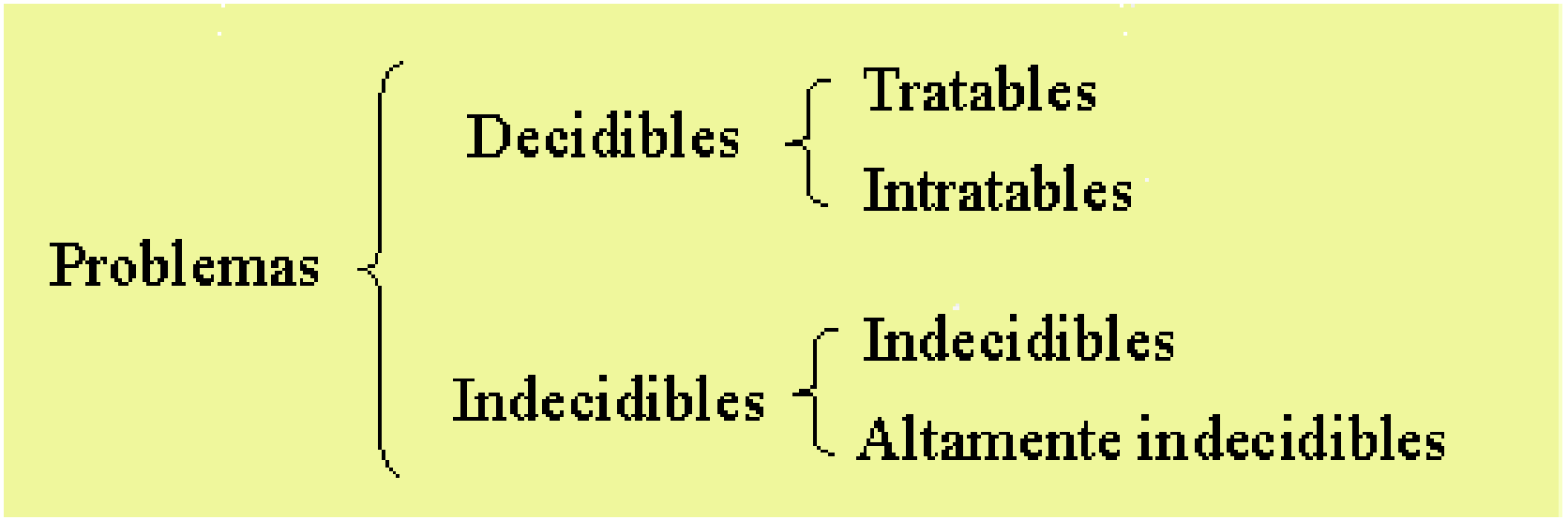
TEMA 4: CLASES DE COMPLEJIDAD DE ALGORITMOS

Objetivo: El alumno describirá los tipos de complejidad P, NP y NPC. Así mismo, será capaz de entender el planteamiento del problema ¿P=NP?

4.1 Generalidades

- Clasificación de tipos de problemas computacionales basado en la posibilidad de resolverlos



4.1 Generalidades

- **Problema decidible**

Un problema decidible es aquel, en el que existe un algoritmo que lo puede resolver.

- Tratable

  La solución del problema se puede obtener en un tiempo razonable.

- Intratable

  A pesar de ser un tipo de problema que cuenta con una solución, obtener ésta implica un tiempo no razonable.

4.1 Generalidades

- **Problema indecidible**

En estos problemas no existe un algoritmo (solución a través de un modelo matemático) que lo pueda resolver.

- Indecidible

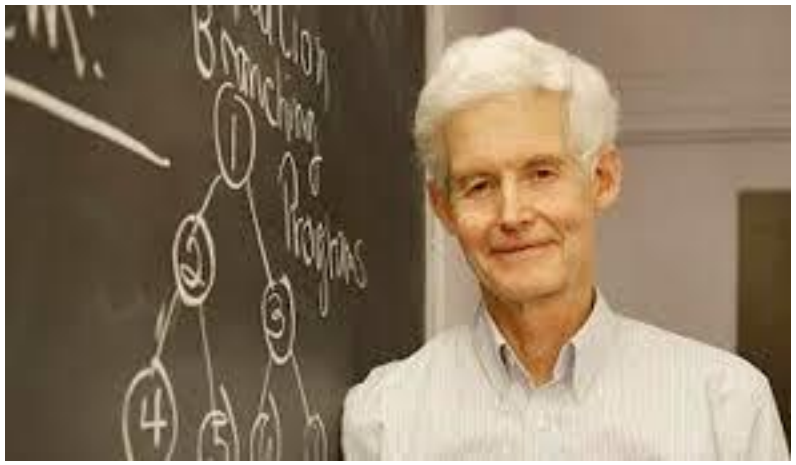
  Se pueden tener aproximaciones razonables en la solución que se puede obtener.

- Altamente indecidible

  Poco probable que algún día se puedan resolver y las aproximaciones suelen ser acotadas.

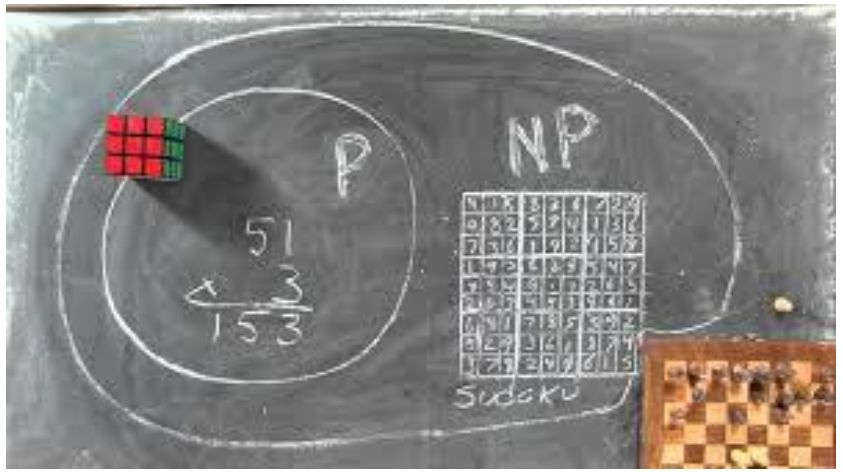
4.2 Clases de complejidad

- Entender las categorías de complejidad de los problemas computacionales, así como el tener claro si puede existir un algoritmo que lo resuelva o no, es la base para las clases de complejidad.
- En 1971 el matemático Stephen Cook, demostró matemáticamente que **problema de satisfacibilidad booleana** es un problema NP.



4.2 Clases de complejidad

- A partir de esta demostración, surge la idea de clases de complejidad para los problemas computacionales.
- Estas clases de complejidad se denominan P, NP y NPC.



4.2.1 Problemas P

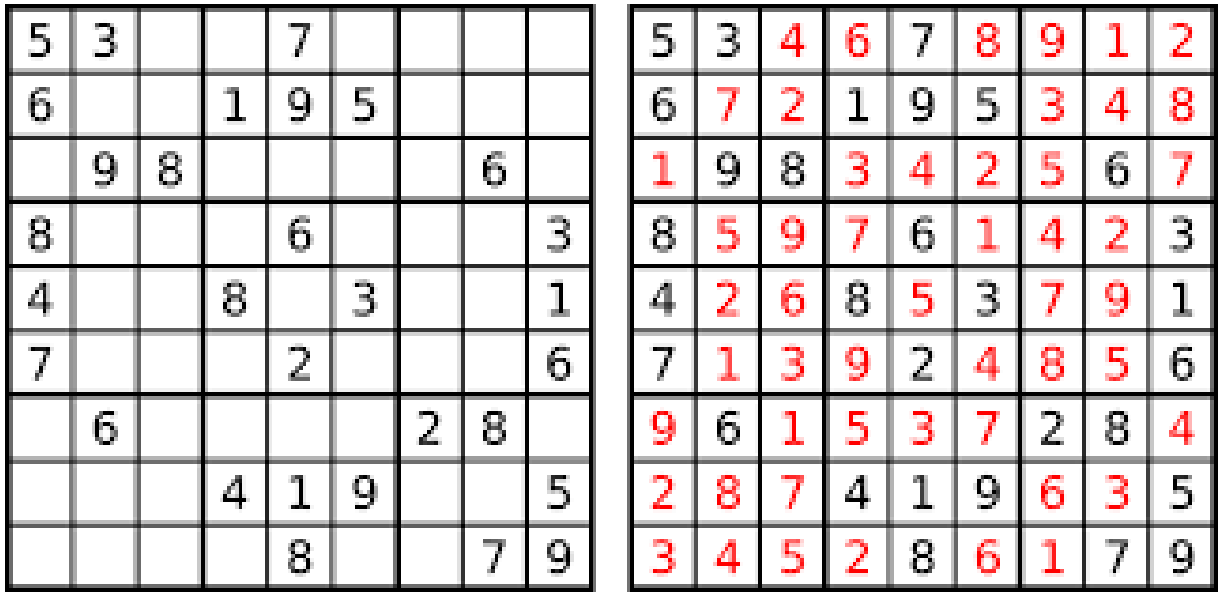
- La clase de complejidad P se compone de los problemas, en los que existe un algoritmo que los puede resolver *eficientemente.*
- Son algoritmos cuya función de complejidad está acotada por algún polinomio.
- En otras palabras, se resuelven en tiempo polinomial en una Máquina de Turing determinística.

4.2.2 Problemas NP

- La clase de complejidad NP es el conjunto de problemas que se pueden resolver en tiempo polinomial no determinístico.
- La solución de estos problemas puede ser comprobada en tiempo polinomial.

4.2.3 Problemas NP-Completos

- La clase de complejidad NP-completa es el conjunto de problemas más difíciles dentro del conjunto NP. Son los problemas que tienen menos posibilidades de encontrarse en el conjunto P.



5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

Ejemplos

P

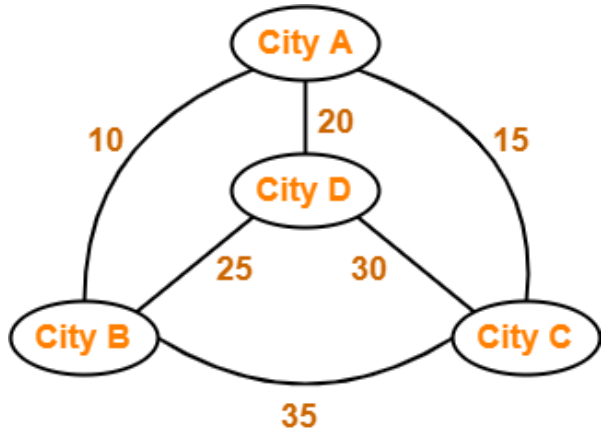
- Búsqueda lineal
- Operaciones matemáticas

NP

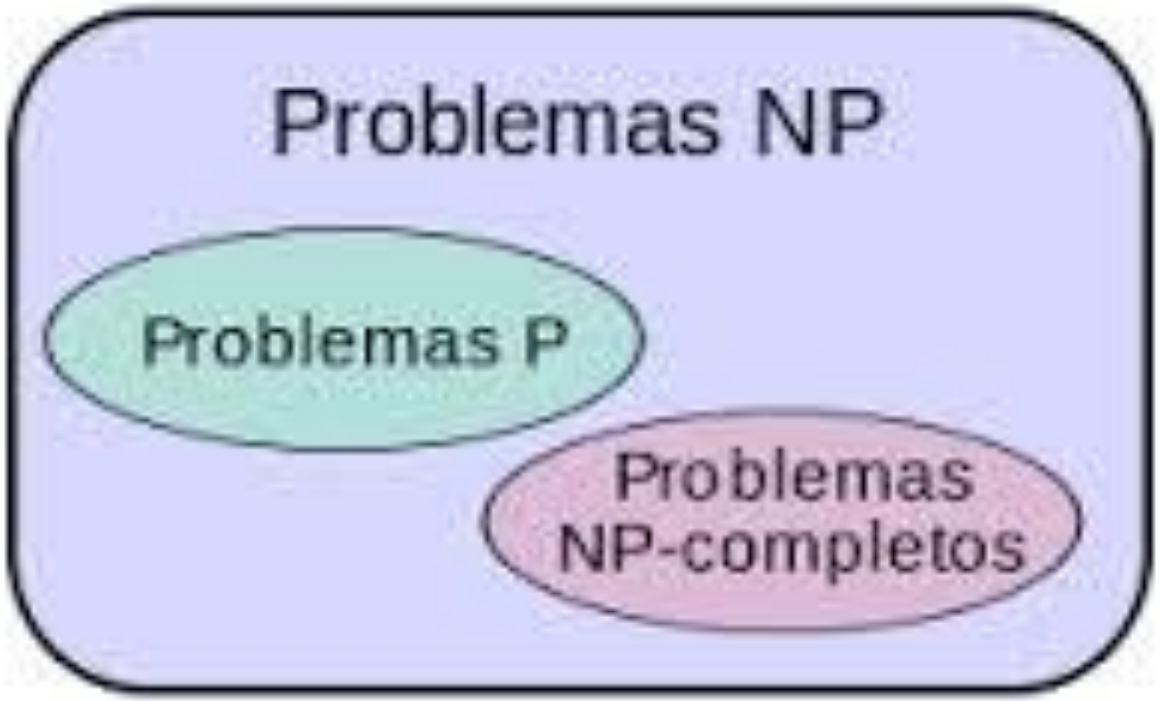
- Cubo de Rubik

NP-Completos

- El agente viajero
- Sudoku



4.2.4 ¿Es P = NP?



4.2.4 ¿Es P = NP?

- En la disciplina de las ciencias de la computación es el problema más importante que aún no se ha podido resolver.
- Es uno de los problemas matemáticos más importantes en la actualidad.
- Millenium problems

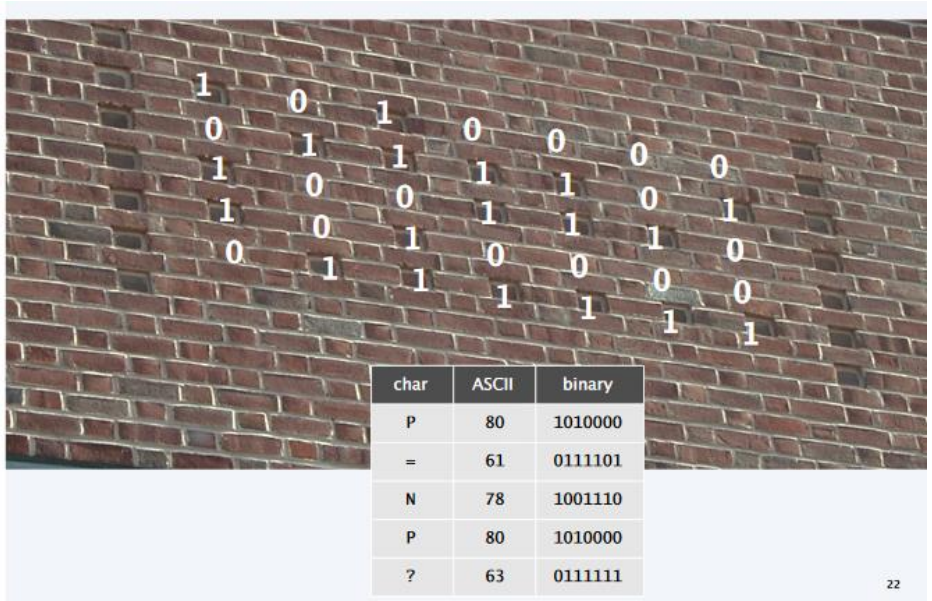
https://www.claymath.org/millennium-problems

4.2.4 ¿Es P = NP?

- ¿Existe un algoritmo que pueda resolver de manera eficiente los problemas de tipo NP?
- Eficiente es una solución determinista obtenida en tiempo polinomial.
- Matemáticamente, los problemas NP son equivalentes.

4.2.4 ¿Es P = NP?

Una de las preguntas abiertas más importantes en la actualidad es descubrir si estas clases son diferentes o no ¿P=NP?



4.2.4 ¿Es P = NP?

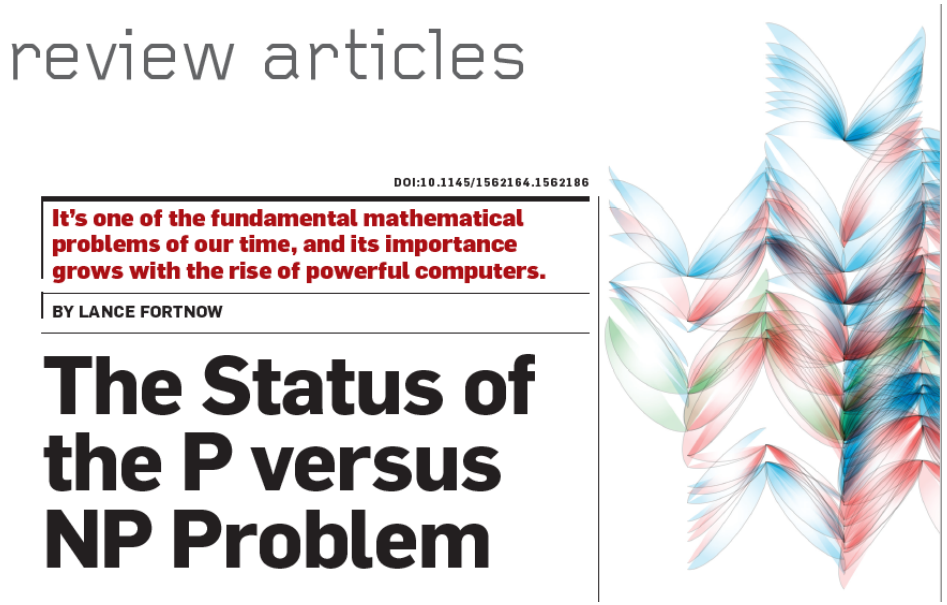
Proof by contradiction:
If P = NP
P-NP = 0
P(1-N) = 0
P = 0 or N = 1
We know P ≠ 0 and N ≠ 1
Therefore P ≠ NP

4.2.4 ¿Es P = NP?

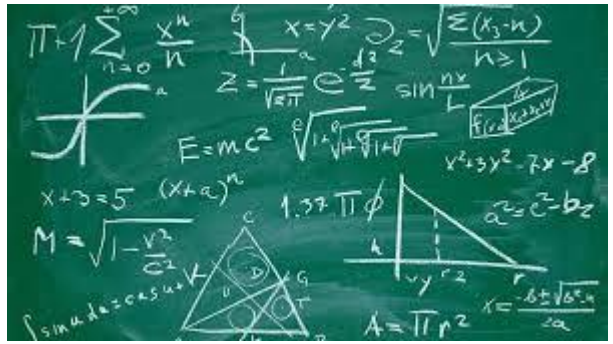
- La tecnología actual así como el conocimiento que se tiene es resultado de cientos de años de conocimiento
- Hasta este punto, hay pocas posibilidades de resolverlo.
- ¿Cuáles son las implicaciones de P=NP o de P≠NP?

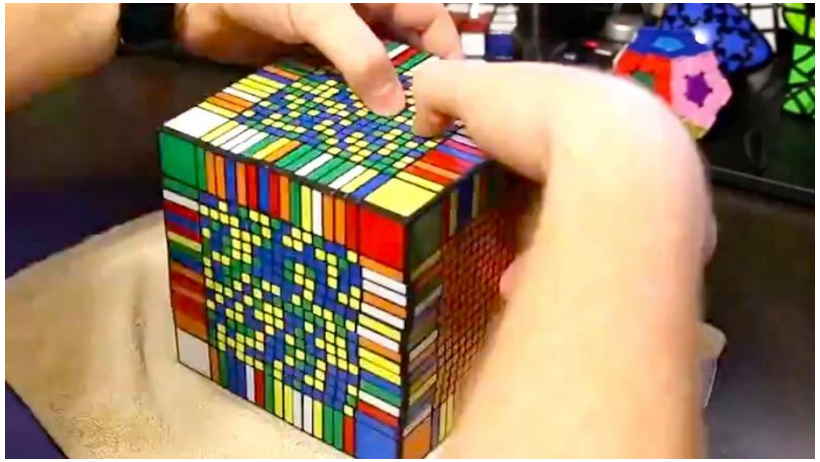
4.2.4 ¿Es P = NP?

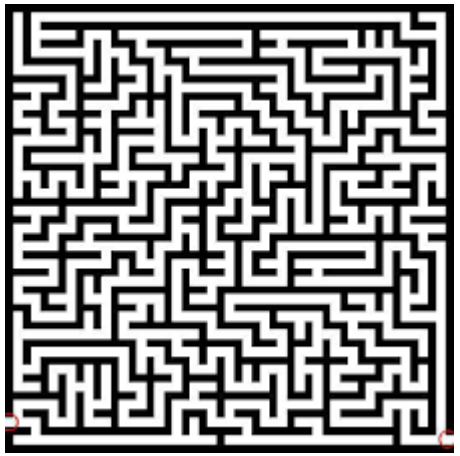
LANCE FURTNOW describe las más importantes en el artículo



4.3 Métodos para encontrar soluciones a problemas no factibles







4.3 Métodos para encontrar soluciones a problemas no factibles

- Actualmente, todos los algoritmos conocidos para problemas NP-Completos utilizan tiempo exponencial con respecto al tamaño de la entrada.
- No se sabe si hay algoritmos eficientes (o no) por lo cual , para resolver un problema NP-Completo de tamaño arbitrario se utilizan diferentes enfoques:

4.3 Métodos para encontrar soluciones a problemas no factibles

- **Aproximación:**

  En este tipo de algoritmos se hace un análisis matemático para determinar qué tan lejos se encuentra la solución a partir de modelos de entradas y salidas de datos.

  Se precisa encontrar el valor de la “constante de aproximación del problema que se resuelve”

4.3 Métodos para encontrar soluciones a problemas no factibles

Para un algoritmo ρ-aproximado, ρ es el factor de aproximación del algoritmo.

Donde:

$\rho > 1$ para problemas de minimización,

$\rho < 1$ para problemas de maximización.

Ejemplo: Un algoritmo 0.4-aproximado para un problema de maximización siempre devuelve una solución cuyo valor se aproxima a un 40% del valor óptimo.

4.3 Métodos para encontrar soluciones a problemas no factibles

- **Probabilístico**

  Un algoritmo probabilístico utiliza aleatoriedad para obtener en promedio una buena solución al problema planteado con una probabilidad de fallar/acertar para una cierta distribución de los datos de entrada dada.

  Las Vegas. (la probabilidad cambia en el tiempo)

  Montecarlo. (generación de números aleatorios)

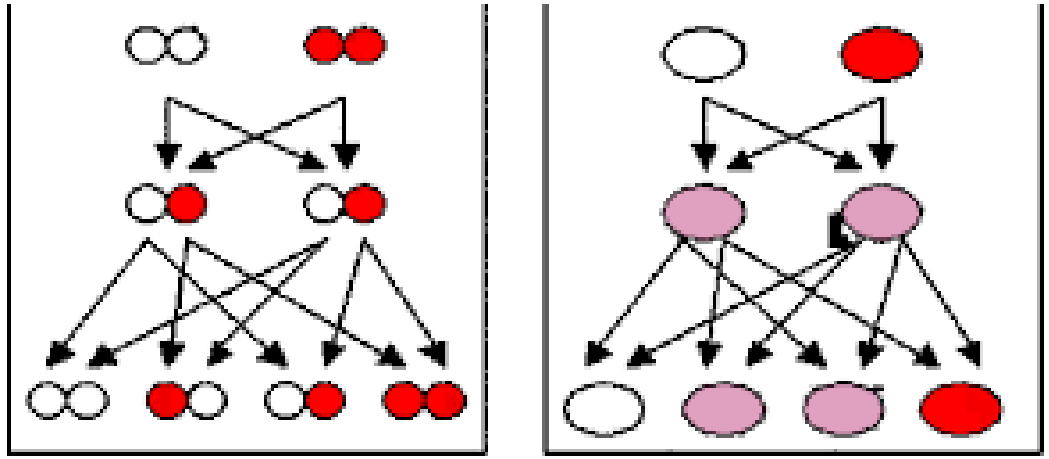
4.3 Métodos para encontrar soluciones a problemas no factibles

- **Algoritmos Genéticos**

  Están basados en el proceso genético de los organismos vivos.

  Mejoran las posibles soluciones hasta encontrar una que esté cerca del óptimo.

  Son métodos adaptativos que pueden usarse para resolver problemas de búsqueda y optimización.



4.3 Métodos para encontrar soluciones a problemas no factibles

- **Restricciones**

  Restringiendo la estructura de las entradas se pueden encontrar soluciones en ciertos contextos

- **Heurísticos**

  Las resolución de problemas se realiza de forma intuitiva o basadas en experiencia previa.

4.3 Métodos para encontrar soluciones a problemas no factibles

- **¿Otros modelos de cómputo?**

  En algunos casos, un modelo de cómputo diferente al de las máquinas de Turing puede resultar una buena opción.

  Actualmente se realiza investigación en modelos alternativos como el cómputo con ADN o el cómputo cuántico