PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

Apuntes Investigación de Operaciones

GONZALO TELLO VALENZUELA

Docente: Ricardo Soto
Docente: Leslie Perez

2do Semestre, 2019

Índice general

Lista de Figuras

Lista de Tablas

Capítulo 1

Programación Lineal

1.1. Introducción al método gráfico

La programación lineal es una técnica de optimización proveniente del área de la programación matemática. Esta técnica permite maximizar o minimizar una función objetivo sujeta a un conjunto de restricciones. Si bien las bases que sustentan a la programación lineal son previas al siglo XX, es durante la segunda guerra mundial cuando la programación lineal se establece formalemente, principalmente para la reducción de costos militares. Luego de la guerra, la programación lineal ha sido ampliamente utilizada en la industria para resolver distintos problemas de optimización. Esta guía se centra en el método gráfico, la cual es una técnica básica de la programación lineal y de la optimización en general, para resolver problemas que contienen 2 variables.

Ejemplo 1 - Paso a Paso.
Considere una fábrica de muebles que recibe partes pre-fabricadas para la elaboración de sillas. Existen 2 modelos de sillas que pueden fabricarse con estas partes: silla 'Valparaíso' y silla 'Olmué'. La fabrica puede destinar 10 horas semanales de ensamble y 8 horas de barniz para elaborar la mayor cantidad de sillas posibles. Considere además que la silla 'Valparaíso' requiere 2 horas de ensamble y 1 de barniz; la silla 'Olmué' requiere 1 hora de ensamble y 2 de barniz. Si la utilidad por cada silla 'Valparaíso' es de $10.000 y la utilidad por cada silla 'Olmué' es de $8.000. ¿Cuántas sillas de cada tipo se deben fabricar semanalmente para maximizar las utilidades de la empresa?.

1.1.1. Definición del modelo matemático

El primer para para resolver el problema corresponde a la definición del modelo de programación lineal o también conocido como modelo matemático en el campo de la optimización. El modelo matemático está compuesto por variables de decisión, restricciones y una función objetivo. Las variables de decisión corresponden a la incognitas para las cuales debemos encontrar una solución, las restricciones limitan los valores que estas variables pueden adoptar y la función objetivo es la expresión matemática que será minimizada o maximizada según el problema. Estos elementos deben ser identificados desde el enunciado del problema.

- Variables de decisión.

  Las variables del problema también conocidas como variables de decisión corresponden a las incógnitas para las cuales se debe encontrar un valor. Este problema plantea dos incógnitas, la cantidad de sillas 'Valparaíso' y la cantidad de sillas 'Olmue' que deben ser fabricadas para maximizar las utilidades, por ende se identifican 2 variables de decisión:

  ```
  x: cantidad de sillas Valparaíso
  y: cantidad de sillas Olmué
  ```

- Restricciones.

  Las restricciones limitan los valores que estas variables pueden adoptar. En este problema se identifican 2 restricciones que limitarán la producción de sillas. Una restricción correspondiente a los recursos de ensamble y la otra a los recursos de barniz que dispone la empresa. Por ejemplo, se disponen 10 horas de ensamble y la silla Valparaíso requiere 2 horas y la silla Olmué sólo de 1 hora, esto se puede representar de la siguiente forma:

  $$R1 : 2x + y \leq 10$$

  donde x se multiplica por 2 dado que la silla Valparaíso requiere 2 horas de ensamble. La suma de horas requeridas por ambas sillas debe ser menor o igual a 10. La segunda restricción se construye de manera análoga, en este caso es la silla Olmué que requiere de 2 horas de barniz, por ende se multiplica por 2.

$$R2 : x + 2y \leq 8$$

- Función objetivo.

  La función objetivo es la expresión matemática que será minimizada o maximizada según el problema. En este problema se deben maximizar las utilidades de la empresa, de las cuales se obtienen de las utilidades obtenidas por cada silla. Asumiendo que se venden todas las sillas, la utilidad se obtiene con la siguiente expresión.

  $$MaximizarZ = 10000x + 8000y$$

  Donde 10000 corresponde a la utilidad obtenida por la silla Valparaíso y 8000 por la silla Olmué. Finalmente, el modelo matemático resultante es el siguiente:

  Variables
  x, y

  Restricciones
  $R1 : 2x + y \leq 10$
  $R2 : x + 2y \leq 8$

  Función objetivo
  $MaximizarZ = 10000x + 8000y$

1.1.2. Resolución del modelo matemático con el método gráfico

El segundo paso corresponde a la resolución del problema, como es mencionado anteriormente se utilizará el método gráfico, la cual es una técnica básica de la programación lineal y de la optimización en general, para resolver que contienen 2 variables.

- Graficar restricciones.

  La primera tarea para resolver el problema corresponde a graficar las restricciones. Esto permitirá visualizar los valores que satisfacen las restricciones. La figura 1.1 muestra las dos restricciones del problema en el gráfico. El área de bajo la recta corresponde a los valores que satisfacen la restricción, por ende los valores que satisfacen el problema se encuentran en la interesección de ambas áreas.

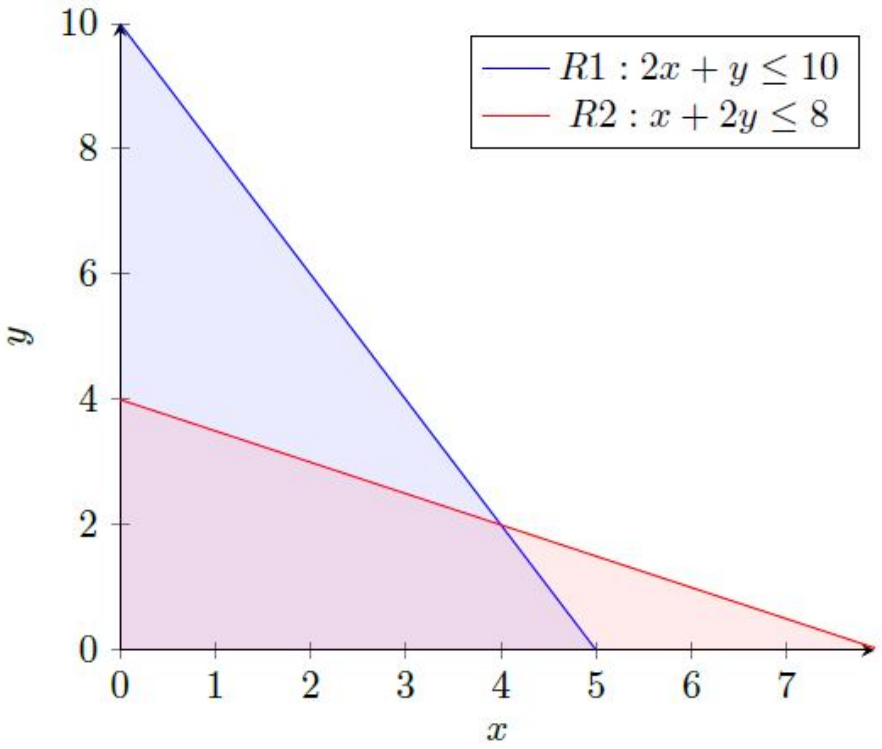


Figura 1.1: Restricciones

NOTA: Para graficar una recta basta con encontrar 2 puntos que estén contenidos y trazar una línea recta entre ellos. Por ejemplo, para la recta $2x + y = 10$ identificamos los puntos (0,10) y (5,0).

- Encontrar la solución óptima.

  La solución óptima corresponde a la que maximiza o minimiza la función objetivo dependiendo del problema en cuestión. El presente problema es de maximización por ende se debe buscar la solución que maximiza el valor de la función objetivo. La solución óptima debe satisfacer además las restricciones del problema, por lo cual debe buscarse en el área generada por la intersección de ambas restricciones. Las posibles soluciones óptimas se encuentran en los vértices de esta área como se muestra en la figura 1.2. Los vértices corresponden a los puntos: (0,0), (0,4), (4,2), (5,0). Para identificar la solución óptima es necesario evaluar estos puntos en la función objetivo, donde el punto (4,2)

es el que maximiza la función objetivo como se visualiza a continuación.

punto(0,0):$10000 * 0 + 8000 * 0 = 0$
punto(0,4):$10000 * 0 + 8000 * 4 = 32000$
punto(4,2):$10000 * 4 + 8000 * 2 = 56000$
punto(5,0):$10000 * 5 + 8000 * 0 = 500000$

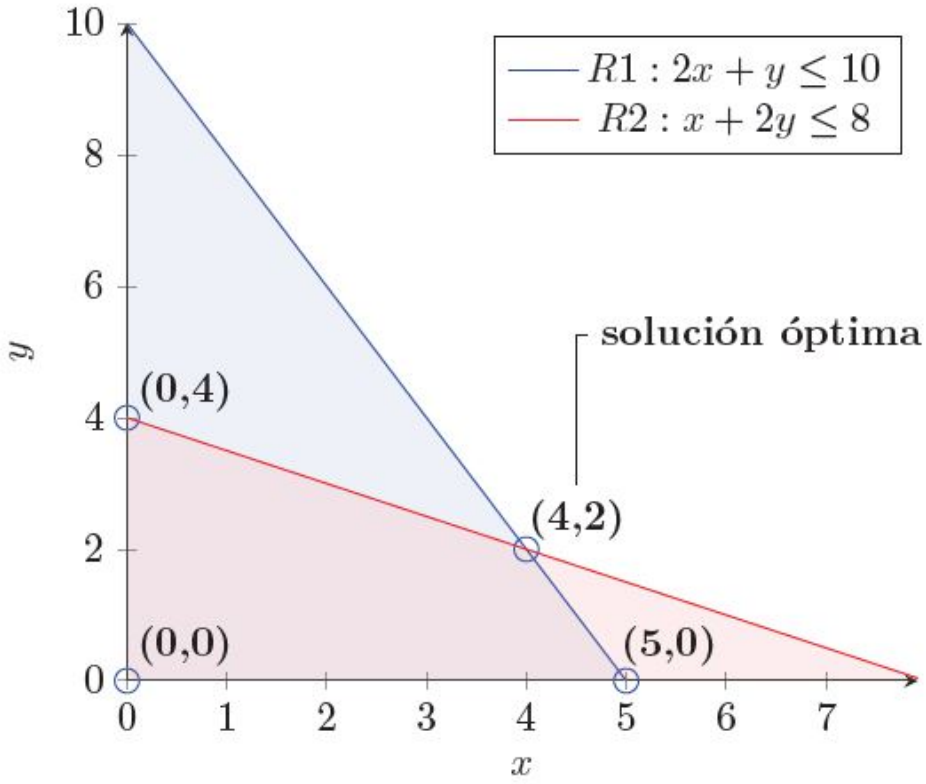


Figura 1.2: Encontrando la solución óptima

- **Indentificación del vértice**.

  Para encontrar el punto correspondiente al vértice generado por la intersección de 2 rectas basta formar un sistema de ecuaciones con las rectas involucradas y resolverlo.

$$2x + y = 10 \Longleftrightarrow y = 10 - 2x$$

$x + 2y = 8$
$x + 2(10 - 2x) = 8$
$x + 20 - 4x = 8$
$x + 2y = 8$
$-3x = -12$
$x = 4$

$2(4) + y = 10$
$y = 2$

1.1.3. Alternativa para identificar el óptimo

Existe una forma alternativa para encontrar el óptimo y consiste en graficar la función objetivo (línea verde discontinua en figura 1.3) incrementando el valor de z, dado que el problema es de maximización. La solución óptima corresponderá al vértice en el espacio de soluciones más allá del cual cualquier incremento adicional de z producirá una solución infactible.

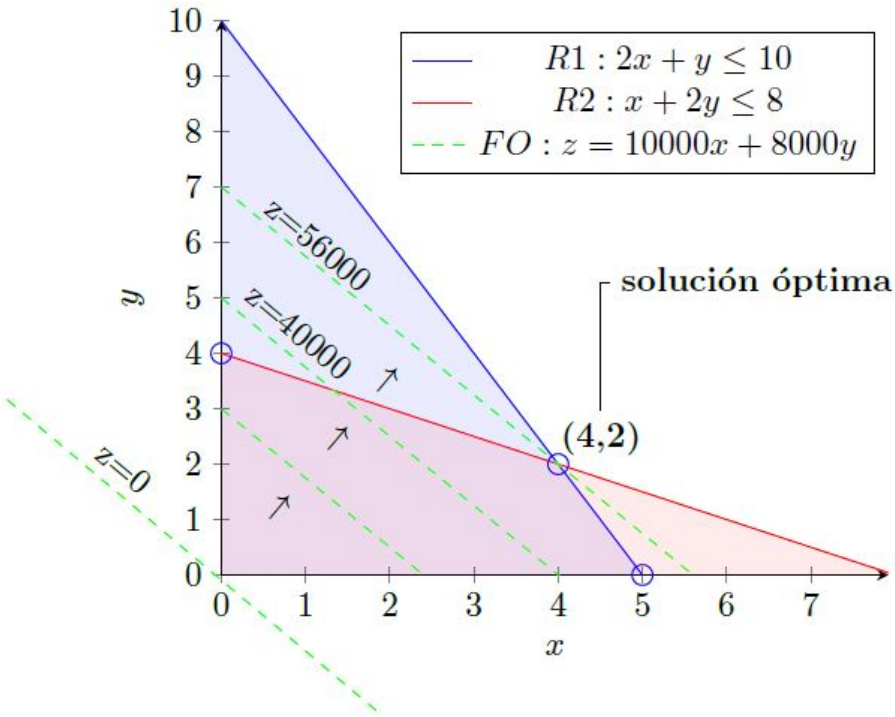


Figura 1.3: Alternativa para identificar el punto óptimo.

1.2. Solver Excel

- **Ejemplo**.

  Considere una empresa electrónica que desea fabricar 2 tipos de computadores. Cada uno debe pasar por 3 fases: ensamblaje, control de calidad y embalaje. Los computadores A requieren de 20 minutos de ensamblaje, 30 de control de calidad y 10 de embalaje. Los computadores B requieren de 30 minutos de ensamblaje, 50 de control de calidad y 15 de embalaje. La empresa dispone de 300 minutos diarios para ensamblaje, 800 para control de calidad y 80 para embalaje. Si la utilidad por computador A es de $80.000 y la utilidad por computador B es $90.000:

  1. Defina un modelo que determine cuántos computadores A y B deben producirse diariamente para maximizar las utilidades de la

empresa. Especifique claramente variables, restricciones y función objetivo del modelo.

2. Resuelva el modelo mediante programación lineal utilizando el método gráfico.

- **Solución**

Modelo matemático

variables:
x : *cantidad computadores A*
y : *cantidad computadores B*

restricciones:
$R1 : 20x + 30y \leq 300$
$R2 : 30x + 50y \leq 800$
$R2 : 10x + 15y \leq 80$

función objetivo:
$FO : Max\ z = 80000x + 90000y$

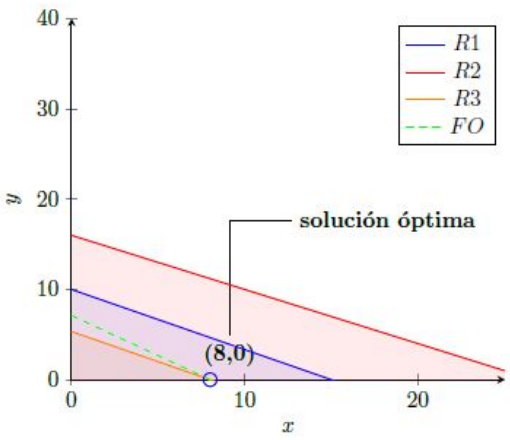
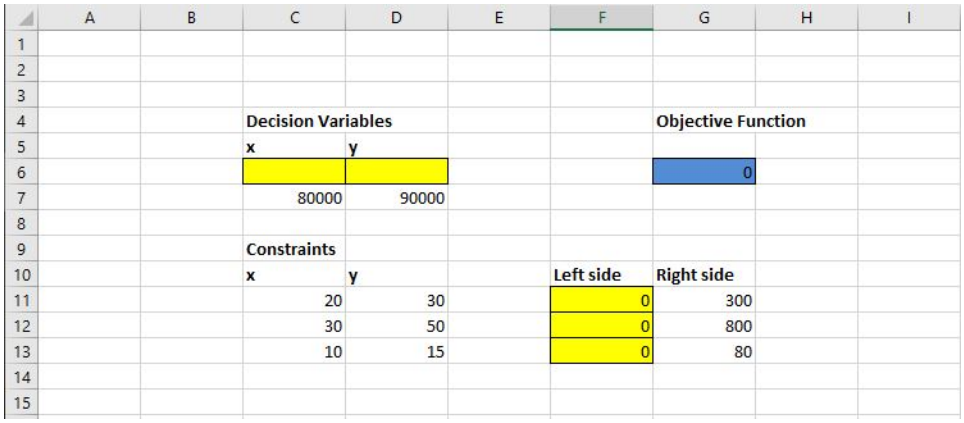


Figura 1.4: Modelo matemático ejemplo solver excel

- **Observación:**

  La solución óptima es el punto (8,0), no obstante se podrían distribuir mejor los recursos para aumentar los ingresos. Por ejemplo, si el negocio lo permite se podrían disminuir las horas ociosas destinadas a ensamblaje y control de calidad para destinarlas a embalaje.

Esta sección se centra en la utilización del solver MS EXCEL para resolver problemas de optimización utilizando programación lineal.



	A	B	C	D	E	F	G	H	I
1									
2									
3									
4			**Decision Variables**				**Objective Function**		
5			**x**	**y**					
6							0		
7			80000	90000					
8									
9			**Constraints**						
10			**x**	**y**		**Left side**	**Right side**		
11			20	30		0	300		
12			30	50		0	800		
13			10	15		0	80		
14									
15									

Figura 1.5: Ejemplo de Solver en excel

El ejemplo anterior contiene 3 secciones:

- Decision variables
- Objective function
- Constrains

En la zona **Decision variables** se establecen 2 valores: 80.000 y 90.000, los cuales corresponden a los factores que multiplican a las variables x e y en las función objetivo ($80{,}000x + 90{,}000y$). En esta misma zona, la solución para las variables x e y se arrojará en las celdas amarillas C6 y D6, una vez se haya ejecutado el solver. En la zona **Objetive function**, el solver entregará el resultado de la evaluación de la solución en la función objetivo, especificamente en la celda azul G6. Finalmente en la zona **Constraints**, se define las restricciones, por ejemplo la línea 11 representa la restricción $20x + 30y \leq 300$. En **Left side** el solver entregará el valor resultante de la evaluación de la solución en la restricción.

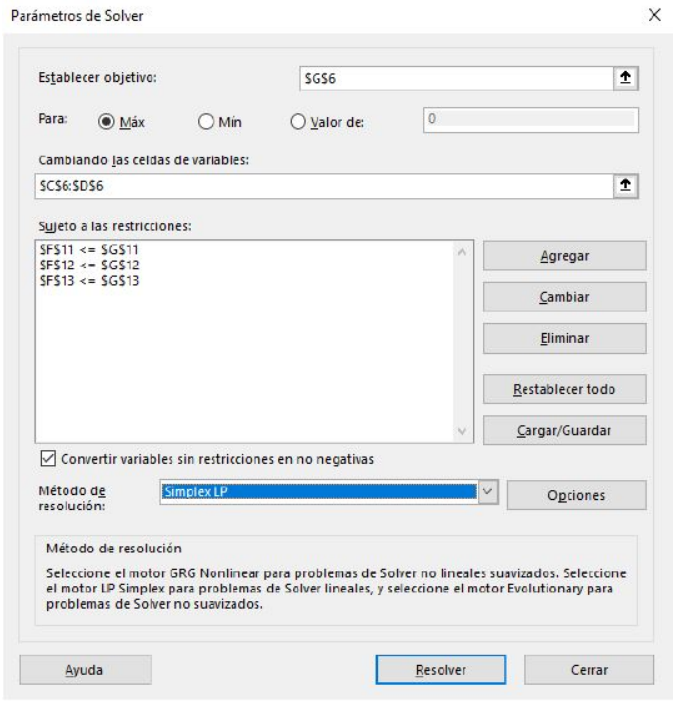


Figura 1.6: Parámetros del solver

Los operadores relaciones de las restricciones se definen en los parámetros del solver como se visualiza en la figura 1.6. La instrucción $F11 \leq $G11 establece el operador menor igual para la restricción definida en la linea 11. En los parámetros del solver también se define si el problema es de maximización o minimización, se establece además la celda objetivo ($G6) y las celdas de las variables($C6:$D6).

Capítulo 2

Programación y satisfacción de restricciones

2.1. Introducción

La satisfacción de restricciones es una técnica de búsqueda que se origina en los años 70 en el área de la inteligencia artificial. Esta técnica se enfoca en la resolución de problemas de satisfacción de restricciones conocidos comúnmente en el ámbito de la investigación de operaciones e inteligencia artificial como CSP (Constraint Satisfaction Problem).

Un CSP, está compuesto por variables, dominio y restricciones. Las variables son las incógnitass para las cuales se debe encontrar una solución, los dominios contienen los valores que pueden ser asignados a las variables y el conjunto de restricciones limita los valores que las variables pueden adoptar.

El problema se resuelve cuando se encuentra un conjunto de valores para las variables de tal manera que todas las restricciones se satisfacen. Durante los años 80, la satisfacción de restricciones se incorpora a lenguajes de programación dando origen a la programación con restricciones, o constraint programing en inglés. Este paradigma permite modelar y resolver CSP's permitiendo entre otros eficiencia en la resolución, reducción en los tiempos de desarrollo, así como también códigos más compactos y expresivos.

2.1.1. Definición formal de un CSP

Un CSP $\mathcal{P}$ se define por la tripleta $\mathcal{P} =< \mathcal{X}, \mathcal{D}, \mathcal{C} >$ donde:

- $\mathcal{X}$ es una n-tupla de variables $\mathcal{X} =< x_1, x_2, ..., x_n >$

- $\mathcal{D}$ es la correspondinte n-tupla de dominio $\mathcal{D} =< \mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_n >$ tal que $x_i \in \mathcal{D}_i$
- $\mathcal{C}$ es una m-tupla de restricciones $\mathcal{C} =< C_1, C_2, ..., C_n >$

2.1.2. Ejemplo de definición formal

Considere el siguiente CSP, el cual se conoce como criptoaritmo. La idea es encontrar dígitos distintos entre 0 y 9 para cada una de las letras del criptoaritmo de tal manera que la ecuación se satisfaga. En este ejemplo, se solicita modelar el criptoaritmo como un CSP, no resolverlo.

- **Definición del modelo**.

  Como se explicó anteriormente un CSP está compuesto por variables (o variables de decisión), dominios y restricciones. Las variables son las incógnitas para las cuales se debe encontrar una solución, los dominios contienen los valores que pueden ser asignados a las variables y el conjunto de restricciones limita los valores que las variables pueden adoptar. Estos valores deben ser identificados desde el enunciado del problema para construir el modelo. Comenzaremos por las variables y sus correspondientes dominios.

- **Variables de decisión y dominios**.

  Se identifican como variables a todas las letras del criptoaritmo, dado que es para ellas que debemos encontrar un valor, corresponde a las incógnitas del problema y como indica el mismo adoptar valores entre 0 y 9.

$$
\begin{gathered}
S \in 0,9 \\
E \in 0,9 \\
N \in 0,9 \\
D \in 0,9 \\
M \in 0,9 \\
O \in 0,9 \\
N \in 0,9 \\
Y \in 0,9
\end{gathered}
$$

- **Restricciones**.

  El problema establece que todos los dígitos deben ser distintos, por ende se deben establecer restricciones de desigualdad entre todas las letras de la siguiente forma:

$$S \neq E$$
$$S \neq N$$
$$S \neq D$$
$$...$$

Una de las ventajas de la programación con restricciones es la existencia de restricciones globales, las cuales permiten aumentar la expresividad de modelado y adumenar la eficiencia de la resolución de CSP's. Una restricción global clásica es la denominada *all different constraint*, la cual garantinza que todas las variables involucradas en la restricción sean distintas. En este ejercicio se utiliza para expresar las restriciones anteriores de corma más compacta y en la resolución aportará eficiencia.

all_ different(S,E,N,D,M,O,N,Y)

Una vez realizada la restricción de las desigualdades entre letras, es necesario modelar la ecuación. Esto se puede realizar transformando cada conjunto de letras en una expresión aritmética de la siguiente forma, donde cada letra es multiplicada por unf actor en función de su posición en la expresión:

$$SEND \Longleftrightarrow S * 1000 + E * 100 + N * 10 + D * 1$$

La restricción completa entonces queda de la siguiente forma:

	S * 1000 +	E * 100 +	N * 10 +	D * 1 +
	M * 1000 +	O * 100 +	R * 10 +	E * 1 =
M * 10000 +	O * 1000 +	N * 100 +	E * 10 +	Y*1

El modelo resultante se muestra a continuación:

- **Variables**
  $S \in 0, 9$
  $E \in 0, 9$
  $N \in 0, 9$
  $D \in 0, 9$
  $M \in 0, 9$
  $O \in 0, 9$

$N \in 0, 9$
$Y \in 0, 9$

- **Restricciones**
  all_ different(S,E,N,D,M,O,N,Y)
  S x 1000 + E x 100 + N x 10 + D x 1 +
  M x 1000 + O x 100 + R x 10 + E x 1 =
  M x 10000 + O x 1000 + N x 100 + E x 10 + Y x 1

Si $M \neq 0$, el problema tiene una única solución, la cual es:

$$\{S = 9, E = 5, N = 6, D = 7, M = 1, O = 0, R = 8, Y = 2\}$$

Si $M = 0$, las soluciones son las siguientes:

S=8, E=5, N=4, D=2, M=0, O=9, R=1, Y=7
S=8, E=4, N=3, D=2, M=0, O=9, R=1, Y=6
S=8, E=3, N=2, D=4, M=0, O=9, R=1, Y=7
S=7, E=6, N=4, D=9, M=0, O=8, R=1, Y=5
S=7, E=6, N=4, D=3, M=0, O=8, R=2, Y=9
S=7, E=5, N=3, D=4, M=0, O=8, R=2, Y=9
S=7, E=5, N=3, D=9, M=0, O=8, R=1, Y=4
S=7, E=5, N=3, D=1, M=0, O=8, R=2, Y=6
S=7, E=4, N=2, D=9, M=0, O=8, R=1, Y=3
S=7, E=3, N=1, D=6, M=0, O=8, R=2, Y=9
S=6, E=8, N=5, D=3, M=0, O=7, R=2, Y=1
S=6, E=8, N=5, D=1, M=0, O=7, R=3, Y=9
S=6, E=5, N=2, D=4, M=0, O=7, R=3, Y=9
S=6, E=4, N=1, D=9, M=0, O=7, R=2, Y=3
S=6, E=4, N=1, D=5, M=0, O=7, R=3, Y=9
S=5, E=8, N=4, D=9, M=0, O=6, R=3, Y=7
S=5, E=7, N=3, D=2, M=0, O=6, R=4, Y=9
S=5, E=7, N=3, D=1, M=0, O=6, R=4, Y=8
S=3, E=8, N=2, D=9, M=0, O=4, R=5, Y=7
S=3, E=8, N=2, D=1, M=0, O=4, R=6, Y=9
S=3, E=7, N=1, D=9, M=0, O=4, R=5, Y=6
S=3, E=7, N=1, D=2, M=0, O=4, R=6, Y=9
S=2, E=8, N=1, D=9, M=0, O=3, R=6, Y=7
S=2, E=8, N=1, D=7, M=0, O=3, R=6, Y=5

2.1.2.1. Resolución de CSP's

Para la resolución de CSP's se utilizará el solver ECLIPe CLP. Este solver está basdao en la programación lógica donde un modelo general tiene la siguiente forma: Head: - Body, donde Head es un predicado (análogo a una función en programación estructurada) que posee un nombre y argumentos. En body se definen las variables, dominios y restricciones.

El modelo comienza estableciendo la librería a utilizar, la librería 'ic' permite modelar problemas con variables enteras, reales o una combinación de ambas. En este problema en particular sólo se utilizan variables de dominio entero. Luego, en la línea 3, se define el nombre del predicado (test1) y su argumento (list). Este predicado se utilizará posteriormente para ejecutar el modelo, el usuario deberá ingresar el argumento que se instanciará en (list)

En la línea 5 se define un comentario para identificar la zona de variables y dominios. Los comentarios se definen con el símnbolo %. En la línea se establece una lista con cuatro variables (A, B, C, D), los nombres de variables comienzan siempre por una letra mayúscula. Luego, en las líneas siguientes se asocia el dominoi a las variables, las cuales podrán adoptar valores entre 1 y 2.

Las restricciones se establecen posteriormente donde los comentarios indican los operadores relacionales utilizados en cada restricción. Finalmente 'labeling' intentará resolver el problema buscando valores para las variables que satisfagan las restricciones.

```
1.:-lib(ic).
2.
3.  test1(List):-
4.
5.  %variables y dominios
6.  List = [A,B,C,D],
7.  A :: 1..2,
8.  B :: 1..2,
9.  C :: 1..2,
10. D :: 1..2,
11.
12. %Restricciones
13. A #\=B, %Distinto
14. B #>=C, %mayor o igual
15. C #< D, %Menor
16. %Resolver
17. labeling(List)
```

Una vez comprendido el modelo, se debe ejecutar la aplicación TkEclipse, compilar y ejecutar el modelo.

2.2. Problema de n-Reinas

El objetivo de este problema es encontrar la posición de n reinas en un tablero de n x n de tal manera que no se ataquen entre ellas. Las reinas se pueden mover a cualquier casilla, tanto de manera horizontal como vertical o diagonal. En este ejercicio, también nos enfocaremos en el modelamiento no en las técnicas de resolución, las que serán explicitadas más adelante.

- **Variables de decisión y dominios**

  El problema requiere identificar las posiciones de las reinas en el tablero, por ende una forma simple de modelar las variables es representando el tablero con una matriz de variables binaria ((i,j) = 1 indica que hay presencia de una reina en la casilla). No obstante, existen formas que permiten una resolución más eficiente del problema. Por ejemplo, el modelo se puede realizar utilizando un vector de variables, donde cada i represente la posición de la reina i en la columna i, como se muestra a acontinuación.

  $$[2, 4, 1, 3]$$

  El vector indica que la reina 1 está ubicada en la fila 2 de la columna 1, la reina 2 está ubicada en la fila 4 de la columna 2, la reina 3 está ubicada en la fila 1 de la columna 3 y finalmente la reina 4 está ubicada en la fila 3 de la columna 4, de igual forma que en la figura 2.1. Este modelo además de tener un espacio de búsqueda más pequeño, evita la restricción de ataque entre reinas en una misma columna, dado que la misma representación de variables evita que haya más de una reina en una columna.

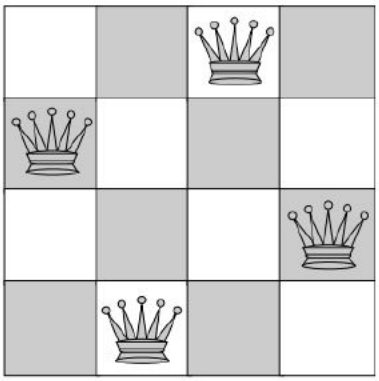


Figura 2.1: Problema de las n-reinas con n=4

- **Tamaño del espacio de búsqueda**

  Cálculo del tamaño del espacio de búsqueda: d^{vars}, donde d es el tamaño del dominio y $vars$ corresponde a la cantidad de variables.

	Modelo con matriz	Modelo con vector
Tamaño del esp. de búsq	2^{n^2}	n^n
n = 4	65536	256
n = 10	127 E30	1.00 E10
Restricciones	filas, columnas diagonales	filas diagonales

Tabla 2.1: Tamaño del espacio de búsqueda para n-reinas

Entonces, utilizando la representación vectorial, las variables se muestran a continuación, donde Q_1 es la reina 1 y puede adoptar valores entre 1 y n.

$$Q_i ... Q_n \in 1, n$$

- **Restricciones**

  El problema requiere que no exista atqeue entre reinas, por ende se deben considerar todas las posibilidades de ataque que este caso es en filas, columnas y diagonales. No obstante, dado que la representación de soluciones es vectorial, sólo se requiere restricciones para filas y diagonales, las cuales se muestran a continuación:

$Q_i \neq Q_j$ (filas)
$Q_i + i \neq Q_j + j$ (diagonal 1)
$Q_i - i \neq Q_j - j$ (diagonal)

$\forall i \in \{1, n-1\}$ and $\forall j \in \{i+1, n\}$

La primera restricción garantiza valores distintos para las filas, por ejemplo para n=4, si $Q_1 = 2$ entonces $Q_2 \neq 2$; $Q_3 \neq 2$ y $Q_4 \neq 2$. La segunda restricción (diagonal 1) evita posiciones en las diagonales derecha-arriba hacia izquierda-abajo. Por ejemplo, si consideramos la solución $[4, 3, 2, 1]$ que es infactible, la infactibilidad será detectada por esta restricción, donde $Q_1 + 1 \neq Q_2 + 2$ no se satisface $(5 = 5)$. Asimismo, la tercera restricción (diagonal 2) evitará posiciones en las diagonales izquierda-arriba hacia derecha abajo.

2.3. Packing Squares

Ubicar un conjunto de cuadrados dentro de una base cuadrada de tal manera que ningún cuadrado se translape con otro.

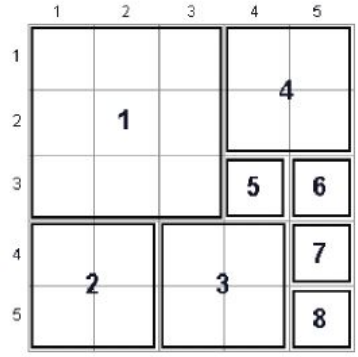


Figura 2.2: Representación gráfica Packing Square

- **Variables**

  $x_1, x_2, ..., x_{squares} \in [1, sideSize]$
  $y_1, y_2, ..., y_{squares} \in [1, sideSize]$

- **Constantes**

$sideSize$
$squares$
$size_1, size_2, ..., size_{squares}$

- **Restricciones** (para $i \in [1, squares]$)
  - Para $i \in [1, squares]$
    $x_i \leq sideSize - size_i + 1$
    $y_i \leq sideSize - size_i + 1$
  - Para $i \in [1, squares]$ y $j \in [i+1, squares]$
    $x_i + size_i \leq x_j$ OR
    $x_j + size_j \leq x_i$ OR
    $y_i + size_i \leq y_j$ OR
    $y_j + size_j \leq y_i$

2.4. Algoritmos de búsqueda y técnicas de filtraje

SOLVING = MODELING + SEARCH

2.4.1. Generate and Test

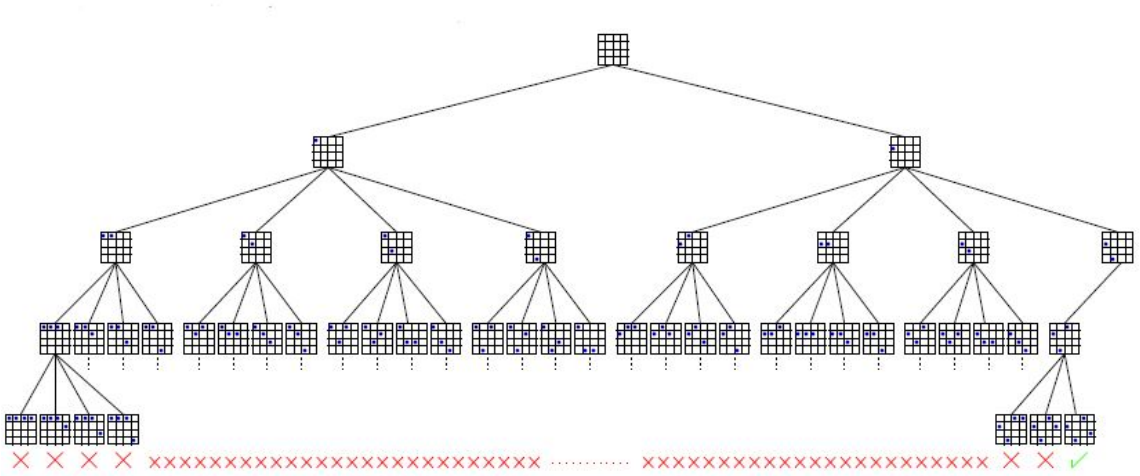


Figura 2.3: Generate and Test

- **Problemas**

- Gran cantidad de instanciaciones que no conducen a una solución.
- Las restricciones se evalúan con todas las variables instanciadas.

- **Solución**
  - Evaluar las restricciones apenas se instancien las variables involucradas.

2.4.2. Backtracking

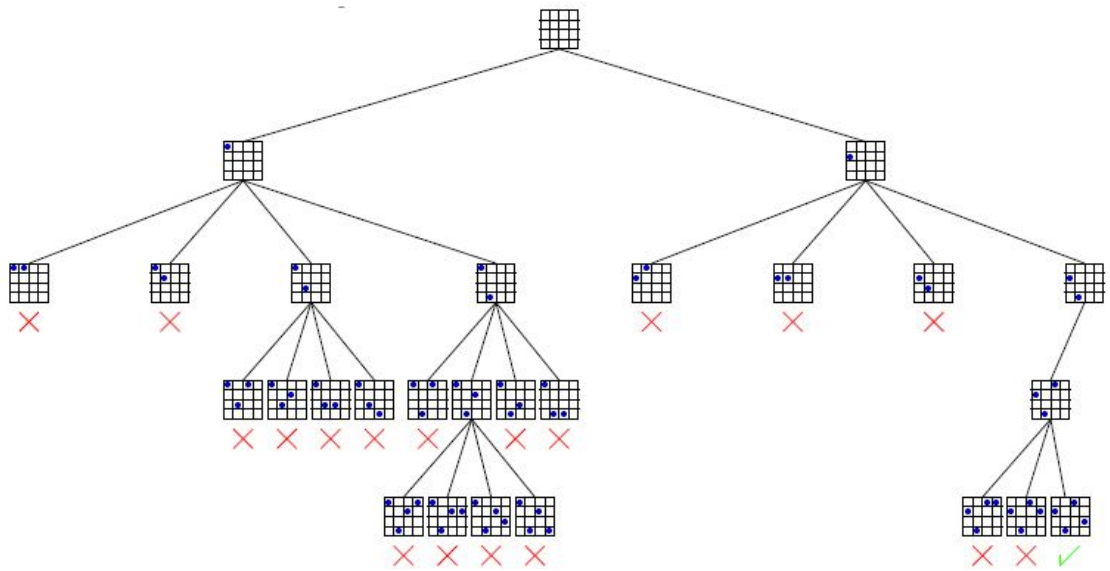


Figura 2.4: Backtracking

- **Principal problema**
  - No se pueden detectar inconsistencias sin instanciar todas las variables involucradas en una restricción.

- **Solución**
  - Eliminar valores temporales de los dominios utilizando técnicas de consistencias (arc-consistency).

2.4.3. Forward Checking

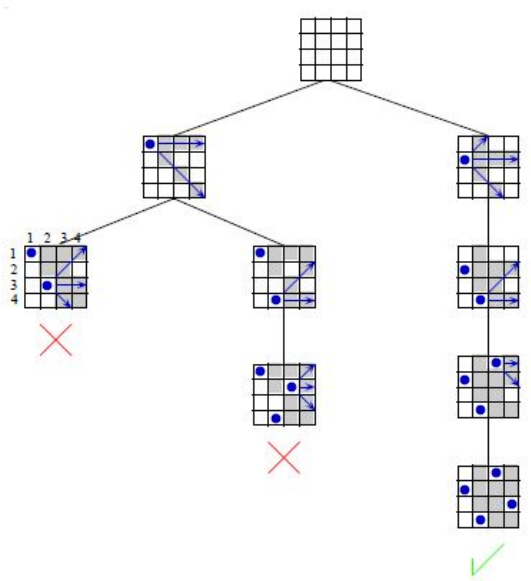


Figura 2.5: Forward Checking

- **¿Se puede mejorar?**
  - Verificar no sólo la consistencia entre la variable actual y las futuras, sino que también entre las futuras...

2.4.4. Maintaining Arc Consistency (full look ahead)

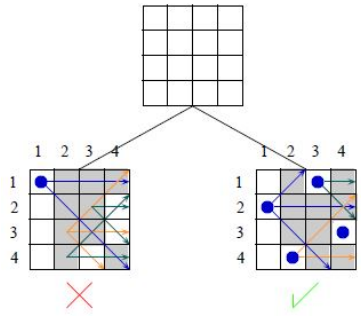


Figura 2.6: Maintaining Arc Consistency (full look ahead)

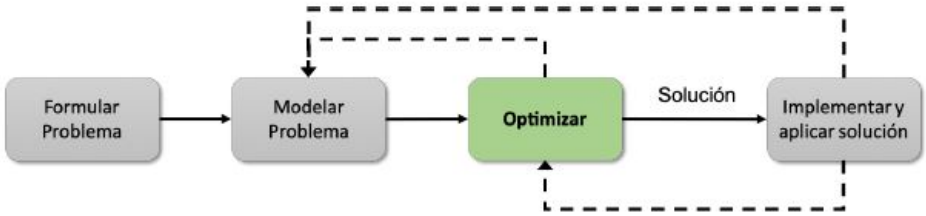
- **Optimización**
  - Basta con extender el algoritmo de búsqueda para considerar la función objetivo.

Capítulo 3

Problemas de optimización

3.1. Introducción

Los problemas de optimización surgen en diversos contextos y hoy en día muchos de ellos se resuelven utilizando técnicas de optimización. En la práctica, encontrar buenas soluciones a problemas de optimización complejos puede ser un desafío, requiriendo de el análisis de un experto y el uso de técnicas adecuadas para su resolución.



Primero, se debe reconocer el problema y formularlo. Una vez definido el problema, este debe ser modelado para luego ser rsuelto con técnicas adecuadas al modelo propuesto. Finalmente, la solución encontrada debe ser implementada y aplicada debe ser implementada y aplicada para dar respuestas al problema real. Es posible que esta solución no sea suficientemente buena por lo que el proceso debe iterar para revisar y mejorar la técnica de optimización aplicada y, si es necesario, cambiar el modelo del problema para lograr mejores soluciones. Un problema de optimización puede modelarse de diversas maneras y por lo tanto puede tener diferentes características en

base al modelo definido para resolverlo.

3.2. Modelo matemático

Un modelo es una representación (matemática) de un problema real. El modelo abstrae las características del problema real y lo simplifica de manera que pueda ser resuelto de forma más sencilla. Los elementos de un modelo son:

- **Variables de decisión:** Las incógnitas para las cuales se debe encontrar un valor.
- **Dominios:** Los posibles valores que pueden adoptar las variables
- **Parámetros:** La información conocida del problema (normalmente constantes).
- **Restricciones:** Las expresiones matemáticas que limitan los valores que las variables pueden adoptar.
- **Función objetivo:** La(s) expresión(es) matemática(s) que será(n) minimizada(s) o maximizada(s).

3.3. Clasificación de problemas de optimizacion

Los problemas se pueden clasificar en base a las características de éstos, estas características se ven reflejadas en los elementos del modelo. Con respecto a las variables de decisión, hay problemas:

- Continuos: Las variables del problema poseen dominios reales
- Discretos, combinatoriales: Las variables del problema poseen dominios enteros o categóricos.
- Mixtos: Poseen dominios reales y enteros o categóricos.

Con respecto a las restricciones, hay problemas:

- Restringidos: El problema posee expresiones que limitan los valores que las variables pueden adoptar.
- No restringidos: El problema no posee limtaciones a los valores de las variables.

Con respecto a la función objetivo, hay problemas:

- De un objetivo: El problema posee una expresión matemática que debe ser maximizada o minimizada.
- Multiobjetivo: El problema posee dos o más expresiones matemáticas que deben ser maximizadas o minimizadas simultáneamente.

3.4. Ejemplos de problemas de optimización.

A continuación se presentan 4 ejemplos de problemas de optimización junto con sus respectivos modelos.

3.4.1. Ejemplo 1: Función

Un cientifico define la siguiente función:

$$f(x) = \sum_{i=1}^{4} 100 * (x_{i+1} - {x_1}^2)^2 + (1 - x_i)^2$$

Se desea encontrar los valores de x_1, x_2, x_3, x_4 y x_5 que minimizan el valor de $f(x)$, considerando que las variables x_i no pueden tener un valor menor que 0 ni mayor que 100.

- **Variables:** x_i, $i = \{1, 2, 3, 4, 5\}$
- **Dominios:** $x_i \in \Re_{>0}, 0 \leq x_i \leq 100$
- **Parámetros:** No hay.
- **Restricciones:** No hay
- **Función objetivo:**

$$Miniminazar f(x) = \sum_{i=1}^{4} 100 * (x_{i+1} - {x_1}^2)^2 + (1 - x_i)^2$$

Este problem es por lo tanto, un problema continuo, no restringido y de un objetivo.

3.4.2. Ejemplo 2: Cilindro

Un emprendedor porteño ha creado una innovadora bebida de frutas y plantas, la cual será producida 100 % con productos locales.

Para su comercialización se desea diseñar un envase de aluminio (cilíndrico) el cual deberá poseer un volumen total de 128 ml.

Se desea encontrar un diseño para el envase de aluminio que minimice la superficie total de aluminio de este.

- **Variables**

  x: Radio del cilindro.
  y: Altura del cilindro.

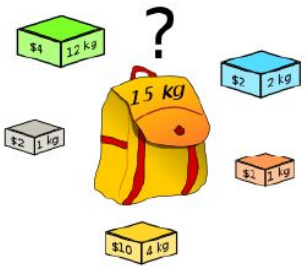
- **Dominios** $x, y \in \Re_0$
- **Parámetros:** No hay
- **Restricciones:** $2\pi x^2 y = 128$ #Área del circulo x altura del cilindro
- **Función objetivo:**

$$Minimizar f(x, y) = 2\pi x^2 y + 2\pi xy$$

El problema es por lo tanto, un problema continuo, restringido y de un objetivo.

3.4.3. Ejemplo 3: Mochila

Dos exploradores encuentran un tesoro en una caverna que está a punto de colapsar. Cada objeto tiene un peso y un valor diferente:



	o_1	o_2	o_3	o_4	o_5	o_6	o_7	o_8
valor	32	47	18	26	85	33	45	59
peso	26	48	21	22	95	43	55	52

Se desea seleccionar los objetos que se deben llevar en sus mochilas considerando:

- El peso total de los objetos seleccionados no debe ser más de 10 KG (de lo contrario no alcanzarán a escapar de la cueva).
- Maximizar el valor de los objetos seleccionados (no podrán volver por el resto del tesoro).

Modelo matemático:

- **Variables:** x_i: indica si se selecciona el objeto i, $i = \{1, 2, ..., 8\}$
- **Dominios:** $x_i \in \{0, 1\}$
- **Parámetros:**

  $v_i = \{32, 47, 18, 26, 85, 33, 45, 59\}$, valor del objeto i
  $p_i = \{26, 48, 21, 22, 95, 43, 55, 52\}$, peso del objeto i
- **Restricciones:**

$$\sum_{i=1}^{8} x_i * p_i \leq 101$$

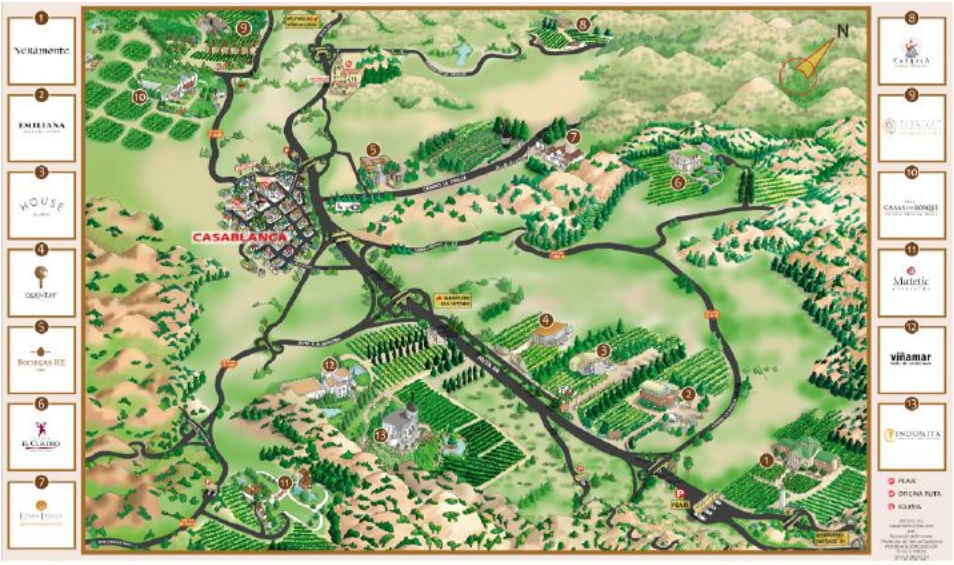
- **Función objetivo:**

$$max f(x) = \sum_{i=1}^{8} x_i * v_i$$

Este problema es por lo tanto, un problema combinatorial, restringido y de un objetivo.

3.4.4. Ejemplo 4 - Enólogo Viajero(alias vendedor viajero o TSP)

Un grupo de amigos interesados en el enoturismo desea recorrer en bicicleta 13 viñas del valle de Casablanca.



Se desea encontrar la tura más corta que pasa por todas las viñas, y termina en la viña inicial.

Modelo matemático

- **Variables:**

  x_i : indica la viña que se visita en la posicion i, $i = \{1, 2, ..., 13\}$

- **Dominios:**

  $x_i \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 12\}$

- **Parámetros:**

  d_{ij}: distancia de la viña i a la viña j

- **Restriciones**

  $x_i \neq x_j$, $\forall ij$, $i \neq j$

- **Función objetivo:**

  $min\ d(x_1, x_2) + d(x_2, x_3) + d(x_3, x_4) + d(x_4, x_5) + d(x_5, x_6) + d(x_6, x_7) + d(x_7, x_8) + d(x_8, x_9) + d(x_9, x_{10}) + d(x_{10}, x_{11}) + d(x_{11}, x_{12}) + d(x_{12}, x_{13}) + d(x_{13}, x_1)$

Este problema es por lo tanto, un problema combinatorial, restringido y de un objetivo.

3.5. Problemas vs Instancias.

Existen problemas que aparecen en varios contextos, estos problemas son entonces definidos en forma general para facilitar su estudio. Ejemplos clásicos de estos problemas son el problema de la mochila y del vendedor viajero (TSP). A continuación, se describe su formulación general:

Problema de la mochila

Dado un numero de objetos n, sus valores $v_i = \{v_1, v_2, ..., v_n\}$, sus costos $p_i = \{p_1, p_2, ..., p_n\}$ y un máximo de recursos disponibles W. Encontrar una selección de estos objetos que:

$$maximizar f(x) = \sum_{i=1}^{n} v_i x_i$$

Sujeto a:

$$\sum_{i=1}^{n} p_i x_i \leq W$$

$$x_i \in \{0, 1\}$$

Problema del vendedor viajero

Dado un número de ciudades n y una matriz de distancias entre ellas $d(a, b)$. Encontrar un recorrido x que pase por todas las ciudades y que empiece y termine en la misma ciudad de manera que:

$$minimizar f(x) = \sum_{i=1}^{n-1} d(x_i, x_{i+1}) + d(x_n, x_1)$$

Sujeto a:

$$x_i \neq x_j, \forall ij, i \neq j$$

$$x_i \in \{1, 2, ..., n\}$$

En la práctica, cuando se resuelve un problema, realmente se resuelve una **instancia** del problema. La instancia define la cantidad de variables, sus dominios y los valores para los parámetros del modelo. Los problemas descritos al inicio de este capituo son por lo tanto instancias de cada uno de los problemas generales recién presentados.

3.6. Espacio de búsqueda.

El espacio de búsqueda de un problema es el espacio de todas las posibles soluciones. Este espacio está caracterizado por los elementos del modelo. Dentro del espacio de búsqueda se pueden encontrar soluciones:

- **Factibles:** Soluciones que cumplen con todas las restricciones del problema.
- **Infactibles:** Soluciones que no cumplen con todas las restricciones del problema.

Entre más variables tenga un problema y más grande sean los dominios de las variables, el espacio de búsqueda será más grande, dificultando la búsqueda de buenas soluciones. En el peor de los casos, deberemos revisar todas las posibles soluciones a un problema para encontrar la mejor solución (óptimo). Este procedimiento se llama comúnmente resolución por fuerza bruta y sólo es posible para resolver problemas que poseen espacios de búsqueda relativamente pequeños debido a su excesivo costo computacional.

¿Cuál es el tamaño del espacio de búsqueda del problema de la mochila?

El problema presentado en el ejemplo 3.4.3 posee 8 variables discretas, cada una con un dominio de tamaño 2 (valores {0,1}). Las cantidades de posibles soluciones es por lo tanto:

$$2^8 = 256 \text{ soluciones}$$

Si pensamos en la formulación general del problema, la cantidad de soluciones es: 2^n

$$\begin{array}{lcll} n = 10 & \rightarrow & 2^{10} = & 1{,}024 \\ n = 15 & \rightarrow & 2^{15} = & 32{,}768 \\ n = 20 & \rightarrow & 2^{20} = & 1{,}048{,}576 \\ n = 30 & \rightarrow & 2^{30} = & 1{,}073{,}741{,}824 \end{array}$$

Si el problema requiere que seleccionemos entre 30 objetos debemos revisar más de 1000 millones de soluciones.

¿Cuál es el tamaño del espacio de búsqueda del problema del vendedor viajero?

El problema presentado en el ejemplo 3.4.4 posee 13 variables discretas, cada una con un dominio de tamaño 13 (número de viñas a visitar). Por ejemplo, la solución:

$$10 - 6 - 3 - 8 - 5 - 11 - 2 - 13 - 7 - 12 - 9 - 1 - 4$$

Indica que inicialmente se visitará la viña 10, luego la viña 6 y el recorrido continuará hasta llegar a la viña 4 para luego retornar a la viña 10. La cantidad de soluciones posibles es entonces:

$$13 * 12 * 11 * .... * 1 = 13!$$

Note que, es posible asumir que la distancia desde la viña i a la j es igual a la distancia de ir de la viña j a la viña i, en este caos el problema es simétrico. En la práctica, esta propiedad no se cumple siempre debido a que muchas veces una ruta es sólo posible en una direccion. Si asumimos simetría, la solución:

$$4 - 1 - 9 - 12 - 7 - 13 - 2 - 11 - 5 - 8 - 3 - 6 - 10$$

Corresponde al mismo recorrido anterior, pero realizado en el orden inverso, por lo tanto tendrá el mismo valor de la función objetivo. Si se quitan las soluciones simétricas del total de soluciones posibles, la cantidad de soluciones en el espacio de búsqueda es:

$$\frac{(13-1)!}{2} = \frac{12!}{2} = 239{,}500{,}800 \text{ soluciones}$$

Si pensamos en la formulación general del problema la cantidad de soluciones es:

$$\frac{(n-1)!}{2} \text{ soluciones}$$

$$\begin{array}{lcll} n = 7 & \rightarrow & \frac{6!}{2} = & 360 \\ n = 10 & \rightarrow & \frac{9!}{2} = & 181{,}440 \\ n = 15 & \rightarrow & \frac{14!}{2} = & 43{,}589{,}145{,}600 \\ n = 20 & \rightarrow & \frac{19!}{2} = & 60{,}822{,}550{,}204{,}416{,}000 \end{array}$$

Sobre n=70 superamos el número estimado de átomos en el universo.

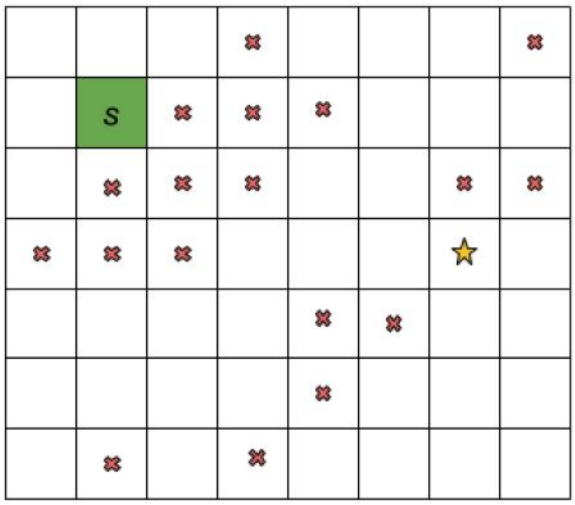
3.7. Restricciones y funciones de evaluación

Las funciones de evaluación son funciones que evalúan soluciones y que pueden incluir en la evaluación de una solución otros criterios que no son directamente la función objetivo, pero que pueden ayudar a guíar la búsqueda hacia buenas soluciones. Muchos de estos criterios se relacionan con la satisfacción de las restricciones.

Como ya hemos visto, las restricciones son expresiones que restringen que soluciones son válidas para el problema (soluciones factibles) o inválidas para un problema (soluciones infactibles). Una opción para manejar las restricciones al ejecutar una metaheurística es simplemente descartar las soluciones infactibles no tomándolas en cuenta como soluciones durante el proceso de búsqueda.

Por otra parte, las **metaheurísticas** en general definen movimientos u operadores para generar nuevas soluciones y así moverse a través del espacio de búsqueda. Si queremos aplicar la técnica de descartar soluciones infactibles descrita arriba, dependiendo del movimiento que utilicemos, las restricciones pueden dificultar la búsqueda de soluciones.

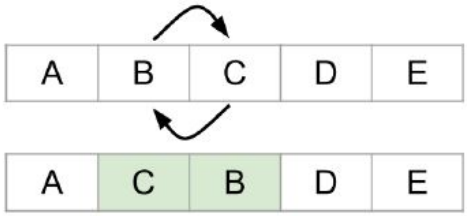
Veamos un ejemplo, supongamos un espacio de búsqueda sencillo como el de a continuación donde todas las soluciones marcadas con una cruz roja son infactibles y la solución marcada con una estrella es el óptimo.



Si nos encontramos en la solución S (color verde) y el movimiento (u operador) definido permitirá que las nuevas soluciones se muevan sólo un espacio en cualquier dirección, entonces la búsqueda no podrá avanzar hacia

el óptimo por que las soluciones que se encuentran en el camino a recorrer son soluciones factibles.

Una estrategia para enfrentar este problema es diseñar el movimiento u operador para que no genere soluciones infactibles, un buen ejemplo de tal movimiento es el movimiento swap que estudiamos para el problema del vendedor viajero. Las restricciones de este problema corresponden a la estructura de permutación de la solución y como el movimiento swap intercambia ciudades no puede generar soluciones infactibles.



En otros casos, crear este tipo de movimiento es complejo, debido a que las restricciones pueden ser complejas y difíciles de satisfacer. En estos casos, para evitar que el algoritmo no pueda avanzar es posible permitir que las soluciones violen restricciones agregando una penalización a la función objetivo. De esta manera, el algoritmo sabrá que una solución infactible tiene una función evaluación mala y naturalmente se moverá a través de soluciones infactibles hasta encontrar una que no viole restricciones.

Por ejemplo, para el problema de la mochila, con los siguientes objetos disponibles para ser seleccionados de manera que no exceda $W = 6KG$ de peso.

Supongamos la siguiente solución:

$s_1 =$ | 1 | 1 | 0 | 0 | 0 | función objetivo: 5300 peso: 9kg

Si solo tomaramos en cuenta la función objetivo pensaríamos que esta es una solución muy buena, pero al revisar la restricción del peso máximo

vemos que no es factible:

$$TW = \sum_{i=0}^{n} x_i w_i = 6 + 3 = 9 > W$$

Para aplicar una metaheurística para el problema podemos definir la siguiente función de evaluación:

$$\sum_{i=1}^{n} x_i v_i - P(x)$$

Donde $P(x)$ es una función de penalización que en este caso, reduce el valor de la función objetivo en caso de que la solución sea infactible:

$$P(x) = \begin{cases} \rho(TW - W) & , si \quad TW > W \quad x\ es\ infactible \\ 0 & , si \quad TW \leq W \quad x\ es\ factible \end{cases}$$

Y ρ es una constante que corresponde al máximo valor entre el valor (v_i) y el peso (w_i) de un objeto:

$$\rho = \max_{i=1..n} \frac{v_i}{w_i}$$

Ahora, calculando la función de evaluación de la solución, tenemos que:

$$\rho = \frac{2500}{3}$$

$$\sum_{i=1}^{n} x_i v_i - P(x) = 5300 - \left(\frac{2500}{3} * (9 - 6) \right) = 2800$$

Las funciones de evaluación son muy utilizadas en metaheurísticas, debido a que los movimientos y operadores a menudo pueden generar soluciones infactibles, haciendo imposible la búsqueda en los casos en que existen restricciones muy difíciles de satisfacer. Las metaheurísticas optimizan la función de evaluación moviéndose en el espacio de búsqueda pudiendo atravesar zonas de infactibilidad inherentemente intentando reparar las restricciones no cumplidas.

Capítulo 4

Heurísticas

4.1. Introduccion

En este módulo nos enfocaremos en los problemas que son difíciles de abordar por las técnicas de resolución estudiadas en los módulos anteriores, las cuales entran en la categoría de técnicas completas o exactas. La figura 4.1 entrega una clasificación general de las técnicas utilizadas para la resolución de problemas de optimización.

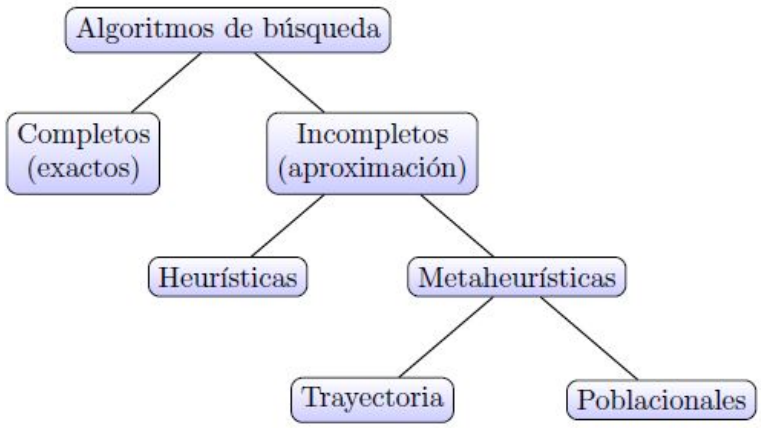


Figura 4.1: Clasificación de técnicas de resolución de problemas

Las técnicas completas o exactas son capaces de entregar la mejor solución posible del espacio de búsqueda (óptimo). Para ello, la búsqueda de soluciones que realizan estas técnicas abarca todo el espacio de búsqueda, de

ahí el nombre de *completas*. Cuando los problemas poseen grandes espacios de búsqueda, aplicar este tipo de técnicas se vuelve imposible debido al tiempo requerido para llevar a cabo la gran cantidad de computación requerida para obtener una buena solución, o simplemente una solución factible. En resumen, estas técnicas se caracterizan por:

- Ser capaces de entregar una solución óptima al problema
- Ser capaces de averiguar que no existe una solución factible al problema.
- Requerir un tiempo de ejecución que crece rápidamente en relación a la:
  - Cantidad de variables
  - Cantidad de valores de los dominios de las variables.

Las técnicas incompletas o de aproximación son técnicas que no tienen como meta obtener la mejor solución posible del espacio de búsqueda (óptimo), si no que apuntan a obtener una **buena solución** utilizando la cantidad de recursos computacionales disponibles. Es así como las técnicas incompletas se mueven a través del espacio de búsqueda intentando revisar la mayor cantidad posible de soluciones y utilizando la información obtenida sobre el espacio de soluciones para guíar la búsqueda.

Debido a los grandes espacios de búsqueda, muchas de estas técnicas utilizan la aleatoriedad para moverse a través del espacio recogiendo información de distintas áreas del espacio de búsqueda. En resumen, estas técnicas se caracterizan por:

- Ser comúnmente estocásticas (utilizan aleatoriedad).
- Entregar la mejor solución encontrada sin garantía de optimalidad.
- Ser capaces de manejar grandes espacios de búsqueda.
- No requerir gran conocimiento a priori del problema.

4.2. Definicion

El término heurística deriva del griego *heuriskein* que significa encontrar; descubrir. Es un concepto utilizado en varias disciplinas para identificar las

estrategias que se utilizan para tomar decisiones en base a cierta información o suposición sobre la sitaución o problema a resolver.

En optimización, las heurísticas son diseñadas para aplicarse a un problema en particular, es decir, no es común que sea posible aplicar una heurística de un problema a otro. Esto se debe a que para tomar decisiones (por ejemplo, que valor elegir para una variables) se utiliza cierta información de la estructura del problema a resolver.

El objetivo principal de las heurísticas es dar una solución - si es posible relativamente buena- a un problema de forma rápida. Las heurísticas pueden ser constructivas o perturbativas dependiendo si están diseñadas para construir o modificar una solución.

4.3. Tipos de heurísticas

4.3.1. Heurísticas constructivas

Las heurísticas constructivas son reglas que nos permiten generar soluciones de manera rápida.

- Suponga el siguiente problema.

  Se desea crea un personaje para un juego de aventuras, el personaje sólo puede llevar un máximo de 6 KG de peso de equipamiento. Para eso, se debe seleccionar el equipamiento del personaje a partir de los objetos descritos en la figura 4.2 con el objetivo de maximizar el valor de los objetos equipados.

  Note que este problema corresponde al problema de la mochila.

Figura 4.2: Objetos disponibles para el equipamiento del personaje

Hay varias posibles heurísticas que es posible aplicar para generar una solución:

1. Elegir en orden los objetos más valiosos:

2. Elegir en orden los objetos más livianos:

Note que para la selección del tercer objeto hay dos opciones con mismo peso (3 KG). Por lo tanto, para seleccionar uno de estos objetos se necesita otra heurística, en este ejemplo, se seleccionó el objeto más valioso. Otra opción sería seleccionar un objeto aleatoriamente.

3. Elegir en orden los objetos con mayor razón valor/peso:

Para aplicar esta heurística calculamos la razón valor peso de cada objeto:

$$
\begin{array}{ll}
\text{Estatuilla:} & \frac{2800}{6} = 466,67 \\
\text{Espada:} & \frac{2500}{3} = 833,33 \\
\text{Monedas:} & \frac{1000}{3} = 333,33 \\
\text{Mazo:} & \frac{1200}{1,5} = 800 \\
\text{Botas:} & \frac{500}{1} = 500
\end{array}
$$

Las heurísticas generan soluciones en base a los criterios definidos por ellas, se obtuvo una mejor solución en este caso con la heurística de la razón valor/peso y, debido a la heurística para seleccionar objetos empatados, la heurística que selecciona los objetos más livianos

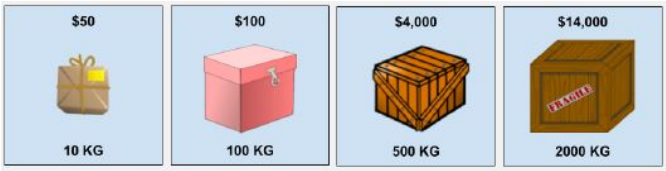
- Veamos otro ejemplo, esta vez estamos cargando un camión y el peso máximo es de 5000 KG. La figura 4.3 muestra los objetos disponibles para ser cargados. Se desea encontrar una solución para este problema intentando maximizar el valor de los objetos cargados en el camión.

Figura 4.3: Objetos disponibles para cargar el camión

1. Elegir en orden los objetos más valiosos:

Valor total: 28 000

2. Elegir en orden los objetos más livianos:



Valor total: 18 150

3. Elegir en orden los objetos con mayor razón valor/peso:

Valor total: 18 150

A diferencia del primer ejemplo, la mejor solución fue encontrada por la heurística que selecciona los objetos más valiosos. Esto evidencia que la heurísticas constructivas sólo proporcionan una estrategia para intentar generar una solución de forma 'rápida', pero estas no garantizan que la calidad de la solución.

- Realicemos el mismo ejercicio con un problema diferente: el vendedor viajero.

  Suponga que tenemos un grupo de 5 ciudades a visitar {A,B,C,D,E} y que se ha entrega una matriz de las distancias entre ciudades. Podemos visualizar las ciudades y sus distancias en un grafo tal como se presenta en la figura

	A	B	C	D	E
A	-	4	5	3	7
B	4	-	7	5	12
C	5	7	-	6	3
D	3	5	6	-	8
E	7	12	3	8	-

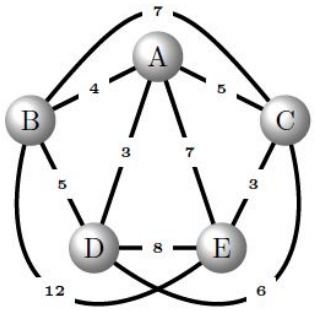


Figura 4.4: Tabla y grafo de distancia entre ciudades

Nota: Las heurísticas propuestas para el problema anterior no pueden ser aplicadas a este problema ya que los conceptos utilizados para decidir que componente de la solución seleccionar en cada paso no tienen sentido para el problema del vendedor viajero. Se debe por lo tanto, pensar en una nueva heurística.

Podemos generar una solución utilizando la heurística del vecino más cercano siguiendo los siguientes pasos

1. Seleccione una ciudad inicial aleatoriamente.
   $\Rightarrow$ seleccionamos la ciudad B
2. Iterativamente seleccione las ciudades más cercanas a la última ciudad seleccionada:
   - La ciudad más cercana a B es A con una distancia de 4

     $B - A$ Distancia preliminar: 4

   - La ciudad más cercana a A es D con una distancia de 3

     $B - A - D$ Distancia preliminar: 4+3 = 7

- La ciudad más cercana a D es C con una distancia de 6

$B - A - D - C$ Distancia preliminar: 4+3+6 = 13

- La ciudad más cercana a C es E con una distancia de 3

$B - A - D - C - E$ Distancia preliminar: 4+3+6+3 = 16

- Retornamos finalmente a B

$B - A - D - C - E - (B)$ Distancia preliminar: 4+3+6+3+12 = 28

4.3.2. Heurísticas perturbativas

Las heurísticas perturbativas son reglas que se pueden aplicar para modificar una solución y obtener una nueva solución.

Ejemplo 1

Recuerde el problema de la mochila para el cual se deben seleccionar objetos para equipar a un personaje con un máximo de 6KG

Figura 4.5: Objetos disponibles para el equipamiento del personaje

Supongamos que tenemos la siguiente solución para este problema:

Valor total: 1 700

Recuerde el modelo propuesto para este problema en el capítulo anterior: en el modelo se representan las soluciones con las variables x_i las cuales pueden tomar el valor 0 si el elemento i no está seleccionado y 1 si el elemento 1 está seleccionado.

Esto puede ser visto como un vector de 0's y 1's del tamaño del número de elementos, por lo tanto la solución puede ser representada de la siguiente manera:

0	0	0	1	1

En este vector la posición 1 del vector indica si el objeto 1 (estatuilla) es seleccionada o no, la posición 2 corresponde al objeto 2 (espada), etc.

Además, dicho vector es una posible representación computacional de la solución del problema y en base a la representación de la solución se deben luego diseñar algoritmos para buscar soluciones a un problema.

Una regla heurística de perturbación podría ser:

- *Sacar un objeto ya seleccionado en la solución o poner un objeto no seleccionado en la solución.*

  - Solución inicial:

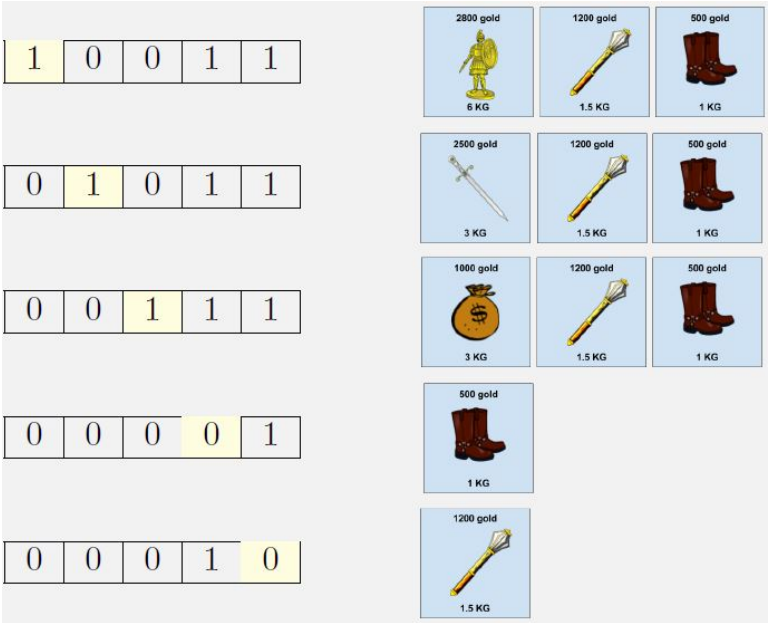
  - Solución aplicando la heurística para agregar el objeto 1:

  Note que la nueva solución generada difiere solo en una variable del vector (x_1) cuando la comparamos con la solución inicial. Así, podemos generar el grupo de todas las posibles soluciones que pueden ser generadas a partir de la solución inicial dada, cambiando sólo una variable de esta solución.

- *Sacar un objeto ya seleccionado en la solución o poner un objeto no seleccionado en la solución*

- Solución inicial:

- Soluciones generadas:

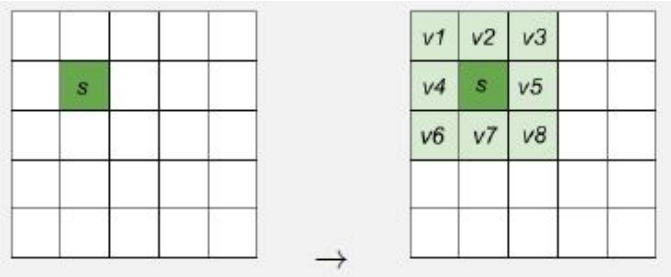


4.4. Vecindarios

Las heurísticas perturbativas son comúnmenete llamadas movimientos ya que permiten moverse de una solución a otra aplicando una regla de perturbación. El conjunto de soluciones que se pueden obtener a partir de una solución inicial aplicando un movimiento recibe el nombre de vecindario. En este caso, las soluciones generadas en el ejemplo anterior para cada problema se encuentran en el vecindario de la respectiva solución inicial. Veamos un ejemplo un poco más visual para comprender este concepto:

- Suponga que esta grilla representa el espacio de búsqueda (todas las posibles soluciones a un problema). Cada una de las celdas es una solución, la solución marcada en verde oscuro (s) es una solución inicial,

imagine que en esta celda se encuentra nuestro personaje de juego. Definimos el movimiento disponible para personaje como el cambio de posición en una unidad de forma horizontal, vertical o diagonal. Las soluciones en color verde claro corresponden a los vecinos (v_i) de la solución actual s, obtenidos a partir de la aplicación del movimiento.



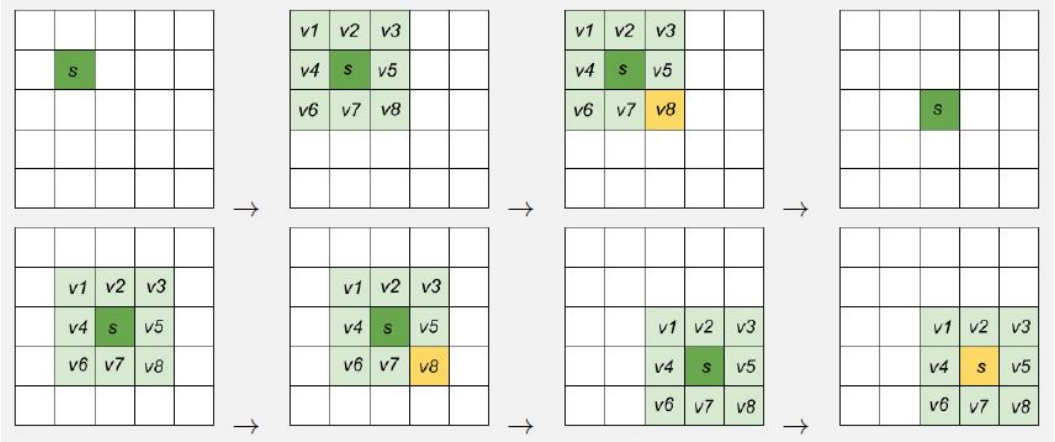
- El personaje podría moverse a cualquiera de los vecinos de su posición actual.

4.5. Óptimos Locales

Podemos movernos de una solución a otra en base a un criterio definido o criterio de aceptación. El criterio más simple es moverse a la mejor solución del vecindario. De esta manera, podemos movernos a través de diferentes soluciones del espacio de búsqueda aplicando repetitivamente un movimiento y un criterio de aceptación.

- Suponga que las celdas marcadas en amarillo en cada vecindario representan las soluciones de mejor función objetivo del vecindario, incluyendo dentro del vecindario a la solución s. Se define el criterio de aceptación como: *cambiar siempre a la mejor solución disponible en el vecindario*. De esta manera nos moveremos:

  1. Generando el vecindario
  2. Identificando al mejor vecino
  3. Designando como nueva solución actual al mejor vecino para luego generar su vecindario.



- Este procedimiento termina al encontrarnos en una celda (solución) la cual posee la mejor función objetivo de su vecindario.

Este procedimiento se denomina búsqueda local, debido a que finaliza en una solución que es un óptimo local.

Un óptimo local es una solución desde la cual, aplicando un movimiento definido, no es posible encontrar una solución con mejor calidad entre sus vecinos.

En un espacio de búsqueda pueden existir múltiples óptimos locales. La figura 4.6

Figura 4.6: Espacio de búsqueda y óptimos

4.6. Exploración y explotación

Los conceptos de exploración y explotación son la base de la estrategia de búsqueda de las metaheurísticas. El balance entre estas dos estrategias es primordial para encontrar buenas soluciones en espacios masivos. La figura 4.7 ilustra los conceptos de exploración y explotación en un espacio de búsqueda simplificado de manera que tiene una dimensión.

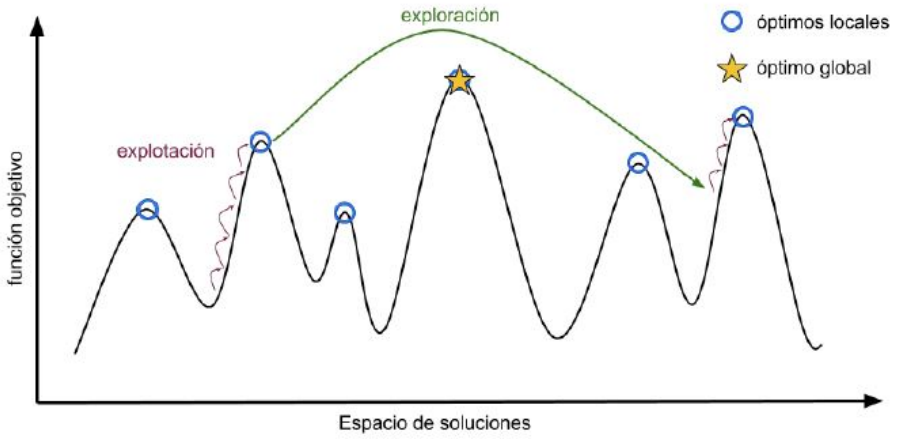


Figura 4.7: Exploración y explotación

La exploración (también llamada diversificación) consiste en diferentes áreas de búsqueda con el objetivo de identificar áreas prometedoras donde podría ser posible encontrar buenas soluciones. De esta manera, la exploración corresponde a la capacidad de un proceso de búsqueda de abarcar un área extendida del espacio de búsqueda. Este componente de la estrategia de búsqueda es muy importante cuando se conoce poco del espacio de búsqueda de un problema.

La explotación (también llamada intensificación) consiste en revisar exhaustivamente un área acotada del espacio de búsqueda, en la cual se cree pueden encontrarse soluciones de alta calidad. La explotación comunmente se realiza buscando alrededor de soluciones de buena calidad y revisando los vecindarios de estas soluciones.

En la figura 4.7, la explotación está representada por las flechas de color rosa, el movimiento es general pequeño ya que el objetivo de esta estrategia es explotar el conocimiento de las actuales soluciones. La explotación consistirá en realizar pequeños cambios a las solucione actuales para revisar

si es posible encontrar mejores soluciones a partir de ellas. La exploración está representada por la flecha verde. El movimiento asociado a esta es más grande, permitiendo alejarse de la solución de actual y explorando áreas más alejadas del espacio de búsqueda.

Capítulo 5

Algorítmos genéticos

5.1. Introduccion

Los algoritmos de población o poblaciones mantienen una población de soluciones durante el proceso de búsqueda. Comparados con los algoritmos de trayectoria, los algoritmos poblacionales al mantener varias soluciones pueden seguir diversas trayectorias a la vez, siendo la población la que se mueve a través del espacio de búsqueda evolucionando en conjunto en cada paso.

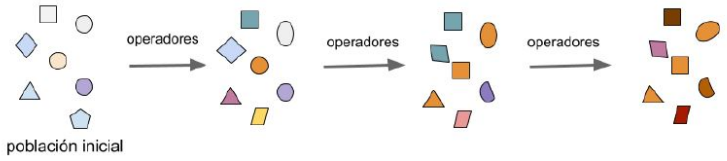


Figura 5.1: Ilustración de la evolución de una población

Los algoritmos genéticos basan la búsqueda de soluciones en el proceso de evolución Darwiniana.

5.2. Soluciones en un algoritmo genético

Desde la perspectiva de los algoritmos genéticos, las soluciones son individuos en las cuales se puede reconocer el genotipo y el fenotipo.

El genotipo corresponde a la representación de la solución, por ejemplo, para el problema de la mochila, el genotipo correspondería a la representación

binaria de las soluciones:

0	0	0	1	1

El fenotipo a su vez, corresponderá a la solución representada por este genotipo:

Valor total: 1 700

Cada individuo es evaluado de manera que se obtiene un valor que define su aptitud o fitness. El fitness corresponde al valor de la función objetivo de las soluciones al problema a resolver o a una función de evaluación.

5.3. El algoritmo genético

En los algoritmos genéticos los individuos son sometidos a un proceso de reproducción, mutación y selección de manera de los mejores individuos posean mayores posibilidades de pasar su material genético a las siguientes interacciones. La figura muestra el diagrama de un algoritmo genético. El algoritmo genético inicia creando una población de soluciones iniciales (G), en cada iteración (o generación) se selecciona un número p de individuos para ser padres (P), estas soluciones son cruzadas o recombinadas de manera que su material genético es combinado para formar nuevas soluciones (G'). Las nuevas soluciones pasan por un proceso de mutación y finalmente se seleccionan las soluciones que formarán la población en la siguiente generación, para luego recomenzar el proceso.

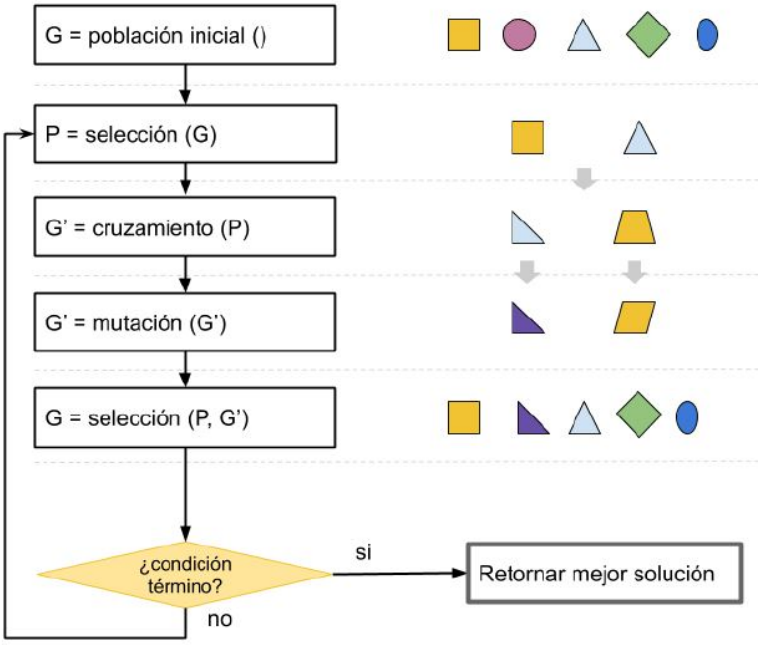


Figura 5.2: Diagrama general de un algoritmo genético

A continuación reviisaremos cada uno de los componentes de los algoritmos genéticos también llamados en este contexto *operadores*.

5.4. Generación inicial de soluciones

Tal como en las metaheurísticas de trayectoria, la generación de soluciones para integrar la población inicial puede hacerse de diversas formas: aplicando heurísticas, definiendo soluciones iniciales o generando soluciones aleatorias.

5.4.1. ¿Qué características debería tener la población inicial de soluciones?

La diversidad de la población inicial es muy importante debido a que favorece la exploracio´n del espacio de búsqueda. Si la soluciones de la población inicial son muy parecidas, entonces los hijos generados por ellas también serán muy parecidos por lo que no se podrá explorar inicialmente el espacio de búsqueda tal como es recomendado para los procesos de búsqueda

incompleta.

5.5. Selección de padres en algoritmos genéticos

En cada iteración, un número de individuos de la población deben ser seleccionado para convertirse en padres de nuevas soluciones. Esta selección normalmente se realiza de manera de favorecer a los individuos que posean mejor fitness, de forma que el material genético de los mejores individuos pase a las siguientes generaciones. Dos tipos de operadores de selección utilizados comúnmente son:

- **Selección elitista:** Selecciona directamente los mejores individuos de la población.

  Suponga que los siguientes individuos de una población y su correspondiente fitness:

  $$s_1 = \boxed{1}\boxed{0}\boxed{0}\boxed{0}\boxed{0}\ Fitness : 2800$$

  $$s_2 = \boxed{0}\boxed{1}\boxed{0}\boxed{0}\boxed{0}\ Fitness : 2500$$

  $$s_3 = \boxed{0}\boxed{0}\boxed{1}\boxed{0}\boxed{0}\ Fitness : 1000$$

  $$s_4 = \boxed{0}\boxed{0}\boxed{0}\boxed{1}\boxed{0}\ Fitness : 1200$$

  $$s_5 = \boxed{0}\boxed{0}\boxed{0}\boxed{0}\boxed{1}\ Fitness : 500$$

  La selección elitista selecciona los mejores individuos, por lo tanto se seleccionarán:

  $$s_1 = \boxed{1}\boxed{0}\boxed{0}\boxed{0}\boxed{0}\ Fitness : 2800$$

  $$s_2 = \boxed{0}\boxed{1}\boxed{0}\boxed{0}\boxed{0}\ Fitness : 2500$$

- **Selección de la ruleta:** Selecciona individuos con una probabilidad proporcional a su fitness, de manera que los individuos con mejor fitness tendrán más probabilidad de ser seleccionados. Este es un tipo de selección probabilistica proporcional.

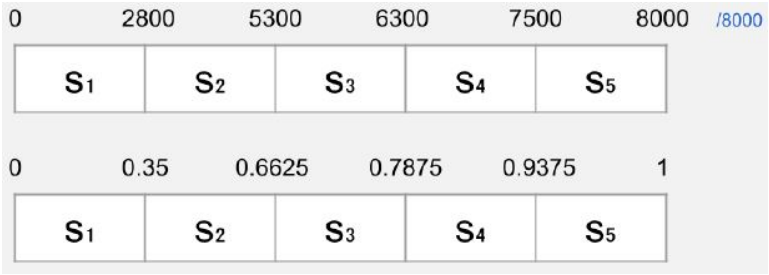
  Utilizando la población descrita anteriormente:

  - Primeramente se debe calcular una probabilidad para la selección de cada uno de los individuos, para esto calculamos la suma de los fitness de todos los individuos de la población:

$$sumaFitness = 2800 + 2500 + 1000 + 1200 + 500 = 8000$$

- Dividamos el intervalo de 1 a 8000, sumando acumulativamente el fitnes de cada solución hasta llegar al fin del intervalo:

- Luego, dividimos todo por la suma del fitness para normalizar a un intervalo $[0, 1]$



- Luego para seleccionar individuos se deben utilizar números aleatorios(supongamos, 0,47 y 0,83).

$$0{,}47 \rightarrow s_2 = \boxed{0}\boxed{1}\boxed{0}\boxed{0}\boxed{0} \; Fitness : 2500$$

$$0{,}83 \rightarrow s_4 = \boxed{0}\boxed{0}\boxed{0}\boxed{1}\boxed{0} \; Fitness : 1200$$

5.5.1. ¿Cuál es el objetivo de los operadores de selección de padres?

En general, los operadores de selección favorecen la intensificacion de la búsqueda alrededor de las mejores soluciones. La selección elitista genera una mayor presión de selección ya que selecciona sólo los mejores individuos intensificando la búsqueda con mayor intensidad que la selección de la ruleta, la cual permite selccionar con una probabilidad menor los peores individuos de la población.

- ¿Porqué podría ser beneficioso que los peores individuos sean seleccionados para ser padres?

  Permitir que todos los individuos de la población sean seleccionados con una probabilidad que depende de su fitnes, permite al algoritmo mantener ciertos niveles de experiencia, de esta manera se evitará que el algoritmo se estanque rápidamente generando soluciones muy parecidas a las mejores.

5.6. Cruzamiento y Mutación

5.6.1. Cruzamiento

Los operadores de cruzamiento combinan el material genético de dos individuos con el objetivo de combinar sus mejores genes para, si es posible, generar una solución que posee un mejor fitness. Es posible aplicar cruzamiento a más de dos soluciones padres, el nombre de estos operadores que combinan el material genético de n soluciones es operadores de recombinación. Los operadores de cruzamiento son por lo tanto un subconjunto de los operadores de recombinación.

Tres tipos de operadores de cruzamiento utilizados comúnmente son:

- **Cruzamiento en un punto:** Este operador selecciona un punto en las soluciones a cruzar y se generan dos individuos combinando las secciones que se generan a partir de este punto de corte.

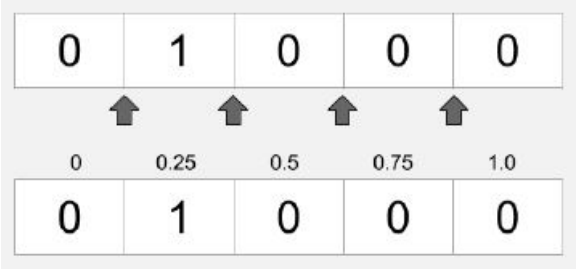
  Suponga los siguientes individuos padres:

$$s_1 = \boxed{0}\boxed{1}\boxed{0}\boxed{0}\boxed{0}$$

$$s_2 = \boxed{0}\boxed{0}\boxed{0}\boxed{1}\boxed{0}$$

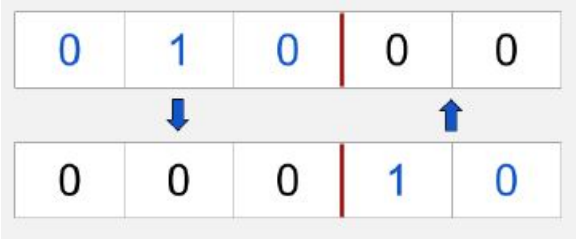
  Apliquemos el operador de cruzamiento en un punto utilizando el número aleatorio 0.58

  - Primero seleccionamos un punto de corte al azar, utilicemos la primera solución para ejemplificar:

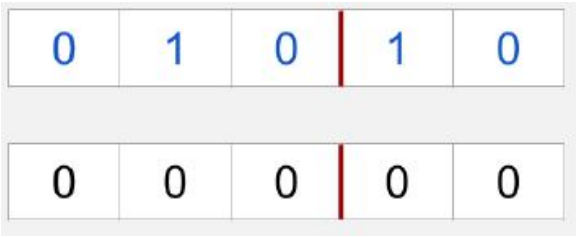


- Con el número aleatorio 0.58 seleccionamos un punto:

- El operador intercambia las secciones de las soluciones generadas por el punto de cruce seleccionado:



- Las soluciones hijas generadas corresponden a las dos combinaciones de las secciones:

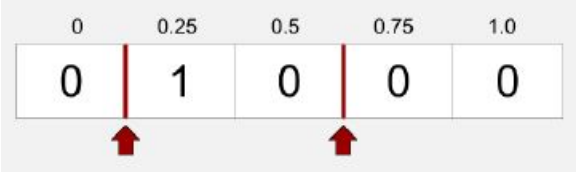


- **Cruzamiento en dos puntos:** Este operador selecciona aleatoriamente dos puntos en las soluciones a cruzar y se generan dos individuos

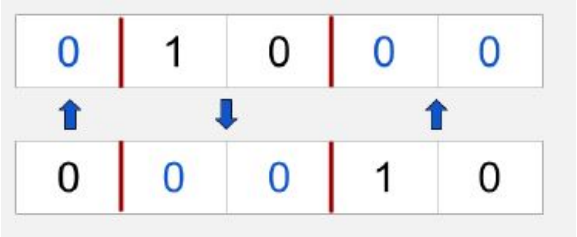
combinando las secciones que se generan a partir de estos puntos de corte.

Apliquemos el operador de cruzamiento en dos puntos utilizando los números aleatorios 0.58, 0.21

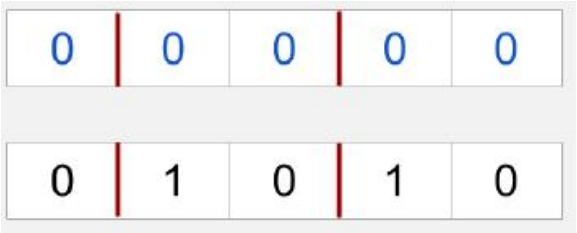
- Seleccionar en base a los números aleatorios dos puntos de corte:



- El operador intercambia las tres secciones obtenidas a partir de los puntos de corte:



- Las soluciones hijas generadas corresponden a las dos combinaciones de las secciones:



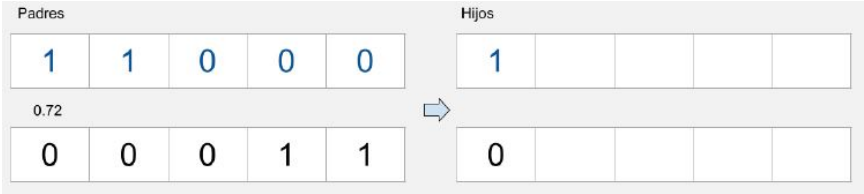
- **Cruzamiento uniforme:** Este operador cruza aleatoriamente a los padres en todos los puntos de corte. Para esto, se define una probabilidad de cruzamiento p_{cruz} que define la probabilidad de que el valor de una variable se pase a un hijo o a otro. Suponga los siguientes individuos padres:

$$s_1 = \boxed{1}\boxed{1}\boxed{0}\boxed{0}\boxed{0}$$

$$s_2 = \boxed{0}\boxed{0}\boxed{0}\boxed{1}\boxed{1}$$

Aplique el operador de cruzamiento uniforme utilizando los números aleatorios 0.72, 0.41, 0.25, 0.96, 0.13 y probabilidad de cruzamiento $p_{cruz} = 0.5$.

- Se generarán dos hijos decidiendo que valores utilizar en base a la probabilidad de cruzamiento, en este caso si el número aleatorio obtenido es mayor que p_{mut} el valor del primer padre se copiará en el primer hijo y si es menor se copiará el valor del segundo padre al primer hijo. Comencemos por la primera variable:



- Luego continuamos con la segunda:

- Y continuamos hasta completar los nuevos individuos:

5.6.1.1. ¿Cuál es el objetivo principal de los operadores de cruzamiento?

En general, los operadores de cruzamiento poseen el objetivo de explotar la información que se tiene del espacio de búsqueda a debido a que intentan encontrar la mejor combinación de los mejores individuos de la población. Desde el punto de vista de las *restricciones* de un problema, el operador de cruzamiento causa muchas veces la violación de restricciones, por lo que esto debe ser considerado en su diseño de manera que se define si las restricciones se penalizarán en la función de evaluación o si asegurara que el operador generara siempre soluciones factibles.

5.6.2. Mutación

El operador de mutación cambia aleatoriamente los nuevos individuos obtenidos en el cruzamiento. Esto se realiza definiendo una probabilidad para que la información genética de los individuos cambie aleatoriamente. Tal como sucede en la naturalez, normalmente esta probabilidad se define con un valor no muy alto para generar cambios pequeños que permitan guiar la búsqueda. Un ejemplo de operador de mutación es el operador **bit flip** el cual cambia cada gen (variable) de la solución con una probabilidad definida p_{mut}.

Considere la siguiente solución:

0	0	0	1	1

Aplique el operador de mutación bit flip con $p_{mut} = 0{,}1$ a la solución entregada utilizando los números aleatorios 0.39, 0.08, 0.65, 0.84 y 0.12.

5.6.2.1. ¿Cuál es el objetivo principal de los operadores de mutación?

La mutación dentro del algoritmo genético tiene el objetivo de explorar el espacio de búsqueda, insertando nuevos genes de manera aleatoria y aumentando la diversidad de los individuos de la población. La probabilidad de mutación define que tan explorativo será el comportamiento del algoritmo, si la probabilidad es muy alta, muchos genes cambiarán su valor, mientras que si es demasiada baja puede que no se produzcan cambios.

5.6.3. Selección de padres en cruzamiento/mutación

Al final del proceso de selección, cruzamiento y mutación, se obtendrá una población de individuos hijos los cuales deberán ser evaluados. Para continuar a la siguiente generación es necesario que se seleccionen los individuos que integrarán la siguiente población. Al igual que en la selección realizada con los padres se pueden utilizar diversos métodos para seleccionar los individuos, por ejemplo:

- Selección elitista
- Selección de la ruleta

Para definir desde cuales individuos seleccionaremos los integrantes de la nueva población, tomamos prestada la notación en otros algoritmos relacionados con los genéticos llamados *estrategias evolutivas*. Definamos μ como el número de individuos en un generación y λ como el número de hijos generados en una generación. Se pueden definir los esquemas de la selección:

- (μ, λ): de una generación de μ individuos se generan λ y la nueva población se selecciona sólo desde los nuevos λ individuos. Note que por lo tanto, debido a que la población debe ser de tamaño μ, el valor de λ debe ser siempre $\mu \leq \lambda$. En este caso, los individuos de la población anterior no pueden pasar directamente de una generación a la siguiente.
- $(\mu + \lambda)$: De una generación de μ individuos se generan λ nuevos individuos y la nueva población se selecciona desde el conjunto de los individuopos en la población original y los nuevos individuos $(\mu + \lambda)$. En este caso, los individuos pueden ser parte de la población por varias generaciones mientras sean seleccionados. En algunos casos es posible definir una cantidad de generaciones máximas que los individuos pueden permanecer en la población (edad) de manera de no permitir que la búsqueda se estanque en las mismas soluciones.

Suponga la siguiente población y los respectivos hijos generados:

Población G:

$s_1 =$	1	0	0	0	0	fitness: 2800
$s_2 =$	0	1	0	0	0	fitness: 2500
$s_3 =$	0	0	1	0	0	fitness: 1000
$s_4 =$	0	0	0	1	0	fitness: 1200
$s_5 =$	0	0	0	0	1	fitness: 500

Nuevos individuos G':

$s_6 =$	0	0	0	0	0	fitness: 0
$s_7 =$	1	0	0	0	1	fitness: 3200
$s_8 =$	0	0	1	1	1	fitness: 2700
$s_9 =$	0	0	1	0	1	fitness: 1500

Seleccione los individuos de la nueva población utilizando (5+4) elitista. Note que en este caso $\mu = 5$ y $\lambda = 4$, debido a que la selección elitista define seleccionar los mejores individuos, la nueva población será:

$s_1 =$	1	0	0	0	1	fitness: 3200
$s_2 =$	1	0	0	0	0	fitness: 2800
$s_3 =$	0	0	1	1	1	fitness: 2700
$s_4 =$	0	1	0	0	0	fitness: 2500
$s_5 =$	0	0	1	0	1	fitness: 1500

Capítulo 6

Simulated Annealing

6.1. Introduccion

En los capítulos anteriores, nos hemos enfocado en problemas que se caracterizan por su dificultad y gran espacio de búsqueda. Estos problemas: el problema de la mochila y el problema del vendedor viajero, pertenecen al grupo de los problemas que en teoría de la complejidad se clasifican como NP, para los cuales no existen algoritmos eficientes para resolverlos a optimalidad. La figura 6.1 ilustra la clasificación de problemas en términos de su complejidad.

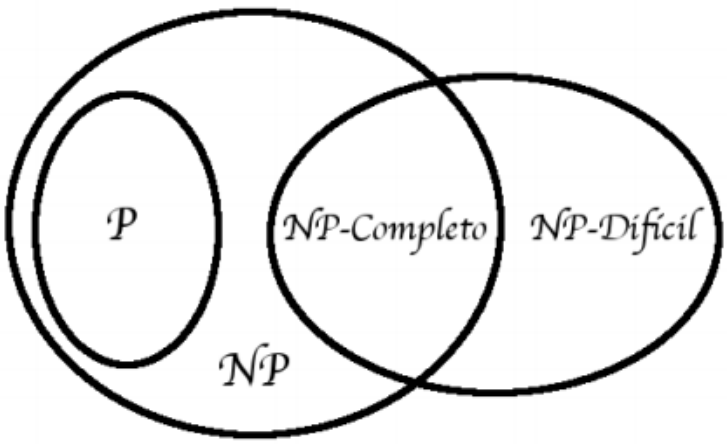


Figura 6.1: Clasificación de problemas en base a su complejidad.

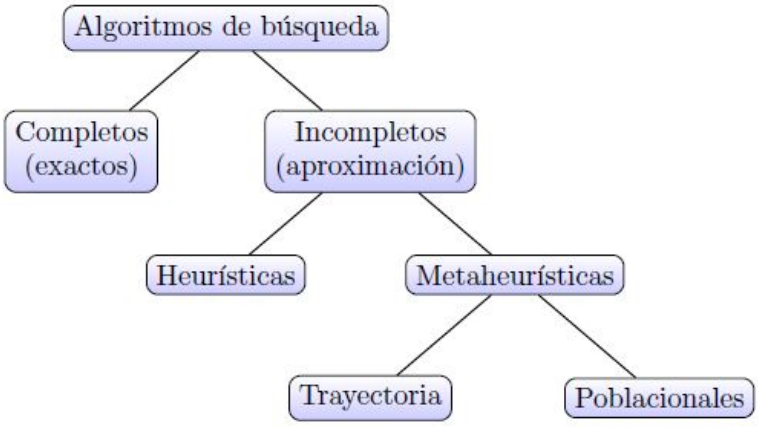
En los capitulos anteriores aprendimos sobre las heurísticas, ellas definen reglas para construir soluciones y modificarlas de manera sencilla y rápida. Aplicando iterativamente heurísticas de perturbación, es posible moverse de una solución a otra en el espacio de búsqueda, este proceso se puede repetir hasta que no es posible continuar el movimiento debido a que se ha obtenido una buena solución llamada *óptimo local.* Este proceso de búsqueda heurístico es llamado búsqueda local. En muchos casos, este tipo de búsqueda no es suficiente para encontrar soluciones de buena calidad a problemas complejos. Para poder resolver de manera relativamente efectiva estos problemas es que existen las **metaheurísticas**.

Las metaheurísticas son métodos generales que guían la búsqueda de soluciones aplicando heurísticas y balanceando la *exploración e explotación* de la búsqueda para evitar el estancamiento en óptimos locales. A diferencia de las heurísticas, las metaheurísticas son métodos generales, que por lo tanto, se pueden aplicar a diversos problemas. Su aplicación muy comunmente requiere de la implementación de heurísticas para generar y modificar soluciones, las cuales son utilizadas en base a una estrategia de búsqueda. Las metaheurísticas implementan un proceso de búsqueda global, moviéndose a través de diferentes óptimos locales en el espacio de búsqueda y evitando quedar atrapadas en ellos como sucede en la búsqueda local.

La mayoría de las metaheurísticas son procedimientos *estocásticos*, esto quiere decir que utilizan aleatoriedad para tomar decisiones durante el proceso de búsqueda.

Las metaheurísticas involucran la aleatoriedad en el proceso de búsqueda de manera de permitir *explorar* diferentes áreas del espacio de búsqueda. Esto es especialmente importante cuando no se tiene información sobre las áreas que pueden contener soluciones de buena calidad, y por consiguiente, la mejor estrategia es explorar aleatoriamente para obtener información lo más diversa posible del espacio de búsqueda y luego pasar a una fase de explotación de esta información.

Existen muchas formas de clasificar las metaheurísticas, la siguiente figura ilustra los dos tipos de metaheurísticas que definiremos en este momento: metaheurísticas de **trayectoria** y **poblacionales**.



La clasificación propuesta se basa en la cantidad de soluciones que la metaheurística maneja durante el proceso de búsqueda.

6.1.1. Metaheurísticas de trayectoria

Son métodos de búsqueda global, los cuales definen una estrategia de búsqueda que parte de **una solución inicial**, y aplica iterativamente heurísticas perturbativas (movimientos) para transformar esta solución, definiendo una trayectoria en el espacio de búsqueda.

Las metaheurísticas de trayectoria manejan sólo una solución a la vez (solución actual), en cada iteración la solución actual es modificada aplicando un movimiento definido, generando una nueva solución de manera que se define una trayectoria en el espacio de búsqueda, tal como lo ilustra la figura 6.2. El proceso de búsqueda termina al cumplirse una condición o criterio de término.

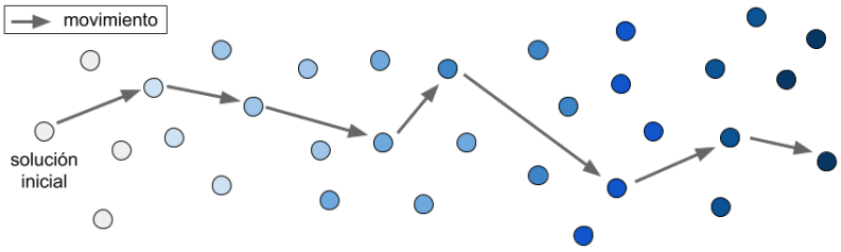


Figura 6.2: Ilustración de una trayectoria de búsqueda

6.1.2. Metaheurísticas de población

Son métodos de búsqueda global los cuales definen una estrategia de búsqueda que mantiene un grupo de soluciones, las cuales son iterativamente modificadas y refinadas de manera que estas soluciones mejoran en conjunto a través de las iteraciones.

Las metaheurísticas de población manejan varias soluciones, grupo de soluciones, el cual se denominan comunmente población. La población representa la información que la metaheurística tiene del espacio de búsqueda y debido a que una población contiene más de una solución, estas metaheurísticas tienen acceso a más áreas del espacio de búsqueda a la vez que las metaheurísticas de trayectoria.

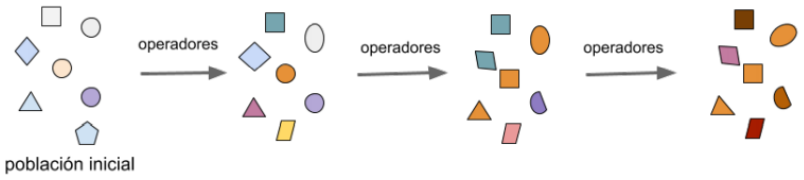


Figura 6.3: Ilustración de la evolución de una población.

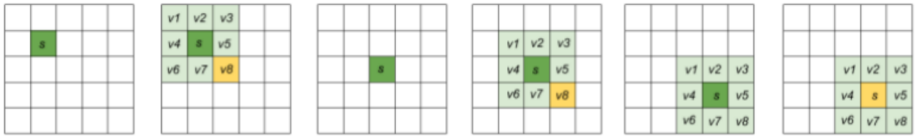
Al igual que en las metaheurísticas de trayectoria, las soluciones de la poblacion son iterativamente modificadas aplicando operadores. La población en cada iteración como ilustra la figura 6.3. El tamaño de la población puede ser fijo o variable y las soluciones pueden permanecer en la población con y sin modificaciones, o pueden ser reemplazadas por nuevas soluciones. El proceso de búsqueda termina cuando se cumple una condición o criterio de término definido.

6.2. Simulated Annealing - Definición

Simulated annealing o recocido simulado es una metaheurística de trayectoria inspirada en el proceso de templado de metales. Simulated annealing evita el estancamiento de un óptimo local propio de las búsqueda localees, implementando un criterio de aceptación de nuevas soluciones inspirados en el manejo de la temperatura en el templado de metales.

El templado de metales es el proceso que se utiliza para alterar las propiedades físicas de los metales. el metal se calienta a una alta temperatura y luego se reduce esta de manera controlada, lo cual permite que ciertas estructuras moleculares se formen en el mental. Recordemos el ejemplo de

los capítulos anteriores para ilustrar un procedimiento de búsqueda local. La búsqueda local siempre cambia la solución actual por la mejor solución del vecindario, una vez que ya no encuentra una solución mejor que la actual, la búsqueda se detiene (estancamiento en un óptimo local) y no podrá continuar buscando otras soluciones que pudieran ser mejores.



Para evitar el estancamiento, Simullated annealing define un criterio llamado *criterio de metrópolis* el cual permite aceptar soluciones que no son necesariamente las mejores del vecindario. Utilizando este criterio, el algoritmo puede seguir una trayectoria diferente a la de una búsqueda local, permitiendo la exploración del espacio de búsqueda. Para ilustrar el procedimiento, la figura muestra el pseudocódigo de Simulated Annealing. El algoritmo comienza generando una solución inicial que es punto de partida del proceso de búsqueda, la variable s corresponderá a la solución actual, y la variable s^* representa la mejor solución encontrada por el algoritmo.

Se simula el proceso de templado definiendo una variable que representa la temperatura, se le asisgna un vlaor inicial alto para representar el momento en que el metal ha alcanzado su temperatura máxima.

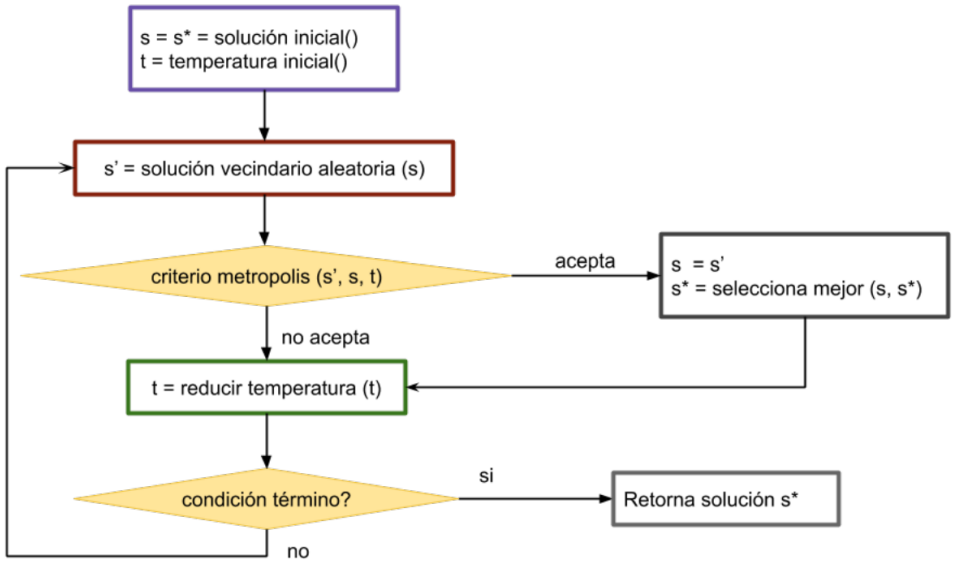


Figura 6.4: Pseudocódigo de Simulated Annealing

6.3. Criterio de metrópolis

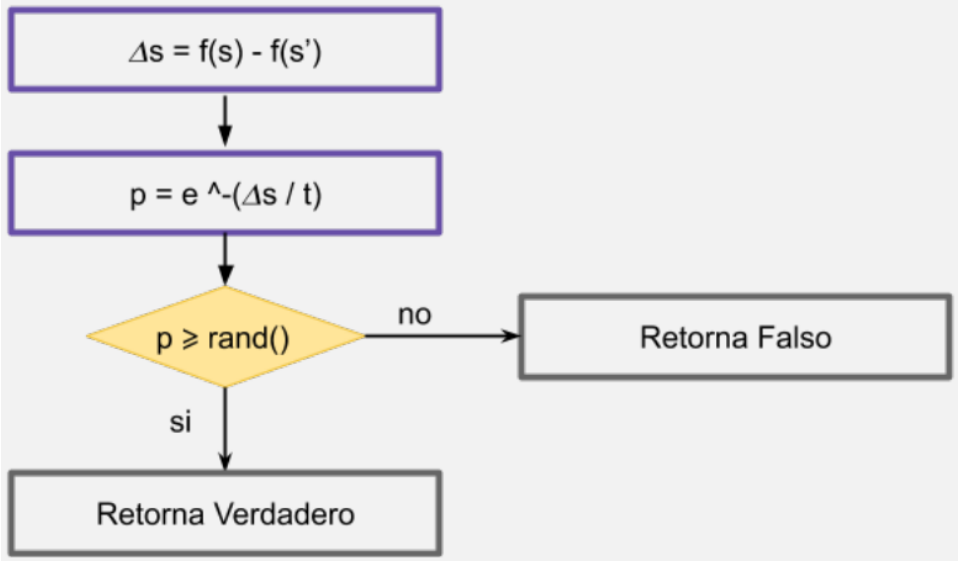
El criterio de metrópolis define una probabilidad para aceptar una nueva solución. Esta probabilidad depende de la función objetivo de la nueva solución $f(s')$, la función objetivo de la solución actual $f(s)$ y la temperatura:

$$probabilidad = e^{-\frac{\Delta s}{t}}$$

Donde:

- Minimización: $\Delta s := f(s') - f(s)$
- Maximización: $\Delta s := f(s) - f(s')$

Calculada esta probabilidad, el algoritmo utiliza aleatorio para determinar si la solución será aceptada o no. El siguiente pseudocódigo ilustra el criterio:



Veamos un ejemplo, se tiene la siguiente solución actual para el problema de la mochila:

$s =$ 

valor: 4500

Al generar una solución del vecindario se obtiene la siguiente solución:

$s' =$ 

valor: 4200

¿Cúal es la probabilidad de que esta solución sea aceptada como nueva solución actual en un Simulated Annealing con temperatura actual de $t = 300$ y $t = 600$?

- Primero calculamos Δs, cómo el problema de la mochila es un problema

de maximización:

$$\Delta s = f(s) - f(s') = 4500 - 4200 = 300$$

- Calculamos la probabilidad con $t = 300$:

$$p = e^{-\frac{\Delta s}{t}} = e^{-\frac{300}{300}} = e^{-1} = 0{,}368$$

  Con $t = 300$ la solución será aceptada con un 36,8 % de probabilidad.

- Calculamos la probabilidad con $t = 600$:

$$p = e^{-\frac{\Delta s}{t}} = e^{-\frac{300}{600}} = e^{-0{,}5} = 0{,}606$$

  Con $t = 600$ la solución será aceptada con un 60,6 % de probabilidad.

En cada iteración de Simulated Annealing, la temperatura es reducida, siguiendo un esquema de enfriamiento. Note que si la temperatura es alta, la probabilidad obtenida por el criterio de metrópolis será alta, por lo cual el algoritmo aceptará soluciones de peor calidad que la actual con más facilidad. Si la temperatura tiene un valor pequeño, la probabilidad también lo será, por lo cual el algoritmo tenderá a no aceptar soluciones que empeoran la calidad de la solución actual.

6.4. Esquema de enfriamiento

El esquema de enfriamiento define como se reducirá en cada iteración la temperatura utilizada por el criteriode metrópolis para calcular la probabilidad de aceptación de nuevas soluciones. Un ejemplo es el esquema geométrico:

$$t = \alpha * t$$

Para visualizar como cambia la probabilidad de aceptar soluciones la figura 6.5 muestra el valor de la temperatura a medida que avanzan las iteraciones del algoritmos. Para este ejemplo se utiliza un enfriamiento geométrico con $\alpha = 0{,}85$ y se define una temperatura inicial de 1000. Para ilustrar la probabilidad obtenida, se calcula la probabilidad con el criterio de metrópolis suponiendo una solución s' peor que la actual s con $\Delta s = 300$

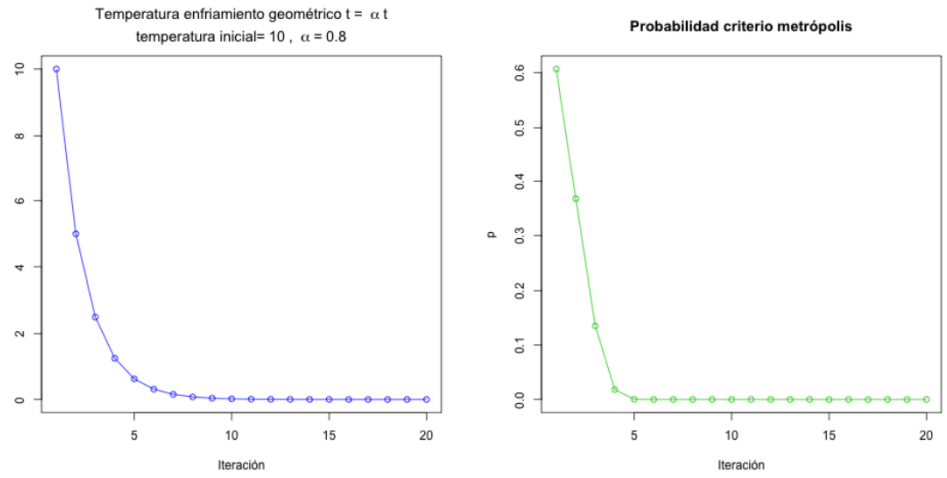
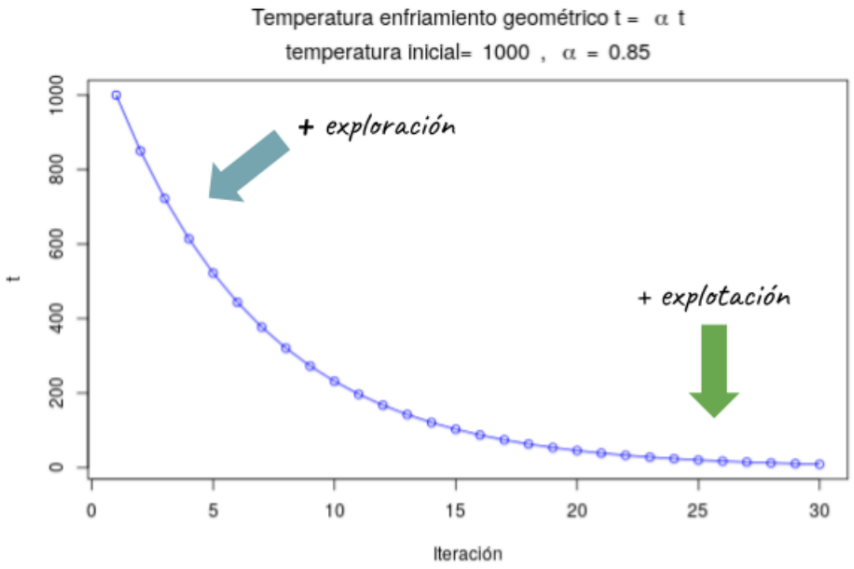


Figura 6.5: Ejemplo de temperatura y probabilidad de aceptación Simulated Annealing

Piense un momento sobre como afecta este comportamiento de aceptación de soluciones a la exploración y la explotación del espacio de búsqueda.

Simulated Annealing comienza con una temperatura inicial alta, por lo tanto al inicio de la búsqueda el algoritmo aceptará soluciones que son peores que la solución actual con una alta probabilidad. Esto por lo tanto privilegia la exploración del espacio de búsqueda, permitiendo acceder a diferentes áreas al escapar constantemente de las trayectorias de búsquda local.



Cuando la temperatura es reducida, las posibilidades de explorar nuevas áreas del espacio de búsqueda se reducen y el algoritmo comienza a aceptar solamente soluciones que son mejores que la solución actual. De esta manera, se privilegia la explotación de la información siguiendo trayectorias como la de una búsqueda local.

6.5. Ejemplo

Suponga que deseamos resolver la siguiente instancia del problema del vendedor viajero. Dada la siguiente matriz de la distancias entre ciudades:

	A	B	C	D	E
A	-	8	4	9	9
B	8	-	6	7	10
C	4	6	-	5	6
D	9	7	5	-	4
E	9	10	6	4	-

Encuentre un tour que pase por todas las ciudades y retornando finalmente a la ciudad inicial que minimice la distancia total viajada. Suponga que decidimos aplicar *Simulated annealing* con las siguientes características:

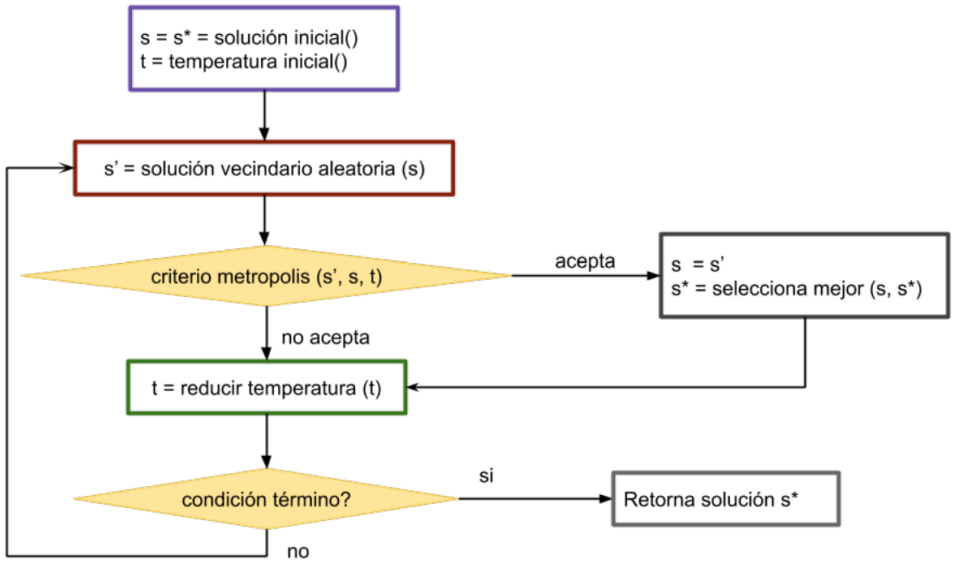
- **Solución inicial:** Aleatoria

- **Temperatura inicial:** 10
- **Criterio de aceptación**: Metrópolis
- **Esquema de enfriamiento**: Geométrico, $\alpha = 0{,}4$
- **Movimiento**: Swap
- **Condición de término:** Temperatura mínima = 0,2

Suponga que la ejecución del algoritmo que se tiene la siguiente secuencia ordenada de números aleatorios:

{ 0.11, 0.20, 0.07 0.44, 0.36, 0.25, 0.72, 0.43, 0.54, 0.97, 0.03, 0.59, 0.36, 0.41, 0.62, 0.29, 0.85, 0.70 }

Sigamos el algoritmo descrito anteriormente:



A continuación se presenta un resumen de la ejecución de este algoritmo. Puede encontrar un desarrollo paso a paso de este ejercicio en el siguiente recuadro

```
//Solución inicial
s = s* = A - B - C - D - E - (A), costo = 32
//Temperatura inicial
t: 10

---------------------------------------------------------------------
ITERACION 1
---------------------------------------------------------------------
//Aplicación aleatoria de swap
s'= A - D - C - B - E - (A), costo = 39  // Es peor que solución s

//Criterio de metrópolis
p = e^{-7/10} = 0.5
rand() = 0.43 -> acepta!

// Cambiar solución
s = s'

//Enfriar temperatura
t = 0.4 * 10 = 4

---------------------------------------------------------------------
ITERACION 2
---------------------------------------------------------------------
s  = A - D - C - B - E - (A), costo = 39

//Aplicación aleatoria de swap
s' =  A - D - E - B - C - (A), costo = 33 // Es mejor que solución s

// Cambiar solución
s = s* = s'

//Enfriar temperatura
t = 0.4 * 4 = 1.6

---------------------------------------------------------------------
ITERACION 3
---------------------------------------------------------------------
s  = A - D - E - B - C - (A), costo = 33

//Aplicación aleatoria de swap
s' =  E - D - A - B - C - (E), costo = 33 // Es peor que solución s

//Criterio de metrópolis
p = e^{-0/1.6} = 1.0
rand() = 0.36 -> acepta!

//Enfriar temperatura
t = 0.4 * 1.6 = 0.64
```

```
-----------------------------------------------------------------
ITERACION 4
-----------------------------------------------------------------
s  = E - D - A - B - C - (E), costo = 33

//Aplicación aleatoria de swap
s' =  E - D - B - A - C - (E), costo = 29 // Es mejor que solución s

// Cambiar solución
s = s* = s'

//Enfriar temperatura
t = 0.4 * 0.64 = 0.256

-----------------------------------------------------------------
ITERACION 5
-----------------------------------------------------------------
s  = E - D - B - A - C - (E), costo = 29

//Aplicación aleatoria de swap
s' =  E - C - B - A - D - (E), costo = 33 // Es peor que solución s

//Criterio de metrópolis
p = e^{-0/0.256} = 1.6 \cdot 10^{-7}
rand() = 0.7 -> NO acepta!

//Enfriar temperatura
t = 0.4 * 1.6 = 0.1024 // Criterio de termino t < 0.2!

-----------------------------------------------------------------
RETORNA SOLUCIÓN: s* = E - D - B - A - C - (E), costo = 29
-----------------------------------------------------------------
```