**4 - Búsqueda informada y exploración**

**4.3 – Algoritmos de búsqueda local y problemas de optimización**

* Tenemos problemas donde el camino al objetivo es irrelevante
* Los algoritmos de **búsqueda local** funcionan con un solo **estado actual** y generalmente se mueve solo a los vecinos del estado
* Los caminos seguidos no se retienen
* Son útiles para resolver problemas de optimización puros, en los que e, objetivo es encontrar el mejor estado según la función objetivo
* El paisaje de espacio de datos tiene:
  + Posición (definido por el estado)
  + Elevación (definido por el valor de función de costo de heurística o función objetivo)
  + Si la elevación es un costo, el objetivo es encontrar el valle mas bajo, mínimo global.

**Búsqueda en ascensión de colinas**

* Es un bucle que continuamente se mueve en dirección del valor creciente
* Termina cuando alcanza un pico en donde ningún vecino tiene un valor más alto
* Los algoritmos de búsqueda local típicamente usan una **formulación de estados completa**
* Destaca por:
  + **Máximo local**
  + **Cresta:** crean una secuencia de máximos locales que hace muy difícil la navegación para los algoritmos avaros
  + **Meseta:** área del paisaje de espacio de estados donde la función de evaluación es plana
* El éxito de la ascensión de colinas depende muchísimo de la forma del paisaje del espacio de estados

**Búsqueda de temple simulado**

* Un algoritmo de ascensión de colinas que nunca hace movimientos hacia estados con un valor inferior garantiza ser incompleto porque puede estancarse en un máximo local
* Parece razonable intentar combinar la ascensión de colinas con un camino aleatorio de algún modo que produzca tanto eficacia como completitud, el **temple simulado** papus.
  + Hay que cambiar el punto de vista a **gradiente descendente**
  + El truco es sacudir con bastante fuerza para echar la pelota de mínimos locales no lo suficiente para desalojarlo del mínimo global
  + Empieza sacudiendo con fuerza, y luego gradualmente reduce la intensidad de la sacudida

**Búsqueda por haz local**

* Guarda la pista de k-estados
* Comienza con estados generados aleatoriamente
* En cada paso, se generan todos los sucesores de los k estados
* Si alguno es un objetivo, paramos el algoritmo.
* La información útil es pasada entre los k hilos paralelos de búsqueda

**Algoritmos genéticos**

* Es una variante de la búsqueda de haz estocástica (escoge a k sucesores aleatorios)
* Los estados sucesores se generan combinando dos estados padres
* Comienzan con un conjunto de k estados generados aleatoriamente llamado **población**
* Cada **individuo** está representado como una cadena sobre un alfabeto finito
* Cada estado se tasa con la función de evaluación o **función idoneidad**
* Finalmente cada posición está sujeta a una **mutación** aleatoria con una pequeña probabilidad independientes

**Palabras clave**

|  |  |
| --- | --- |
| **Búsqueda local** | funcionan con un solo estado actual y generalmente se mueve solo a los vecinos del estado |
| **Ascensión de colinas de reinicio aleatorio** | Si al principio no tiene éxito, intente otra vez |
| **gradiente descendente** | Minimizar el coste |
| **Población** | Conjunto de k estados generados aleatoriamente |