Metaheurísticas

Búsqueda local, recocido simulado y búsqueda tabú.

Ángel García Baez & Ulises Jiménez Guerrero

Optimización inteligente
Maestría en Inteligencia Artificial

13 de diciembre de 2024

Contenido

Heurísticas

Definición

Una heurística es una técnica de búsqueda directa que utiliza reglas prácticas (intuición) para localizar soluciones mejoradas. La ventaja de la heurística es que en general determina (buenas) soluciones con rapidez, utilizando reglas simples. La desventaja es que la calidad de la solución (con respecto a la óptima) suele desconocerse. [Taha, 2012]

Las primeras generaciones de heurísticas se basan en la regla de **búsqueda codiciosa**.

Metaheurísticas

El término fue acuñado por Fred Glover en 1986, y existen diferentes definiciones [Zakizadeh and Erfani, 2024].

- Un proceso iterativo que combina inteligentemente algunos heurísticos para buscar e investigar en todo el espacio de búsqueda. (Laporte & Osman).
- Un *framework* algorítmico de alto nivel, independiente del problema, que da una serie de guías o estrategias para desarrollar algoritmos de optimización heurísticos. (Sörensen & Glover).

Definición

Metaheurística

Un proceso iterativo de alto nivel que modifica y categoriza soluciones heurísticas de bajo nivel para generar soluciones de alta calidad. También puede manipular una o más soluciones completas o incompletas, iterativamente. (Voss et al).

¿Por qué necesitamos metaheurísticas?

- Su objetivo principal es ayudar a los métodos heurísticos a escapar de los óptimos locales.
- Encuentran una solución suficientemente buena en un tiempo aceptable.
- Altamente adaptables para los requerimientos del problema a optimizar.
- No son dependientes de la formulación del problema.
- Compromiso entre calidad de las soluciones y tiempo de cómputo.

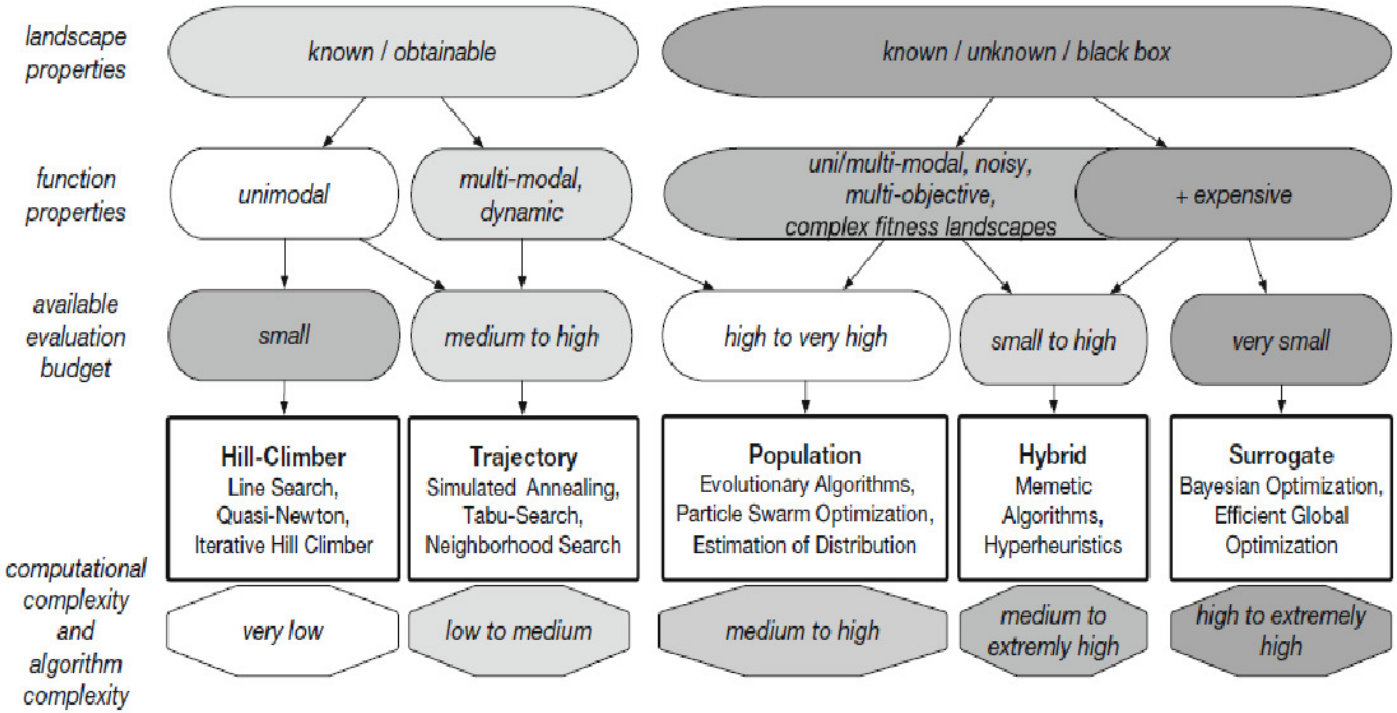
Tipos de metaheurísticas

[Glover and Sörensen, 2015] da una clasificación a grandes rasgos de los tipos de metaheurísticas, en base a cómo se manipulan las soluciones generadas.

1. De búsqueda local. Cambios pequeños de forma iterativa a una solución.
2. Constructivas. Generación de nuevas soluciones a partir de otras.
3. Basadas en poblaciones. Combinar otras soluciones de forma iterativa.

Selección del algoritmo

[Zakizadeh and Erfani, 2024] lista más de 120 algoritmos de optimización metaheurísticos. Por tanto, se recomienda un análisis de las características del problema para elegir el tipo de algoritmo adecuado.



Búsqueda local

- **Idea pricipal:** Encontrar buenas soluciones aplicando iterativamente cambios pequeños a una única solución, llamada **solución actual**. [Glover and Sörensen, 2015]
- Los cambios son llamados **movimientos**, y deben ser pequeños, de forma que las soluciones adyacentes sean cercanas de acuerdo a alguna métrica.
- El conjunto de soluciones al que se puede llegar a partir de una solución dada se llama la **vecindad** de esta solución. Es dependiente de cómo se definan los movimientos.
- La regla utilizada para obtener una nueva solución actual es la **estrategia de búsqueda**. Estas suelen depender de una **función heurística**, eligiendo la mejor solución posible.

Búsqueda local II

Una gran cantidad de algoritmos pertenecen a la familia de búsquedas locales. Entre ellos [Luigi Ceccaroni, 2007]:

- **Hill Climbing.** Selecciona cualquier movimiento que mejore la solución actual.
- **Steepest-ascent Hill Climbing.** Selecciona el mejor movimiento perteneciente a la vecindad actual.
- **Algoritmos genéticos.** Hill Climbing en paralelo con inspiración biólogica.
- **Búsqueda tabú.**
- **Recocido simulado.**

Búsqueda tabú I

Mejora sobre la búsqueda local, evitando estancarse en óptimos locales. Para esto se utiliza una lista tabú (tabu tenure), que recuerda (memoria a corto plazo) los movimientos recientes y prohíbe su exploración durante un número de mientras permanezcan en esta lista. Esto puede llevarnos a soluciones peores, pero evita ciclarse alrededor de óptimos locales.

Búsqueda tabú II

Se toma una solución inicial y se mejora iterativamente con búsqueda local. Se toma la mejor solución de las posibles, incluso si es peor que la actual. Si la solución encontrada pertenece a la lista tabú, se evalúa un criterio de aspiración para decidir si se permite elegirla.
Las soluciones se eliminan de la lista tabú con un criterio *first-in first-out*, después de un periodo de permanencia.

Algoritmo básico

Paso 0. Seleccionar solución inicial, posiblemente de manera aleatoria. Iniciar la lista tabú y especificar su tamaño, tiempo de permanencia y el criterio de aspiración.

Paso 1. Determinar la vecindad de soluciones alrededor de la solución actual. Descartar aquellas que pertenezcan a la lista tabú.

Paso 2. Seleccionar el siguiente movimiento a partir de los candidatos actuales, o posiblemente de la lista tabú de acuerdo al criterio de aspiración. Actualizar la lista tabú.

Paso 3. Si se llega a una condición de salida, terminar. Si no, ir al paso 1.

[Taha, 2012]

Condiciones de salida: Máximas iteraciones, vecindad vacía, solución actual aceptable.

Ejemplo básico

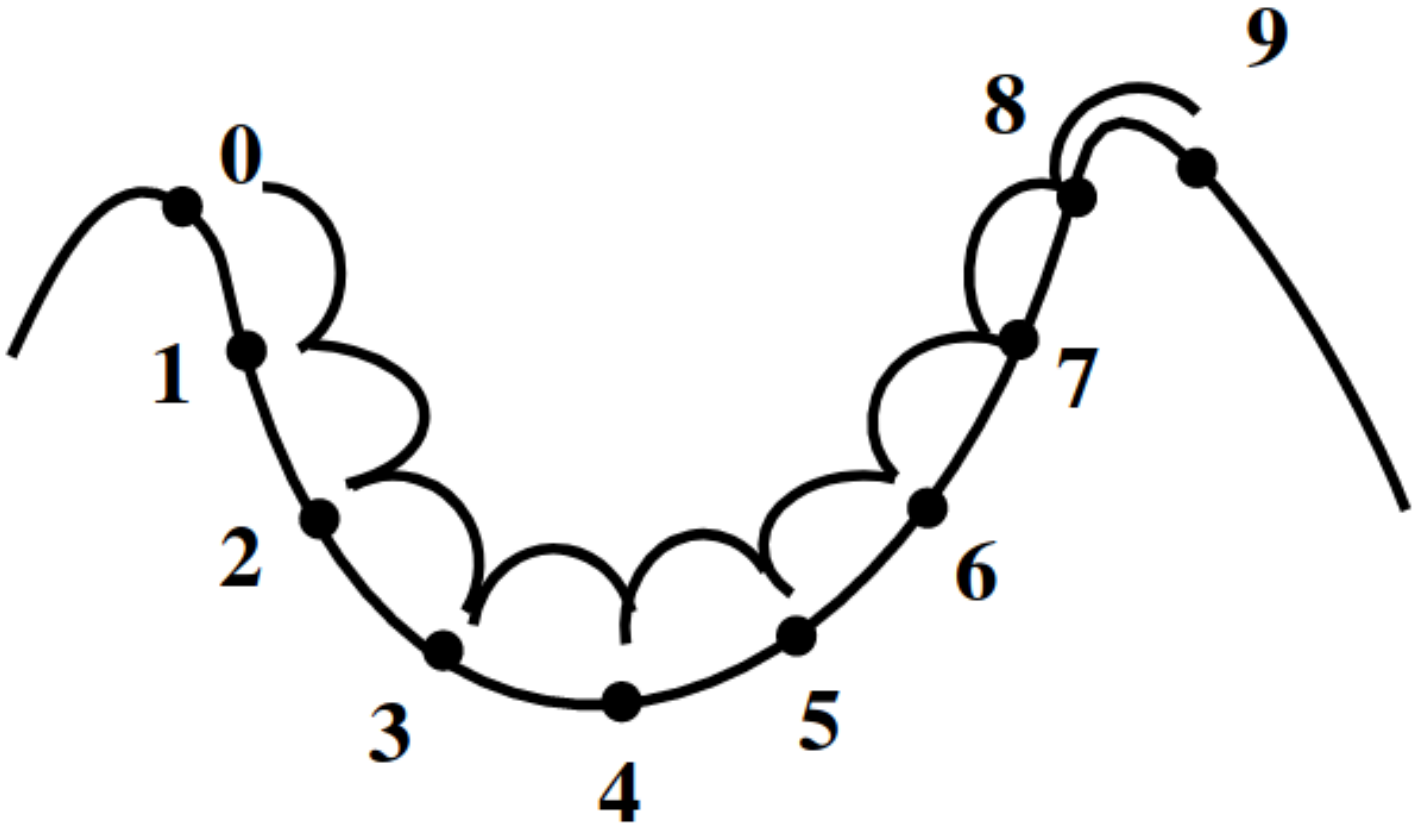


Figura: Ejemplo básico de búsqueda tabú [Morales, 2004]

Parámetros a optimizar.

- **Permanencia en la lista tabú.** Una menor permanencia permite más exploración pero puede llevar a ciclarse. Una mayor permanencia proueve la diversificación pero ralentiza la búsqueda. Se recomienda un valor entre 7 y 20 iteraciones.
- **Criterio de aspiración.** El más común es permitir un movimiento tabú si lleva a una solución mejor que todas las que se han visitado.

El algoritmo puede necesitar un alto número de iteraciones para converger a soluciones de alta calidad. Su desempeño depende de los parámetros dados. Presenta dificultades en problemas con muchas restricciones o muchos óptimos locales.

Ejemplo aplicado

Problema. [Taha, 2012] Secuenciar tareas en un máquina. Cada tarea tiene un tiempo de procesamiento y una fecha límite. Se tiene un costo de almacenamiento si se termina antes de la fecha límite, y un costo de penalización si se termina después. A continuación se muestran los datos para 4 tareas.

Tarea, j	Tiempo de procesamiento en días, T_j	Fecha límite, d_j	Costo de retención por día, h_j	Costo de penalización por día, p_j
1	10	15	\$3	\$10
2	8	20	2	22
3	6	10	5	10
4	7	30	4	8

Ejemplo aplicado II

- **Vecindad.** Permutación del orden de dos tareas sucesivas.
- **Selección del siguiente movimiento.** Aleatoria.
- **Periodo de permanencia.** Dos iteraciones.
- **Secuencia inicial.** (1-2-3-4).
- **Número de iteraciones.** Cinco.

Ejemplo aplicado III

Iteración, k	Secuencia, s_k	Costo total (retención) + (penalización)	z^*	Lista tabú $L(s_k)$	R	Vecindades $N(s_k)$*
(Inicio)0	(1-2-3-4)	$(5\times3+2\times2)+(14\times10+1\times8)=167$	167		.5124	(**2-1**-3-4) (1-**3-2**-4)✓ (1-2-**4-3**)
1	(1-**3-2**-4)	$(5\times3)+(6\times10+4\times22+1\times8)=171$		{3-2}	.3241	(**3-1**-2-4)✓ ~~(1-**2-3**-4)~~ (1-3-**4-2**)
2	(**3-1**-2-4)	$(4\times5)+(1\times10+4\times22+1\times8)=126$	126	{3-2, 3-1}	.2952	~~(**1-3**-2-4)~~ (3-**2-1**-4)✓ (3-1-**4-2**)
3	(3-**2-1**-4)	$(4\times5+6\times2)+(9\times10+1\times8)=130$		{3-1, 2-1}	.4241	(**2-3**-1-4)✓ ~~(3-**1-2**-4)~~ (3-2-**4-1**)
4	(**2-3**-1-4)	$(12\times2)+(4\times10+9\times10+1\times8)=162$		{2-1, 2-3}	.8912	~~(**3-2**-1-4)~~ (2-**1-3**-4) (2-3-**4-1**)✓
(Termi-5 nación)	(2-3-**4-1**)	$(12\times2+9\times4)+(4\times10+16\times10)=260$		{4-1, 1-3}	.0992	(**3-2**-4-1)✓ (2-**4-3**-1) (2-3-**1-4**)

Mejoras: intensificación y diversificación

Intensificación. Periodo centrado en explorar regiones promisorias. Se utiliza una memoria de mediano plazo, buscando atributos comunes entre las mejores soluciones.

Diversificación. Periodo centrado en explorar regiones poco explotadas. Usa una memoria de largo plazo para diversificar la búsqueda en zonas que contrasten fuertemente con los atributos explorados hasta el momento. No es diversidad aleatoria.

Se puede dar un número de iteraciones determinado para el periodo de intensificación, seguido de un periodo de diversificación, o dar alguna condición de cambio, como el no encontrar una mejor solución durante varias iteraciones.

Recocido simulado

Origen. El recocido simulado es una técnica meta-heuristica propuesta por [Kirkpatrick et al., 1983] el cual permite buscar en el espacio de soluciones simulando el proceso del recocido de los materiales. Dicha técnica nace bajo la necesidad de dar soluciones competentes a problemas combinatorios en el contexto del diseño de circuitos electrónicos (VLSI)

Recocido simulado II

El recocido simulado tiene la particularidad de que escapa de quedar atrapado en optimos locales, utilizando una condición probabilista con la cual se acepta o se rechaza un movimiento inferior, salvo que sea un mejor movimiento. La propuesta nace para el tratamiento de problemas combinatorios pero fue extendida a problemas continuos y multivariables.

Algoritmo básico

Paso 0. Seleccione una solución de inicio $s_0 \in S$. Establezca $k = 0, p = 0$ e $i = 0$.

Paso 1. Genere la cercanía $N(s_k)$ y establezca la temperatura $T = T_i$.

Paso 2. Determine la solución s_{k+1} al azar desde $N(s_k)$. Si s_{k+1} no es peor que la solución última aceptada o si $R < P$ acepte s_{k+1}, luego acepte s_{k+1}, establezca $p = p + 1$ y vaya al paso 3. De lo contrario, rechace s_{k+1} y establezca $N(s_{k+1}) = N(s_k)$ Establezca $k = k + 1$ y vaya al paso 1

Paso 3. Si se llega a una condición de salida, terminar. Si no, establezca $k = k + 1$. Si $p = t$ entonces establezca $i = i + 1$. Volver al paso 1.

[Taha, 2012]

Condiciones de salida: Temperatura mínima, máximas iteraciones,.

Ejemplo básico

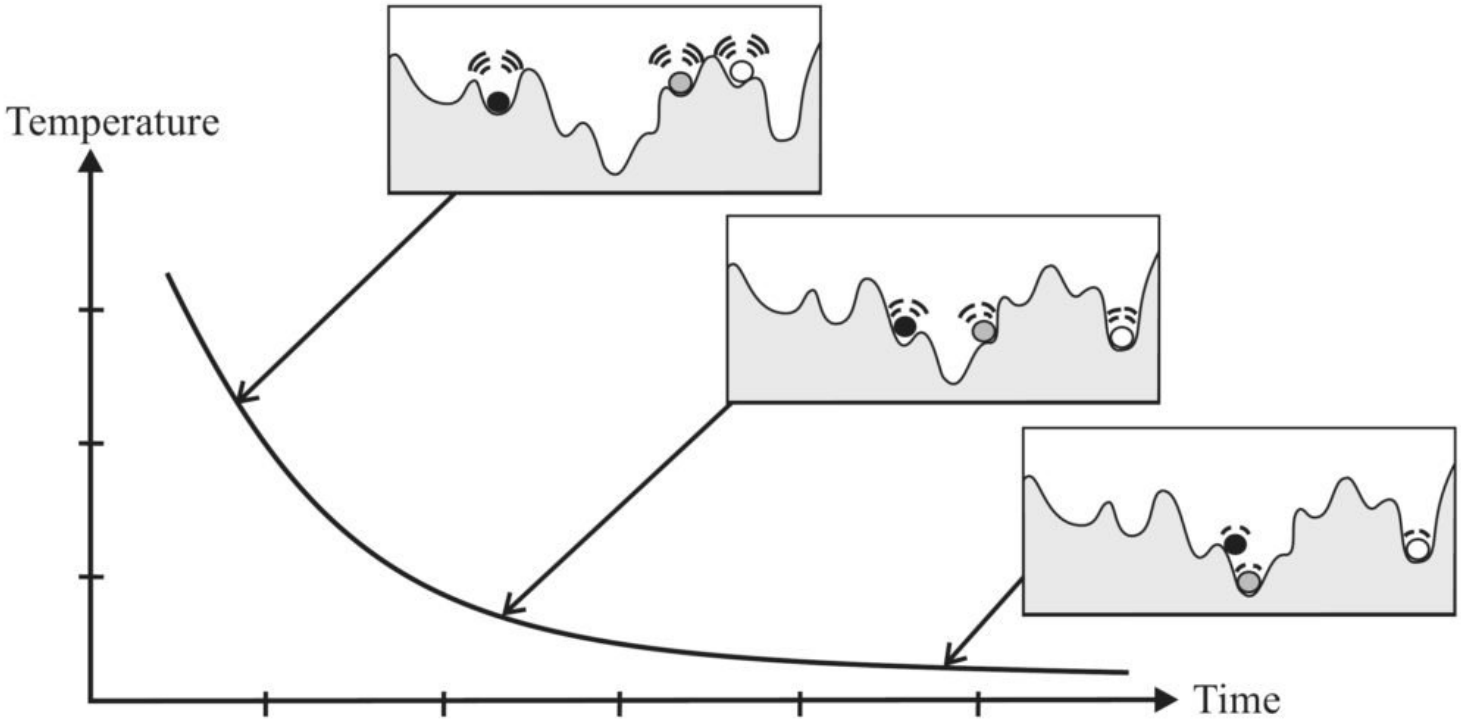


Figura: Ejemplo básico de búsqueda recocido simulado [Guerard, 2016]

Ejemplo básico II

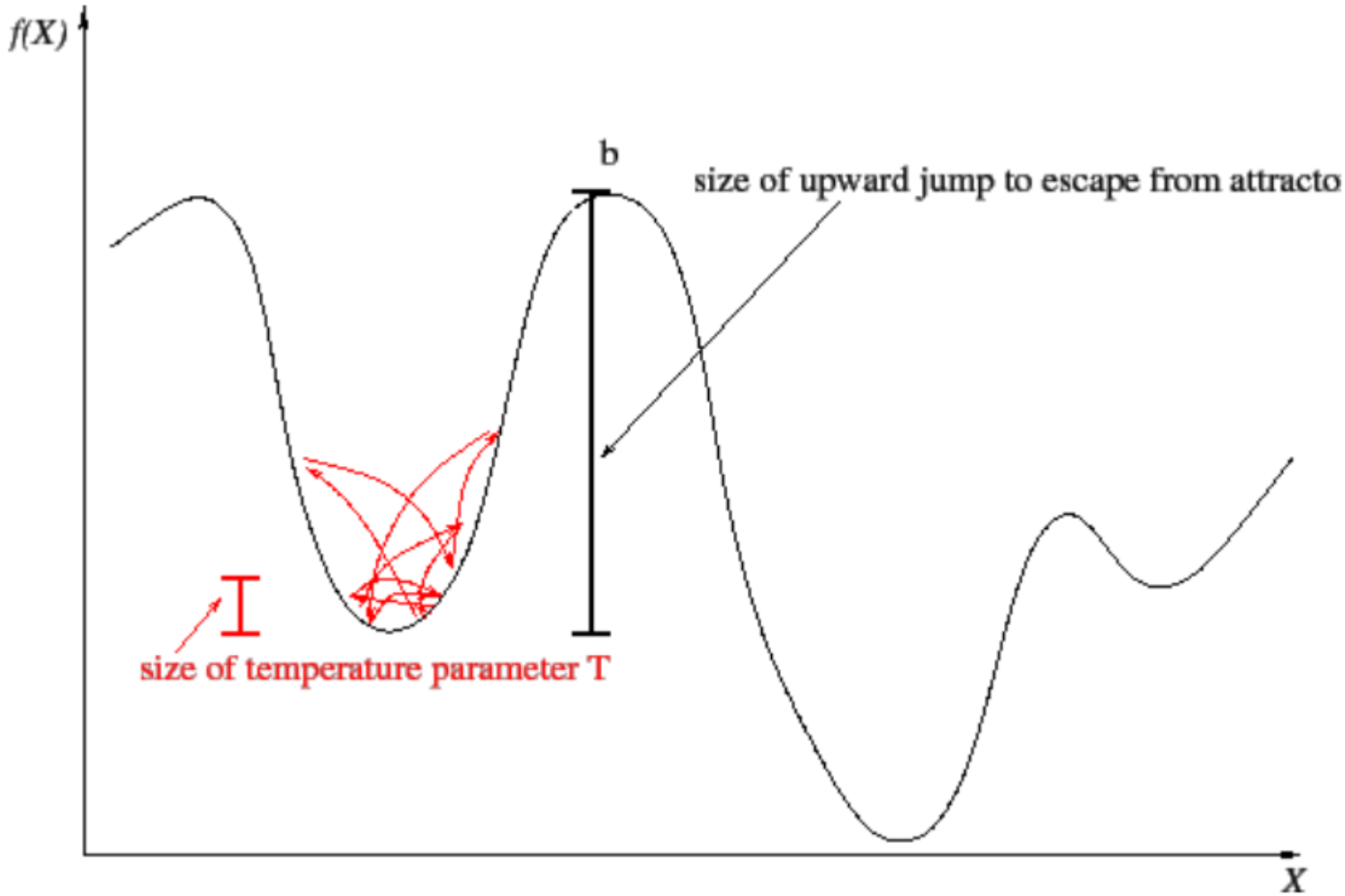


Figura: Baja temperatura en recocido simulado [Guérard, 2016]

Ejemplo aplicado

Problema. [Taha, 2012] Secuenciar tareas en un máquina. Cada tarea tiene un tiempo de procesamiento y una fecha límite. Se tiene un costo de almacenamiento si se termina antes de la fecha límite, y un costo de penalización si se termina después. A continuación se muestran los datos para 4 tareas.

Tarea, j	Tiempo de procesamiento en días, T_j	Fecha límite, d_j	Costo de retención por día, h_j	Costo de penalización por día, p_j
1	10	15	\$3	\$10
2	8	20	2	22
3	6	10	5	10
4	7	30	4	8

Ejemplo aplicado II

- **Vecindad.** Permutación del orden de dos tareas sucesivas.
- **Selección del siguiente movimiento.** Aleatoria.
- **Periodo de permanencia.** Dos iteraciones.
- **Secuencia inicial.** (1-2-3-4).
- **Número de iteraciones.** Cinco.
- **Temperatura inicial.** 83.5.

Ejemplo aplicado III

Iteración k	Secuencia s_k	Costo total c_k = (retención) + (penalización)	T_k	$z = \frac{\lvert\text{Cambio de costo}\rvert}{T_k}$	e^{-z}	R_{1k}	Decisión	R_{2k}	Cercanía $N(s_k)$*
(Inicio)0	(1-2-3-4)	$(5\times3+2\times2)+(14\times10+1\times8)=167$	83.5					.5462	(**2-1**-3-4) (1-**3-2**-4) ✓ (1-2-**4-3**)
1	(1-**3-2**-4)	$(5\times3)+(6\times10+4\times22+1\times8)=171$	83.5	.0479	.9532	.5683	Aceptar: $R_{11} < e^{-z}$	.7431	(**3-1**-2-4) (1-**2-3**-4) (1-3-**4-2**)✓
2	(1-3-**4-2**)	$(5\times3+7\times4)+(6\times10+11\times22)=345$	83.5	2.083	.1244	.3459	Rechazar: $R_{12} > e^{-z}$	.1932	(**3-1**-2-4) ✓ (1-**2-3**-4) (1-3-**4-2**)
3	(**3-1**-2-4)	$(4\times5)+(1\times10+4\times22+1\times8)=126$	83.5				Aceptar: $c_3 < c_1$	.6125	(**1-3**-2-4) (3-**2-1**-4) ✓ (3-1-**4-2**)
4	(3-**2-1**-4)	$(4\times5+6\times3)+(9\times10+1\times8)=130$	83.5	.0479	.9532	.6412	Aceptar: $R_{14} < e^{-z}$	.2234	(**2-3**-1-4) ✓ (3-**1-2**-4) (3-2-**4-1**)
(Termi-nación)5	(**2-3**-1-4)	$(12\times2)+(4\times10+9\times10+1\times8)=162$	41.75	.766	.4647	.5347	Rechazar: $R_{15} > e^{-z}$	.8127	(**2-3**-1-4) (3-**1-2**-4) (3-2-**4-1**)✓

Búsqueda de la mejor solución: (3-1-2-4) con costo de 126 en la iteración 3.

Referencias I

Glover, F. and Sörensen, K. (2015).
Metaheuristics.
Scholarpedia, 10(4):6532.

Guerard, G. (2016).
Recocido simulado.
Sistemas complejos e IA.
Accedido: 2024-12-13.

Kirkpatrick, S., Gelatt, C. D. J., and Vecchi, M. P. (1983).
Optimization by simulated annealing.
Science, 220(4598):671–680.

Referencias II


Luigi Ceccaroni (2007).
Inteligencia Artificial.
https://www.cs.upc.edu/ luigi/II/IA-2007-fall/2c-Busqueda-local-(es).pdf.


Morales, E. (2004).
Búsqueda tabú.
https://ccc.inaoep.mx/~emorales/Cursos/Busqueda/btabu.pdf.


Taha, H. (2012).
Investigación de Operaciones.
Pearson, 9th edition.


Zakizadeh, M. and Erfani, H. (2024).
Meta-Heuristic Algorithms A Comprehensive Review.
Islamic Azad University.