V Congreso de Ingeniería de Organización
Valladolid-Burgos, 4-5 Septiembre 2003

Índice

Comparativa de las fases constructivas de las metaheurísticas ACO y GRASP para el problema CARP.

Joaquín Bautista Valhondo[1], Jordi Pereira Gude[2]
[1] Departamento de Organización de Empresas. UPC (Doctor Ingeniero Industrial, ETSEIB, joaquin.bautista@upc.es)
[2] Departamento de Organización de Empresas. UPC (Ingeniero en Organización Industrial, ETSEIB, jorge.pereira@upc.es)

RESUMEN

En los años 80 y 90 han aparecido diversas metaheurísticas para la resolución de problemas de optimización. Entre ellas, la metaheurística GRASP, Greedy Randomized Adaptive Search Procedure, y los algoritmos de hormigas, ACO Ant Colony Optimization, comparten un esquema similar dividido en una fase constructiva que permite la generación de soluciones distintas mediante la incorporación de un factor de azar, seguidas por una fase de mejora de éstas. En el presente trabajo se comparan los resultados ofrecidos por las fases constructivas de ambas metaheurísticas para un problema en concreto, el diseño de itinerarios con servicio asociado a los arcos y restricciones de capacidad de los vehículos.

***Palabras clave**: Diseño de itinerarios, CARP, metaheurísticas.*

1. Introducción.

Muchos problemas de organización pueden plantearse como problemas de optimización combinatoria. Desde un enfoque práctico y debido a la dificultad para resolver de forma exacta muchos de estos problemas, se han desarrollado diversos procedimientos heurísticos para obtener soluciones con una calidad aceptable.

Dentro de los procedimientos heurísticos desarrollados, destacan las heurísticas constructivas de tipo *greedy*, comilón, basadas en el empleo de reglas de prioridad para discriminar entre las diferentes opciones que se presentan en un proceso constructivo de la solución. Estas heurísticas suelen ofrecer soluciones aceptables, en promedio, tanto más cuanto mayor sea el número de aspectos (relacionados con las propiedades que satisfacen las soluciones óptimas) que combina la regla; no obstante, no puede concluirse que exista una única regla que domine a las demás ante cualquier ejemplar de un problema y, por otra parte, salvo que se incorpore el azar en el procedimiento, la heurística siempre ofrecerá la misma solución.

Una segunda clase de heurísticas, es la constituida por las metaheurísticas GRASP, [8], (*Greedy Randomized Adaptative Search Procedure*) y ACO, [3], que permiten generar distintas soluciones gracias a la incorporación de azar al procedimiento anterior. En cada paso de construcción de un procedimiento *greedy*, se establece una selección de alternativas entre el conjunto de las posibles y se someten a sorteo obteniendo la decisión en el paso, con probabilidades que dependen, o no, de una regla de prioridad y de la información obtenida de soluciones anteriores. A las soluciones generadas por estos procedimientos se acostumbra aplicar un procedimiento de mejora local para obtener soluciones de mayor calidad.

Finalmente, una tercera clase de heurísticas se conforma por métodos de búsqueda local o procedimientos de exploración de entornos; entre ellos se encuentran: los procesos de escalado (HC), el recocido simulado (SA), la búsqueda tabú (TS) y los algoritmos genéticos (GA). Esta clase de heurísticas proporciona vías alternativas para buscar soluciones en un espacio delimitado por la definición de un vecindario.

El presente trabajo se centra en un estudio de la fase constructiva de la segunda clase de heurísticas expuestas, formada por los algoritmos ACO y GRASP. En el siguiente apartado se plantea el problema CARP que servirá como base para el estudio. El apartado tres lo dedicaremos a las heurísticas constructivas presentes en la literatura y que se modificarán para su uso por ambos esquemas metaheurísticos. El apartado cuatro se centrará en la metaheurística ACO, mientras que el apartado cinco mostrará el esquema general de las propuestas realizadas de la metaheurística GRASP, comparando los resultados obtenidos, en el apartado seis, por ambas propuestas, para finalmente mostrar las conclusiones del trabajo.

2. Los problemas de diseño de itinerarios con servicio asociado a los arcos.

Los problemas de diseño de itinerarios con servicio asociados a arcos se refieren a los problemas donde la actividad principal de servicio es el cubrimiento de los arcos en la red de transporte. En contraste con los problemas de diseño de itinerarios clásicos como el viajante de comercio, donde el servicio a cubrir se encuentra localizado en los vértices de la red y los arcos representan los caminos que conectan los vértices, el diseño de itinerarios en arcos se centra en atravesar ciertos arcos que tienen un servicio asociado. Ejemplos prácticos incluyen la limpieza de aceras y calles, la retirada de nieve de las carreteras, la inspección de carreteras para su mantenimiento y la entrega de correo. En todos ellos los segmentos deben ser recorridos completamente, o bien la presencia de demandantes en el arco obliga a recorrer todo el arco. Se debe tener presente que la diferencia entre un problema de diseño de itinerarios en arcos o en vértices sólo pertenece a una decisión en el modelado del problema, que debe tener en cuenta la dificultad de resolución del modelo resultante.

El diseño de itinerarios con servicio asociado a los arcos y restricciones de capacidad, conocido como CARP [4], del inglés *Capacitated Arc Routing Problem*, es el problema análogo en el campo del diseño de itinerarios en arcos al clásico *Vehicle Routing Problem* (VRP), donde existen restricciones explícitas asociadas a cada ruta. Para su resolución se han planteado diversos procedimientos heurísticos y exactos. Entre los procedimientos heurísticos presentados, se encuentran tres procedimientos constructivos conocidos como el algoritmo de escaneado de caminos, debido a Golden, DeArmon y Baker [4], el algoritmo de aumento e inserción, Pearn, [7], así como la tradicional heurística de *Nearest Neighbourhood* , véase [6], que centrarán la comparativa realizada en este trabajo.

A continuación se presenta la nomenclatura que se utilizará, para posteriormente comentar en el apartado tres los algoritmos anteriormente señalados.

Se considerará a $G=(V,E)$ un grafo no dirigido, con $|V|$ vértices y $|E|$ arcos. Se define la matriz de caminos mínimos $SP(i,j)$ entre el vértice i y j en el grafo G. Por convención, se considerará el vértice 1 como el depósito del que parten y retornan los vehículos con una capacidad asociada Q. Cada uno de los arcos $e(i,j) \in E$ tiene asociada una pareja de valores (c_{ij},q_{ij}) que

indican, respectivamente, el coste de atravesar y la demanda asociada al arco. Se denomina *R* al conjunto de arcos que requieren servicio (aquellos que $q_{ij}>0$) y se establece una lista *RE* para su uso durante los procedimientos constructivos formada por los arcos de *R* aún no cubiertos.

3. Procedimientos heurísticos para el problema.

Entre los diversos algoritmos presentados en la literatura, se han escogido dos de ellos, el algoritmo de escaneado de caminos y el algoritmo de aumentado y combinación, por su sencillez y facilidad de hibridación en procedimientos constructivos aleatoriezados derivables de los esquemas generales de las metaheurísticas que se pretenden comparar posteriormente, así como por su representatividad como dos de las formas más estandarizadas de construir soluciones, ya sea mediante la construcción sucesiva o bien mediante la inserción en una parte de las soluciones ya construidas.

Adicionalmente a estos procedimientos, se detalla una heurística basada en visitar el vecino más próximo semejante a la heurística más básica existente para el problema del viajante de comercio. En el contexto del problema del viajante de comercio, la heurística inicia la ruta en una ciudad al azar y escoge, una a una, la ciudad más cercana a la actual a visitar entre aquellas aún no visitadas. Cuando se han visitado todas las ciudades que componen el problema, se añade el arco que une la última ciudad visitada con la ciudad inicial. Para el problema CARP, la adaptación inicia la ruta desde el depósito, vértice 1, en vez de una ciudad al azar, y teniendo en cuenta las restricciones de capacidad de los vehículos, que limitan la lista de ciudades candidatas a ser visitadas a aquellas que cumplan con la restricción de capacidad impuesta por la parte de la solución construida y el retorno al depósito en caso de no existir ningún candidato, se escoge visitar el arco requerido aún no presente en la parte de solución construida tal que el vértice más cercano al último vértice visitado en la solución sea el más cercano. Al elegir el vértice por el que se iniciará la visita a un arco, se está fijando el sentido en que el arco es atravesado. A la pareja formada por el arco y el sentido en que se visita, se le denominará arco-sentido. A continuación se muestra el algoritmo implementado.

Algoritmo 1: Algoritmo de vecinos más próximos para el problema CARP.

1. Sea el depósito el último vértice de la solución.
2. Formar un ciclo *C* añadiendo de forma secuencial el arco en la lista de arcos requeridos tal que uno de sus vértices sea el más cercano al último vértice en la solución, y sea el otro vértice del arco el último vértice de la solución. Cuando la capacidad del vehículo se agote, no existe un arco no visitado cuya demanda asociada sea inferior al remanente de carga del camión, volver al depósito por que camino más corto ofrecido por la matriz de distancias desde el último vértice de la solución hasta el depósito y sea el depósito el último vértice de la solución.
3. Eliminar el arco escogido de la lista de arcos requeridos. Si la lista de arcos requeridos está vacía finalizar; si no, volver a 2.

El algoritmo de vecinos más próximos es un ejemplo de los algoritmos constructivos basados en la concatenación sucesiva a la parte de la solución ya construida por el procedimiento. Otro ejemplo, procedente en este caso de la literatura del problema CARP, es el algoritmo propuesto por [4] y que se compone de los siguientes pasos:

Algoritmo 2: Algoritmo de escaneado de caminos para el problema CARP

1. Formar un ciclo *C* añadiendo secuencialmente el arco en *RE* escogido según un criterio de selección de arcos (ver más abajo). Parar cuando la capacidad del vehículo se agote, volver al depósito por que camino más corto.
2. Eliminar *C* de *RE* y parar si *RE* es una lista vacía, si no volver a *1*.

El mayor atractivo de este procedimiento es su simplicidad, siendo básicamente idéntico al procedimiento de vecinos más próximos, si ésta es la regla marcada para la selección del siguiente arco. Aún así, Golden, et al., [4], no proponen esta regla, sino cinco diferentes. Entre ellas se ha escogido la siguiente: Escoger el arco de *RE* tal que si la carga acumulada en el camión es menor a la mitad de la carga *C*, se escoja el arco tal que maximice la distancia con el depósito, o se minimice en caso contrario.

La otra clase de procedimientos constructivos existentes, se basan en la inserción de las candidatas entre las partes de la solución ya construida, a continuación se detalla el algoritmo propuesto por [5] y adaptado de la siguiente forma:

Algoritmo 3: Algoritmo de aumentado y combinación para el problema CARP

1. Construir tantos ciclos separados como rutas deben construirse, seleccionando aquellos arcos del conjunto de arcos requeridos tales que la distancia mínima de los ciclos posibles sea máxima. (El ciclo de distancia mínima para el servicio de un único arco corresponde al ciclo más corto entre los posibles según sentido de circulación).
2. Aumentar: Empezando por el ciclo más largo y para cada ciclo construido, tantos como rutas, extender el servicio a aquellos arcos requeridos pertenecientes al ciclo, que son aquellos arcos que formen parte de la matriz de traza de los caminos mínimos ya sea para el trayecto entre el vértice y el arco como del arco al vértice, mientras la capacidad del vehículo lo permita.
3. Combinar: En orden creciente a la distancia adicional necesaria para introducir un arco en cualquier posición de uno de los ciclos con capacidad remanente, incluir el arco en el ciclo y modificar el ciclo correspondiente.
4. En caso que aún queden arcos por servir, volver a 1 eliminando de la lista del conjunto de arcos requeridos

aquellos que ya forman parte de alguno de los ciclos creados, y estableciendo la construcción de un único ciclo adicional.

Después de mostrar los tres procedimientos constructivos implementados, los dos apartados siguientes muestran su adaptación a las metaheurísticas ACO y GRASP.

4. Procedimientos basados en la metaheurística ACO.

Muchas especies de hormigas tienen un comportamiento de obtención de alimento basado en el seguimiento y depósito de rastro a través de una sustancia química denominada feromona. Dada una fuente de alimento, la colonia de hormigas tiende a encontrar de forma natural el camino más corto entre la fuente y su nido, a través de dos procesos. Primero, las hormigas depositan feromonas durante el trayecto, y segundo, las hormigas siguen, normalmente, el camino dónde encuentran más feromonas depositadas con anterioridad. Si las hormigas encuentran un camino más corto, más hormigas circularán por ese camino y más feromonas se depositará por él.

Teniendo en cuenta todas las características observadas en las hormigas, este fenómeno natural es adaptable a obtener una solución de un problema mediante una secuencia de decisiones tales que, en cada etapa del proceso, la decisión tomada depende sólo de las decisiones anteriores y las alternativas actuales. Un procedimiento de resolución basado en este comportamiento de las hormigas, véase los procedimientos AS, [2], o ACS, [3] , deberá poseer las siguientes características:

- Las hormigas depositan feromona artificial en alguna de las características del problema asociada con la decisión en curso. Inicialmente se inicializa esta feromona artificial a un valor suficientemente pequeño y posteriormente, cada vez que se obtiene una solución para el problema, se deposita feromona asociada a las características presentes en la nueva solución.

  En los procedimientos implementados, se utilizan dos tipos de elementos asociados al mantenimiento de la feromona. El procedimiento mantiene feromona, τ, entre parejas de arcos y sentidos correlativos, i y j, según su aparición en las rutas solución, véase fórmula 1 donde fo es el valor de la función objetivo obtenido y fo^* el valor de la mejor solución obtenida por el procedimiento para el problema.

$$\tau_{ij} \leftarrow \tau_{ij} + \frac{fo^*}{fo} \cdot \rho \qquad (1)$$

  El procedimiento de aumentado y combinación mantiene adicionalmente información de feromona, τ', asociada a la ciudad que se ha seleccionado para iniciar la formación de los ciclos, véase fórmula 2.

$$\tau'_i \leftarrow \tau'_i + \frac{fo^*}{fo} \cdot \rho \qquad (2)$$

- Las hormigas incluyen una información heurística denominada visibilidad relacionada

con el problema a resolver, asociada a las reglas utilizadas en un proceso constructivo para el problema.

En los procedimientos implementados, se utiliza como visibilidad, η, el valor utilizado para discriminar en cada decisión del procedimiento constructivo. En la heurística de vecinos más próximos se utiliza la inversa de la distancia entre los arcos y sentidos, primando aquellos arcos más cercanos, mientras que en el procedimiento de escaneado de caminos se utiliza la inversa de la distancia y se duplica el valor de visibilidad en caso de cumplir la condición de lejanía o cercanía, según el caso, que se explota durante el procedimiento constructivo. Finalmente, en cada una de las fases del procedimiento de aumentado e inserción se utiliza la distancia entre el arco-sentido y el depósito, para el primer caso, y la inversa del aumento de recorrido que supone la inserción de la candidata entre una pareja de ciudades presentes en la ruta.

- Las hormigas construyen soluciones mediante una secuencia de decisiones. La elección entre las alternativas se basará en una lista de decisiones candidatas, y una ley probabilística que pondere la cantidad de feromonas presentes, mediante un peso denominado α, y la información heurística, mediante un peso denominado β.

  La lista de decisiones candidatas está formada por todos aquellos arcos-sentido tales que una vez insertado el arco en la solución continúe siendo factible, o bien, por un grupo restringido de candidatos formado por el subgrupo con mejor criterio de selección, numerador de la fórmula 3. En cualquiera de los dos casos, la probabilidad de escoger una de las decisiones candidatas entre el conjunto de candidatas será:

$$p_{ij} = \frac{\left[\tau_{ij}\right]^{\alpha}\left[\eta_{ij}\right]^{\beta}}{\sum_{h\in D}\left[\tau_{ih}\right]^{\alpha}\left[\eta_{ih}\right]^{\beta}} \qquad (3)$$

  Dónde D es el conjunto de decisiones candidatas, j representa el arco-sentido candidato, e i representa el último arco-sentido, o el depósito en su caso, presente en la parte de la solución construida para las heurísticas de vecinos más próximos o de escaneado de caminos, la posición a insertar el candidato en el caso de aumentado y combinación durante el proceso de combinación, ó ϕ durante la elección de los arcos-sentido del paso 1 del mismo procedimiento.

- La feromona depositada se evapora a lo largo del tiempo, evitando el estancamiento en una única solución.

  A efectos prácticos, una vez se ha construido una solución y antes de depositar feromona relativa a la solución construida, depositada como se ha citado anteriormente, se eliminará una cantidad proporcional de feromona entre todos los elementos de la solución mediante la fórmula 4.

$$\tau_{ij} \leftarrow (1-\rho)\cdot\tau_{ij} \qquad (4)$$

En general, un procedimiento basado en hormigas, ACO, tendrá la siguiente estructura:

Algoritmo 4: Esquema general de las metaheurísticas ACO.

```
1.  Para todo i, 1≤i≤máximo_iteraciones
     a. Generar una solución mediante un procedimiento
        constructivo específico para el problema
     b. Aplicar mejora local
     c. Actualizar mejor solución
     d. Actualizar feromonas
2.  Fin Para
3.  Mostrar resultados
```

Para el presente trabajo, al pretender estudiar el comportamiento del procedimiento constructivo se ignora el punto 1.b.

5. Procedimientos basados en la metaheurística GRASP.

La metaheurística GRASP, *greedy randomized adaptive search procedure*, es un procedimiento para la resolución de problemas de optimización combinatoria. En general, un algoritmo GRASP se compone por dos fases, en una primera fase constructiva, se genera una solución mediante una heurística greedy aleatoriezada, mientras que en una segunda fase se realiza una mejora local iterativa partiendo de la solución generada en la primera fase, hasta que se encuentra una solución localmente óptima. A continuación se muestra el esquema de la metaheurística:

Algoritmo 5: Bloque principal de la metaheurística GRASP.

```
1.  Inicialización.
2.  Para todo k, 1≤k≤numero_iteraciones
     a. Solucion←Fase_constructiva(semilla)
     b. Solucion←Mejora_local(Solucion)
     c. Actualizar_solucion(Solucion,Mejor_solucion)
3.  Fin Para
4.  Mostrar mejor_solución
```

Pueden verse más detalles de esta metaheurística en Resende y Ribeiro (2001).

Básicamente, los procedimientos constructivos son similares a los implementados para los algoritmos basados en hormigas, ya que el concepto de visibilidad de los algoritmos de hormigas es equiparable al concepto de criterio de idoneidad de los algoritmos GRASP, aunque no se almacena ningún tipo de información parcial sobre las soluciones ya construidas, papel de las feromonas en los algoritmos ACO, y se utiliza una lista de candidatos restringida en todos los casos, véase algoritmo 6, en que se muestra el esquema general de la fase constructiva.

Algoritmo 6: Bloque constructivo de la metaheurística GRASP.

```
1.  Solución←∅
```

2. Evaluar la función de calidad de los elementos candidatos
3. Mientras Solución no esté completa
   a. Construir una lista de candidatos reducidos (*RCL*)
   b. Escoger un elemento *s* de *RCL* al azar
   c. Solución←Solución∪{*s*}
   d. Reevaluar función de calidad de los elementos candidatos
4. Fin
5. Retornar solución

El esquema mostrado en el algoritmo 6 es un esquema semejante al de los algoritmos 1 y 2, mientras que el algoritmo 3 puede dividirse en dos fases, la primera fase formada por los pasos 1 y 2, mientras la segunda se conforma por los pasos 3 y 4, que pueden realizarse mediante dos aplicaciones del algoritmo 6, el primero encargado de la elección de arcos y sentidos semilla de la solución, y equivalente al paso 1 del algoritmo original, mientras que el segundo, equivalente al paso 3, realizará el paso de combinación.

Como función de calidad de los elementos candidatos, puede utilizarse el mismo elemento utilizado como visibilidad en los algoritmos de hormigas, y al igual que en estos algoritmos no se utiliza ninguna mejora local.

6. Experiencia computacional.

Para analizar el comportamiento de los procedimientos constructivos propuestos, así como los diferentes parámetros de cada uno de estos algoritmos, se compara su comportamiento frente a la colección presentada por [4] de problemas de referencia para el problema CARP. La colección está compuesta por 23 ejemplares con un número de vértices comprendidos entre siete y veintisiete y un número de arcos entre once y cincuenta y cinco, y se compara el resultado ofrecido por cada procedimiento metaheurístico utilizando cada procedimiento constructivo con diferentes juegos de parámetros, y la mejor solución conocida para el problema.

Se ha denotado cada procedimiento, y su juego de parámetros de control, de la siguiente manera: se ha denominado (1) ACO y (2) ACS a los dos procedimientos basados en colonias de hormigas presentados anteriormente y que corresponden respectivamente a (1) el procedimiento que no utiliza una lista restringida de candidatas y (2) al procedimiento con una lista de candidatos restringido formada de la misma forma que para los algoritmos GRASP. Cada uno de estos procedimientos ha sido hibrizado con una de las tres heurísticas constructivas, ya sea la heurística de escaneado de caminos, utilizando PS como acrónimo, la heurística de inserción, I, o de vecinos más próximos, N. Se han probado cinco juegos de valores para los parámetros α, β, y ρ iguales a (0.2,1,0.1), (1,5,0.1), (1,1,0.1), (5,1,0.1) y (1,0.2,0.1) que se denotarán, respectivamente, en la experiencia computacional como 0, 1, 2, 3 y 4. Adicionalmente, el algoritmo ACS posee otro parámetro de control, número de elementos que componen la lista de candidatos restringida, que ha sido testeado con dos valores (5 y 10).

Cada algoritmo basado en hormigas, por tanto, puede denotarse mediante un juego de 3 o 4 elementos, el primero representando el tipo de procedimiento, el segundo el tipo de algoritmo heurístico en que se basa, el tercero indicando la terna de parámetros de control del algoritmo y el cuarto que representará el elemento adicional de control presente en el algoritmo ACS.

Por su parte los algoritmos GRASP, cuentan con tres parámetros para su definición. El primero equivalente al existente en los algoritmos de hormigas representa el tipo de algoritmo heurístico con que se ha hibridizado la metaheurística. El segundo y el tercero están relacionados a la construcción de la lista de candidatos restringidos. El primer parámetro marca el tipo de corte, 0 porcentual y 1 por cardinalidad, y el tercer parámetro marca en caso porcentual la máxima diferencia aceptable para entrar en la lista restringida entre el mejor indicador y el candidato, y en el caso de cardinalidad, la cardinalidad del conjunto.

A continuación se estudian las desviaciones obtenidas por los procedimientos con las mejores soluciones conocidas de cada colección de problemas, marcando en negrita el mejor valor de discrepancia con el óptimo para cada número de iteraciones.

Algoritmo	Desv.0	Desv.10	Desv.100	Desv.1000	Desv.10000	Desv.50000
ACO_PS_0	88.63	74.16	61.21	40.95	29.70	24.80
ACO_PS_1	45.57	30.57	21.42	20.81	20.79	18.99
ACO_PS_2	88.63	72.50	42.68	33.70	28.64	21.00
ACO_PS_3	88.63	72.14	60.88	59.55	58.73	55.78
ACO_PS_4	108.87	90.99	78.47	67.72	56.70	53.76
ACO_I_0	65.83	52.35	37.76	28.45	20.28	18.33
ACO_I_1	**34.78**	**19.28**	**14.01**	**13.92**	**13.92**	13.92
ACO_I_2	65.83	52.23	37.56	37.22	37.22	37.22
ACO_I_3	65.83	55.85	55.39	54.92	54.39	54.39
ACO_I_4	96.51	84.43	73.01	71.42	70.44	69.28
ACO_N_0	89.35	75.83	61.40	40.66	26.12	20.17
ACO_N_1	41.98	25.74	14.52	14.39	14.32	**13.08**
ACO_N_2	89.35	76.40	44.92	33.67	25.76	17.40
ACO_N_3	89.35	73.99	59.05	58.11	57.90	57.22
ACO_N_4	105.39	89.71	75.99	66.62	56.60	53.16

Tabla 1: Desviación (%) de los algoritmos ACO respecto a la mejor solución conocida para la colección

Algoritmo	Desv.0	Desv.10	Desv.100	Desv.1000	Desv.10000	Desv.50000
GRASP_PS-0-90	32.78	23.27	18.38	16.70	15.41	15.28
GRASP_I-0-90	30.47	20.26	11.99	**8.31**	5.87	4.88
GRASP_N-0-90	**23.62**	**15.41**	14.69	13.27	12.83	12.76
GRASP_PS-0-95	32.63	24.73	22.79	21.24	21.05	20.95
GRASP_I-0-95	31.58	19.97	**11.77**	**8.31**	**5.79**	**4.66**
GRASP_N-0-95	25.02	19.54	17.74	17.43	17.25	17.23
GRASP_PS-1-5	58.33	45.30	37.54	31.12	26.03	22.86
GRASP_I-1-5	54.98	42.54	31.65	25.72	21.15	18.75
GRASP_N-1-5	53.64	42.34	35.09	29.34	24.28	21.56
GRASP_PS-1-10	76.66	61.52	52.80	45.35	39.53	35.42
GRASP_I-1-10	67.79	51.40	42.71	35.59	29.73	27.37
GRASP_N-1-10	74.43	60.80	51.10	43.96	39.54	34.58

Tabla 2: Desviación (%) de los algoritmos GRASP respecto a la mejor solución conocida para la colección

En igualdad de condiciones entre los algoritmos GRASP y ACO, son los primeros los que muestran mejores resultados utilizando sólo la fase constructiva del algoritmo, lo que al procedimiento GRASP a un procedimiento de muestreo, siendo, al igual que en los algoritmos

de hormigas, los algoritmos de inserción y vecinos más próximos los más prometedores, optando por versiones que discriminen los candidatos mediante una margen de empeoramiento aceptable.

El otro algoritmo propuesto, denotado como ACS, hace uso de ambas técnicas, un rastro para guiar el proceso constructivo y un procedimiento de limitación de las candidatas en cada paso de la iteración. A continuación se muestran las desviaciones sobre la mejor solución conocida por las variantes más prometedoras de esta propuesta.

Algoritmo	Desv.0	Desv.10	Desv.100	Desv.1000	Desv.10000	Desv.50000
ACS_PS_01-5	38.09	25.43	19.88	18.14	17.91	16.72
ACS_PS_02-5	49.05	37.44	23.43	19.66	18.53	17.45
ACS_PS_04-5	51.62	44.01	25.64	22.02	21.22	18.96
ACS_I_00-5	32.80	19.78	**12.43**	**8.70**	**5.67**	**4.16**
ACS_I_01-5	**28.50**	16.18	11.51	11.25	11.25	11.25
ACS_I_02-5	32.80	20.21	14.47	14.47	14.47	14.47
ACS_N_00-5	47.34	31.08	21.90	11.85	7.88	5.26
ACS_N_01-5	31.77	19.10	13.82	12.52	12.52	12.25
ACS_N_02-5	47.34	29.51	16.25	13.63	13.51	12.96
ACS_N_04-5	50.64	36.70	19.49	17.38	16.56	16.56
Algoritmo	Desv.0	Desv.10	Desv.100	Desv.1000	Desv.10000	Desv.50000
ACS_PS_01-10	43.30	29.13	21.17	19.06	18.46	16.30
ACS_I_00-10	41.03	30.50	20.35	12.55	9.69	8.12
ACS_I_01-10	31.08	**15.86**	12.52	12.14	12.14	12.14
ACS_I_02-10	41.03	29.17	20.74	19.95	19.95	19.95
ACS_N_00-10	62.97	46.96	34.56	18.84	10.55	7.78
ACS_N_01-10	35.38	21.46	13.82	13.14	13.05	12.35
ACS_N_02-10	62.97	43.24	22.47	17.45	16.46	14.11

Tabla 2: Desviación (%) de los algoritmos ACS respecto a la mejor solución conocida para la colección

Como puede verse la combinación de ambos procedimientos mejora los resultados obtenidos en ambos casos.

Cabe citar, finalmente, que la inclusión de la fase constructiva de los procedimiento favorece los resultados obtenibles por parte de los algoritmos de hormigas, que depositan rastro una vez se ha realizado la fase de mejora local, aunque los resultados de su fase constructiva básica no se alejan de los procedimientos más simples de muestreo. Por tanto, puede concluirse que en ciertas ocasiones los procedimientos de muestreo básicos pueden resultar mejores que algunas heurísticas más elaboradas para la resolución de problemas combinatorios.

Referencias

[1] Benavent E., V. Campos, A. Corberán y E. Mota (1992) "The capacitated arc routing problem. Lower bounds" *Networks*, 22:669-690.

[2] Colorni A., M. Dorigo y V. Maniezzo (1992). "Distributed Optimization by Ant Colonies". *Proceedings of the First European Conference on Artificial Life, Paris, France*, F.Varela and P.Bourgine (Eds.), Elsevier Publishing, 134-142.

[3] Dorigo M. y M. Gambardella (1997) "Ant Colony System: A cooperative learning approach to the traveling salesman problem". *IEEE Transactions on Evolutionary Computation*, 1(1) 53-66.

[4] Golden B.L., J.S. DeArmon y E.K.Baker (1983) "Computational experiments with algorithms for a class of routing problems" *Computers and Operations Research,* 11(1), pp. 49-66

[5] Golden B.L. y R.T.Wong (1981) "Capacitated arc routing in the soft drink industry". *Operations Research,* 35(1), 6-17

[6] Johnson D.S. y McGeoch L.A. (1997) "The traveling salesman problem: a case study", en Local Search in Combinatorial Optimization. E.Arts y J.K.Lenstra, eds., John Wiley & Sons Ltd., pp.215-310.

[7] Pearn W.L. (1991) "Augment-insert algorithms for the capacitated arc routing problem". *Computers and Operations Research,* 18(2), 189-198

[8] Resende M.G.C. y C.C.Ribeiro, (2002) "Greedy randomized adaptive search procedures", *en Handbook of Metaheuristics*, F. Glover and G. Kochenberger, eds., Kluwer Academic Publishers, pp. 219-249.