Métodos Constructivos

Camilo Andrés Rodríguez Garzón

March 20, 2018

1 Descripción

En el presente trabajo se desea resolver el problema del viajante. Para esto, se elaboró tres métodos aleatorios que permitiecen obtener una ruta y de esta manera resolver dicho problema. Para este trabajo se eligieron el método Búsqueda Aleatoria en la región factible, Estrategia Evolutiva (1+1) y Recocido Simulado (url con implementación de los métodos Tsp). A continuación se presenta el algoritmo correspondiente a cada uno de los métodos:

Algorithm 1 Random Search Tsp

```
1: procedure BUILD ROUTE(N)
2:     d, coorR ← generateRandomSolution(coordinates)      ▷ obtener una solución inicial aleatoria y su distancia
3:     k ← 1                                               ▷ inicializa número de iteraciones
4:     while k < N do
5:         dt, rs ← generateRandomSolution(coordinates)    ▷ obtener una solución aleatoria y su distancia
6:         if dt < d then                                  ▷ si la nueva solución es mejor
7:             d ← dt
8:             coorR ← rs
9:         end if
10:        k ← k + 1
11:    end while
12:    drawTsp(coorR, d)                                   ▷ se dibuja la solución obtenida
13: end procedure
```

Algorithm 2 Evolutionary Strategy Tsp

procedure BUILD ROUTE(N)
    $newCoor \leftarrow copy(coordinates)$ ▷ obtener una copia de las coordenadas
    $dF, coorF \leftarrow generateFather(coordinates)$ ▷ obtener una solución inicial padre y su distancia
    $k \leftarrow 1$ ▷ inicializa número de iteraciones
    while $k < N$ **do**
        $dS, coorS \leftarrow generateMutation(dF, coorF)$ ▷ obtener una solución hija y su distancia con una mutación dada
        if $dt < d$ **then** ▷ si la nueva solución es mejor
            $dF \leftarrow dS$
            $coorF \leftarrow coorS$
        end if
        $k \leftarrow k + 1$
    end while
    $drawTsp(coorR, d)$ ▷ se dibuja la solución obtenida
end procedure

Algorithm 3 Simulated Annealing Tsp

```
1: procedure BUILD ROUTE(N)
2:     tk ← 100                                   ▷ la temperatura se inicializa en 100
3:     alpha ← 0.98   ▷ la tasa de variación de la temperatura se inicializa en 0.98
4:     newCoor ← copy(coordinates)   ▷ obtener una copia de las coordenadas
5:     dF, coorF ← generateFather(coordinates)   ▷ obtener una solución inicial padre y su distancia
6:     k ← 1                                      ▷ inicializa número de iteraciones
7:     while k < N do
8:         dS, coorS ← generateMutation(dF, coorF)   ▷ obtener una solución hija y su distancia con una mutación dada
9:         if dt < d then                         ▷ si la nueva solución es mejor
10:            dF ← dS
11:            coorF ← coorS
12:        else
13:            pa ← e(− |dF − dS| /tk)            ▷ probabilidad de aceptación
14:            tk ← tk ∗ alpha                    ▷ decrementar la temperatura
15:            ra ← random(1, 0)                  ▷ random de aceptación
16:            if ra < pa then                    ▷ aceptar solución menos factible
17:                dF ← dS
18:                coorF ← coorS
19:            end if
20:        end if
21:        k ← k + 1
22:    end while
23:    drawTsp(coorR, d)                          ▷ se dibuja la solución obtenida
24: end procedure
```

Es de resaltar que para obtener una solución hija se realizo swaping entre dos coordenadas, una coordenada inicial aleatoria del padre y una coordenada cercana a la coordenada inicial y así posteriormente obtener una solución hija recalculando la distancia total.

En relación a la temperatura y a decremente de este (alpha) se realizaron varias muestras que permitieron ver mejores comportamientos con dichos valores iniciales.

La solución inicial se obtiene atra vez de la construcción del Tsp usando como algoritmo base nearest neighbors y esta cota primar se toma como pie de apoyo para la obtención de nuevas soluciones.

Se eligió dichos métodos y esto se resume al hecho de aprender en un ambiente más real cuales son las dificultades de implementarlos así como también de adquirir la experiencia de poder construir cada unos de estos.

2 Resultados

2.1 Datos linhp318

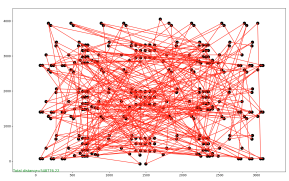

Figure 1: Tsp random search

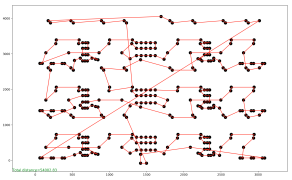

Figure 2: Tsp evolutionary strategy

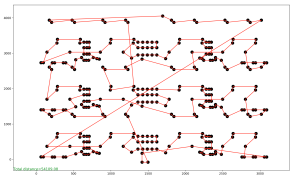

Figure 3: Tsp simulated annealing

Table 1: Comparativa de tiempo de ejecución y distancia recorrida de Algoritmos para linhp318

Algoritmo	Tiempo Inicial	Tiempo final	Distancia
Random search	2018-03-20 01:42:36.915734	2018-03-20 01:42:37.509704	548776.22
Evolutionary strategy	2018-03-20 01:38:42.038569	2018-03-20 01:38:45.149977	54002.83
Semi-pilot	2018-03-20 01:44:23.633350	2018-03-20 01:44:26.755945	54109.08
TSPLIB	-	-	41345

2.2 Datos berlin52

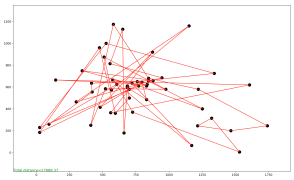

Figure 4: Tsp random search

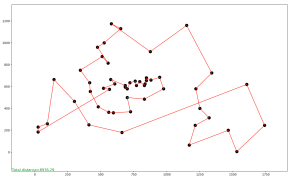

Figure 5: Tsp evolutionary strategy

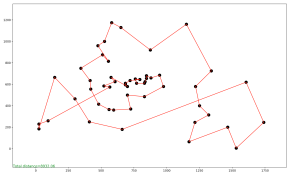

Figure 6: Tsp simulated annealing

Table 2: Comparativa de tiempo de ejecución y distancia recorrida de Algoritmos para berlin52

Algoritmo	Tiempo Inicial	Tiempo final	Distancia
Random search	2018-03-20 01:33:02.365071	2018-03-20 01:33:02.702718	27688.37
Evolutionary strategy	2018-03-20 01:33:50.417394	2018-03-20 01:33:52.613921	8976.29
Simulated annealing	2018-03-20 01:34:58.561924	2018-03-20 01:35:00.775077	8932.06
TSPLIB	-	-	7542

3 Conclusiones

- El algoritmo de simulado recodido arrojó mejor solución en los datos de berlin52 pero en los datos de linhp318 lo hizo la estrategía evolutiva, lo que nos hace pensar que siempre hay que calibrar la temperatura y el decremento de está para que el simulado recocido se comporte de una mejor manera.
- Los tiempos de ejecución son más extensos en el algoritmo de simulado recocido y la estrategia evolutiva, esto es debido a la evaluación de el camino padre e hijo.
- El método del vecino más cercano es un algoritmo muy potente en relación a los tiempos de ejecución y tomando este como cota primal se hace muy dificil mejorar la solución ya obtenida.
- Ambos algoritmos por la estrategia greedy que manejan son castigados al unir la coordenada final con la inicial.
- Las implementaciones realizadas en TSPLIB superan significativamente a las soluciones construidas.