OPTIMIZACIÓN INTELIGENTE

Dr. en C. Luis Fernando Gutiérrez Marfileño
Universidad Autónoma de Aguascalientes

Centro de Ciencias Básicas
Departamento de Ciencias de la Computación
Academia de Inteligencia Artificial y Fundamentos Computacionales

7 de Agosto de 2023

Índice general

Índice de figuras

Índice de tablas

Unidad 1

INTRODUCCIÓN A LA OPTIMIZACIÓN

"Enunciar conceptos básicos relacionados con la optimización, clasificar problemas y técnicas y formular problemas, y aplicar métodos de optimización numérica unidimensional y multidimensional no restringidas."

Objetivos específicos 1 y 2

1.1. Introducción a la optimización

En el lenguaje común, **optimizar** es buscar la mejor manera de realizar una actividad [DRALE], aquí el término $mejor$ depende del contexto en el que se trabaje, podría significar una solución que minimiza los costos, o maximiza los beneficios, o que hace que la distancia recorrida sea mínima, etc..
En matemáticas, la Optimización es una disciplina que se ocupa de encontrar los extremos (mínimos y máximos) de números, funciones o sistemas.
Los filósofos y matemáticos antiguos crearon sus cimientos definiendo el óptimo (como un extremo, máximo o mínimo) sobre varios dominios fundamentales como números, formas geométricas, óptica, física, astronomía, la calidad de la vida humana, el gobierno estatal, y varios otros.
El abordar un problema real de Optimización supone básicamente dos etapas:

- Determinar el modelo matemático que rige el problema.
- Resolver dicho problema usando una serie de técnicas matemáticas.

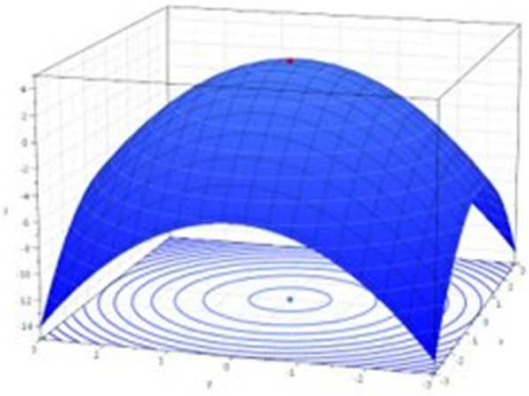


Figura 1.1: Gráfico de un paraboloide

Los problemas que pueden ser resueltos mediante la Optimización Matemática son aquellos que pueden expresarse en la forma:

$MAX f(X)$ $\quad$ $MIN f(X)$

$X \in CS$ $\quad$ $X \in CS$

Donde CS corresponde al conjunto solución.

La optimización matemática de una sola variable se clasifica en:

- Optimización sin restricciones
- Con restricciones de igualdad
- Con restricciones de desigualdad

1.2. Antecedentes matemáticos

EN matemáticas, algunas de las principales técnicas de solución para problemas de optimización son:

- Programación lineal
- Programación lineal entera mixta
- Programación cuadrática
- Programación no lineal
- Optimización estocástica
- Programación dinámica
- Teoría de grafos

Definición (Problemas de optimización)

Son aquellos que se ocupan de elegir la decisión óptima de solución de un problema, es decir, encontrar cual es el máximo o mínimo de un determinado criterio (una función) sujeto a las condiciones dadas por el problema.

Clasificación

Los problemas de optimización se pueden dividir en dos categorías, dependiendo de si las variables son **continuas** o **discretas**:

1. Un problema con variables continuas se conoce como **optimización continua**, en la que se debe encontrar un valor óptimo de una función continua. Pueden incluir problemas restringidos y problemas multimodales.
2. Un problema de optimización con variables discretas se conoce como **optimización discreta**, en la que un objeto como un número entero, una permutación o un gráfico se debe encontrar en un conjunto contable.

Modelado del problema

Un modelo de optimización es la representación matemática de un problema real, en el cual se identifican aspectos de la realidad y se representan mediante fórmulas.

Una vez representado el problema, se pueden utilizar algoritmos para encontrar rápidamente las mejores soluciones.

Los modelos de optimización tienen 4 elementos principales:

- Parámetros
- Variables
- Restricciones
- Función objetivo.

Descripción matemática

Maximizar: $f(x)$ función objetivo
Sujeto a:
$h(x) = 0$ Restricción de igualdad
$g(x) \geq 0$ Restricción de desigualdad
Donde:
- $x \in \Re^n$ es un vector de n variables como: $\{x_0, x_1, x_2, ..., x_n\}$
- $h(x)$ es un vector de igualdades de dimensión m_1
- $g(x)$ es un vector de desigualdades de dimensión m_2

Solución

Para resolver un problema de optimización de forma correcta se va a establecer una serie de pasos que harán más sencillo el planteamiento y la resolución:

1. Establecer cuál o cuáles son las **incógnitas** que plantea el problema.
2. Buscar y plantear qué es lo que se tiene que maximizar o minimizar: $f(x, y)$.
3. Buscar la condición que se plantea. En la mayoría de los problemas la función a maximizar o minimizar dependerá de dos variables, por tanto, la condición permitirá relacionar estas dos variables para poner una en función de la otra.
4. Una vez, despejado una variable en función de la otra, suponer y en función de x. Sustituir la función a optimizar, quedándose ahora en función de una sola variable: $f(x)$.
5. **Derivar** la función e igualar a cero: $f'(x) = 0$.
6. Una vez obtenidas las soluciones, comprobar si realmente se trata de un máximo o un mínimo, para ello, realizar la segunda derivada de tal forma que:

   - si $f''(x) = 0$, entonces se trata de un **mínimo**.
7. Una vez teniendo x, ir al paso 3, donde se despeja y, hallar el valor de y, y dar la solución.

Características de las técnicas de optimización matemáticas

- Buscan el óptimo localmente
- Garantizan el óptimo numérico
- Permiten un elevado número de restricciones

1.3. Métodos numéricos de optimización unidimensional no restringida

Muchos métodos de optimización de problemas con restricciones (univariables y multivariables) involucran la resolución de un problema de optimización en una dimensión.
Existen dos tipos de métodos numéricos, a saber:

- **Métodos directos**: sólo utilizan los valores de la función objetivo.
- **Métodos indirectos**: utilizan las condiciones necesarias, las derivadas (analíticas o numéricas) y la función objetivo.

Los métodos indirectos requieren el cálculo de las derivadas primeras y segundas. Sin embargo, muchas veces obtener las derivadas es una tarea difícil, y hasta es posible que ni siquiera se conozca la forma analítica de la función objetivo.
Esto plantea la necesidad de contar con métodos capaces de trabajar únicamente con los valores (experimentos) de la función objetivo.

Métodos numéricos para optimización de funciones de una variable

Para la aplicación de estos métodos es necesario conocer el intervalo inicial Δ^0, donde está contenido el óptimo de la función objetivo, y asegurar la unimodalidad de la función en el intervalo en estudio.
Un método de optimización para una función de una sola variable podría ser determinar una matriz (tan fina como se quiera) de valores de x y calcular los valores de $f(x)$ en cada valor de la matriz, el óptimo sería el mejor valor de $f(x)$.

Métodos indirectos

Método de Newton

El objetivo de este método es calcular el máximo o mínimo de una función, haciendo uso de una aproximación cuadrática dada por la serie de Taylor.
Dicha aproximación cuadrática es:

$$q(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x)}{2}(x - x_0)^2$$

El máximo o mínimo de la función $q(x)$ se calcula haciendo $q'(x) = 0$.
Con lo que se obtiene:

$$f'(x_0) + f''(x_0)(x - x_0) = 0$$

$$x = x_0 - \frac{f'(x_0)}{f''(x_0)}$$

Algoritmo

1. Dado un valor inicial x_0 y una función cuyas primera y segunda derivada existan
2. Se calcula la i-ésima aproximación utilizando la fórmula:

$$x_i = x_{i-1} - \frac{f'(x_{i-1})}{f''(x_{i-1})}$$

3. Se repite el paso 2 hasta lograr la convergencia.

Ejemplo

Use el método de Newton para encontrar el máximo de la función:

$$f(x) = 2seno(x) - x^2/10$$

con el valor inicial $x_0 = 2{,}5$.

Solución

i	x_0	x_1
1	2.5000	0.9951
2	0.9951	1.4690
3	1.4690	1.4276
4	1.4276	1.4276

Método de Cuasi-Newton

Este método es una solución a las limitaciones del método de Newton.
En el caso en que la función objetivo no sea conocida o no puedan evaluarse las derivadas, estas pueden reemplazarse por aproximaciones de diferencias finitas:
La desventaja adicional de este método consiste en la necesidad de evaluar funciones adicionales en cada iteración, también es necesario conocer el valor de h (paso de la diferencia finita).

Método de la Secante

El método de la secante combina el método de Newton con un esquema de reducción de intervalo para encontrar, si existe, la raíz de la ecuación $f'(x) = 0$, en el intervalo (a, b).

$$f'(x^k) + m(x - x^k) = 0$$

En este método la condición necesaria se resuelve mediante la siguiente expresión: donde m es la pendiente de la recta que une los puntos x^p y x^q, dada por:

$$m = \frac{f'(x^q) - f'(x^p)}{x^q - x^p}$$

Este método aproxima la derivada de la función a una línea recta, m aproxima la segunda derivada de la función.

Métodos directos

Eliminación de regiones

Este tipo de métodos se centra en la búsqueda de las soluciones óptimas mediante sucesivas reducciones del intervalo de estudio y en la eliminación de subintervalos.

Si la función es unimodal, se puede definir un criterio para eliminar regiones donde seguro el óptimo no se encuentra.

Para ello necesitamos evaluar la función en dos puntos y aplicar algo de lógica. En la figura siguiente se indica cual sería la región eliminada para los tres casos posibles en la búsqueda de un máximo.

Figura 1.2: Criterio de eliminación de regiones

Es fundamental el hecho de que la función estudiada sea unimodal, al menos dentro del dominio de interés.

La utilidad de esta propiedad radica en el hecho de que si $f(x)$ es unimodal, entonces solamente es necesario comparar $f(x)$ en dos puntos diferentes para predecir en cuál de los subintervalos definidos por esos puntos no se va a encontrar el óptimo.

Cuando el subintervalo $sobreviviente$ tenga una longitud suficientemente pequeña, la búsqueda termina. La gran ventaja de estos métodos de búsqueda es que solamente requieren evaluaciones de la función y no necesitamos ninguna hipótesis adicional acerca de la derivabilidad de la misma.

1.4. Métodos numéricos de optimización multidimensional no restringida

UNa forma de clasificación de las múltiples técnicas de optimización no restringida multidimensional tiene que ver con si requieren la evaluación de la derivada o no.
Los procedimientos que no requieren dicha evaluación se llaman **métodos directos** o sin gradiente.
Aquellos que requieren derivadas se conocen como **métodos de gradiente**.

Métodos directos

Búsqueda aleatoria

La búsqueda aleatoria implica generar y evaluar entradas aleatorias a la función objetivo.
Es eficaz porque no asume nada sobre la estructura de la función objetivo.
Esto puede ser beneficioso para problemas en los que hay mucha experiencia en el dominio que puede influir o sesgar la estrategia de optimización, lo que permite descubrir soluciones no intuitivas.
La búsqueda aleatoria también puede ser la mejor estrategia para problemas muy complejos con áreas ruidosas o no uniformes (discontinuas) del espacio de búsqueda que pueden generar algoritmos que dependen de gradientes confiables.
Se puede generar una muestra aleatoria de un dominio usando un generador de números pseudoaleatorios.
Cada variable requiere un límite o rango bien definido y se puede muestrear un valor aleatorio uniforme del rango y luego evaluarla
Generar muestras aleatorias es sencillo desde el punto de vista computacional y no ocupa mucha memoria, por lo tanto, puede ser eficiente generar una muestra grande de entradas y luego evaluarlas.
Cada muestra es independiente, por lo que las muestras se pueden evaluar en paralelo si es necesario para acelerar el proceso.

Método de búsqueda en rejilla

Los métodos básicos de diseño de experimentos discutidos en muchos textos de estadística se pueden aplicar también a minimización de funciones.
Se pueden seleccionar una serie de puntos alrededor de un punto base de referencia (creando una rejilla) de acuerdo a algunos de los diseños del tipo que se muestra en la figura 1.3.
Después se pasa al punto que más mejora la función objetivo y se continua la búsqueda.

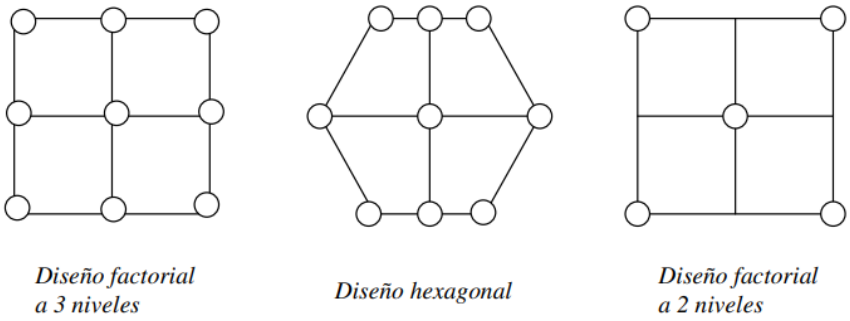


Figura 1.3: Métodos de búsqueda en rejilla

Sin embargo, el sistema es muy ineficaz, por ejemplo, con $n = 10$ y una búsqueda factorial a tres niveles se deben realizar $310 - 1 = 59,048$ evaluaciones de la función objetivo, lo cual es obviamente impráctico.

Métodos indirectos

Los métodos indirectos, en contraste con los métodos anteriores hacen uso de las derivadas en la determinación de la dirección de búsqueda (aunque no todos).
Una buena dirección de búsqueda debería reducir (para minimización) la función objetivo, de tal manera que si x^0 es el punto inicial y x^1 es un nuevo punto:

$$f(x^1) \leq f(x^0)$$

Método del Gradiente

El gradiente es un vector en un punto x que proporciona la dirección (local) de máxima variación de la función.
El vector gradiente es un vector ortogonal al contorno de la función en el punto. Por lo tanto, en la búsqueda de un mínimo la dirección de movimiento será contragradiente:

$$\mathbf{s}^k = -\nabla f(\mathbf{x})$$

En el método de máximo descenso la transición de un punto x^k a otro x^{k+1} viene dada por la siguiente expresión:

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \Delta\mathbf{x}^k = \mathbf{x}^k + \lambda^k\mathbf{s}^k = \mathbf{x}^k - \lambda^k\nabla f(\mathbf{x}^k)$$

donde:

$\Delta\mathbf{x}^k$ = Vector desde $\mathbf{x}^k$ hasta $\mathbf{x}^{k+1}$
$\mathbf{s}^k$ = Dirección de búsqueda de máximo descenso
λ^k = Escalar que determina la longitud de paso en la dirección s

El gradiente negativo da la dirección de movimiento, pero no la longitud de dicho movimiento.
Por lo tanto, existen varios procedimientos posibles dependiendo de la elección de λ^k .

Algoritmo

1. Elegir un valor inicial $\mathbf{x}^0$. En pasos sucesivos será $\mathbf{x}^k$.
2. Calcular, analítica o numéricamente las derivadas parciales $\frac{\partial f(\mathbf{x})}{\partial x_j}\ j = 1, 3, 4, ..., n$
3. Calcular el vector de búsqueda: $\mathbf{s}^k = -\nabla f(\mathbf{x})$
4. Usar la relación $\mathbf{x}^{k+1} = \mathbf{x}^k + \lambda^k \mathbf{s}^k$ para obtener el siguiente punto. El valor de ? k puede ser de valor fijo o calculado en cada paso mediante una búsqueda unidireccional.
5. Comparar $\mathbf{x}^{k+1}$ con $\mathbf{x}^k$. Si el cambio es menor que una tolerancia preespecificada terminar, en caso contrario volver al paso dos y continuar con las iteraciones.

Método del Gradiente conjugado

El método del gradiente conjugado debido a Fletcher y Reeves (1964) combina las características de la convergencia cuadrática del método de las direcciones conjugadas con las del método del gradiente.
El método supone una importante mejora del método del gradiente con sólo un pequeño incremento en el esfuerzo de cálculo.
Combina la información obtenida del vector gradiente con la información acerca del vector gradiente de iteraciones previas.
Lo que hace el método es calcular la nueva dirección de búsqueda utilizando una combinación lineal del gradiente en la etapa considerada y el de la etapa anterior.

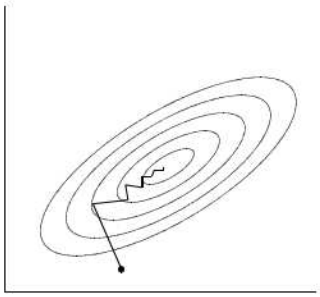


Figura 1.4: Oscilación del método de máximo descenso

Unidad 2

OPTIMIZACIÓN BASADA EN ALGORITMOS EVOLUTIVOS

"Describir las partes que constituyen la modelación metaheurística de un problema e implementar el algoritmo genético como un algoritmo evolutivo clásico."

Objetivos específicos 1 y 2

2.1. Naturaleza de las metaheurísticas

Los algoritmos de optimización inteligente (**metaheurísticas**) son algoritmos que simulan fenómenos y comportamientos naturales mediante iteraciones basadas en poblaciones.
Son una solución para problemas NP-duros difíciles de resolver con los algoritmos deterministas clásicos.

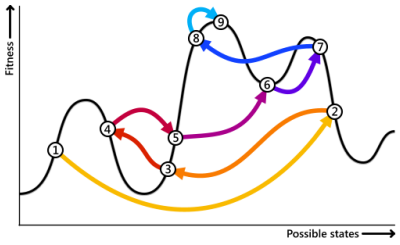


Figura 2.1: Problema de optimización

Y pueden encontrar soluciones subóptimas factibles para problemas complejos en un tiempo relativamente corto.
La gran versatilidad, la alta velocidad y la solidez del algoritmo de optimización inteligente brindan una variedad de soluciones de toma de decisiones para problemas de optimización combinatoria y numéricos complejos con múltiples restricciones, como la composición de servicios con múltiples objetivos, la programación del flujo de trabajo, la asignación de recursos de fabricación y la evaluación y el control de la calidad del producto, etc. en un sistema de fabricación orientado a servicios en red.
Como se mencionó, los problemas de optimización se encuentran en muchos dominios: ciencia, ingeniería, administración y negocios.
Un problema de optimización puede estar definido por el par (S, f), donde S representa el conjunto de soluciones factibles, y $f : S \rightarrow R$ la función objetivo a optimizar.
La función objetivo asigna a cada solución $s \in S$ del espacio de búsqueda un número real que indica su valor y permite definir una relación de orden total entre cualquier par de soluciones en el espacio de búsqueda.

Características de las metaheuristicas

- Las metaheurísticas son estrategias generales que guían el proceso de búsqueda.
- El objetivo es una búsqueda eficiente que encuentre soluciones casi óptimas.
- Pueden incorporar mecanismos para evitar la exploración en regiones del espacio de búsqueda no óptimas.
- El procedimiento de cualquier metaheurística es genérico, no depende del problema.
- Las metaheurísticas utilizan métodos heurísticos específicos que son controlados por una estrategia de más alto nivel.
- Las metaheurísticas utilizan funciones de bondad para cuantificar el grado de adecuación de una determinada solución.

Hay diferentes formas de clasificar las metaheurísticas: basadas en la naturaleza (algoritmos bioinspirados) o no basadas en la naturaleza, basadas en memoria o sin memoria, con función objetivo estática o dinámica, etc.
La clasificación más empleada es la que se basa en si la técnica utiliza un único punto del espacio de búsqueda o trabaja sobre un conjunto o población.
Según esta clasificación las metaheurísticas se dividen en las basadas en trayectoria y las basadas en población.

Metaheurísticas basadas en poblaciónes

Las técnicas metaheurísticas basadas en población trabajan con un conjunto de individuos que representan otras tantas soluciones. Su eficiencia y resultado depende fundamentalmente de la forma con la que se manipula la población en cada iteración.
A continuación algunas de éstas técnicas:

Algoritmos Evolutivos

Están basadas en un aspecto de la naturaleza, en este caso, en la evolución de las especies.
Se inspiran en la capacidad de la evolución de seres o individuos para adaptarlos a los cambios de su entorno.
Cada individuo representa una posible solución.
El funcionamiento básico de estos algoritmos es el siguiente:
La población se genera de forma aleatoria.
Cada individuo de la población tiene asignado un valor de su bondad con respecto al problema considerado, por medio de una función de aptitud, capacidad, adaptabilidad o estado, también denominada con bastante frecuencia por la palabra inglesa $fitness$.
El valor de la $aptitud$ de un individuo es la información que el algoritmo utilizar para realizar la búsqueda.
La modificación de la población se efectúa mediante la aplicación de tres operadores: selección, recombinación y mutación.
En estos algoritmos se pueden distinguir la fase de selección, explotación de buenas soluciones, y la fase de reproducción, búsqueda de nuevas regiones.
Se debe de mantener un equilibro entre estas dos fases.
La política de reemplazo permite la aceptación de nuevas soluciones que no necesariamente mejoran las existentes.
Los algoritmos evolutivos se pueden clasificar en las siguientes tres categorías:
Programación Evolutiva (PE), Estrategias Evolutivas (EE), y los Algoritmos Genéticos (AG), que constituyen una de las técnicas más conocidas, y que fueron introducidos por Holland.

2.2. Elementos para la modelación de problemas

ETapas para construir un buen modelo de un problema:

1. **Fase de Conceptualización**. Tener un profundo conocimiento de la realidad que se trata de modelar, es decir, ser capaces de representar conceptualmente el problema sin ningún tipo de contradicciones lógicas ni de errores de análisis.

2. **Fase de Formalización**. Establecer de forma clara y correcta (matemáticamente) las relaciones entre los elementos, para que, además, sea fácilmente entendible y detectar rápidamente los errores.

3. **Fase de Evaluación**. En esta fase, además de establecer la forma en la que debe ser el procedimiento de resolución a emplear, se debe poder interpretarlo correctamente.

Para la aplicación practica para modelar un problema de optimización se pueden seguir los siguientes reglas basadas en la experiencia:

Reglas de modelado

1 Análisis del problema. Establecer claramente cuál es el objetivo que se persigue, qué limitaciones existen, etc. Todo ello debe tenerse en cuenta aunque no esté formalizado, sino simplemente una relación de las diferentes condiciones.

2 Definición de las variables. Identificar las posibles decisiones. Esta es una de las fases críticas de la modelización, por ello es conveniente prestar mucha atención a esta definición. En esta fase hay que identificar (e interpretar el significado) y denominar a las variables que intervienen. Este segundo aspecto, aunque puede parecer trivial, es también de gran importancia.

3 Identificación y formalización de las restricciones. Identificar cuáles son las limitaciones a las que está sujeto el problema, y plantearlas matemáticamente. A veces esto no resulta muy sencillo. En esta fase hay que denominar e identificar a las restricciones con los nombres adecuados, de forma que sea fácil interpretar los resultados obtenidos.

4 Identificar la función objetivo. La cuantificación de los resultados que se desean alcanzar. Aunque no en todos los problemas es inmediato definir el objetivo, siempre es posible encontrar una función que permita evaluar los resultados de cada una de las acciones.

A continuación algunos principios básicos de modelado:

Principios de modelado

I. Formular un modelo simple y agregar características junto con la ejecución de la optimización.

Un modelo a optimizar debe desarrollarse logrando un equilibrio razonable entre los objetivos de precisión mejorada en el modelo (lo que generalmente implica mayor complejidad en la formulación) y mayor facilidad de optimización.

II. Uso de funciones derivables.

Probablemente, la propiedad más fundamental de las funciones del problema con respecto a la facilidad de optimización, es la derivabilidad, que es importante porque los algoritmos se basan en el uso de funciones disponibles.

III. Evitar definir funciones de problemas que sean el resultado de algún procedimiento iterativo (como la solución de una ecuación diferencial o la evaluación de una integral.

Las funciones del problema definidas por un procedimiento iterativo son a menudo la fuente de sutiles discontinuidades que pueden obstaculizar el progreso de la optimización. La solución de estos subproblemas con la máxima precisión de la máquina (incluso si es posible) generalmente requiere un esfuerzo computacional considerable.

IV. Análisis cuidadoso sobre la naturaleza de las restricciones.

No siempre se aprecia que pueden producirse mejoras sustanciales en el rendimiento y la robustez cuando los métodos explotan las diferentes propiedades de los límites simples, las restricciones lineales y las restricciones no lineales. Siempre que sea posible, se deben aislar las restricciones lineales de las no lineales y utilizar un software que diferencie entre los tipos de restricciones durante la optimización.

V. No intente eliminar las restricciones de igualdad del problema.

Los modeladores a menudo asumen que dado que puede no haber significado físico para un punto en el que se violan las restricciones de igualdad no lineal, dichas restricciones deben cumplirse exactamente en todas las etapas de la optimización. En consecuencia, los usuarios a menudo intentan .elïminar"las restricciones de igualdad no lineal del problema mediante el siguiente método. Las variables se dividen en conjuntos independientes y dependientes.

VI. Distinguir entre restricciones duras y suaves

En muchos problemas, las restricciones pueden clasificarse por lotes como duras o blandas. Por ejemplo, las restricciones duras podrían especificar la región en la que las funciones del problema están bien definidas. Una restricción suave podría representar un límite cuya violación puede tolerarse si esto resulta en una gran disminución en la función objetivo, en algunos casos, las variables pueden tener un límite flexible y un límite rígido.

VII. Evite modelar restricciones de igualdad casi dependientes.

Las restricciones ocurren en las formulaciones de problemas por una variedad de razones. A menudo, la naturaleza misma de las variables impone una restricción

de igualdad; por ejemplo, si las variables zi representan proporciones o probabilidades, esto da lugar a una restricción. Las restricciones de este tipo son igualdades genuinas, en el sentido de que la solución calculada debe satisfacerlas exactamente (donde exactamente significa dentro de la precisión de trabajo).

VIII. Usar información sobre el problema para escalar las variables y restricciones.

Es bien sabido que la escala correcta de variables y restricciones puede mejorar drásticamente la eficiencia y precisión de los métodos de optimización. La escala de un problema es la medida de la importancia relativa de las variables y restricciones o, de manera equivalente, la escala de un problema es una declaración de lo que es grande y lo que es pequeño en un problema.

IX. Trate de proporcionar tanta información sobre la función como sea posible.

Como regla general, los algoritmos tienden a ser más eficientes y robustos cuando se proporciona mayor información sobre el problema. Por ejemplo, si las funciones del problema son simples, los algoritmos que utilizan la primera y la segunda derivada realizan mejores algoritmos mixtos que usan valores de función únicamente.

En general, se percibe que un algoritmo de segundas derivadas rara vez es útil porque el costo de calcular las segundas derivadas puede ser varios órdenes de magnitud mayor que el de calcular las derivadas primeras.

X. Tener especial cuidado en verificar que las funciones del problema y sus derivadas estén programadas correctamente.

Antes de emprender una serie de ejecuciones de optimización, el usuario debe verificar que el código que define las funciones del problema es correcto. Una verificación obvia es evaluar las funciones del problema en un punto donde se conocen sus valores.

Los errores en la programación de la función pueden ser bastante sutiles en el sentido de que el valor de la función puede casi correcto.

2.3. Algoritmo genético

Los Algoritmos Genéticos son métodos de optimización heurística que, entre otras aplicaciones, pueden usarse para encontrar el valor o los valores que consiguen maximizar o minimizar una función.
En este método los individuos de una población se reproducen generando nuevos descendientes cuyas características son combinaciones de las características de sus progenitores (más ciertas mutaciones), de ellos solo los mejores sobreviven y pueden reproducirse de nuevo.

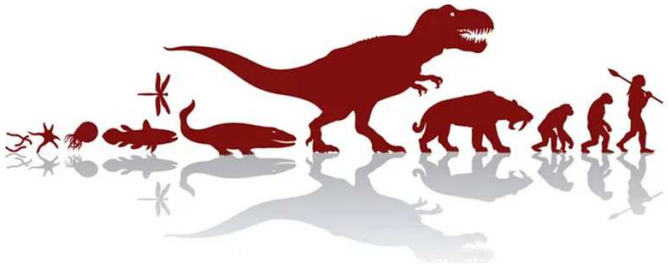


Figura 2.2: Proceso evolutivo natural

Algoritmo
En términos generales la estructura de un AG para optimizar una función con una o múltiples comprende los siguientes pasos:

1. Crear una población inicial de P individuos, donde cada uno representa una combinación de valores de variables.

2. Calcular la aptitud (fitness) de cada individuo de la población. El fitness está relacionado con el valor de la función para cada individuo. Si se quiere maximizar, cuanto mayor sea el valor de la función para el individuo, mayor su fitness. En el caso de minimización, ocurre lo contrario.

3. Crear una nueva población vacía y repetir los siguientes pasos hasta que se hayan creado P nuevos individuos.

    a) Seleccionar dos individuos de la población existente, donde la probabilidad de selección es proporcional al fitness de los individuos.

    b) Cruzar los dos individuos seleccionados para generar un nuevo descendiente (crossover).

    c) Aplicar un proceso de mutación aleatorio sobre el nuevo individuo.

    d) Añadir el nuevo individuo a la nueva población.

4. Reemplazar la antigua población por la nueva.

5. Si no se cumple un criterio de parada, volver al paso 2

Población inicial

En el contexto del AG, el término **individuo** hace referencia a cada una de las posibles soluciones del problema que se quiere resolver.
En el caso de maximización o minimización de una función, cada individuo representa una posible combinación de valores de las variables.
Para representar dichas combinaciones, se pueden emplear vectores, cuya longitud es igual al número total de variables, y cada posición toma un valor numérico.
Por ejemplo, supóngase que la función objetivo J(x,y,z) depende de las variables x,y,z . El individuo 3,9.5,-0.5, equivale a la combinación de valores x=3,y=9.5,z=-0.5 . El primer paso del algoritmo genético consiste en crear una población inicial aleatoria de individuos.

Fitness de un individuo

Cada individuo de la población debe ser evaluado para cuantificar cómo de bueno es como solución al problema, a esta cuantificación se le llama (fitness). Dependiendo de si se trata de un problema de maximización o minimización, la relación del fitness con la función objetivo f puede ser:
Maximización: el individuo tiene mayor fitness cuanto mayor es el valor de la función objetivo f(individuo)
Minimización: el individuo tiene mayor fitness cuanto menor es el valor de la función objetivo f(individuo), o lo que es lo mismo, cuanto mayor es el valor de la función objetivo, menor el fitness.
Tal y como se describe más adelante, el algoritmo genético selecciona los individuos de mayor fitness, por lo que, para problemas de minimización, el fitness puede calcularse como -f(individuo) o también 1/(1+f(individuo))

Seleccionar individuos

La forma en que se seleccionan los individuos que participan en cada cruce difiere en las distintas implementaciones de los algoritmos genéticos. Por lo general, todas ellas tienden a favorecer la selección de aquellos individuos con mayor fitness. Algunas de las estrategias más comunes son:
Método de ruleta: la probabilidad de que un individuo sea seleccionado es proporcional a su fitness relativo, es decir, a su fitness dividido por la suma del fitness de todos los individuos de la población. Si el fitness de un individuo es el doble que el de otro, también lo será la probabilidad de que sea seleccionado. Este método presenta problemas si el fitness de unos pocos individuos es muy superior (varios órdenes de magnitud) al resto, ya que estos serán seleccionados de forma repetida y casi todos los individuos de la siguiente generación serán **hijos** de los mismos **padres** (poca variación).

Método rank: la probabilidad de selección de un individuo es inversamente proporcional a la posición que ocupa tras ordenar todos los individuos de mayor a menor fitness. Este método es menos agresivo que el método ruleta cuando la diferencia entre los mayores fitness es varios órdenes de magnitud superior al resto.
Selección competitiva (tournament): se seleccionan aleatoriamente dos parejas de individuos de la población (todos con la misma probabilidad). De cada pareja se selecciona el que tenga mayor fitness. Finalmente, se comparan los dos finalistas y se selecciona el de mayor fitness. Este método tiende a generar una distribución de la probabilidad de selección más equilibrada que las dos anteriores.
Selección truncada (truncated selection): se realizan selecciones aleatorias de individuos, habiendo descartado primero los n individuos con menor fitness.
Cuando no existe una gran diferencia entre el individuo de mayor fitness y el resto, con el método rank, el individuo con mayor fitness se selecciona con mucha más frecuencia que el resto.
Con los otros dos métodos, la probabilidad de selección decae de forma gradual.
Cuando existe una gran diferencia entre el individuo de mayor fitness y el resto (uno o varios órdenes de magnitud), con el método ruleta, el individuo con mayor fitness se selecciona con mucha más frecuencia que el resto. A diferencia del caso anterior, en esta situación, la probabilidad de selección decae de forma más gradual con los métodos rank y tournament.
Teniendo en cuenta los comportamientos de selección de cada método, el método tournament parece ser la opción más equilibrada.

Cruzar dos individuos (crossover, recombinación)

El objetivo de esta etapa es generar, a partir de individuos ya existentes (parentales), nuevos individuos (descendencia) que combinen las características de los anteriores. Este es otro de los puntos del algoritmo en los que se puede seguir varias estrategias. Tres de las más empleadas son:
Cruzamiento a partir de uno solo punto: se selecciona aleatoriamente una posición que actúa como punto de corte. Cada individuo parental se divide en dos partes y se intercambian las mitades. Como resultado de este proceso, por cada cruce, se generan dos nuevos individuos.
Cruzamiento a partir múltiples puntos: se seleccionan aleatoriamente varias posiciones que actúan como puntos de corte. Cada individuo parental se divide por los puntos de corte y se intercambian las partes. Como resultado de este proceso, por cada cruce, se generan dos nuevos individuos.
Cruzamiento uniforme: el valor que toma cada posición del nuevo individuo se obtiene de uno de los dos parentales. Por lo general, la probabilidad de que el valor proceda de cada parental es la misma, aunque podría, por ejemplo, estar condicionada al fitness de cada uno. A diferencia de las anteriores estrategias, con esta, de cada cruce se genera un único descendiente.

Mutar individuo

Tras generar cada nuevo individuo de la descendencia, este se somete a un proceso de mutación en el que, cada una de sus posiciones, puede verse modificada con una probabilidad p. Este paso es importante para añadir diversidad al proceso y evitar que el algoritmo caiga en mínimos locales porque todos los individuos sean demasiado parecidos de una generación a otra.

Existen diferentes estrategias para controlar la magnitud del cambio que puede provocar una mutación.

Distribución uniforme: la mutación de la posición i se consigue sumándole al valor de i un valor extraído de una distribución uniforme, por ejemplo, una entre [-1,+1].

Distribución normal: la mutación de la posición i se consigue sumándole al valor de i un valor extraído de una distribución normal, comúnmente centrada en 0 y con una determinada desviación estándar. Cuanto mayor la desviación estándar, con mayor probabilidad la mutación introducirá cambios grandes.

Aleatorio: la mutación de la posición i se consigue reemplazando el valor de i por nuevo valor aleatorio dentro del rango permitido para esa variable. Esta estrategia suele conllevar mayores variaciones que las dos anteriores. Hay que tener en cuenta que, debido a las mutaciones, un valor que inicialmente estaba dentro del rango permitido puede salirse de él. Una forma de evitarlo es: si el valor tras la mutación excede alguno de los límites acotados, se sobrescribe con el valor del límite. Es decir, se permite que los valores se alejen como máximo hasta el límite impuesto.

2.4. Ejemplos y ejercicios

Problema de las 8 reinas. Un tablero de ajedrez consta de 8 columnas y 8 filas, en ellas la reina se puede desplazar tantas casillas como quiera en cualquiera de los siguientes ocho sentidos mostrados (fig.2.3).

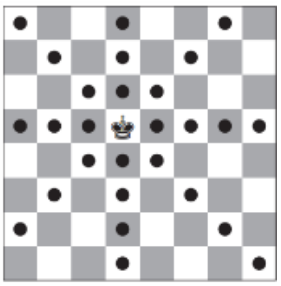


Figura 2.3: Desplazamientos permitidos de la reina en ajedrez

En este problema la solución obtenida debe tener únicamente una reina por columna, así, se representa un individuo mediante un vector de ocho elementos en el que cada elemento representa una de las columnas del tablero, y el valor del elemento representa la fila en la que se encuentra la reina dentro de esa columna. Numerando las filas de 0 a 7 (en lugar de 1 a 8) la figura 2 recoge el ejemplo de un individuo cuya representación vectorial sería (2,4,0,6,1,3,5,1) fig.2.4.

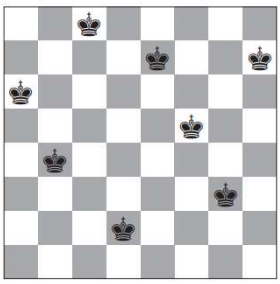


Figura 2.4: Problema 8 reinas una solución

Como el AG trabaja con estos números, pero en formato binario, para representar ocho valores diferentes se utilizan tres bits (23 =8), así, el cromosoma constará de un total de 8 x 3 = 24 bits.
En la fig.2.4, el vector queda representado por la siguiente cadena de bits: 010 100 000 110 001 011 101 001. Numerando las filas de 0 a 7 en lugar de 1 a 8 se aprovechan al máximo los 3 bits (ya que el número 8 en binario es 1000, número de 4 bits).

Así, ya se ha definido el **genotipo** de un individuo, que consistirá en una cadena de veinticuatro bits, la interpretación de estos datos, es decir, la disposición de las reinas sobre el tablero sería el **fenotipo**.
El siguiente paso es definir la **función de evaluación**. Esta nos dará una valoración de lo bien que un individuo se adapta al medio, es decir, lo cerca o lejos que se encuentra de la solución al problema.
La elección de esta función, junto con la definición del cromosoma, son los dos pasos más críticos, y son los que marcarán en gran medida el éxito del algoritmo, así como su rapidez.
La función de evaluación debe tratar de penalizar aquellas disposiciones de las reinas en las cuales una fila o una diagonal estén ocupadas por más de una reina, se sabe que una reina situada en la fila x y en la columna y ocupará también las diagonales x + y y x - y.
En el tablero de ajedrez existen ocho filas, ocho columnas, quince diagonales en una dirección y quince en la otra.
La función de evaluación debe utilizar tres vectores, uno con ocho elementos representando las filas y los otros dos, con quince elementos cada uno, que representan las diagonales. Inicialmente todos los elementos valen -1. Para evaluar una posible solución se atiende a las posiciones de las reinas. A partir de las coordenadas de cada una de ellas se obtienen tres valores: el número de fila que ocupa, y los números de las dos diagonales.
Con estos valores se incrementa en una unidad los elementos que le correspondan en los vectores anteriormente mencionados.
Tras estas operaciones, se realiza la suma de todos los elementos mayores que cero, siendo la mejor solución aquella que se encuentre más cercana al valor cero, pues sólo la coincidencia de dos o más piezas en una fila o diagonal producirá valores mayores que cero en el vector correspondiente.

2.5. Implementación

Problema de implementación de un algoritmo genético para la asignación de aulas en un centro de estudio.
Los AGs han demostrado ser una herramienta muy eficiente para resolver problemas de optimización.
Por otra parte, la asignación de aulas en cualquier centro educativo, en particular, aquellos centros que no disponen de gran cantidad de aulas para hacer frente a la demanda periódica de cursos se convierte en un problema de optimización.
En la mayoría de los centros educativos esta asignación se realiza semestre a semestre, en forma manual, por lo que se hace necesaria la asignación de personal dedicado sólo a esta labor por varios días.
Se busca proponer un sistema que reduzca el tiempo de respuesta a unos cuantos segundos, y que además encuentre una solución óptima en la mayoría de las pruebas realizadas.
Además, que ofrezca facilidades adicionales como flexibilidad a la hora de definir horarios, cursos y tipos de aulas, así como la capacidad para probar formas diversas de asignación de aulas dependiendo de los requisitos de cada curso.

Resultados requeridos

1. Automatización de una tarea que hasta ahora se ha realizado de forma manual.
2. Reducción del tiempo necesario para encontrar una solución (actualmente el tiempo requerido es de varios días).
3. La solución obtenida debe ser muy cercana a la óptima, si no es que la óptima.
4. La herramienta debe ser muy sencilla de comprender y de utilizar.
5. La herramienta debe ser muy versátil: con pequeñas y sencillas modificaciones que permita ajustarse a cambios en la descripción original del problema.

Descripción de la herramienta: asignador de aulas

A continuación, se describen los componentes de la herramienta, denominada de ahora en adelante: **asignador de aulas**.
Como primer componente se tienen los **datos de entrada**, la información que aquí se brinda es indispensable, debe ser **completa y correcta**, ningún dato puede **omitirse**, pues no permitiría el adecuado funcionamiento de la herramienta.
Inicialmente debe indicarse el **número de aulas**, la respectiva **identificación** y el **tipo** de cada aula, así como la **cantidad de tipos distintos** que componen este grupo.

Posteriormente se debe editar, dentro de un formato determinado, el **catálogo** de cada uno de los cursos a los que se le desea asignar un aula.
Es importante aclarar que la edición de este catálogo sólo se realiza la primera vez, en usos posteriores sólo será necesario actualizarlo si fuera el caso.
Como segunda parte se tiene la **ejecución del AG**, que, con base en los datos suministrados, encuentra una solución utilizando las operaciones que más adelante se describirán.
Finalmente, se tienen los resultados del programa, los cuales brindan la **configuración completa** de los horarios para cada una de las aulas especificadas en la entrada de datos.
Es decir, para cada aula se configura un horario que indica las horas en que los distintos cursos serán impartidos.
Además, debe permitir ver todos aquellos cursos del catálogo a los cuales **no les fue posible asignarle** un aula.

Composición del AG

Individuo

Este es el objeto o clase principal del algoritmo, representa la información de una posible solución, ya que posee toda la configuración de los horarios de las aulas.
El método de representación utilizado permite una mayor precisión y complejidad que el método comparativamente restringido de utilizar sólo números binarios.
Su composición es la siguiente:
I. Un conjunto de cromosomas representado por un arreglo bidimensional de hileras llamado Cromosoma, cada fila almacena la información propia y detallada de un curso de la siguiente manera:

1. La sigla del curso. Ej.: *CI - 201*.
2. El grupo del curso. Ej.: *01*.
3. El primer día de su horario. Ej.: *L* (Lunes).
4. Las horas del primer día. Ej.: *07:00- 8:50*.
5. El último día de su horario. Ej.: *J* (Jueves).
6. Las horas del segundo día. Ej.: *07:00-8:50*.
7. El requisito de tipo de aula. Ej.: *Laboratorio*.
8. El aula asignada.

II. Un campo numérico entero que indica la cantidad de **cromosomas** (cursos) del individuo.

III. Un campo numérico entero que indica la cantidad de choques que se presentan a la hora de crear el horario; un **choque** se da cuando no es posible asignarle el aula designada que contiene en su cromosoma.

Padres

Este objeto es un arreglo de individuos, que contiene los padres iniciales que se han generado al azar y empezarán a cruzarse cuando dé inicio el algoritmo.

Élite

Es un arreglo de individuos, que almacena los dos individuos más aptos que se van generando a través del algoritmo.

Aulas

Este objeto manipula toda la información referente a las aulas, entre los campos que maneja se encuentran:

- Un arreglo de hileras bidimensional que almacena todos los horarios de las aulas a las que se les desea asignar cursos. Cada horario de un aula abarca 7 columnas (días) y 14 filas (horas).
- Otro arreglo bidimensional que guarda las aulas según el tipo.
- Un campo numérico que almacena la cantidad de aulas.
- Un campo numérico que indica la cantidad de tipos de aulas.

Operadores básicos

Aptitud

La aptitud del individuo está basada en el número de choques de horario que presente a la hora de realizar los horarios.
La función de aptitud en la implementación de este algoritmo genético se basa en contar el número de choques presentes en la asignación de las aulas con respecto a la hora.
Dicho en otras palabras, se da cuando dos cursos con una misma hora y tipo de requisito fueron asignados a la misma aula y, por lo tanto, se genera un choque.

Cruce

La función de cruce es la que permite la creación de una nueva generación deindividuos; en nuestro caso, se implementó un criterio del cruce básico. De esta manera, el punto de cruce se establece en cuatro distintas formas:

- Se toma la primera mitad del cromosoma padre y se la segunda mitad del cromosoma madre para generar un nuevo hijo.

- Se toma la primera mitad del cromosoma madre y se la segunda mitad del cromosoma padre para generar un nuevo hijo.
- Se divide el cromosoma en cuatro partes y se asigna la sección 1.3 del padre y 2.4 de la madre.
- Se divide el cromosoma en cuatro partes y se asigna la sección 2.4 del padre y 1.3 de la madre.

Elitismo

La función de elitismo se basa en escoger los dos mejores padres de cada generación, o sea, aquéllos que poseen la mayor aptitud, y por lo tanto, representan buenas soluciones al problema. ?

Esta élite podría no sufrir alteraciones si la nueva generación no produce individuos con una aptitud mayor que la que poseen los individuos de la élite.

Unidad 3

OPTIMIZACION BASADA EN PROCESOS FÍSICOS

"Describir algunas técnicas para el control de la exploración del espacio de soluciones y el funcionamiento del algoritmo de recocido simulado."

Objetivos específicos 1 y 2

3.1. Introducción. Recocido como proceso en la metalurgia

EL **recocido** es un proceso térmico que sirve para obtener estados de baja energía para un sólido en un baño caliente o en fundición. Este proceso altera las propiedades físicas y, a veces, químicas de un material para aumentar su ductilidad y reducir su dureza.
El proceso consta de dos pasos:
Aumentar la temperatura hasta un valor máximo en el que el solido casi se derrite.
Decrementar suavemente la temperatura hasta que las partículas se agrupen en el estado fundamental del sólido (estado estable).
En la fase cuasi- liquida, las partículas del sólido se agrupan aleatoriamente.
En la fase estable, las partículas se agrupan en un distribución altamente estructurada, en este estado, la energía del sistema es mínima.
El estado estable del sólido se alcanza solo si la temperatura máxima es suficientemente alta y el enfriamiento se hace suficientemente suave.

En otro caso, el sólido se distribuye en un estado meta-estable.
Estado fundamental = Las partículas forman retículas perfectas y el sistema está en su mas bajo nivel energético.
Metropolis, Rosenbluth y Teller introdujeron un algoritmo simple para simular la evolución de un sólido en un baño caliente.
El algoritmo se basa en técnicas Monte Carlo y genera una secuencia de estados del sólido de la forma siguiente.
Dada una sustancia, no todas las moléculas tienen la misma energía, sino que se encuentra en diferentes niveles, el menor de los cuales se denomina estado fundamental.
Si la sustancia esta a $0°$K todas las moléculas están en su estado fundamental.
Si está a temperaturas mayores las moléculas ocupan estados superiores de energía.
La distribución de partículas en los diferentes niveles sigue la distribución de Probabilidad de Boltzman.
El algoritmo de Metropolis puede aplicarse a la resolución de un problema combinatorio si se realiza la siguiente analogía:

- Las soluciones del problema son equivalentes a los estados del sistema físico
- El costo de una solución es equivalente a la energía de un estado
- Un parámetro de control juega el papel de la temperatura

En una iteración del proceso de Enfriamiento una solución actual x es perturbada para producir una nueva alternativa que puede reemplazarla o no.
El reemplazo será aceptado o no en función de una regla de aceptación.
El proceso se repite iterativamente hasta que se considere que se llego al optimo (la solución converge).
Perturbar = obtener una nueva solución x' a partir de x (es decir, generar un elemento de N(x)).
Enfriamiento: En el transcurso del proceso usar un parámetro dinámico T que toma valores cada vez menores.

Regla de aceptación (Algoritmo de Metropolis)
Si la nueva solución x' tiene mejor valor Z que la anterior, la reemplaza

$$x = x'$$

si no, la probabilidad de que x' se acepte es:

$$P(\Delta Z) = exp(\frac{\Delta Z}{CT}) \tag{3.1}$$

Donde:

ΔZ = (Z(x') - Z(x))

$C = 1{,}38054 \times 10^{-23}$ (Cte. de Boltzman)

T es la temperatura

La distribución da la probabilidad de que el sólido esté en el estado i con energía Ei a temperatura T. Esta distribución se denomina Probabilidad de Boltzman.
Se escoge un numero aleatorio n uniformemente distribuido en el intervalo (0,1), ese numero es comparado con P(ΔZ).
Si $n < P(\Delta Z)$, x' reemplaza a x como solución actual.
Si $n \geq P(\Delta Z)$, x se usa de nuevo como paso inicial de una próxima iteración.
Al principio T es un valor alto (fundición), y luego va disminuyendo (enfriamiento), es decir, cada vez es menor la probabilidad de que la nueva alternativa reemplace a la anterior.

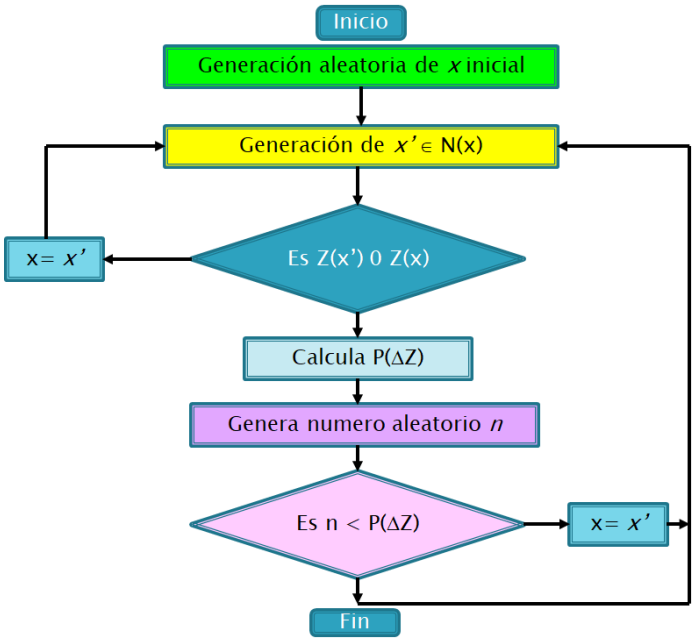


Figura 3.1: Diagrama de flujo del Algoritmo de Mteropolis

Un plan de enfriamiento especifíca:
Una sucesión infinita de valores del parámetro de control, es decir:

- Un valor inicial del parámetro, c0
- Una función para decrementar el parámetro de control
- Un valor final del parámetro de control

- Un número infinito de transiciones a cada valor del parámetro de control

Parámetros

1. T_0 = temperatura inicial (alta)
2. Velocidad de enfriamiento $L(T)$ = número de iteraciones en que se usa la misma temperatura antes de disminuirla para otras $L(T)$ iteraciones
3. Enfriamiento α = grado de disminución de temperatura, se recomienda un $\alpha \rightarrow$ (0,8, 0,99)
4. T_f = Temperatura final ($T_f \rightarrow 0$)

A partir de una temperatura inicial, disminuir lentamente la temperatura hasta alcanzar el punto de congelación.

Figura 3.2: Enfriamiento

3.2. Recocido simulado: la metaheurística

UN modo de evitar que una búsqueda local finalice en óptimos locales (suele ocurrir con los algoritmos tradicionales de búsqueda local), es permitir que algunos movimientos sean hacia soluciones peores.
Pero si la búsqueda está avanzando realmente hacia una buena solución, estos movimientos *de escape de óptimos locales* deben realizarse de un modo controlado.
En el caso del Recocido Simulado (RS), esto se realiza controlando la frecuencia de los movimientos de escape mediante una función de probabilidad que hará disminuir la probabilidad de estos movimientos hacia soluciones peores conforme avanza la búsqueda (y por tanto se está más cerca, previsiblemente, del óptimo local).
Así, se aplica la filosofía habitual de búsqueda de diversificar al principio e intensificar al final.
Con el aumento de iteraciones disminuye la probabilidad de aceptar soluciones peores que la actual.

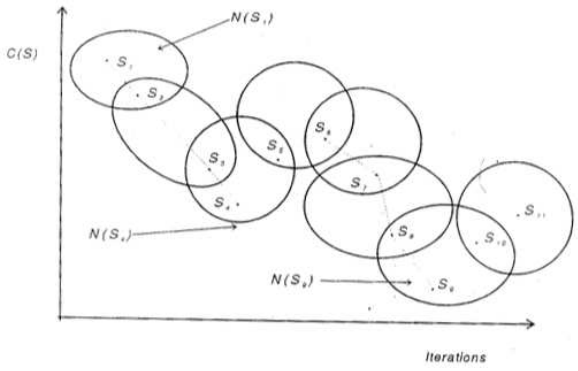


Figura 3.3: Espacio de soluciones

Algoritmo básico de RS

El algoritmo de RS es un método de búsqueda por entornos caracterizado por un criterio de aceptación de soluciones vecinas que se adapta a lo largo de su ejecución.
Hace uso de una variable llamada Temperatura, T, cuyo valor determina en qué medida pueden ser aceptadas soluciones vecinas peores que la actual.
La variable Temperatura se inicializa a un valor alto, denominado Temperatura inicial, T_0, y se va reduciendo cada iteración mediante un mecanismo de enfriamiento de la temperatura, $\alpha()$, hasta alcanzar una Temperatura final, T_f.
Esta probabilidad depende de la diferencia de costes entre la solución actual y la vecina, δ, y de la temperatura T:

$$P_{aceptacion} = exp^{(-\delta/T)}$$

A mayor temperatura, mayor probabilidad de aceptación de soluciones peores. Así, el algoritmo acepta soluciones mucho peores que la actual al principio de la ejecución (exploración) pero no al final (explotación).
A menor diferencia de costes, mayor probabilidad de aceptación de soluciones peores.
Una vez finalizada la iteración, es decir, tras generar L(T) soluciones vecinas, se enfría la temperatura y se pasa a la siguiente iteración.

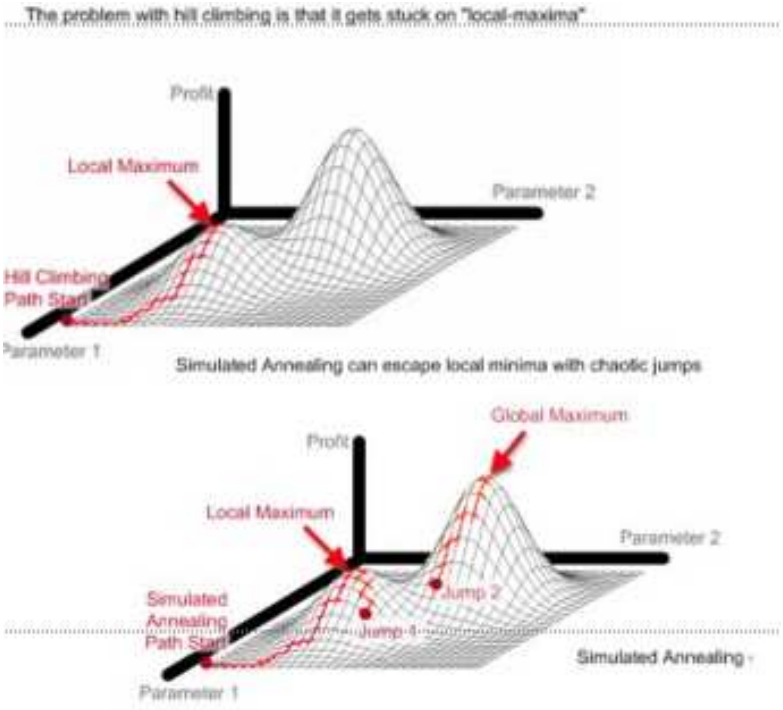


Figura 3.4: Comparación entre Ascenso de Colina y RS

Mecanismo de transición de soluciones

1. Generación de una nueva solución
   - Definición del conjunto de vecinos
   - Selección de un elemento de dicho conjunto
2. Cálculo de la diferencia de costos entre la solución actual y la vecina
3. Aplicación del criterio de aceptación

Secuencia de enfriamiento

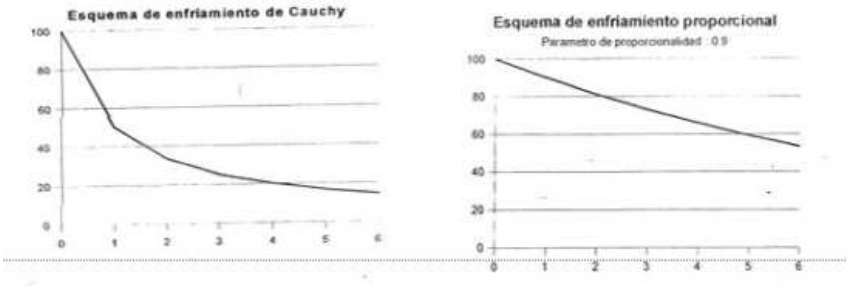
1. **Valor Inicial** del parámetro de control (temperatura)
   - No parece conveniente considerar valores fijos independientes del problema.
   - PROPUESTA: $T_0 = (\mu - ln(\varphi))C(S_0)$
   - Tanto por uno φ de probabilidad de que una solución sea un μ por uno peor que la solución inicial S
2. **Mecanismo de enfriamiento**

   Exisisten varios mecanismo de enfriamiento
   - Enfriamiento basado en sucesivas temperaturas descendentes fijadas por el usuario
   - Enfriamiento con descenso constante de temperatura
   - Descenso exponencial $T_{k+1} = \alpha \cdot T_k$

     k = no. de iteración actual, α consante cercana a 1 (por lo general, $\alpha \in [0{,}8 - 0{,}99]$)
   - Criterio de Boltzman $T_k = \frac{T_0}{1+log(k)}$
   - Esquema de Cauchy $T_k = \frac{T_0}{1+k}$
   - Para controlar el número de iteraciones (Cauchy modificado):

     $T_{k+1} = \frac{T_k}{1+\beta \cdot T_k)}$

     Para ejecutar exactamente M iteraciones $\Longrightarrow \beta = \frac{(T_0 - T_f)}{(M \cdot T_0 \cdot T_f)}$



3. **Velocidad de enfriamiento**
   - L(T) debe de ser lo suficientemente grande para que el sistema llegue a alcanzar su estado estacionario para esa temperatura
   - Lo habitual es que sea un valor fijo pero hay una variante que permite decidir mejor cuando alcanzar la iteración actual y enfriar

- Consiste en enfriar cuando se dé una de las dos situaciones siguientes:
  - Se ha generado un número máximo de vecinos (*max_vecinos*)
  - Se han aceptado un número máximo de vecinos (*max_exitos*)
- Lógicamente, *max_vecinos* tiene que ser mayor que *max_exitos*. Una buena proporción puede ser: *max_exitos*=0.1**max_vecinos*

4. **Condición de parada**

- En teoría, el algoritmo debería finalizar cuando T = 0. En la practica, se para:
  - cuando T alcanza o está por debajo de un valor final T_f, fijado previamente, o
  - después de un número fijo de iteraciones
- Como es difícil dar valor de T_f se suele usar un número fijo de iteraciones
- Una buena opción es parar cuando no se haya aceptado ningún vecino de los L(T) generados en la iteración actual (*max_exitos*=0)

  En ese caso es probable que el algoritmo se haya estancado y no vaya a mejorar lasolución obtenida

  Combinando este criterio de parada y la condición de enfriamiento de los *max_vecinos* y *max_exitos* se obtiene un equilibrio en la búsqueda que evita malgastar recursos

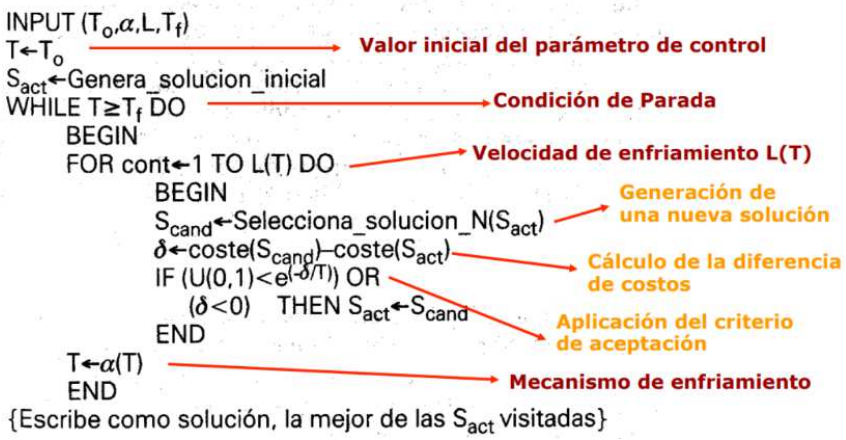


Figura 3.5: Pseudo-código de Recocido Simulado

3.3. Ventajas y retos del uso del RS

EL RS es un método metaheurístico que permite solucionar problemas de optimización global. Debido a su naturaleza estocástica, este método garantiza que, en sentido probabilístico, el óptimo global será alcanzado.
Su principal desventaja es que el tiempo de búsqueda puede tornarse infinito.
El RS es especialmente efectivo para resolver problemas de larga escala (multidimensionales) en los que el óptimo global se encuentra escondido detrás de muchos óptimos locales.
En el recocido simulado, tres factores son cruciales para la convergencia al óptimo global:

I. Mecanismo de caminada o distribución de visitación;
II. Mecanismo de decisión o probabilidad de aceptación;
III. Mecanismo de direccionamiento o esquema de enfriamiento.

El **mecanismo de caminada** se refiere a la búsqueda de una nueva solución a partir de la solución actual, por medio de un muestreo inteligente del espacio solución.
Lo que caracteriza a un buen mecanismo de caminada es que cuando la temperatura de recocido es alta, el algoritmo es capaz de dar saltos largos desde la solución actual, mientras que cuando la temperatura es baja, el algoritmo apenas consigue saltar a puntos vecinos, indicando la presencia del algoritmo en la cuenca de atracción al óptimo global.
El **mecanismo de decisión** o probabilidad de aceptación indica la regla de cumplimiento mediante la cual una nueva solución puede ser aceptada. ?
Una nueva solución será aceptada siempre que disminuya el valor de la función objetivo.
Sin embargo, algunas de aquellas soluciones indeseadas que aumentan el valor de la función objetivo pueden ser eventualmente aceptadas.
Esto último es lo que permite al algoritmo evitar quedar atrapado en óptimos locales.
El **mecanismo de direccionamiento** o esquema de enfriamiento se refiere a la manera en que la temperatura dirige el proceso de enfriamiento.
Cuanto más alta es la temperatura, más fácil es dar saltos largos.
En cambio, cuanto menor es, los saltos en torno a la vecindad de la solución actual son recurrentes.
Un buen mecanismo de direccionamiento evitará movimientos alrededor de un óptimo local a fin de acortar el tiempo de convergencia al óptimo global.
En otras palabras, la temperatura actúa como una fuente de estocasticidad extremadamente conveniente para rehuir de óptimos locales, tal que, cerca del final del proceso de optimización, el sistema esté inmerso en la cuenca de atracción hacia el óptimo global.

Con base en lo anterior el proceso del algoritmo de RS tiene las siguientes ventajas principales:

1. La eficiencia de búsqueda iterativa es alta y se puede paralelizar;
2. Existe una cierta probabilidad en el algoritmo de aceptar una solución que es peor que la solución actual, por lo que se puede alejar de un óptimo local en cierta medida;
3. La solución obtenida por el algoritmo no tiene nada que ver con el estado de solución inicial S, por lo que tiene un cierto grado de robustez;
4. Tiene convergencia asintótica y se ha demostrado teóricamente que es un algoritmo de optimización global que converge a la solución óptima global con probabilidad l.

El algoritmo de RS también tiene las siguientes desventajas:

1. **El problema de establecer el valor inicial de temperatura T**: El ajuste del valor inicial de la temperatura T es uno de los factores importantes que afectan el rendimiento de búsqueda global del algoritmo de RS. Si la temperatura inicial es alta, es probable que se busque la solución óptima global, pero costará mucho tiempo de cálculo; por el contrario, el tiempo de cálculo se puede guardar, pero el rendimiento de la búsqueda global puede verse afectado.
2. **Problema de velocidad de recocido**: el algoritmo de RS también está estrechamente relacionado con la velocidad de recocido. En términos generales, la búsqueda $completa$ a la misma temperatura es bastante necesaria, pero también requiere tiempo de cálculo. El aumento en el número de ciclos aumentará inevitablemente la sobrecarga computacional.
3. **Problemas de gestión de temperatura**: La gestión de la temperatura también es uno de los problemas con los que es difícil tratar el algoritmo de RS. En aplicaciones prácticas, dado que debe considerarse la viabilidad de la complejidad del cálculo, a menudo se utilizan los siguientes métodos de enfriamiento:

3.4. Implementación

SE tiene una instancia del problema del agente viajero con 25 ciudades, el objetivo es encontrar el menor recorrido, en el que, partiendo de una ciudad se pase una vez por todas las demás y se regrese a la ciudad inicial.
El número de combinaciones es del orden de:

15,511,210,043,330,985,984,000,000

Es decir $1{,}5 \times 10^{24}$

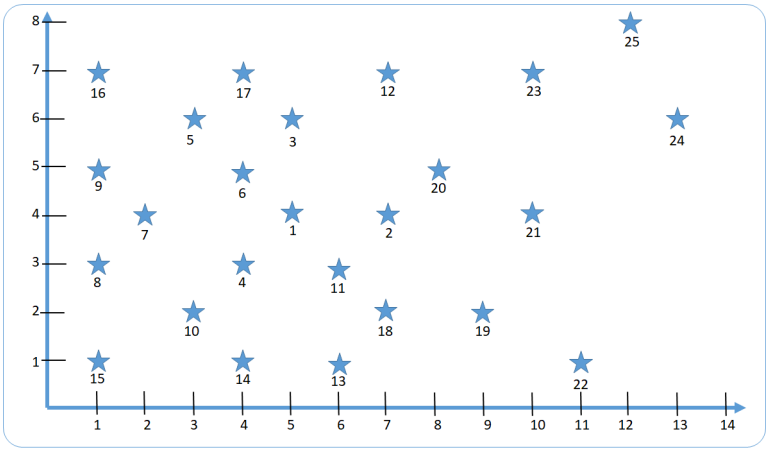


Figura 3.6: Mapa de ciudades (espacio de búsqueda)

Ciudad	Coordenadas	Ciudad	Coordenadas
1	5,4	14	
2	7,4	15	
3	5,6	16	
4	4,3	17	
5	3,6	18	
6		19	
7		20	
8		21	
9		22	
10		23	
11		24	
12		25	
13	6,1		

Tabla 3.1: Ubicación de las ciudades

Unidad 4

OPTIMIZACION BASADA EN COLONIA DE HORMIGAS

"Describir el funcionamiento de un algoritmo genérico de optimización con colonia de hormigas."

Objetivo específico 1

4.1. Introducción

La principal cualidad de los llamados insectos sociales (hormigas, abejas, avispas, etc.) es la de formar parte de un grupo auto-organizado, cuya palabra clave es **simplicidad**.
Las hormigas resuelven todos los días problemas complejos gracias a la suma de interacciones simples entre cada individuo.

Figura 4.1: Hormiga

Por ejemplo, siguiendo la pista de feromonas dejadas por sus congéneres, una hormiga es capaz de tomar el camino mas corto para ir del nido al alimento.

La localización de éste es fácil pero la elección es mucho menor.
Resulta realmente interesante analizar como las hormigas buscan su alimento y logran establecer el camino mas corto entre las fuentes de alimento (F) para luego regresar a su nido (N).

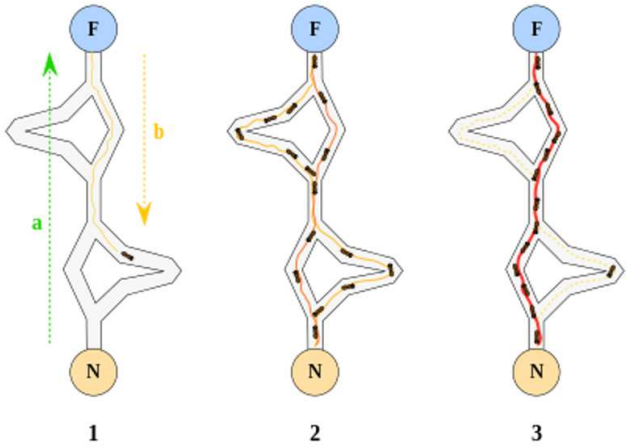


Figura 4.2: Comportamiento de la Hormiga

Para esto, al moverse una hormiga, deposita una sustancia química denominada **feromona** como una señal odorífera para que las demás puedan seguirla (línea roja en la fig.4.2).
Las feromonas son un sistema indirecto de comunicación química entre animales de una misma especie que transmiten información acerca del estado sicológico, reproductivo y social, así como la edad, el sexo y el parentesco del animal emisor, las cuales son recibidas en el sistema olfativo del animal receptor quien interpreta esas señales, jugando un papel importante en la organización y la supervivencia de muchas especies.
Estos insectos son considerados uno de los grupos más exitosos entre los insectos, cuentan con aproximadamente unas 14.000 especies descritas, aunque se estima que pueden ser 20.000 especies; distribuidas en 350 géneros y 12 subfamilias.
Se distribuyen en prácticamente todos los ecosistemas terrestres menos en la Antártida y algunas pocas islas como Groenlandia, Islandia y Polinesia; tan amplia distribución se debe a la diversidad de sus hábitos alimenticios, su organización social, su capacidad de adaptación y poblaciones estables, que les permite ocupar una amplia variedad de nichos ecológicos.
El comportamiento de las hormigas y su estructura social han sido ampliamente estudiado, siendo considerado uno de los sistemas sociales más complejos en la naturaleza.
Las colonias de hormigas están conformadas principalmente por hembras estériles y algunos pocos individuos reproductores.

Estas diferencias, que no solo son fisiológicas, sino también morfológicas, y que implican una diferenciación en el comportamiento de las hormigas, su división de trabajo y funciones, permiten catalogarlas en **castas**. -

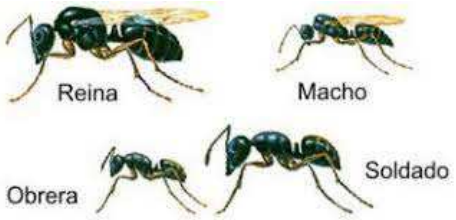


Figura 4.3: Castas de Hormigas

Por lo general, cada casta presenta características morfológicas y fisiológicas propias, que las diferencia una de las otras; pero eso no implica que dentro de cada casta se observe polimorfismo, entre la casta de las obreras podemos identificar distintos tipos de obreras que se diferencian principalmente por su tamaño y el trabajo que desempeñan para la colonia.

4.2. Hormigas artificiales y naturales

Hormiga natural

SE trata de insectos pequeños, cuyas tallas varían desde los 0,70 mm a los 60 mm; y como todos los insectos cuentan con un exoesqueleto que les brinda una cubierta protectora.

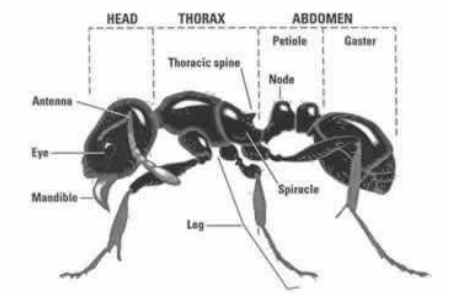


Figura 4.4: Anatomía de Hormiga

Una característica interesante del comportamiento de las colonias de hormigas es cómo pueden encontrar los caminos más cortos entre el hormiguero y la comida. Para realizar esto en su recorrido, depositan una sustancia química llamada feromona que todas pueden oler (estimergia).
Este rastro permite a las hormigas volver a su hormiguero desde la comida. Cada vez que una hormiga llega a una intersección, decide el camino a seguir de un modo probabilístico, dando mayor probabilidad los caminos con un alto rastro de feromona.
El comportamiento de las hormigas dentro de las colonias es tan organizado y unificado, que parecieran conformar un superoganismo, demostrando un alto grado de cooperación y trabajo colectivo, con división de trabajo y diferentes vías de comunicación que permiten la coordinación y cooperación, así como la resolución de problemas.
Las colonias pueden estar conformadas desde unas pocas decenas de individuos hasta millones de ellos, en su gran mayoría hembras estériles; que, por lo general, habitan $nidos$ subterráneos u hormigueros, conformados por galerías que se conectan a la superficie mediante túneles y cuyas entradas se reconocen por la presencia de montículos de tierra, arena o arcilla.
Los hormigueros pueden ser simples o complejos, aunque no todas las especies construyen hormigueros terrestres, algunas pueden llegar a elaborar los hormigueros en los árboles, aprovechar los refugios naturales o construir estructuras vivas, enlazando sus cuerpos. Incluso algunas especies no requieren nidos, ya que son nómadas.

Hormiga artificial

Los algoritmos de Optimización por Colonia de Hormigas reproducen el comportamiento de las hormigas reales en una **colonia artificial de hormigas** para resolver problemas complejos de camino mínimo.

Figura 4.5: Hormiga artificial

Cada hormiga artificial es un mecanismo probabilístico de construcción de soluciones al problema (un agente que imita a la hormiga natural) que usa: - Unos rastros de feromona (artificiales) τ que cambian con el tiempo para reflejar la experiencia adquirida por los agentes en la resolución del problema. - Información heurística η sobre la instancia concreta del problema.
Busca soluciones validas de coste mínimo para el problema a resolver.
Tiene una memoria en la que almacena el camino recorrido para poder reconstruir soluciones validas, evaluar las soluciones generadas, o bien reconstruir el camino completo llevado por la hormiga.
Lleva a cabo los movimientos aplicando reglas de transición, que en función de los rastros de feromona que están disponibles, de los valores heurísticos, de la memoria privada de la hormiga y de las restricciones del problema.
Durante su recorrido, modifica el entorno depositando una cantidad de feromona.
Una vez que una hormiga ha construido una solución valida puede recorrer el camino de vuelta actualizando los espacios visitados.
Para el desarrollo del simulador de colonia de hormigas, se puede elegir el modelo del autómata celular, donde cada hormiga lleva a cabo un movimiento cada vez (no concurrentemente), y se actualiza el entorno.
Las premisas que sigue la construcción de la hormiga artificial son las siguientes:
Cada hormiga realiza un evento unitario que consiste en desplazarse a una celda adyacente (arriba, abajo, izquierda o derecha), y después cede turno a otra hormiga.
La hormiga solo es sensible a las señales químicas encontradas en el medio (la feromona depositada en el espacio).

Las señales a las que es sensible la hormiga son una cantidad de feromona depositada que indica la presencia del hormiguero, y otra que indica la presencia de comida.
Una hormiga puede tener tres estados:

1. **Hambrienta**: Cuando una hormiga está en estado hambrienta, y a falta de señal de comida cercana, deambula al azar con igual probabilidad de desplazarse a cualquiera de las casillas adyacentes, excepto de la casilla de la que procede (cuya probabilidad es menor). Cada paso que da una hormiga hambrienta deja en la celda en que se encuentra una señal de hormiguero. En el paso siguiente, dejar a una unidad menos de la señal hormiguero, cuando termine la señal de hormiguero, la hormiga pasara a estado perdida, hasta que encuentre el hormiguero o, en su deambular aleatorio, encuentre comida.

2. **Alimentada**: Cuando una hormiga encuentra comida, cambia su estado a alimentada, y trata de volver atrás usando el mismo camino que siguió a la ida. En cada paso depositará una señal de comida, que ira disminuyendo paso a paso.

3. **Perdida**: Las hormigas perdidas intentan regresar al hormiguero asignando mas probabilidad de movimiento a las celdas adyacentes con mayor cantidad de feromona de hormiguero. Las hormigas hambrientas se mueven con mayor probabilidad a casillas con cantidad de señal de comida mas alta. Cuando una hormiga perdida llega al hormiguero se convierte en una hormiga hambrienta.

En cada iteración la cantidad de feromona de cada celda disminuye, siendo multiplicada por un numero menor que 1 (para evitar quedar en mínimos locales).

4.3. Estructura genérica de un algoritmo de optimización con colonia de hormigas

La optimización por Colonia de Hormigas (ACO) es una metaheurística en la que una colonia de hormigas artificiales cooperan para encontrar buenas soluciones a problemas difíciles de optimización discreta.
De este modo, asignando recursos a un grupo de agentes simples que se comunican indirectamente por medio del ambiente, surgen buenas soluciones por su interacción cooperativa.
Esta meta-heurística emula el comportamiento de las colonias de hormigas que buscan la ruta más corta entre su nido y la fuente de comida.
Las hormigas artificiales en un algoritmo AC O son procedimientos de construcción estocástica que incrementalmente construyen una solución agregando componentes de solución a una solución parcial.
Lo que principalmente compone a un algoritmo ACO es la Construcción de Soluciones, la Actualización de Feromonas y las Acciones Independientes:

- **Construcción de Soluciones**: maneja una colonia de hormigas que asincrónicamente visitan estados adyacentes del problema moviéndose hacia nodos vecinos del grafo de la construcción del problema. Se mueven aplicando una política de decisión local estocástica usando el camino de feromonas e información heuróstica. Con esto, las hormigas incrementalmente construyen la solución al problema de optimización. Una vez que una hormiga ha construido una solución, la hormiga evalúa la solución que será usada para que el procedimiento Actualización de Feromonas decida cuanta feromona se depositara.

- **Actualización de Feromonas**: es el proceso mediante el cual los caminos de feromona se modifican. Los valores de los caminos se pueden incrementar conforme las hormigas depositan la feromona, o pueden disminuir por su evaporación. El depósito de nueva feromona aumenta la probabilidad que los componentes que fueron usados por muchas hormigas, o por una sola con una muy buena solución, vuelvan a ser usados otra vez por hormigas en el futuro; mientras que la evaporación es una forma útil de olvidar, es decir, de permitir que soluciones que ya no han visitado pierdan gradualmente probabilidades de que vuelvan a ser utilizadas, de este modo, se evita una convergencia demasiado rápida hacia regiones subóptimas, y por lo tanto, favorece la exploración de nuevas áreas en el espacio de búsqueda.

- **Acciones Independientes**: son procedimientos utilizados para implementar acciones centralizadas que no pueden ser hechas por una sola hormiga. Un ejemplo de acción independiente sería la aplicación de un algoritmo de Diferenciación Evolutiva una vez que el algoritmo se haya quedado estancado.

En un algoritmo de Diferenciación Evolutiva (DE), cada valor de una variable en el cromosoma es representada por un número real. La aproximación trabaja creando una población inicial aleatoria de posibles soluciones factibles.
Un individuo es seleccionado para un remplazo aleatorio y tres individuos diferentes son seleccionados como padres.
Uno de estos individuos es seleccionado como el padre principal. Con cierta probabilidad, cada variable en el padre principal es cambiada. El cambio es tomado agregando a cada valor de la variable la diferencia entre los dos valores de esta variable en los otros dos padres.
En escencia, una fracción del vector padre principal es perturbado por los otros dos vectores.
Un algoritmo DE es un método de búsqueda en paralelo que utiliza p vectores de parámetros como población por cada generación G. La población inicial es elegida aleatoreamente y deberá cubrir el espacio de búsqueda de manera uniforme.

4.4. Implementación

LAS hormigas se comunican mediante sustancias químicas llamadas **feromonas**.
A medida que las hormigas se mueven, pueden dejar feromonas en el suelo, explica Marco Dorigo. *Y si detectan [un rastro de] feromona ya en el suelo, tienen una mayor probabilidad de seguir ese rastro en lugar de ir en una dirección diferente*.
Si muchas hormigas favorecen un sendero en particular, se produce un circuito de retroalimentación: las feromonas que han depositado atraerán a más hormigas al sendero, quienes aumentarán la concentración de feromonas a lo largo de él, lo que a su vez atraerá a más hormigas.

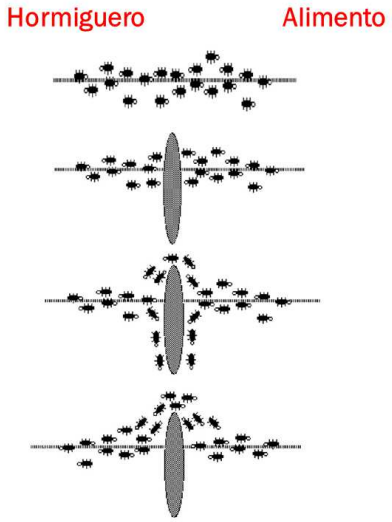


Figura 4.6: Flujos de hormigas

En 1990, Jean-Louis Deneubourg y sus colegas ofrecieron a una colonia de hormigas dos rutas que conectaban su nido con una zona que aún no habían explorado y donde podían encontrar comida.
En un experimento, una de las rutas era dos veces más larga que la otra y resultó que, en la mayoría de las pruebas, todas las hormigas finalmente eligieron el camino más corto.
Por supuesto, esta elección podría deberse a algún tipo de capacidad incorporada de las hormigas para medir la distancia.
Si este fuera el caso, entonces no se esperaría que las hormigas prefirieran una ruta sobre la otra si ambas rutas son igualmente largas, pero en experimentos en los que ambos caminos tenían la misma longitud, las hormigas también terminaron prefiriendo una ruta sobre la otra.

Deneubourg y sus colegas produjeron un modelo matemático que describe el mecanismo de las feromonas.
En el modelo, la probabilidad de que una hormiga elija uno de los caminos depende de la cantidad de feromona que ya tiene.
Al ejecutar el modelo en una computadora, los científicos encontraron exactamente el comportamiento mostrado por las hormigas reales.

Función probabilística de transición

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{k \in allowed_k} [\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}]^{\beta}} & \text{si j} \in \text{allowedk} \\ 0 & \text{de otra forma} \end{cases}$$

Donde:
- $p_{ij}^{k}(t)$ es la probabilidad de que la hormiga k seleccione la ruta que va de i a j en el tiempo t.
- $\tau_{ij}(t)$ es la cantidad de feromona que existe actulamente en la ruta que va directamente de i a j en el tiempo t.
- Sea $\tau_{ij}(t)$ que denota la intensidad de la ruta (i,j) en el tiempo t.
- α y β son parametros que llamaremos *fuerza heuristica*
- η_{ij} es el valor heuristico de esta ruta - en la aplicación clasica del TSP, esta es la distancia seleccionada para ser $1/distancia_{(i,j)}$, es decir, la distancia más corta, que es el valor más alto de la heuristica.

El sistema comienza con la misma pequeña cantidad de feromona artificial en todos los enlaces y los agentes toman decisiones aleatorias sobre por qué enlace viajan.
Cuando un agente ha encontrado una solución, por ejemplo, una ruta desde el nodo inicial hasta el nodo final evaluará la calidad de la solución.
Luego agregará una cantidad de feromona artificial a los enlaces contenidos en la solución que sea proporcional a la calidad de la solución.
La cantidad de feromona artificial asociada a un vínculo influye en la elección de un agente artificial ubicado en un nodo adjunto a ese vínculo: cuanta más feromona, mayor será la probabilidad de que el agente elija viajar por ese vínculo.
Si soy un agente artificial sentado en un nodo, al principio las opciones que tengo son todas las mismas, así que hago una elección aleatoria, explica Dorigo.
Después de un tiempo, algunos de los bordes tendrán más feromonas porque se han usado más o porque pertenecen a soluciones que son mejores que otras.
Entonces tengo una probabilidad ligeramente mayor de elegir esos bordes.
Esto sucede todo el tiempo, con muchos agentes en paralelo, y en algún momento el sistema encuentra una buena solución.
Hay tres factores que guian el modelo probabilistico:

1. **Visibilidad**, denotado η_{ij}, igual a la cantidad $1/d_{ij}$
2. **Camino**, denotado $T_{ij}(t)$
3. **Evaporación**

Estos tres factores juegan un rol esencial en la transición de la función probabilistica del Sistema de Hormigas.
A cambio, el peso de cualquiera de los factores en la función de transición está controlado por las variables α y β, respectivamente.
Los investigadores han realizado un importante estudio para derivar combinaciones óptimas de α:β.
Un valor alto de α significa que el camino es muy importante y por lo tanto las hormigas tienden a elegir bordes elegidos por otras hormigas en el pasado.
Por otro lado, los valores bajos de a hacen que el algoritmo sea muy similar a un algoritmo estocástico multicodicioso.
Después de cada recorrido de hormigas la intensidad del rastro en cada borde se actualiza usando lo siguiente:

Aumento de feromonas al final del tour con la fórmula:

$$\tau_{ij}(t+n) = \rho\tau_{ij}(t) + \Delta\tau_{ij}$$

Fijar $\Delta\tau_{ij}$ para todos los enlaces $\Delta\tau_{ij}$ fijos:
$\Delta\tau_{ij} =$

$$\Delta\tau_{ij} = \begin{cases} \Delta\tau_0 & \text{seleccionar la ruta} \\ 0 & \text{de otra forma} \end{cases}$$

Pseudocódigo

```
procedure ACO_MetaHeuristic
    while(not_termination)
       generateSolutions()
       daemonActions()
       pheromoneUpdate()
    end while
  end procedure
```

Estas son algunas de las variaciones más populares de los algoritmos de colonia de hormigas:

Sistema de Hormigas

El sistema de hormigas es el primer algoritmo OCH propuesto. Se corresponde con el funcionamiento descrito en la anterior sección.

Sistema Elitista de Hormigas

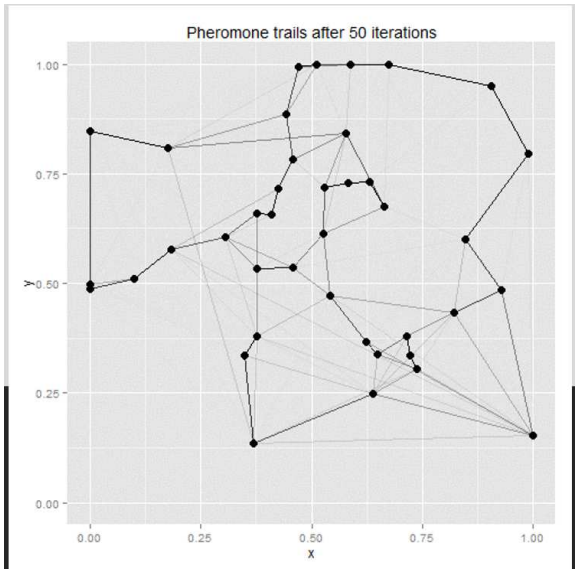


Figura 4.7: Ruteo de hormigas

La mejor solución global deposita feromonas en cada iteración junto con todas las otras hormigas.

Max-Min Sistema de Hormigas (MMAS)

Agregada la cantidad máxima y mínima de feromonas [tmax,tmin] Solamente la mejor iteración deposita feromonas. Todas las aristas son inicializadas con tmax y re-inicializadas con tmax cuando se acerca a un estancamiento.

Sistema de Hormigas Basado en Ranking (ASrank)

Todas las soluciones se clasifican de acuerdo su longitud. La cantidad de feromonas depositadas es ponderada para cada solución, de tal manera que las soluciones con los caminos más cortos depositan más feromonas que las soluciones que con los caminos más largos.

Colonia de Hormigas Ortogonal Continua (COAC)

El mecanismo de depósito de feromonas de COAC es permitir a las hormigas la búsqueda de soluciones en conjunto y efectiva. Usando un método de diseño ortogonal, las hormigas en un dominio factible pueden explorar las regiones elegidas de una manera rápida y eficiente, con mayor capacidad de búsqueda global y precisión.

Optimización de Colonia de Hormigas con Lógica Difusa

Este método introduce inteligencia difusa dentro de las hormigas para acelerar las habilidades de búsqueda.

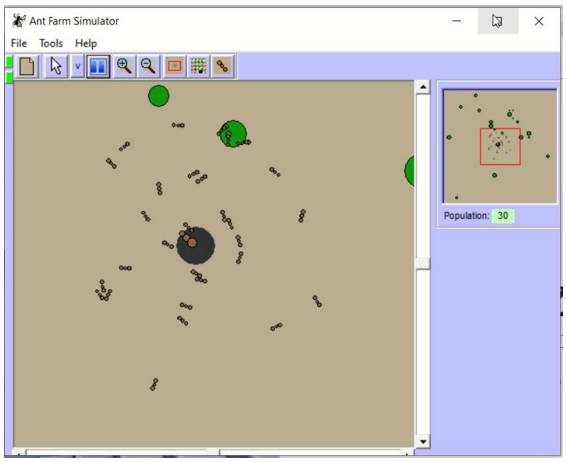


Figura 4.8: Simulador colonia de hormigas

Unidad 5

OPTIMIZACIÓN CON VARIOS OBJETIVOS

"Reconocer problemas de optimización con varios objetivos."

Objetivo específico 1

5.1. Introducción

En la vida real, existen numerosas situaciones y problemas que son reconocidos como **problemas multiobjetivo**, es decir, no poseen un único criterio medible por el cual pueda declararse que una solución sea completamente satisfactoria.
Dicho de otra forma, este tipo de problemas contiene múltiples criterios que han de satisfacerse o que han de ser tenidos en cuenta.
A menudo dichos criterios entran en conflicto unos con otros y no existe una única solución que simultáneamente satisfaga a todos.
Por tanto, la solución que se pretenda obtener debe estar en concordancia con las preferencias del decisor.
En problemas de optimización multiobjetivo con objetivos contradictorios no siempre existe una solución única que pueda ser considerada como la mejor, sino un **conjunto de soluciones** que representan los mejores compromisos entre los distintos criterios.
Dicho conjunto es llamado conjunto Pareto-optimo y su imagen en el espacio objetivo es denominado **Frente de Pareto**.

Mientras que en optimización monobjetivo se busca un vector de decisión n-dimensional que optimice una función escalar, en optimización multiobjetivo se intenta encontrar uno que optimice una **función vectorial** cuyos elementos representan las distintas funciones objetivo.
Un problema de optimización multiobjetivo puede ser definido formalmente como:

$$\mathbf{y} = \mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_k(\mathbf{x})) \quad (1)$$

sujeto a

$$\mathbf{g}(\mathbf{x}) = (g_1(\mathbf{x}), \ldots, g_m(\mathbf{x})) \leq \mathbf{0} \quad (2)$$

donde

$$\mathbf{x} = (x_1, \ldots, x_n) \in \mathcal{X} \subseteq \mathbb{R}^n \; ; \; \mathbf{y} = (y_1, \ldots, y_k) \in \mathcal{Y} \subseteq \mathbb{R}^k$$

x es una variable de decisión vectorial n-dimensional, y **y** es un vector objetivo k-dimensional,
$X \subseteq R^n$ denota el espacio de decisión, e $Y \subseteq R^k$ denota el espacio objetivo.
El conjunto de restricciones dadas por la Ecuación 2 define la región de factibilidad $Xf \subseteq R^n$ y cualquier punto $x \in Xf$ es una solución factible.
En un MOP con objetivos contradictorios, el espacio de búsqueda se encuentra sólo parcialmente ordenado y dos soluciones pueden ser indiferentes entre sí, siendo poco usual que una única variable de decisión optimice de manera simultánea todos los objetivos.
Para la mayoría de los MOPs, el conocimiento del Frente Pareto optimo ayuda al tomador de decisiones a seleccionar aquella solución que representa el mejor compromiso.
Generar dicho frente puede ser computacionalmente costoso o incluso imposible, en especial en problemas reales complejos (ingeniería, diseño, toma de decisiones).
Entonces, lo único que se puede pretender es obtener una buena aproximación al frente Pareto óptimo verdadero.
¿Qué se necesita para resolver este tipo de problemas?

- Un método de búsqueda basado en los múltiples objetivos
- Una política de equilibrio entre los objetivos
- Un orden para este proceso de optimización

En la actualidad, hay unas 30 técnicas clásicas de programación matemática para resolver problemas de optimización multiobjetivo (MO).
Sin embargo, estas técnicas suelen generar los elementos del conjunto de Pareto de uno en uno, requiriendo múltiples ejecuciones para obtener una aproximación.
Además, muchas de ellas son muy sensibles a la forma del frente de Pareto (p.e., no funcionan cuando el frente es cóncavo o está desconectado).

Por estas razones, la mayoría de los enfoques MO existentes están basados en metaheurísticas (MHs), en particular en algoritmos evolutivos (AEs) (70 %).
La mayor parte de ellos (un 90 %) aplican el segundo enfoque, tratando de obtener una buena aproximación del frente de Pareto.
Los AEs son muy buenos optimizadores MO debido a que:

- Son algoritmos poblaciones que permiten obtener múltiples soluciones en una única ejecución
- Se adaptan a buscar en distintas zonas del espacio simultáneamente
- No son sensibles a la forma del frente de Pareto

Definición

Los problemas de optimización multiobjetivo (POM) pueden formularse de la siguiente manera:

$$POM = \begin{Bmatrix} \min F(x) = (f_1(x), f_2(x), \dots, f_n(x)) \\ s.a. \quad x \in X \end{Bmatrix}$$

Donde:

- n ≥ 2 es el número de funciones objetivo.
- x = ($x_1, x_2, \dots, x_r$) es el vector de variables de decisión.
- X es el espacio de soluciones factibles. X es definido habitualmente como el conjunto de restricciones $\{x | g_j(x) \leq 0,\ j = 1,2, \dots m\ y\ h_i(x) = 0,\ i = 1,2, \dots e\}$.
- $F(x)$ es el vector objetivo.
- y = ($y_1, y_2, \dots, y_n$), con $y_i = f_i(x)$, es el valor de una solución en el espacio de soluciones factibles.

ALGORITMOS EVOLUTIVOS

El origen de los algoritmos evolutivos se encuentra en el término algoritmo genético, propuesto en 1975 por Holland en su libro titulado *Adaptation in Natural and Artificial Systems*, Holland (1975), y que fue la base para la creación y desarrollo de lo que actualmente es un activo campo de investigación y con un enfoque sobre aplicaciones mucho más amplio que el que tuvo en su planteamiento inicial.
Muchos investigadores utilizan ahora los términos computación evolutiva o algoritmos evolutivos con el objeto de cubrir los amplios desarrollos habidos en los últimos años.
Sin embargo, en el contexto de las metaheurísticas, es probablemente cierta la afirmación de que los algoritmos genéticos en su forma original abarcarían la mayoría de los conceptos básicos que se utilizan en la actualidad en los algoritmos evolutivos.

El esquema común a estos desarrollos fue la utilización de los conceptos de mutación y selección, fundamentales en la teoría de la evoluci6n neodarwiniana.
Aunque se obtuvieron resultados que parecían prometedores, la computación evolutiva no tomó fuerza hasta mediados de los años ochenta, siendo la escasa potencia de las computadoras de esa época una de las razones. Un aspecto clave para la promulgación de esta metodología fue el libro de Goldberg (1989), en el que ya se planteó su desarrollo formal tanto a nivel teórico como practico.
De forma genérica, apuntemos que los algoritmos evolutivos se apoyan en la denominada población de individuos o soluciones, formada por un subconjunto de la región factible o de la población global.
La idea subyacente es que, manteniendo una población de individuos que irá evolucionando en el tiempo, estos métodos pueden ayudar a encontrar muchas soluciones eficientes a través de unos mecanismos de autoadaptación y cooperación.
La autoadaptación significa que los individuos evolucionan independientemente, mientras que la cooperación implica un intercambio de información entre los individuos.
Con ello, la poblacion global contribuye al proceso de evolución hacia el conjunto eficiente.
Como en la naturaleza, los operadores evolutivos actúan a través del algoritmo evolutivo sobre la población tratando de generar soluciones lo más idóneas posibles.

En la aplicación de un algoritmo genético se parte de un conjunto (población) de n soluciones (individuos) tomadas de la población global, que va cambiando de acuerdo con los operadores evolutivos y manteniéndose su tamaño en las siguientes iteraciones.
Tales operadores evolutivos son: selección de buenos individuos, cruce o recombinación, mutación y selección de malos individuos.
El significado de estas operaciones, que conducen a una nueva generación, es el siguiente:
Selección de buenos individuos. Se escogen con reemplazamiento n individuos de la población, con mayor probabilidad de elección para los individuos de mejor valor.
Cruce o recombinación de individuos. Dos individuos se parten aleatoriamente e intercambian sus partes, es decir, se tiene una forma de recombinación que opera sobre dos individuos (ascendientes) para obtener los nuevos individuos (descendientes), tal como se muestra en forma esquemática en la siguiente figura.

5.2. Conceptos y Problemas con varios objetivos

UNa función $\phi(x)$ es llamada **convexa** sobre el dominio de **R** si para dos vectores cualquiera $\vec{x}_1$ y $\vec{x}_2 \in R$.

$$\phi(\theta\vec{x}_1 + (1-\theta)\vec{x}_2 \leq \theta\phi(\vec{x}_1) + (1-\theta)\phi(\vec{x}_2)$$

donde θ es un escalar en el rango $0 \leq \theta \leq 1$.

La función $\phi(\vec{x})$ es estrictamente convexa si, para $\vec{x}_1 \neq \vec{x}_2$ el signo $\leq$ de la ecuación anterior puede ser reemplazada por la desigualdad ($<$).

Una función convexa no puede tener ningún valor mayor que los valores de la función obtenidos mediante interpolación lineal entre $\phi(\vec{x}_1)$ y $\phi(\vec{x}_2)$.

Un conjunto (o región) de puntos se define como un **conjunto convexo** en un espacio n-dimensional si, para todos los pares de puntos $\vec{x}_1$ y $\vec{x}_2$ en el conjunto, el segmento rectilíneo que los une ésta también enteramente dentro del conjunto.

De tal forma todo punto $\vec{x}$ donde:

$$\vec{x} = \theta\vec{x}_1 + (1-\theta)\vec{x}_2, 0 \leq \theta \leq 1$$

está también en el conjunto.

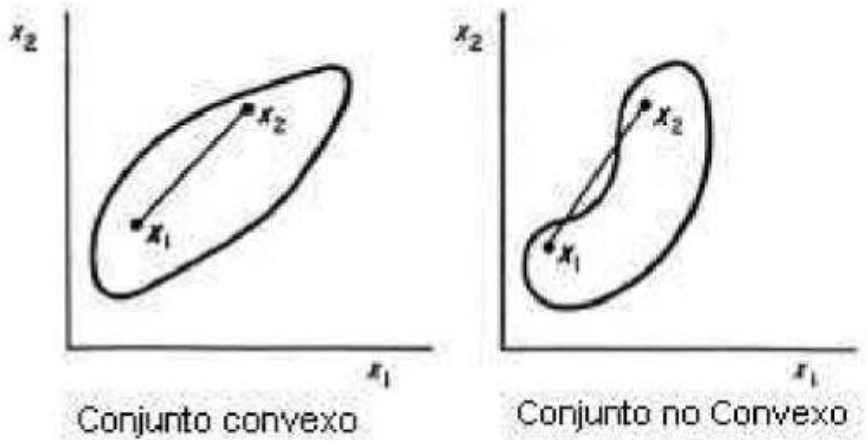


Figura 5.1: Conjunto de Pareto

Si la desigualdad reversa de la ecuación anterior se cumple, la función es **cóncava**.

De tal forma una función $\phi(\vec{x})$ es **cóncava** si $-\phi(\vec{x})$ es convexa.

Las funciones lineales son convexas y cóncavas.

Vector Objetivo Ideal

Vector de variables de decisión correspondientes a los óptimos (factibles) considerando a cada una de las funciones objetivo del problema como aisladas.

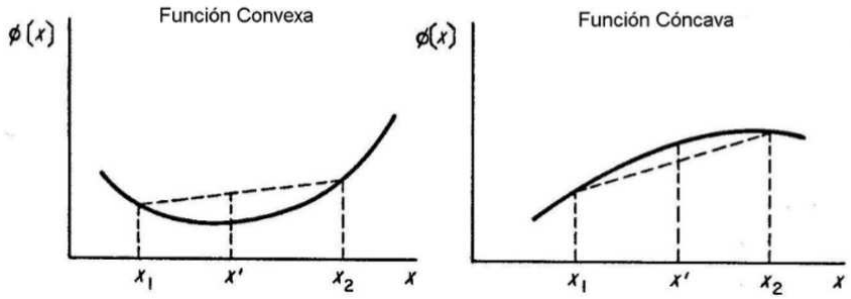


Figura 5.2: Funciones lineales

Nótese que el vector ideal es inalcanzable, excepto en el caso en que no existe ningún conflicto entre las funciones objetivo del problema.

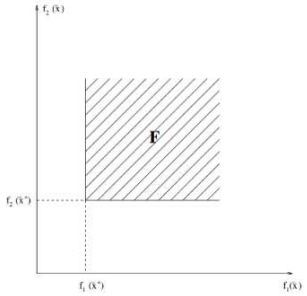


Figura 5.3: Vector ideal

El vector ideal es utilizado por algunos métodos de programación matemática que normalmente tratan de minimizar la distancia (en el espacio de las funciones objetivo) entre una solución dada y el vector ideal.
Determinar el vector objetivo ideal suele ser fácil excepto cuando las funciones objetivo (vistas por separado) presentan multimodalidad.

Vector Objetivo Utópico
El concepto de vector objetivo útopico se suele definir como $z** \in R^k$ el cual es un vector objetivo infactible cuyos componentes se forman mediante:

$$z** = z_i^* - \epsilon_i$$

para toda $i = 1, 2, 3, ..., k$, donde z_i^* es un componente del vector ideal y $\epsilon_i > 0$ esun escalar relativamente pequeño pero computacionalmente significativo.

Vector Objetivo de Nadir

Se denomina así a los límites superiores del conjunto de óptimos de Pareto.
Se suele denotar alvector objetivo de Nadir como z^{nad} y sus componentes son mucho más difíciles de obtener.
Una **tabla de resultados** (*pay of table*) se forma usando los vectores de decisión obtenidos cuando se calcula elvector objetivo ideal.
La fila i de la tabla de resultados muestra los valores de todas las funciones objetivo calculadas en el punto donde la f_i obtuvo su valor mínimo.
Por tanto z_i^* (o sea la componente i del vector objetivo ideal) se encuentra en la diagonal principal de la tabla el valor máximo de la columna i en la tabla de resultados.
Puede seleccionarse como un estimado de la cuota superior del objetivo f_i para $i = 1, ..., k$ sobre el conjunto de óptimos de Pareto.
En la siguiente figura se muestra en negro los vectores objetivos ideales y en gris los vectores objetivo de nadir nótese que los vectores objetivo de nadie pueden ser o no factibles.

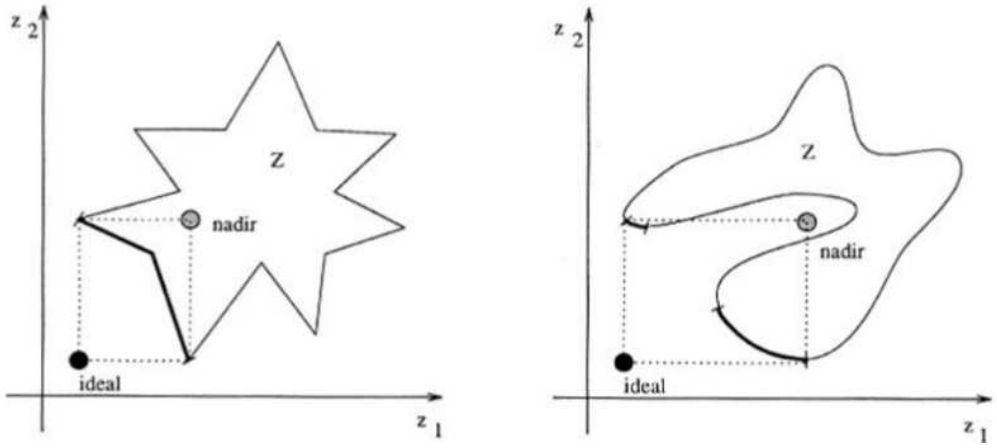


Figura 5.4: Vectores objetivo ideales

La siguiente figura muestra el vector objetivo ideal el utópico y el de nadir ($znadir$).

Tipos de problemas Multiobjetivo

Hay 3 tipos de situaciones que pueden presentarse en un problema multiobjetivo:

- minimizar todas las funciones objetivo
- maximizar todas las funciones objetivo
- minimizar algunas funciones y maximizar otras

Por cuestiones de simplicidad normalmente todas las funciones se convierten ya sea un problema de maximización o a uno de minimización por ejemplo se puede

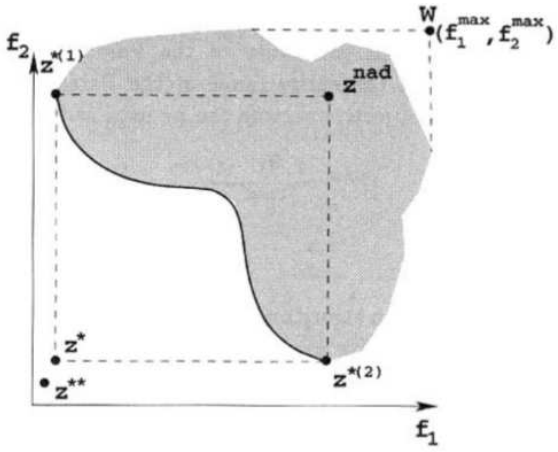


Figura 5.5: Vectores objetivo ideales

usar la siguiente identidad para convertir todas las funciones a maximizarse de manera que correspondan a un problema de minimización :

$$max f_i(\vec{x}) = min(-f_i(\vec{x}))$$

¿Cuál es la noción de óptimo en optimización multiobjetivo?
Al tener varias funciones objetivos la noción de óptimo cambia porque en los POMs realmente se esta tratando de obtener buenos compromisos (llamados *trade-offs* en inglés) en vez de una solución única como en optimización global.
La noción de óptimo que suele adoptarse más comúnmente es aquella propuesta originalmente por Edgeworth en 1881.
Esta lo esta noción fue luego generalizada por Vilfredo Pareto en 1896 aunque algunos autores le llaman a esta nación óptimo de Edgeworth-Pareto el término más común aceptado es el de óptimo de Pareto.

Definicion de Optimalidad de Pareto
Se dice que un vector de variables de decisión $\vec{x} \in \Omega$ (Ω es la zona factible) es un **óptimo de Pareto** si no existe otra $\vec{x} \in \Omega$ tal que $f_i(\vec{x}) \leq f_i(x^*)$ para toda $i = 1, ..., k$ y $f_j(\vec{x}) \leq f_i(j^*)$ para al menos j.
En palabras esta definición dice que $\vec{x} \in \Omega$ es un óptimo de Pareto si no existe ningún vector factible de variables de decisión que decremente algún criterio sin causar un incremento simultáneo en al menos un criterio.
Desafortunadamente este concepto casi siempre produce no una solución única sino un conjunto de ellas a las que se les llama **conjunto de óptimos de Pareto**.

Los vectores $\vec{x}*$ correspondientes a las soluciones incluidas en los conjuntos de óptimo de pareto son llamados **no dominados**.
La gráfica de las funciones objetivo cuyos vectores no dominados se encuentran en el conjunto de óptimos de Pareto se denominan **frente de Pareto**.
Un vector $\vec{u} = (u_1, ..., u_k)$ **domina** a otro $\vec{v} = (v_1, ..., v_k)$ (denotado mediante $u \preceq v$) sí solo sí u es parcialmente menor que v es decir $\forall i \in \{1, ..., k\}$, $u_i \leq v_i \wedge \exists i \in \{1, ..., k\}$: $u_i < v_i$.

Conjunto de Óptimos de Pareto
Para un problema multiobjetivo dado $\vec{f}(x)$ el conjunto de óptimos de Pareto ($\mathcal{P}*$) se define como:

$$\mathcal{P}^* := \{x \in \Omega | \neg \exists x' \in Omega \vec{f}(x') \preceq \vec{f}(x)\}$$

Decisión basada en preferencias
Al ser todas las soluciones Pareto optimas igualmente deseables desde el punto de vista matemático, puede ser útil considerar las preferencias concretas de un decisor, para ello existen múltiples mecanismos.
Uno de los más frecuentes, tanto en **MCDM** (*Multiple Criteria Decision Making*) como en *preference-based* **EMO** (*Evolutionary Multi-objective Optimization*), consiste en especificar el valor deseable para cada función objetivo, determinando un **punto de referencia**.
Este se define como $q = (q_1, ..., q_k)^T$, siendo q_i un valor de aspiración para la función objetivo f_i , para todo $i = 1, ..., k$.
Un punto de referencia se dice que es **alcanzable** si existe alguna solución factible que lo iguale o mejore simultáneamente en todos los objetivos.
Una forma de medir la calidad de la aproximación de la región de interés teniendo en cuenta el punto de referencia se basa en el concepto de **hipervolumen**.
Para un conjunto de soluciones P, su hipervolumen, denotado por HV (P), es el volumen (en el espacio de objetivos) cubierto por las soluciones de P.
Formalmente, para cada solución y ? P, se construye un hipercubo vy acotado por un punto de referencia R y la propia solución y.
El punto de referencia R puede fijarse al vector nadir.?
La unión de todos los hipercubos de las soluciones de P determina el valor de HV (P), tal y como indica la expresión siguiente:

5.3. MOGA (Multiobjective Optimization Genetic Algorithms)

EL objetivo de los problemas de optimización multiobjetivo. es encontrar todas las posibles compensaciones entre múltiples funciones objetivas que suelen estar en conflicto.
Dado que es difícil elegir una solución única para un problema de optimización multiobjetivo sin una interacción iterativa con quien toma las decisiones, un enfoque general es mostrar el conjunto de soluciones óptimas de Pareto a quien toma las decisiones.
Luego se puede elegir una de las soluciones óptimas de Pareto según la preferencia.
Para conocer todas las soluciones óptimas de Pareto mediante algoritmos genéticos, se debe mantener la variedad de individuos en cada generación.
El algoritmo genético de objetivos múltiples (MOGA) es una de las muchas técnicas de optimización de ingeniería, un método de búsqueda aleatoria guiada.
Es adecuado para resolver problemas relacionados con la optimización de objetivos múltiples con la capacidad de explorar las diversas regiones del espacio de solución.
Por lo tanto, es posible buscar un conjunto diverso de soluciones con más variables que se pueden optimizar al mismo tiempo.
Un conjunto óptimo de Pareto es un conjunto de soluciones que son fronteras de soluciones no dominadas.
Con el óptimo de Pareto establecido, los valores de la función objetivo correspondiente en el espacio objetivo se denominan frente de Pareto.
Los métodos convencionales para resolver problemas multiobjetivo consisten en búsquedas aleatorias, programación dinámica y métodos de gradiente, mientras que los métodos heurísticos modernos incluyen paradigmas cognitivos como AGs, recocido simulado y enfoques lagrangianos.
Las soluciones de MOGA se ilustran utilizando los **frentes de Pareto**.
Teniendo en cuenta que el AG es uno de los algoritmos evolutivos más importantes.
Algunos de estos métodos se gestionan para encontrar la solución óptima, pero tienden a tardar más en converger, por lo que necesitan mucho tiempo de cálculo.
Por lo tanto, mediante la implementación del enfoque MOGA que se basa en el principio de evoluación biológica natural se utilizará para abordar este tipo de problemas.

Estructura general del algoritmo genético

(1) Inicialización
  (1.1) Generar la población inicial $.P(0)$
  (1.2) Establezca la tasa de cruce, la tasa de mutación y el tiempo máximo de generación.
  (1.3) Deja $.t \leftarrow 0$

(2) Dado que no se alcanza el tiempo máximo de generación, haga lo siguiente.
  (2.1) Operador cruzado: generar $.O_1(t)$
  (2.2) Operador de mutación: generar $.O_2(t)$
  (2.3) Evaluar y calcular la función de aptitud.$O_1(t)O_2(t)$
  (2.4) Seleccionar operador: construir la siguiente población.
  (2.5) $t \leftarrow t + 1$, ve a .(2.1)
  Fin

Fin.

Del pseudocódigo podemos ver que hay tres operadores importantes en un AG general: los operadores de cruce, mutación y selección.
La implementación de estos operadores depende en gran medida de la forma de codificación.
El algoritmo genético multiobjetivo utiliza una métrica de distancia de aglomeración para crear una distribución homogénea de los puntos no dominados en el frente de Pareto.
El algoritmo genético multiobjetivo finaliza si se cumple una de las siguientes condiciones:
Se cumple el criterio de convergencia, esto ocurre cuando se ejecuta el número mínimo de iteraciones permitidas (Iteraciones mínimas), se encuentran diseños factibles (Tol. de violación de restricciones (%)) y los diseños no dominados no cambiaron en la última iteración.
Se alcanza el número máximo de iteraciones permitidas (Iteraciones máximas).
Un análisis falla y la opción Terminar optimización es la predeterminada (en caso de evaluación fallida).
*Admite restricciones de variables de entrada.
*Aunque el número de evaluaciones por iteración es una combinación de múltiples configuraciones, se ve afectado principalmente por la configuración de Tamaño de la población. Todas las evaluaciones dentro de una iteración se pueden ejecutar en paralelo.
Si se requiere computación paralela, se recomienda utilizar el método Metamodelo o No híbrido.

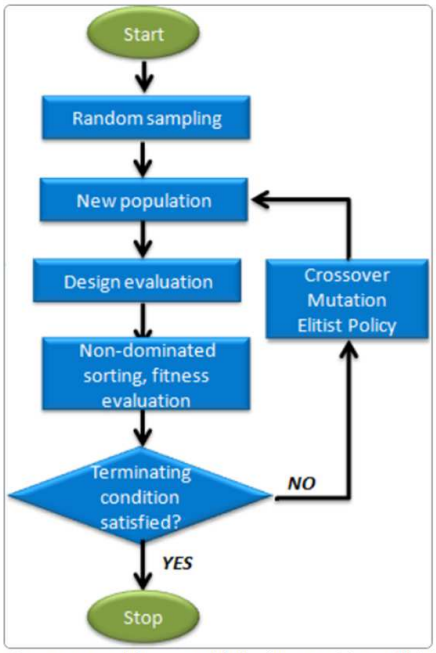


Figura 1. Fases del proceso del algoritmo genético multiobjetivo

Bibliografía

[Diccionario Larousse, 2003] *"Diccionario Larousse, Edición Premium*", EDICIONES LAROUSSE MÉXICO y SPS EDITORIAL BARCELONA, (2003).

[Cambridge Dictionary, 2022] *"Cambridge Dictionary on-line*", `https://dictionary.cambridge.org/`, (2022).

[DRALE] *"Diccionario de la Real Academia de la Lengua Española*", versión on-line (2017).