

Comparación entre el Algoritmo Genético Simple y el Algoritmo de Optimización Inspirado en Magnetismo con Repulsión de Corto Alcance aplicados al Problema del Agente Viajero

Andrea Itzel González Vargas
Carlos Gerardo Acosta Hernández

Cómputo Evolutivo 2017-1
Facultad de Ciencias UNAM

1. Introducción

Panorama general

Para este proyecto final decidimos realizar una comparación entre el Algoritmo Genético Simple (**AGS**) y el Algoritmo de Optimización Inspirado en Magnetismo (**MOA**), con un operador enfocado a la **exploración**, propuesto en el artículo¹ que escogimos para la exposición en clase.

Como punto de referencia, elegimos implementar un programa similar al que hicimos primero para encontrar soluciones óptimas de varias instancias del Problema del Agente Viajero (**TSP**) usando el MOA. Lo anterior por nuestra experiencia previa con el desarrollo del primer proyecto², que nos sirvió de guía para llevar a cabo la construcción de nuestra propia versión de un MOA aplicado a TSP.

La idea es verificar si el MOA, siendo una propuesta reciente³ en el mundo del cómputo evolutivo, es una buena opción para resolver problemas de este tipo como ya hemos visto con el AGS y decidir si es en algún aspecto competitivo respecto a su adversario. La entrada de ambos programas es una instancia del TSP, para la que ambos algoritmos mostrarán la mejor solución encontrada en cada una de sus iteraciones.

Para manejar y procesar las instancias del TSP, proporcionadas por la colección *TSPLIB*⁴ y recopiladas en archivos individuales bajo el directorio *tsp/*, hacemos uso de una biblioteca para *Java*, llamada *TSPLIB4J*⁵.

¹Magnetic-inspired optimization algorithms: Operators and structures

²Referencia al repositorio del primer proyecto

³Propuesto por bla y bla

⁴pagina oficial

⁵referencia a su repo de github

Marco teórico

Creemos que ya no es necesario ser muy específicos respecto al AGS implementado y presentado previamente en nuestro primer proyecto, por lo que esta sección referirá únicamente al MOA.

En la teoría de los campos magnéticos, las partículas pueden atraerse o repelerse unas con otras dependiendo de su polaridad. Como muchos algoritmos en el campo de estudio de la optimización, el MOA es una propuesta inspirada en este principio observado en la naturaleza. Para este algoritmo, el conjunto de posibles soluciones son partículas con propiedades magnéticas, diseminadas en el espacio de búsqueda, donde cada partícula es capaz de incidir una sobre la otra con una fuerza de atracción de largo alcance -tal como el concepto de fuerza electromagnética.

Similar a los algoritmos de optimización por enjambre o nube de partículas (**PSO**⁶), donde los individuos buscan imitar a aquel con mayor grado de adaptación, las partículas en el MOA con mejor calificación poseen masa y campo magnético mayores, resultando en una mayor fuerza de atracción ejercida sobre las otras partículas.

Por un lado, este comportamiento cubre el concepto de explotación revisado en nuestro curso, sin embargo, el MOA cuenta además con una mejora considerable respecto a los PSO convencionales.

Es considerado una de las debilidades de los PSO, el hecho de que sólo las mejores partículas, son las que tienen una repercusión significativa sobre las demás menos adaptadas, que no tienen incidencia en otras. En este sentido, estos algoritmos dejan fuera de consideración, importante información que pueden contener las partículas con bajo grado de adaptación. Es por ello que en el esquema del MOA que revisamos, se considera la posibilidad de que todas las partículas cuenten con cierto potencial de atracción, independientemente de su grado de adaptación (aunque éste lo afecte en magnitud).

Asimismo, se incluye un operador de repulsión de corto alcance (**SRR**⁷). Este operador -que explicaremos más ampliamente en secciones posteriores-, permite que el algoritmo no se concentre únicamente en las partículas más cercanas a la solución, pues actúa como a manera de repulsión cuando la distancia entre dos partículas es menor a cierto umbral predefinido, manteniendo la diversidad y previniendo una convergencia prematura en óptimos locales.

Además del esfuerzo implementado para la optimización de “tours” en las instancias del TSP, el MOA ya ha sido puesto a prueba en varias aplicaciones de Inteligencia Artificial, como el entrenamiento de redes neuronales y ha sido propuesta ya una versión binaria del algoritmo. Las implementaciones de los MOA, resultan en un paradigma dual de interacción entre partículas con la fuerza de repulsión/atraccción, que puede devenir un balance entre exploración y explotación.

Siguiendo esa línea, las virtudes que supone el esquema del MOA parecen prometedoras frente a la contraparte de este proyecto, el AGS.

⁶En inglés: Particle Swarm Optimization

⁷En inglés: Short-Range Repulsion

Herramientas

Como herramienta fundamental, proporcionamos el pseudo-código del MOA básico que no contempla más operadores que el de actualización de posición de una partícula dada su velocidad.

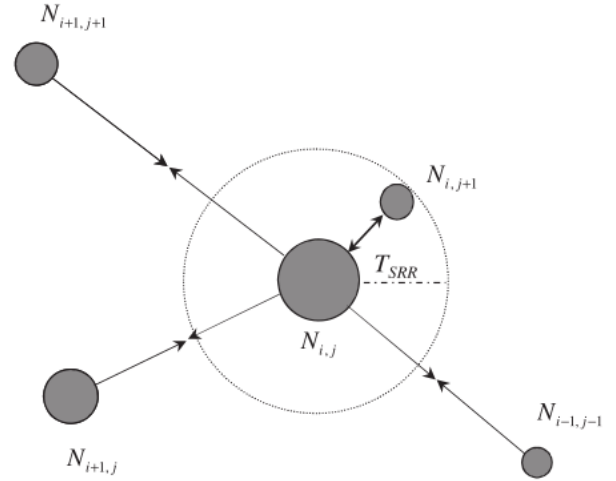
ALGORITHM 1: Magnetic-inspired Optimization Algorithm (MOA)

```

 $t \leftarrow 0$ ;  $X^t \leftarrow \text{randomValidPopulation}()$ ;
repeat
   $t \leftarrow t + 1$ ;
  for each  $x$  in  $X^t$  do
    evaluate  $x$  and store performance
    in magnetic fields  $B^t$ ;
  end
  Normalize  $B^t$ ;
  for each  $x$  in  $X^t$  do
    evaluate mass of  $x$  and store it in
     $M^t$ ;
  end
  for all  $x_{ij}^t$  in  $X^t$  do
     $F_{ij} \leftarrow 0$ ;
    find  $N_{ij}$ ;
    for each  $x_{uv}^t$  in  $N_{ij}$  do
       $F_{ij} \leftarrow F_{ij} + \frac{(x_{uv}^t - x_{ij}^t) \times B^t}{D(x_{ij}^t, x_{uv}^t)}$ ;
    end
  end
  for all  $x_{ij}^t$  in  $X^t$  do
     $v_{ij,k}^{t+1} = \frac{F_{ij,k}}{M_{ij,k}} \times R(u_k, l_k)$ ;
     $x_{ij,k}^{t+1} = x_{ij,k}^t + v_{ij,k}^{t+1}$ ;
  end
until  $t = \text{maxGen}$ ;
```

N_{ij} := Conjunto de vecinos de la partícula x_{ij} .
 F_{ij} := Fuerza de atracción ejercida sobre la partícula x_{ij} .
 $D(x_1, x_2)$:= Distancia entre x_1 y x_2 .

La figura siguiente es el esquema del operador de repulsión SRR,



En ella se puede apreciar el umbral de distancia mínima T_{SRR} , que debe existir entre partículas. Pasado el umbral, se aplica el operador de repulsión.

Más detalles sobre el diseño de la implementación se pueden encontrar en la Sección 3 y siempre se puede consultar el código fuente del proyecto en el directorio *src/*.

A partir de ahora, nos referiremos a la implementación del MOA como SRMOA, por el operador que incluye en su ejecución.

2. Planteamiento

Objetivo

Determinar si el SRMOA (Tipo III) es una opción cuyo desempeño es competente respecto al algoritmo genético simple (AGS) en la resolución del problema del agente viajero (TSP).

Hipótesis

Esperamos que ambos tengan un desempeño similar para las distintas instancias del TSP. Creemos también que conforme la dimensión de las instancias del TSP crezcan, el SRMOA tendrá una mejora significativa respecto a instancias menores y pueda superar al AGS en tales magnitudes. Respalando ésta hipótesis tenemos las conclusiones del artículo sobre el MOA que mostraba que mientras mas crecía el problema, el SRMOA mejoraba su desempeño. Mientras que por otro lado nuestras pruebas con el AGS nos hacían ver que mientras más crecía el problema, disminuía el desempeño de éste algoritmo.

Asimismo, suponemos que la implementación del SRMOA será más sencilla que la del AGS.

Metodología

Implementaremos el SRMOA y utilizaremos el AGS que previamente implementamos para el Proyecto 1. Probaremos los algoritmos con los archivos burma14, ulysses16, gr17, gr21 y ulysses22. Para cada archivo las tuplas de parámetros de las tablas siguientes un total de cinco veces. En conjunto se harán 25 corridas por algoritmo.

Los **parámetros** que se utilizarán para el **AGS** serán:

Pc	Pm
0.6	0.001

Y para el **SRMOA**:

α	ρ	T_{SRR}	C_{SRR}
100	100	1	0.1

Para ambos algoritmos, se fijará el tamaño de la **población** en 225 individuos y se establecerá 1000 como máximo número de **generaciones** (siendo este el criterio de detención de las iteraciones).

Para hacer la comparación entre ambos algoritmos graficaremos el exceso porcentual promedio de peso del mejor circuito vs el número de generaciones de cada archivo para cada algoritmo, a demás del promedio del conjunto de los cinco archivos. Compararemos también el tiempo promedio que se tardó cada algoritmo para correr cada archivo.

3. Especificación

Codificación del problema

La codificación del problema se realizó de manera similar a la codificación del AGS implementado en el primer proyecto de la materia, puesto que cada individuo contaba tanto en su fenotipo como en su genotipo, una permutación de las ciudades sin repeticiones de la instancia del TSP; el arreglo asociado, representa la ruta propuesta por el individuo.

En este caso, se trata de partículas con una ruta asociada, un vector de ciudades que determinan la posición de dicha partícula en el espacio de búsqueda y que será alterada dependiendo de la velocidad calculada por el cociente de la fuerza de atracción ejercida por sus vecinos y su masa, junto con un factor aleatorio (3).

Por ejemplo, para una partícula x con posición i, j en la estructura celular en la generación t , $x_{ij}^t = [1, 3, 2, 4, 5]$, la ruta propuesta sería: $C_1 \rightarrow C_3 \rightarrow C_2 \rightarrow C_4 \rightarrow C_5 \rightarrow C_1$, $C_i \in P$, considerando a P el conjunto de ciudades de la instancia de TSP.

Con el fin de evitar la implementación de un corrector de partículas, la generación de la población inicial aleatoria, se realizó con cuidado de asignar cada ciudad elegida aleatoriamente de P sólo una vez por individuo. Acorde a esa intención, el operador velocidad que afecta la posición de una partícula, i.e. su ruta, realiza permutaciones entre las ciudades de dicha partícula únicamente, por lo que no existe el riesgo de invalidar una propuesta de “tour” al aplicarlo.

Evaluación de la población

La evaluación de la población en un primer momento se realiza calculando el costo de la ruta propuesta por cada partícula.

Campos magnéticos

Masa

Fuerza de atracción

Operadores

Velocidad

SRR

Experimentación

Conclusiones