Vecindades

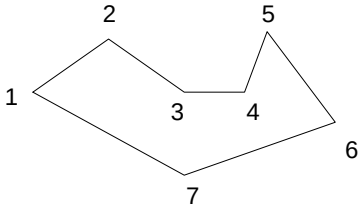
Vecindades

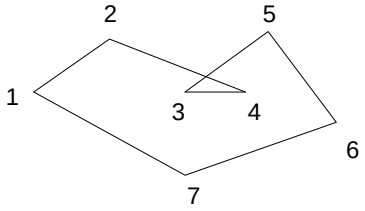
- Dado un punto factible $f \in F$ en una instancia de un problema:
- Una vecindad de f se define como un conjunto $N(f)$ de puntos factibles cerrados a f.
- Dado un problema de optimización con instancias (F,c), una vecindad es un mapeo

$$N: F \rightarrow 2^F$$

Ejemplo de Vecindad

- TSP, definiendo una vecindad:
- $N(f) = \{g \mid g \in F$ y g se obtiene de f como sigue: remover 2 arcos del camino entonces remplazar estos con dos arcos$\}$





Óptimo Local y Global

- Encontrar una solución óptima global puede ser muy difícil en problemas combinatorios.
- Encontrar una solución f que es mejor en $N(f)$, es posible.

Óptimo Local

- Dada una instancia (F,c) de un problema de optimación y una vecindad N, una solución factible $f \in F$, se llama localmente óptima con respecto a N si:

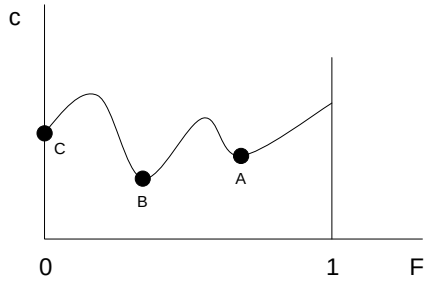
$$c(f) \leq c(g) \forall g \in N(f)$$

Ejemplo de Óptimos

- Considerar la instancia de un problema de optimización (F,c) definido por:

$$F = [0,1] \ sub \ R$$

La función de costo c se presenta en la figura



Ejemplo de Óptimos

- La vecindad se define simplemente por una distancia cerrada para un $\varepsilon > 0$.

$$N_\varepsilon(f) = \{x \mid x \in F \; y \; |x - f| \leq \varepsilon\}$$

Si ε es muy pequeño, los puntos A, B y C son llamados óptimos locales. Pero sólo B es un óptimo global.

Algoritmo de Búsqueda Local

- En TSP, la solución óptima local con respecto a la vecindad N_n con n-cambios, se obtiene con la función *mejorar(t)*, donde $t \in F$:
- *mejorar(t):*
  - Busca cualquier $s \in N_n(t)$ tal que $c(s) < c(t)$ si tal *s* existe, regresa *s*
  - Regresa "*no*" de otra forma

Algoritmo de Búsqueda Local

- *mejorar(t)* busca en $N_n(t)$ una mejor solución s. Si se encuentra, ésta retorna la ruta mejorada. De otra manera retorna "*no*".

```
procedure k-opt
begin
        t:= una ruta inicial
        While mejorar(t) ≠ "no" do
                t:= mejorar(t);
        Return t
end
```

Búsquedas Locales Clásicas

- **Búsqueda local.** es un algoritmo que mejora de forma iterativa la solución inicial. Reemplaza de forma sucesiva la solución actual con una mejor obtenida de su vecindad.
- **Mejoramiento iterativo.** Sólo se aceptan las mejores soluciones. Se comienza con una solución generada de forma aleatoria y se acepta como nueva solución al primer vecino que vaya mejorando el costo de la actual solución.
- **Descenso más rápido.** A diferencia del anterior evalúa a todos los vecinos de una solución y elige al que ofrezca un mayor mejoramiento.
- **Mejoramiento iterativo multi-inicio.** Mejora iterativamente la actual solución como lo hace *mejoramiento iterativo,* cuando esta solución ya no se puede mejorar se tiene un óptimo local, la búsqueda comienza en otro punto de inicio elegido aleatoriamente.
- **Aceptación por Umbral.** Eligen dentro de una vecindad, a los vecinos que se encuentren en un umbral definido por el programador. La comparación con este umbral se hace obteniendo la diferencia en costo del vecino con la solución actual. El valor del umbral se fija alto y con esto todos los vecinos son aceptados, a continuación el umbral se va disminuyendo gradualmente a cero, para que al final sólo se acepten las configuraciones que mejoren la solución.

Ejercicio

- **Hacer un algoritmo con aceptación por umbral para el agente viajero que optimice la solución de la instancia definida.**
- **Desarrollar el programa de tarea y presentarlo para la siguiente clase**