Informe Proyecto
parte 1

Gonzalo Torterolo
Facultad de ingeniería
UDELAR
Montevideo, Uruguay
Email: gonzalo.torterolo@fing.edu.uy

Gisel Cincunegui
Facultad de ingeniería
UDELAR
Montevideo, Uruguay
Email: gisel.cincunegui@fing.edu.uy

***Abstract*—En esta primer entrega se presenta el problema a solucionar en el correr del proyecto, las principales estrategias de resolución, y los principales focos del estudio experimental. En la próxima entrega se intentan aplicar las estrategias planteadas para la resolución práctica del problema. Al final del proyecto se espera lograr una aplicación que logre con recursos computacionales no demasiados elevados y en un tiempo razonable resolver una instancia real o al menos verosímil del problema, junto con un set de datos estadísticos experimentales.**

***Index Terms*—GA, VRP, TSP, Rescheduling, Crowding, GoogleMaps, watchmaker, java**

I. Introducción

El escenario del problema en cuestión puede enunciarse de la manera siguiente:
"Dado un conjunto de pedidos a resolver sobre un mapa, y un conjunto de mandaderos que resuelven los pedidos, se desea encontrar un recorrido para cada mandadero partiendo de un punto inicial, de manera que se resuelvan todos en el menor tiempo posible. A medida que pasa el tiempo y se resuelven pedidos también van llegando nuevos."

El problema puede mapearse a una variante del problema de ruteo de vehiculos (VRP), una generalización de TSP [1], donde además se agrega que el la instancia del problema varíe con el tiempo a medida que llegan nuevos pedidos y se resuelven otros, característica a la que se llama rescheduling de recorridos

La motivación de este problema es realizar una aplicación móvil/web que permita a un conjunto de personas coordinar sus recorridos de manera eficiente. Esto comprende el agregar/quitar pedidos, ó agregar/quitar mandaderos.

II. Análisis

A. Variables de entrada

Considerando el enunciado inicial del problema se define:

- Cantidad de nodos N: Cantidad de puntos en el mapa.
- Cantidad de mandaderos M: Cantidad de mandaderos asignados al problema.
- Mapa Map: Es el conjunto de puntos p_i , $i \in 1..N$
- Distancia $Dist$: La distancia en el mapa del problema, que es una función $Dist(x, y)$ que para dos puntos x, y retorna su distancia en tiempo (es la distancia en tiempo). Todos los pedidos deben ser realizables, por lo que la distancia entre cualquier par de puntos debe ser finita.
- Cola de pendientes Q: La lista global de pedidos pendientes a resolver.
- Tasa de arribos T: Es el tiempo cada cuanto se agrega un nuevo pedido a la cola de pendientes. Los puntos del pedido se generan aleatoriamente según una distribución propia del problema.
- Punto de origen O: Representa la ubicación del local a donde se va comenzar el recorrido.
- Recorrido individual R_i , $i \in 1..M$: Es el recorrido para el mandadero i, una tupla ordenada de puntos del mapa cumpliendo que no tiene puntos en común con ningún otro R_j salvo por el origen.
- Recorridos R: Es el conjunto de todos los recorridos.

El problema busca optimizar el tiempo transcurrido en los recorridos R_i entre todas las instancias R posibles, esto es, intenta minimizar la función objetivo

$$TotTime(R) = \max_{i \in 1..M} (Tiempo(R_i))$$

$$Tiempo(R_n) = Dist(O, R_n[1]) + \sum_{i \in 2..largo(R_n)} (Dist(R_n[i-1], R_n[i]))$$

III. Síntesis

A. Literatura y trabajos previos

No se encontró una aplicación de AG para el problema en particular, si para variantes similares. Existe un estudio para un problema simple de VPR que incluye además capacidades y costos aunque no rescheduling (ver [2]), que propone la utilización de una técnica similar al crowding para mantener la diversidad de la población, a partir del mismo se decide utilizar crowding en el algoritmo.

B. Técnicas a utilizar

La utilización de AG parece adecuada para abordar el problema planteado, por las siguientes razones:

1) Interesa tener un buen tiempo de respuesta y retornar soluciones de calidad relativamente buena en tiempos cortos de ejecución, los AG al ser técnicas metaheurísticas suelen funcionar más rápido que métodos exactos, sobre todo en problemas NP, como es el presentado.
2) Los AG son buenos para problemas combinatorios, pues tienen una formulación algorítmica natural.
3) El entorno varía de manera dinámica, por lo que el AG puede aprender de manera implicita variedad de soluciones y evolucionar junto con el entorno.

Se ha optado por utilizar modelo de islas con la intención de distribuir el cómputo del algoritmo. Esta arquitectura permite paralelizar el cómputo, haciéndola más escalable. Dada esta realidad, puede interesar estudiar el comportamiento de aumentar el tamaño de la población a medida que se incorporan nuevos mandaderos, pues cada uno dispone de un celular que puede aportar en el cómputo. Los parámetros adecuados se estudian en la configuración paramétrica de cada instancia. La migración se realizará de manera asincrónica.

- Codificación
  Se codifican los recorridos de todos los mandaderos utilizando una permutación de puntos del mapa, separando el recorrido de cada mandadero por un separador S, habiendo M-1 separadores.

$$R =< R_{i_1}, S, R_{i_3}, ..., S, R_{i_M} >$$

  Siendo la concatenación de R_i una permutación de todos los puntos del mapa.
- Población inicial
  Se piensa que utilizar alguna heurística como greedy para inicializar la población podría afectar la diversidad, por lo que una inicialización aleatoria parece lo más adecuado.
- Fitness
  El fitness sugerido en una función de minimización es sencillamente la función objetivo:

$$Fitness(R) = TotTime(R)$$

- Operadores evolutivos
  - Mutación
    Para la mutación se utiliza sencillamente el intercambio de lugar de un gen del cromosoma. A los fines prácticos su poder disruptivo es tan bueno como cualquier otro, y asegura la obtención de una solución factible.
  - Selección
    La estrategia de selección será por torneo de tamaño M, para evitar calcular el fitness tantas veces, y mu+lamda.
  - Cruzamiento
    Se tiene algo muy similar a una permutación, a los fines prácticos se puede adaptar Ordered Crossover para la codificación en cuestión, además en comparación con PMX y CX parece ser menos disruptiva. Este tipo de cruzamiento tiene la propiedad de mantener la cantidad de mandaderos en todo momento, lo que asegura una instancia factible.

C. Aspectos y consideraciones tecnológicas

Por su portabilidad al momento de escalar se decide utilizar java como lenguaje de programación. Por su claridad y sencillez se decide utilizar el framework watchmaker para implementar el AG. Son muchas las librerías para el manejo de mapas y datos geográficos (algunas investigadas son geotools http://www.geotools.org/ y KML), y muchas son también las fuentes desde donde obtener modelos geográficos realistas/reales (se pueden citar GoogleMaps, datos abiertos de montevideo y OpenStreetMap). La más prometedora ha sido Google DistanceMatrixAPI [3] y DirectionAPI [4], las mismas permiten dar la el tiempo real de un recorrido entre dos puntos geográficos, considerando la forma de las calles y otras restricciones reales.

La principal particularidad del problema implica el rescheduling. Para solucionar el mismo, ante el arribo de un nuevo pedido, se mantiene mantiene la población del calculada hasta el momento pero se le agrega a todos los individuos de la población en una posición aleatoria el nuevo punto del pedido; y ante la asignación de un pedido a un mandadero, se eliminan de todos los individuos de la población el punto del pedido. Luego de esto se deja evolucionar el algoritmo para que se ajuste al cambio.

IV. Evaluación experimental

A. Ajuste y configuración paramétrica

En la configuración paramétrica se van a probar diversas instancias de ajuste y comparar los resultados en distintos casos. Una vez determinado los mejores parámetros se inicia la evaluación experimental. Las instancias a utilizar serán generadas a partir de una selección de puntos de un mapa real, y la distribución a tomar de estos se generarán a partir de información de densidad de población ó cantidad de puntos de interés, etc de cada mapa. Se maneja la posibilidad de usar GooglePlaces como fuente de referencia para las distribución de los puntos.

En resumen, los parámetros a configurar dadas las técnicas antes definidas son:

- Tamaño de población:
  El tamaño global de la población.
- Mutación:
  Probabilidad de mutación.
- Reemplazo:
  Crowding: El delta de crowding debe ser determinado.
- Islas:
  - Uso de islas.
  - Número de islas.
  - Número de islas en relación a cantidad de mandaderos.
  - Tasa de migración.
  - Topología.

B. Dimensiones del problema

En una primer etapa de ajustes deberá importar el rendimiento que alcance el algoritmo, pero su principal objetivo es verificar la correctitud funcional del mismo. Al momento de realizar la evaluación experimental se espera poder manejar instancias de dimensiones realistas.

Haciendo un análisis de las variables de decisión, se enfocará el estudio sobre los casos particulares donde se tiene: Una cantidad reducida de mandaderos en comparación a los puntos del mapa, pues el caso contrario tiende a ser cada vez más sencillo y algoritmos greedy alcanzan el óptimo mejor. También a medida que la tasa de arribo de nuevos pedidos se hace más grande, el problema se asemeja a un VPR clásico, no es el principal interés resolver este caso, pero puede ser de utilidad a modo de contrastar la calidad del algoritmo con implementaciones conocidas. Del mismo modo a medida que se reduce la cantidad de mandaderos a 1 el problema se reduce a un TSP clásico.

El tiempo es otra dimensión importante a determinar. Se exige la que el algoritmo pueda arrojar soluciones de calidad en tiempos del orden de pocos minutos.

C. Pruebas

El proceso de evaluación experimental comprenderá:

- Análisis comparativo de calidad de solución contra un algoritmo greedy.
- Análisis comparativo del mismo algoritmo usando islas y sin usar islas.
  - En distintos entornos de ejecución paralelo VS secuencial (Cluster VS Máquinas domésticas).

  - Se evalua calidad de soluciones y speedUp.
- Análisis de evolución del sistema. Tiempo VS mejora de la función de fitness.
- Análisis de sensibilidad a las entradas.
  - Se estudiará cuanto tarda en estabilizarse la solución y una vez que el sistema entre en régimen como varía la calidad de la solución obtenida a lo largo del tiempo.
  - El mismo procedimiento se piensa realizar variando cantidad de mandaderos y la tasa de arribo de pedidos.

V. Expectativas

Se espera al finalizar este proyecto comprobar la adecuación, confirmando o corrigiendo las ideas planteadas en esta primer instancia de análisis. La principal particularidad, sin que presente una novedad, para el algoritmo es la incorporación de rescheduling al clásico problema de VPR, por lo que se espera obtener resultados originales en este sentido. Se esperan comentarios y observaciones sobre las decisiones tomadas para la realización del proyecto.

References

[1] Wikipedia. (2015) Vrp brief. [Online]. Available: http://en.wikipedia.org/wiki/Vehicle_routing_problem

[2] C. P. U. of Technology of Troyes. (2001) A simple and effective evolutionary algorithm for the vehicle routing problem. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.3.7230&rep=rep1&type=pdf

[3] G. Developers. (2015) Distancematrixapi documentation. [Online]. Available: http://developers.google.com/maps/documentation/distancematrix/

[4] ——. (2015) Directionsapi documentation. [Online]. Available: http://developers.google.com/maps/documentation/directions/