (hormiga) es buena, depositará más feromona, influyendo más en las siguientes hormigas



- Existen muchas más metaheurísticas y su número no para de crecer
- Situaciones híbridas: TS+SA, GA+LS,...
- Grandes similitudes entre la mayoría de los métodos
- Algunos:
 - Variable Neighbourhood Search (VNS)
 - Constructive Genetic Algorithms (CGA)
 - Dynamic Local Search (DLS)
 - Adaptive Iterated Construction Search (AICS)
 - Artificial Immune Systems (AIS)
 - Particle Swarm Optimization (PSO)
 - ...



- Metaheurísticas:
 - Más eficaces y más generales que las heurísticas
 - Buenas implementaciones muy cerca de las soluciones óptimas
 - Mucho más complicadas
 - Suelen ser lentas (mucho más que las heurísticas en casi todos los casos, pero más rápidas que los métodos exactos)
 - Hay todo un mundo ahí fuera
 - No muy extendidas, lejos del software comercial
 - Estado del arte en lo que se refiere a scheduling



- Cerný, V. (1985). “A thermodynamical approach to the traveling salesman problem”, *Journal of Optimization Theory and Applications*, 45(1):45-51
- Dorigo, M., Maniezzo, V. y Coloni, A. (1996). “Ant system: Optimization by a colony of cooperating agents”. *IEEE Transactions on Systems, Man, and Cybernetics – Part B*, 26(1):29-41
- Feo, T. A. y Resende, M. G. C. (1995). “Greedy randomized adaptive search procedures”. *Journal of Global Optimization*, 6:109-133
- Glover, F. (1989). “Tabu Search – part I”. *ORSA Journal on Computing*, 1(3):190-206
- Glover, F. (1990). “Tabu Search – Part II”. *ORSA Journal on Computing*, 2(1):4-32
- Goldberg, D. E. (1989). “Genetic Algorithms in Search, Optimization, and Machine Learning”. Addison-Wesley, Reading, MA, USA
- Kirkpatrick, S., Gelatt Jr., C. D., Vecchi, M. P. (1983) “Optimization by Simulated Annealing”. *Science*, 220:671-680



Metropolis, N, Rosenbluth, A. W., Rosenbluth, M, N., Teller, A. y Teller, E. (1953). "Equation of state calculations by fast computing machines". *Journal of Chemical Physics*, 21: 1087-1092

Michalewicz, Z. (1994). "Genetic Algorithms + Data Structure = Evolution Programs", segunda edición. Springer-Verlag. Berlin, Alemania

Stützle, T. (1998). "Applying Iterated Local Search to the Permutation Flowshop Problem". Informe Técnico AIDA-98-04. FG Intellektik, TU Darmstadt, FB Informatik, Darmstadt, Alemania

