

# CC371:3.- Practica Calificada

## Heuristicas & Meta-Heuristicas

Brando Miguel Palacios Mogollon  
[bpalaciosm@uni.pe](mailto:bpalaciosm@uni.pe)



Universidad Nacional de Ingeniería  
Escuela Profesional de Ciencias de la Computacion

August 27, 2020

# Overview

Pregunta 1

Pregunta 2

# Contenido

Pregunta 1

Pregunta 2

# Motivación: Historia

Se dice que fue postulado por Cauchy en 1847, pero las propiedades de optimizaciones fueron recién vista por primera vez por Haskell Curry en 1944.



(a) Augustin Louis Cauchy



(b) Haskell Curry

# Motivación: Analogía



Figure 2: Una persona trata de bajar de una montaña

# Motivación: Analogía



Figure 3: Hay una densa niebla que hace que la visibilidad sea extremadamente baja

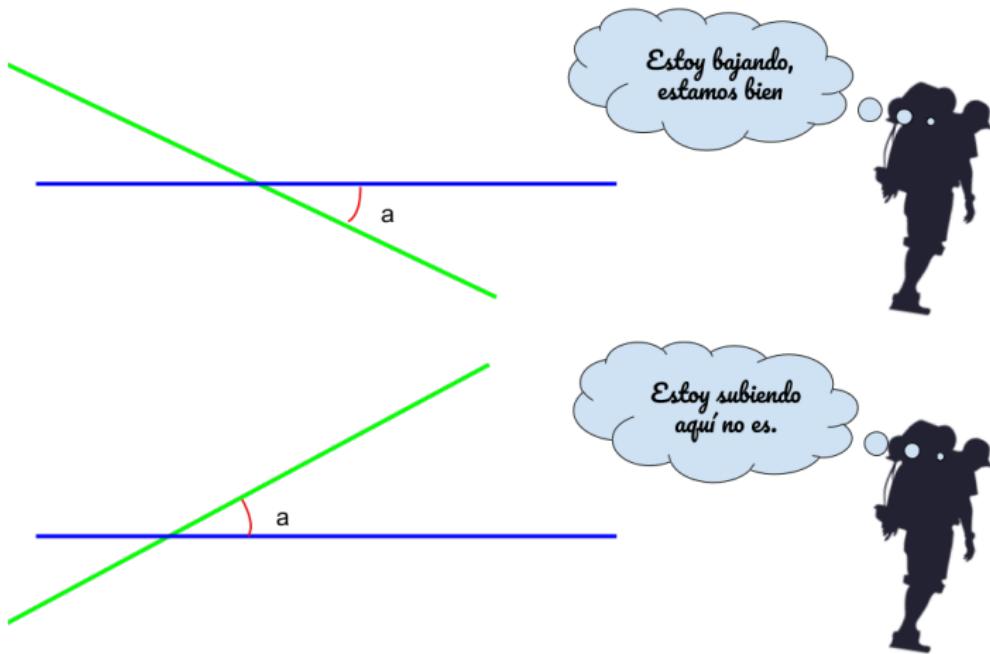
# Motivación: Analogía



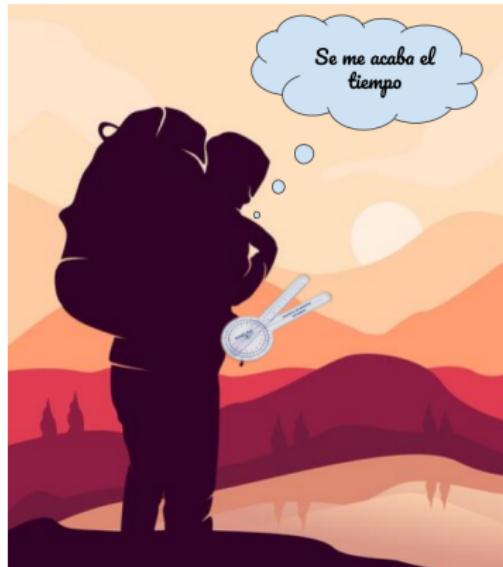
# Motivación: Analogía



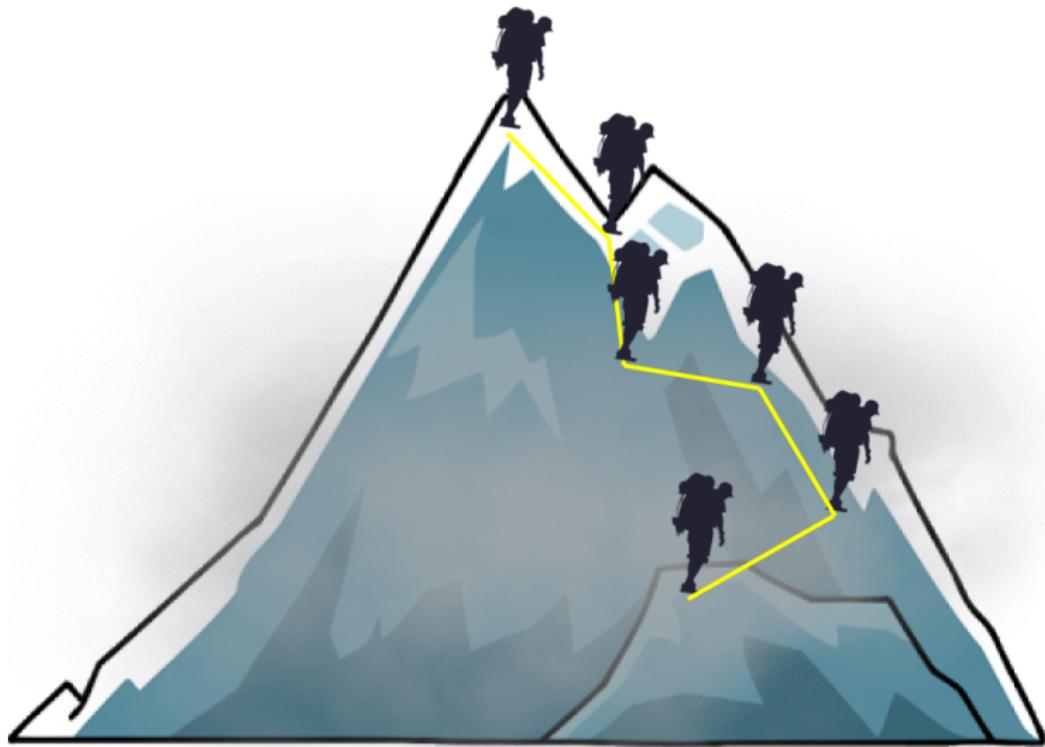
# Motivación: Analogía



# Motivación: Analogía



# Motivación: Analogía



# Motivación: Analogía



Montañista (Algoritmo)



Medición de suelo (Gradiente)

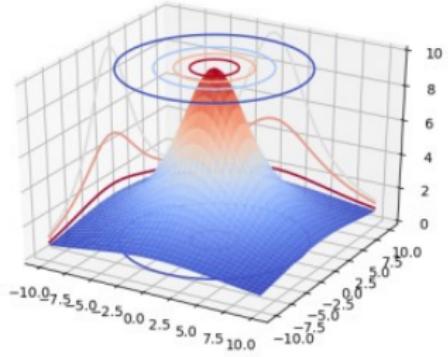
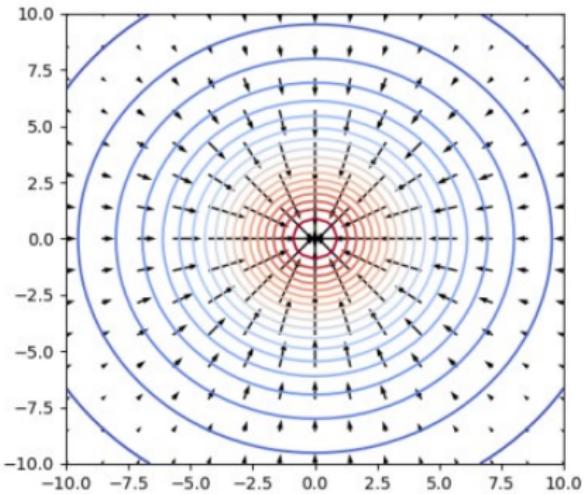


frecuencia de medición ("tasa de aprendizaje")



Montaña ("Función"/Función de coste)

## Gradiente

(a)  $f(x, y)$ (b)  $\nabla f(x, y)$

# Fuerza Bruta

---

**Algorithm:** Algoritmo del Fuerza Bruta

---

**Result:** Encontrar minimo de  $F$

```
 $F \leftarrow \text{array}(F(x));$ 
 $min \leftarrow \infty;$ 
 $i \leftarrow \text{num}(F);$ 
while  $i < 0$  do
|   if  $F[i] < min$  then
|   |    $min \leftarrow F[i];$ 
|   else
|   |
end
```

---

# Descenso de gradiente

Pertenece al conjunto de los heurísticos de optimización de problemas de búsqueda. Este inicializa en un punto aleatorio de la función y se mueve en la dirección opuesta a la gradiente de la función para alcanzar un mínimo local o global en caso sea posible.

## Característica:

- ▶ Es parte de los algoritmos de búsqueda local.
- ▶ Es muy útil en el campo de los métodos numéricos para resolver un sistema de ecuaciones no lineales.

## Definición

Sea una función  $F(x)$  definimos un punto aleatorio de la función  $x = a_0$ , en dirección opuesta a la gradiente de  $F$  ( $-\nabla F(a_0)$ ).

$$a_{n+1} = a_n - \alpha \nabla F(a_n)$$

Donde  $\alpha \in \mathbb{R}^+$

De esto tendremos una secuencia monótona

$$F(x_0) \geq F(x_1) \geq F(x_2) \geq \dots$$

El valor  $\alpha$  es un valor muy importante en este metodo, es también conocido como tasa de aprendizaje o tamaño del paso, el cual determina que tan rápido llegaremos al minimo.

- ▶ Si  $\alpha$  es demasiado grande , el algoritmo puede sobrepasar al minimo, y en el caso de no tenga la "búsqueda de lineas" nunca convergerá a uno y en peor de los casos diverja.
- ▶ De ser demasiado pequeño, el progreso sera muy lento.

# Algoritmo

---

**Algorithm:** Algoritmo del Descenso de Gradiente

---

**Result:** Encontrar minimo local de  $F$

```
x ← random();  
lr ← 0.001 ;  
F ← F(x) ;  
g ← ∇F(x) ;  
while ||∇F(x)|| < ε do  
    x ← x - lr · g ;  
    F ← F(x) ;  
    g ← ∇F(x) ;  
end
```

---

- ▶ La complejidad del descenso de gradiente es de forma exponencial ( $T \in \mathcal{O}(n)$ )

# Aplicación

## Redes Neuronales:

$$J(\theta) = -\frac{1}{m} \sum \left[ y^{(i)} \log(h\theta(x(i))) + (1 - y^{(i)}) \log(1 - h\theta(x(i))) \right]$$

$$J_{regularized} = \underbrace{-\frac{1}{m} \sum_{i=1}^m (y^{(i)} \log(a^{[L](i)}) + (1 - y^{(i)}) \log(1 - a^{[L](i)}))}_{\text{cross-entropy cost}} + \underbrace{\frac{1}{m} \frac{\lambda}{2} \sum_l \sum_k \sum_j W_{k,j}^{[l]2}}_{\text{L2 regularization cost}}$$

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2$$

# Contenido

Pregunta 1

Pregunta 2

# Motivación: Historia

Fue recien introducido por Thomas Feo y Mauricio Resende para el set covering problem en 1989 y formalmente nombrado GRASP y detallado en (1995). Que sus siglas son " *Greedy Randomized Adaptive Search Procedures*" (Procedimiento de búsqueda adaptativa aleatoria codiciosa).

*Journal of Global Optimization* 6: 109–133, 1995.  
© 1995 Kluwer Academic Publishers. Printed in the Netherlands.

109

## Greedy Randomized Adaptive Search Procedures

THOMAS A. FEO

*Operations Research Group, Department of Mechanical Engineering, The University of Texas,  
Austin, TX 78712 U.S.A. (email: feo@emx.utexas.edu)*

and

MAURICIO G.C. RESENDE

*Mathematical Sciences Research Center, AT&T Bell Laboratories, Murray Hill, NJ 07974 U.S.A.  
(email: mgcr@research.att.com)*

(Received: 29 July 1994; accepted: 14 October 1994)

**Abstract.** Today, a variety of heuristic approaches are available to the operations research practitioner. One methodology that has a strong intuitive appeal, a prominent empirical track record, and is trivial to efficiently implement on parallel processors is GRASP (Greedy Randomized Adaptive Search Procedures). GRASP is an iterative randomized sampling technique in which each iteration provides a solution to the problem at hand. The incumbent solution over all GRASP iterations is kept as the final result. There are two phases within each GRASP iteration: the first intelligently constructs an initial solution via an adaptive randomized greedy function; the second applies a local search procedure to the constructed solution in hope of finding an improvement. In this paper, we define the various

Figure 6: Feo, Thomas A.; Resende, Mauricio G. C. (1995). " *Greedy Randomized Adaptive Search Procedures*". *Journal of Global Optimization*.

# Motivación



Figure 7: EL cojo y el ciego



# Características

- ▶ Es considerado una metaheurística iterativa multiarranque, especializado para producir soluciones de buena calidad de problemas de optimización combinatorio.
- ▶ No está inspirado en la naturaleza por lo que no usa el concepto de adaptabilidad.
- ▶ Es también considerado como heurística semi-codiciosa, entrando al debate si este puede ser un algoritmo metaheurístico o semi-heurístico.
- ▶ Se divide en fases: construcción y búsqueda local (mejoramiento).
- ▶ Posee un conjunto solución que irá siendo refinado en cada iteración, realizando muestreos sucesivos.

## Enunciado

Dado un conjunto de soluciones finitas  $X = \{x_1, x_2, \dots, x_n\}$  y una función objetivo de valor real  $f: X \rightarrow R$  para ser minimizado.

Para ello emplearemos 2 fases:

- ▶ Fase de Construcción
- ▶ Fase de Mejoramiento

## Fase de Construcción

- ▶ Esta fase es ciega pero greedy por lo que busca componentes de forma inicialmente aleatoria.
- ▶ Se construye iterativamente se realiza la elección del siguiente elemento ordenando los candidatos en una lista  $C$  definiendo así la función codiciosa como  $g : C \rightarrow R$ .
- ▶ Existirán beneficios para poder seleccionar cada elemento, pero siendo este del tipo adaptativo estos beneficios se actualizaran en cada iteración para reflejar los cambios provocados en la iteración anterior.
- ▶ Su propiedad aleatoria se basa en escoger aleatoriamente uno de los mejores candidatos de la lista previamente creada, el cual sera no necesariamente el mejor.
- ▶ Esta lista de candidatos mejores se llamará *lista de candidatos restringidos* (RCL), esta permitirá obtener diferentes soluciones en cada iteración, pero no necesariamente el mejor componente, siendo así codicioso adaptativo.
- ▶ La técnica mas utilizada para construir el RCL (*MakeRCL*) aplica las reglas de porcentaje "*minmax – α*".



# Fase de Construcción: Algoritmo

```
procedure ConstructGreedyRandomizedSolution(Seed, g(.))
1    $x := \emptyset$ ;
2   Sort the candidate elements  $i$  according to their incremental costs  $g(i)$ ;
3   while ( $x$  is not a complete solution)  $\rightarrow$ 
4       RCL:=MakeRCL();
5        $v := \text{SelectIndex}(RCL, \text{Seed})$ ;
6        $x := x \cup \{v\}$ ;
7       Resort remaining candidate elements  $j$  according to  $g(j)$ ;
8   endwhile;
9   return( $x$ );
end ConstructGreedyRandomizedSolution;
```

## Fase Construcción

En cualquier iteración *GRASP*, sean  $g_{min}$  y  $g_{max}$  los costos son altos y bajos por lo que.

$$g_{min} = \min_{i \in C} g(i)$$

$$g_{max} = \max_{i \in C} g(i)$$

Este mecanismo necesita dos parámetros importantes para construir esta lista: un mecanismo basado en cardinalidad (*CB*) y un mecanismo basado en valores (*VB*).

Para *CB*, el *RCL* esta compuesto por  $k$  elementos con los mejores costos incrementales. En el caso de *VB*, *RCL* esta asociado a un parámetro  $\alpha \in [0, 1]$  y un valor de umbral  $\mu = g_{min} + \alpha(g_{max} - g_{min})$ .

Todo elemento candidato que no sea mayor que el valor umbral se inserta en *RCL*, es decir,  $g(i) \in [g_{min}, \mu]$ . El parámetro  $\alpha$  al ser 0 denotara un algoritmo codicioso puro, mientras al ser 1 sera equivalente a una construcción aleatoria.



# Fase Construcción: Algoritmo Refinado

```
procedure ConstructGreedyRandomizedSolution(Seed,  $\alpha$ ,  $k$ ,  $g(\cdot)$ )
1    $x := \emptyset$ ;
2   Initialize the candidate set  $C$  by all elements;
3   Evaluate the incremental cost  $g(i)$  for all  $i \in C$ ;
4   while ( $|C| > 0$ )  $\rightarrow$ 
5      $g_{min} := \min_{i \in C} g(i)$ ;  $g_{max} := \max_{i \in C} g(i)$ ;
6     if (CB RCL is used) then
7       Sort candidate elements  $i \in C$  according to  $g(i)$ ;
8       RCL :=  $C[1 \dots k]$ ;
9     else RCL :=  $\{i \in C \mid g(i) \leq g_{min} + \alpha(g_{max} - g_{min})\}$ ;
10    endif;
11     $v := \text{SelectIndex(RCL, Seed)}$ ;
12     $x := x \cup \{v\}$ ;
13    Update the candidate set  $C$ ;
14    Reevaluate the incremental costs  $g(i)$  for all  $i \in C$ ;
15  endwhile;
16  return( $x$ );
end ConstructGreedyRandomizedSolution;
```

## Fase de Construcción: $\alpha$

Se puede elegir de diversas maneras:

- ▶  $\alpha$  se elige al azar de una distribución de probabilidad.
- ▶  $\alpha$ , valor fijo cercano al interés puramente codicioso.



## Fase de Búsqueda local

Las soluciones generadas por la construcción no garantiza que sean localmente óptimas. Por lo tanto, se crea una búsqueda local que intentara mejorar cada solución construida. Esta funciona de manera iterativa reemplazando sucesivamente la solución actual por una mejor solución cercana a la solución actual. Termina cuando no se encuentra una solución mejor en el vecindario.

La estructura de vecindad  $N$  para un problema relaciona una solución  $s$  del problema con un subconjunto de soluciones  $N(s)$ .

La clave del éxito de un algoritmo de búsqueda local consiste en la elección adecuada de una estructura de vecindad, técnicas de búsqueda de vecindad eficientes y la solución inicial



## Fase de Búsqueda local

Comenzamos con un  $x_0 \in X$  y generando una secuencia de soluciones mejoradas  $x_1, x_2, \dots, X_M$ . Buscando una solución tal que  $f(x_k) < f(x_{k+1})$ , de encontrarse se convertirá en actual hasta no existir otro mejor.

```
procedure LocalSearch( $x, f(\cdot)$ )
    1 Let  $N(x)$  be the neighborhood of  $x$ ;
    2  $H := \{y \in N(x) \mid f(y) < f(x)\}$ ;
    3 while ( $|H| > 0$ ) →
        4  $x := \text{Select}(H)$ ;
        5  $H := \{y \in N(x) \mid f(y) < f(x)\}$ ;
    6 endwhile
    7 return( $x$ );
end LocalSearch
```

# Algoritmo

```
algorithm GRASP( $f(\cdot)$ ,  $g(\cdot)$ , MaxIterations, Seed)

1    $x_{best} := \emptyset$ ;  $f(x_{best}) := +\infty$ ;
2   for  $k = 1, 2, \dots, \text{MaxIterations}$  do
3        $x := \text{ConstructGreedyRandomizedSolution}(\text{Seed}, g(\cdot))$ ;
4       if ( $x$  not feasible) then
5            $x := \text{repair}(x)$ ;
6       endif
7        $x := \text{LocalSearch}(x, f(\cdot))$ ;
8       if ( $f(x) < f(x_{best})$ ) then
9            $x_{best} := x$ ;
10      endif
11  endfor;
12  return( $x_{best}$ );
end GRASP
```

# Aplicación

Existen muchos campos a los que el metodo GRASP es muy usado:

- ▶ Lógica (*Deshpande A, Triantaphyllou E (1998) Un GRASP para inferir cláusulas lógicas a partir de ejemplos en tiempo polinomial y algunas extensiones. Modelo de cálculo matemático*)
- ▶ Ubicación (*Cravo G, Ribeiro G, Lorena LN (2008) Un GRASP para la colocación de etiquetas cartográficas de características puntuales. Computo Geo-science*)
- ▶ Árbol minimo de Steiner (*Canuto S, Resende M, Ribeiro C (2001) Búsqueda local con perturbaciones para el problema del árbol de Steiner recolector de premios en gráficas. Redes*)
- ▶ Horarios y programación (*Feo T, Bard J (1989) Programación de vuelos y planificación de la base de mantenimiento.* )
- ▶ Trasporte (*Argüello M, Bard J, Yu G (1997) A GRASP para el enrutamiento de aeronaves en respuesta a encallamientos y retrasos.*)