Metaheurísticas

Búsqueda Local en Entornos Continuos

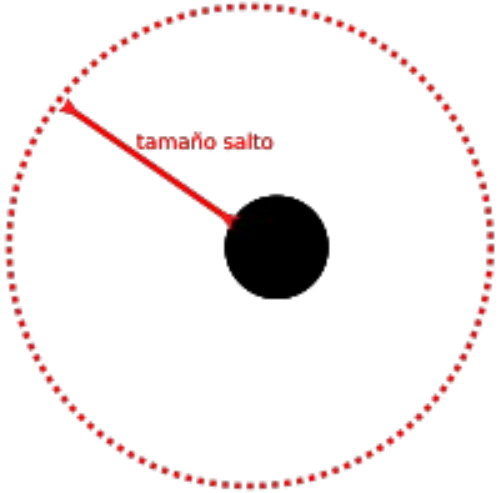
- Esquema de BL en entornos Continuos
- Solis-Wets

Búsqueda Local en continuo

- En optimización continua existen distintos métodos de Búsqueda Local:

Búsqueda Local en continuo

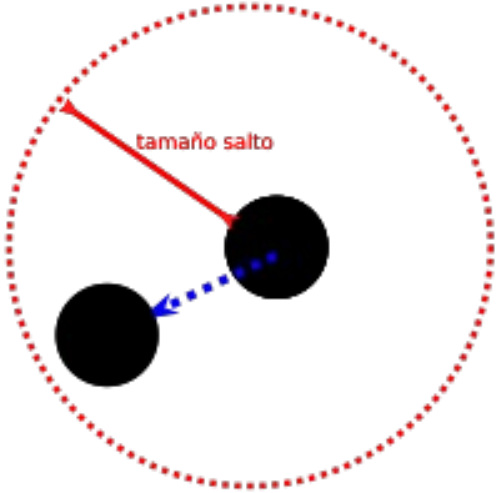
- En optimización continua existen distintos métodos de Búsqueda Local:
  - Trabajan con ta
  - Hasta cuanto permito que se pueda aleja



Búsqueda Local en continuo

- En optimización continua existen distintos métodos de Búsqueda Local:
  - Trabajan con tamaño de salto.
  - Incrementa según vector distancia (normal o uniforme).

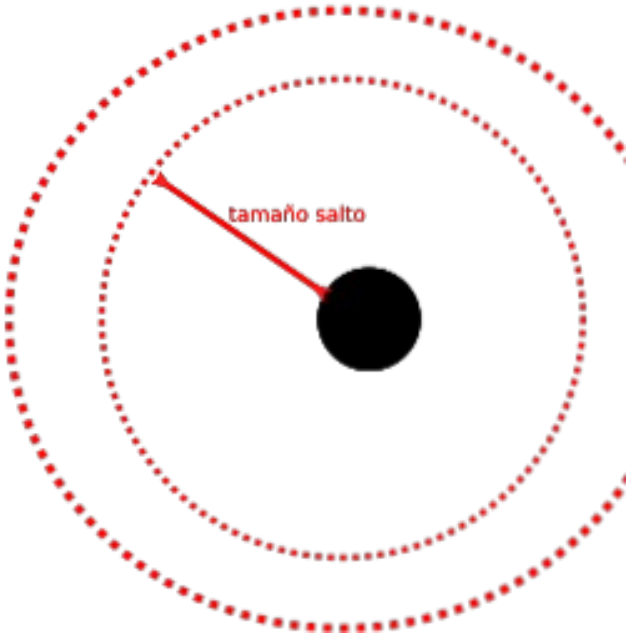
Se usa un número aleatorio para cambiar cada



Búsqueda Local en continuo

- En optimización continua existen distintos métodos de Búsqueda Local:
  - Trabajan con tamaño de salto.
  - Incrementa según vector distancia (normal o uniforme).
  - Si suele mejorar incrementa el tamaño de salto.

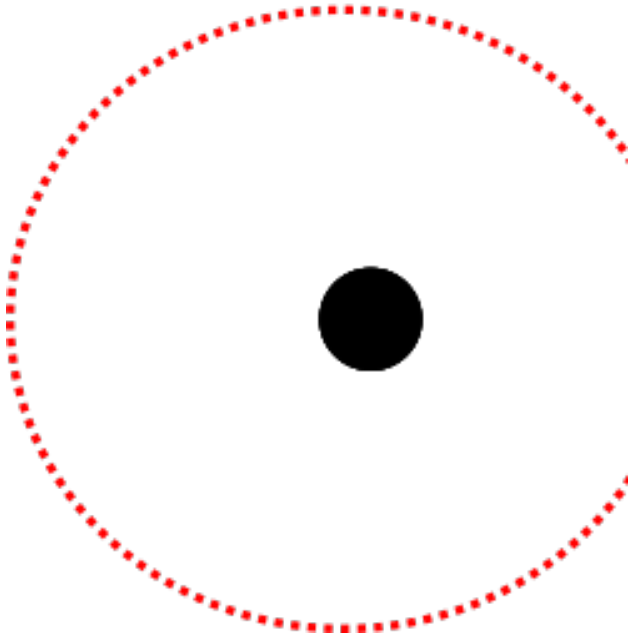
Esto acelera la bú



Búsqueda Local en continuo

- En optimización continua existen distintos métodos de Búsqueda Local:
  - Trabajan con tamaño de salto.
  - Si suele mejorar incrementa el tamaño de salto.
  - Si s .

Que vaya bien o mal varias veces para poder modif



Búsqueda Locales en Continuo

- Distintos algoritmos genéricos muy conocidos:
  - Simplex, muy conocido pero no escala bien.
  - Quasi-Newton, usa la derivada.
    - Versiones que *calculan la derivada.*
  - Solis-Wets, sencillo y no requiere derivada.
    - Ofrecemos el código fuente de Solis-Wets para las versiones meméticas.

Solis-Wets

Procedimiento Solis-Wets *(step_size)*

Start

Evals = 1; Bias = 0; Num_success = num_failed = 0;

Mientras evals < maxevals

Dif = vector aleatorio(0, s tamaño de)

newsol = current + bias + dif; evals += 1

If (COST(newsol) is better than COST(current) then

Current = newsol;

increm_bias // *Incrementa bias*

num_sucess += 1; num_failed = 0;

Else

Si la nueva solución no ha funcionado bien, se comprueba si el

Newsol' = current – bias – dif; evals += 1

If (COST(newsol') is better than COST(current) then

Current = newsol';

decrement_bias // Decrementa bias

num_sucess += 1; num_failed = 0;

Else

Bias /= 2; num_failed += 1; num_success = 0;

End

Update step_size

End

Solis-Wets

Procedimiento update_step_size *(num_failed, num_success)*

Start

```
    if (num_success == 5) then
       step_size *= 2;
       num_success = 0;
    End

    if (num_failed >= 3) then
        step_size /= 2;
        num_failed = 0;
End
```

cinco éxitos conse

Procedimiento increment_bias

Start

```
    Bias[i] = 0.2*bias[i] + 0.4*(dif[i]+bias[i]);
End
```

Procedimiento decrement_bias

Start

```
    Bias[i] = bias[i] - 0.4*(dif[i]+bias[i]);
End
```

Influencia de step_size

- El step_size inicial posee mucha influencia.
  - Un valor muy alto influye poco la solución inicial.
  - Un valor muy bajo hace la BL más lenta.
- ¿Cómo establecer el step_size inicial?
  - Valor fijo pequeño.
  - En algoritmos poblacionales considerar la distancia entre soluciones (como asignar la distancia al más cercano).
  - Combinar ambos (Considerar un tamaño fijo y la distancia entre valores).

Implementación Solis-Wets

- Ofrecemos dos implementaciones disponibles en el repositorio:
  - **Python**: solis.py, función para usarlo con numpy.
  - **C++**: testsolis.cc, que incluye la función y programa main de ejemplo.

El profesor le suele meter 500 evaluaciones a la

Hay que asegurar de no salirs

Hay que poner seguro un máximo de evaluaciones q

Hay un warning que nos avisa si nos pas