COMPARACIÓN DE DOS TIPOS DE VECINDARIOS EN PROBLEMAS DE OPTIMIZACIÓN DE RUTAS A TRAVÉS DE DIFERENTES LIBRERÍAS DE PROBLEMAS

AUTORA: **Cristina R. DELGADO SERNA**

Profesora Asociada del Departamento de Economía Aplicada

Facultad de Ciencias Económicas y Empresariales

Universidad de Burgos

cdelgado@ubu.es

Resumen.-

En los problemas de optimización combinatoria siempre han tenido gran importancia los Algoritmos de Búsqueda Local (o de Búsqueda Vecinal) dentro de las técnicas heurísticas. Sin embargo, a la hora de su aplicación a diferentes modelos siempre aparecen dos inconvenientes: su convergencia a óptimos locales que, en la mayoría de los casos, no son globales; y su dependencia de la solución inicial. En los últimos años han aparecido diferentes estrategias denominadas metaheurísticas que en muchos casos han logrado superar, al menos en parte, estos inconvenientes de la Búsqueda Local y que se pueden considerar una extensión suya al estar basados también en movimientos entre soluciones vecinas. Destacamos entre estas estrategias el Temple Simulado, Búsqueda Tabú, GRASP y, de más reciente aparición, Concentración Heurística y Scatter Search. Sin embargo, un aspecto importante en el diseño de estas estrategias en cada caso, es además de una discusión sobre los parámetros que las definen es una comparación de diferentes tipos de vecindarios para elegir el más adecuado. En este trabajo se comparan dos tipos de vecindarios para el VRPTW Mixto (carga y descarga) a través de instancias recogidas en diferentes librerías disponibles en la red, así como de problemas simulados.

1. INTRODUCCIÓN

El Problema de Rutas de Vehículos con Ventanas de tiempo y con Carga y Descarga Simultánea o sencillamente VRPTW Mixto (Mixed VRPTW) con flota heterogénea puede ser descrito de la forma siguiente: considérese un conjunto de puntos {2, 3, ..., *n*1} donde hay que entregar unas determinadas cantidades de mercancía *q(i)*, *i* = 2, ..., n1, en forma de *palés*, desde un origen 1; además considérese otro conjunto de puntos {*n*1+1, *n*1+2, ..., *n*1+*n*2} donde hay que recoger otras cantidades de *palés q(i)*, *i* = *n*1+1, ..., *n*1+*n*2, y llevarlas al origen 1. Para cumplir estos requerimientos se dispone de una flota heterogénea con diferentes tipos de vehículos; cada tipo de vehículo tiene una capacidad de carga diferente; cada punto del problema lleva asociado un intervalo de tiempo de visita $[e_i, l_i]$, *i* = 2, ..., *n*1+*n*2, (si se llega a *i* antes del instante e_i hay espera, y no puede visitar más tarde del instante l_i). Las distancias d_{ij} y tiempos t_{ij} entre cada par de puntos *i*, *j* $\in$ {1, 2, ..., *n*1+*n*2} son conocidas. (A partir de ahora *n* = *n*1+*n*2). El número de tipos de vehículos disponibles se denota por *ntipos* y las capacidades y costes de cada tipo por *capactipo(i)* y *costetipo(i)* , *i* = 1, ..., *ntipos*; obviamente a más capacidad mas coste.

En este problema se ha de diseñar un conjunto de rutas de coste mínimo verificando las siguientes restricciones:

- Cada ruta comience y finalice en el punto 1;
- Se entrege la mercancía correspondiente a cada uno de los puntos del conjunto {2, .., *n*1}, y se recoja de cada uno de los puntos del conjunto {*n*1+1, *n*1+2,..., *n*1+*n*2};
- El número de *palés* que en cada momento deba transportar cada vehículo no supere su capacidad máxima;
- Cada punto *i*, *i* = 2, .., *n*1+*n*2, sea visitado exactamente una vez:
- Se respeten los intervalos de tiempo de visita.

El coste total a minimizar *f* se descompone en dos partes:

- Una parte proporcional a la distancia total recorrida.
- El coste fijo por cada vehículo (según su tipo).

Este modelo es una generalización de dos subproblemas muy estudiados a lo largo de los años, el VRP ($n2 = 0$; *ntipos* = 1; e_i = -inf.; l_i = +inf.; para $i = 1, ..., n1$), y el VRPTW ($n2 = 0$, *ntipos* = 1),

Existen muchos algoritmos de solución para el VRP y el VRPTW en la literatura. Se pueden encontrar recopilaciones de los principales en trabajos como los de Bodin y Golden (1.981), Desrochers y otros (1.988), Haouri y otros (1.990), y Laporte (1.992). En los últimos años han tomado importancia el desarrollo de algoritmos basados en procesos denominados Metaheurísticos como *Temple Simulado, Búsqueda Tabú, GRASP...* especialmente a partir del trabajo de Gendreu y otros (1.991); como así se refleja en el trabajo de Osman, (1.993), y más recientemente en el de Campos y Mota (1.995) o Kantoravdis (1.995), Rego (1.998).

Como se ha comentado en el resumen, estas estrategias están basadas en muchos casos en el movimiento de soluciones vecinas; por tanto un aspecto fundamental en el diseño de estas estrategias (tan importante sino más que la elección de parámetros), es la definición de la estructura vecinal. En este trabajo se comparan dos tipos de vecindarios que se proponen para este modelo y los resultados obtenidos al insertarlos en procedimientos de Búsqueda Local y Búsqueda Tabú.

En las dos siguientes secciones, se describen las diferentes definiciones de *vecindarios* usados. En la cuarta sección se describe la estructura de los algoritmos empleados. En la quinta se compara los resultados obtenidos en un conjunto de instancias simuladas por los dos tipos de vecindarios al insertarlos en un procedimiento de Búsqueda Local. En la sexta se compara los resultados obtenidos al insertarlo en un procedimiento de Búsqueda Tabú para resolver una serie de instancias del VRP de la conocida librería TSPLIB.

A partir de ahora se denotará por S el conjunto de soluciones factibles del problema y por f la función de costes a minimizar definida en S.

2. VECINDARIO TIPO OR

Se van a considerar dos tipos de soluciones vecinas, unas basadas en la idea propuesta por Or (1.976) para el *Problema del Viajante* o TSP y otras más recientes usadas en los trabajos de Gendreu et al. (1.991), Osman (1.993), Campos y Mota (1.995),... que se explicarán posteriormente en la siguiente sección.

En primer lugar se van a definir como soluciones vecinas las obtenidas por el método de intercambio propuesto por Or (1.976) que sean factibles. El método de intercambio Or es una variante de los conocidos intercambios r-óptimos desarrollados por Lin (1.965) y Lin & Kernighan (1.973) para el TSP simétrico. El método de Or se puede utilizar en problemas asimétricos. La eficacia de este método para el TSP ha sido contrastada en trabajos como el de Nurmi (1.991).

Obsérvese que una solución puede expresarse de forma sencilla, como una única secuencia de puntos; por ejemplo, la solución formada por las dos rutas siguientes

ruta 1: 1 - 3 - 5 - 1; y *ruta 2*: 1 - 4 - 2 - 1

puede expresarse como:

1 - 3 - 5 - 1 - 4 - 2 - 1

los '1' representan la vuelta de un vehículo al origen y la salida del siguiente, (obviamente el último y el primer elemento siempre serán '1'). De esta forma, como se ilustra a continuación, se puede aplicar los Or-intercambios para obtener soluciones vecinas de la actual, considerando a esta como una única secuencia.

Or propone restringir la búsqueda de intercambios a los 3-intercambios en los que cadenas[1] de uno, dos o tres puntos consecutivos son recolocadas entre otros dos. Nótese que con estos intercambios no se cambia el sentido de los diferentes tramos.

---

[1] En este trabajo se denomina *cadena* a toda secuencia de puntos consecutivos en la solución actual.

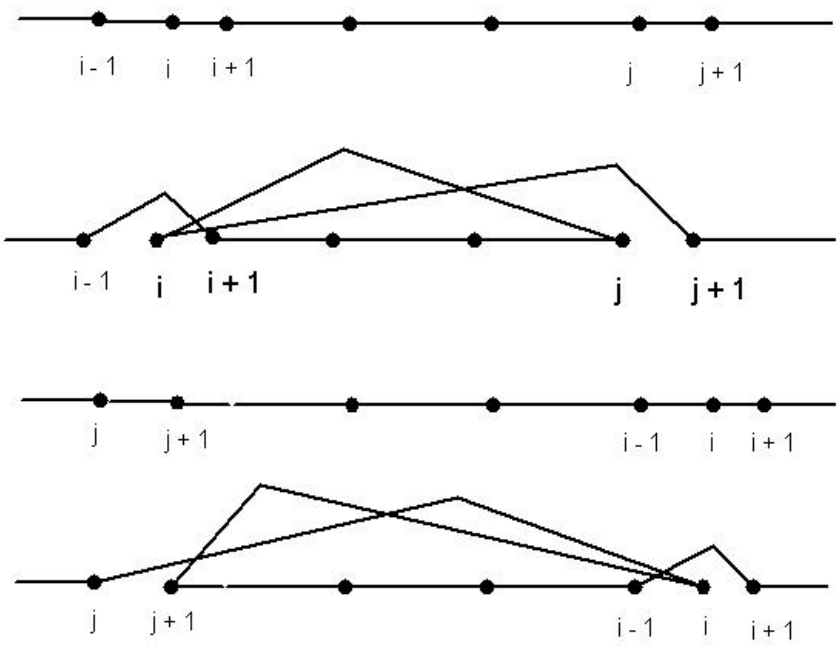


Figura 2.- Posible recolocación del elemento i hacia adelante y hacia atrás entre j y j+1.

En nuestro caso seguiremos la misma idea, pero sólo se consideraremos *recolocaciones hacia adelante* y no pondremos límite al tamaño de la cadena a recolocar (salvo el determinado por el tamaño del problema); además se debe chequear la factibilidad de cada posible recolocación respecto a las restricciones del problema. A continuación se ilustra la recolocación de una cadena de *k* elementos comenzando en *i* entre *j* y *j*+1.

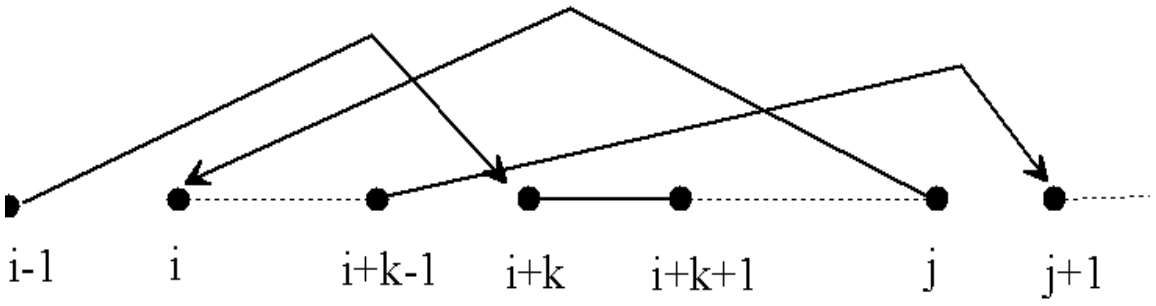


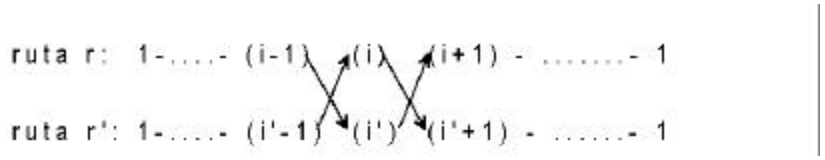
Figura 3.- Recolocación de una cadena de k elementos.

Para cada $s \in S$ se va a denotar por $N^k_1(s)$ al conjunto de soluciones factibles obtenidas por recolocaciones hacia adelante de cadenas de a lo sumo *k* elementos en *s*. Se denota por $N^{\infty}_1(s)$ el conjunto de soluciones factibles obtenidas por todas las recolocaciones.

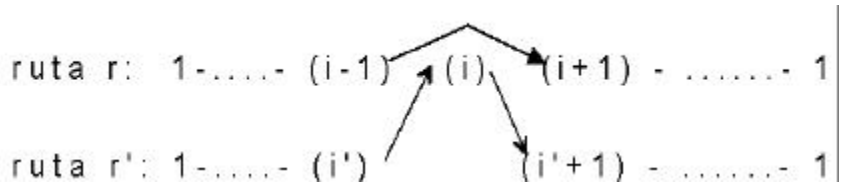
3. VECINDARIOS TIPO GENDREU-CLARKE

El segundo tipo de vecindarios no considera las soluciones como una única secuencia de puntos, sino que sólo considera 3 tipos de intercambios entre 2 rutas diferentes:

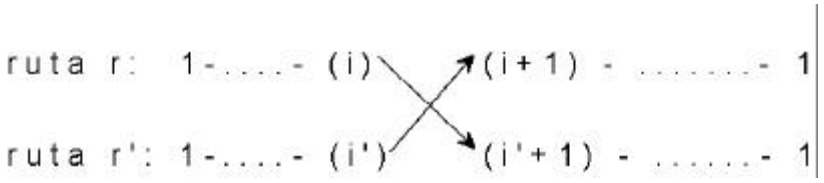
- Tipo I: Intercambio del elemento *i* de la ruta *r* y con el elemento *i'* de la ruta *r'*:
  - Eliminación de los arcos (*i*-1, *i*), (*i*, *i*+1), (*i'*-1, *i'*), (*i'*, *i'*+1)
  - Incorporación de los arcos (*i*-1, *i'*), (*i'*, *i*+1), (*i'*-1, *i*), (*i*, *i'*+1)



- Tipo II: Inserción del elemento *i* de la ruta *r* entre los elementos *i'* e *i'*+1 de la ruta *r'*:
  - Eliminación de los arcos (*i*-1, *i*), (*i*, *i*+1), (*i'*, *i'*+1)
  - Incorporación de los arcos (*i'*, *i*), (*i*, *i'*+1), (*i*-1, *i*+1)



- Tipo III: Cruce de las rutas *r* y *r'* por los elementos *i* e *i'* según la figura:
  - Eliminación de los arcos (*i*, *i*+1), (*i' i'*+1)
  - Incorporación de los arcos (*i'*, *i*+1), (*i*, *i'*+1).



Los dos primeros tipos aparecen en el trabajo de Gendreu et al. (1.991) y otros mencionados anteriormente; el tercero sin embargo no aparece en dichos trabajos: se basa algunas de las ideas propuestas por Clarke and Wright (1.964).

Obsérvese que el tipo I es en realidad un 4-intercambio, el tipo II un 3-intercambio y el tipo III un 2-intercambio. Para cada $s \in S$, se denotará a los conjuntos de soluciones factibles generados por cada uno de estos 3 tipos de intercambio como $N^1_2(s)$, $N^2_2(s)$ y $N^3_2(s)$ respectivamente; asi mismo se denota por $N_2(s)$ a la unión de estos 3 conjuntos.

El chequeo de la factibilidad y la valoración de cada intercambio (tanto de tipo Or como de tipo I, II o III) puede suponer un tiempo de computación excesivo. En los trabajos de Pacheco y Delgado, (1.996) y (1.997b) se propone el uso de variables globales, con lo que el número de operaciones para chequear y evaluar cada intercambio es constante, es decir, independiente del tamaño del problema.

4. DISEÑO DE LOS ALGORITMOS A USAR

Como se ha comentado en la introducción se van a insertar estos vecindarios en procedimientos de Búsqueda Local, y Búsqueda Tabú. A continuación se van a definir los siguientes procedimientos que indican como actúan los respectivos procedimientos de Búsqueda Local.

⇒ Procedimiento Movimiento_Vecinal_Or (*k*, *sf*)

Determinar $s' \in N^k_1(sf)$ *verificando* $f(s') = min\{f(s) / s \in N^k_1(sf)\}$
Si $f(s') < f(sf)$ *entonces hacer* $sf = s'$

⇒ Procedimiento Búsqueda_Local_Or (*k*, *sf*)

Repetir
Ejecutar Movimiento_Vecinal_Or (*k*, *sf*)
hasta que $f(s) \geq f(sf), \forall s \in N^k_1(sf)$.

y análogamente:

⇒ Procedimiento Movimiento_Vecinal_Ge (*k*, *sf*)

Determinar $s' \in N_2(s)$ *verificando* $f(s') = min\{f(s) / s \in N_2(sf)\}$
Si $f(s') < f(sf)$: *hacer* $sf = s'$
Ejecutar Búsqueda_Local_Or (*k*, *r'*) y Búsqueda_Local_Or (*k*, *r''*)
donde r' y r''son las dos rutas de sf modificadas para dar lugar a s'.

⇒ Procedimiento Búsqueda_Local_Ge (*k*, *sf*)

Repetir
Ejecutar Movimiento_Vecinal_Ge(*k*, *sf*)
hasta que $f(s) \geq f(sf), \forall s \in N^k_1(sf)$.

Por su parte las técnicas basadas en procesos de Búsqueda Tabú, Glover, (89) y (90), actúan básicamente de la forma siguiente:

⇒ Procedimiento Búsqueda_Tabú_Básico (*s**)

Hacer $s_0 = s^*$; $T = \varnothing$, *niter = 0, kiter = 0*
Repetir
niter:=niter+1;
Seleccionar $s \in N(s_0)$ / $s \notin T$ *o s verifica criterio de 'aspiración' con f(s) mínimo*
Hacer $s_0 = s$ (*)
Si $f(s_0) < f(s^*)$ *entonces: hacer* $s^* = s_0$ *y kiter = niter*
Actualizar T
hasta niter - kiter $\geq$ *maxiter*

T es el conjunto de *movimientos tabús*. Se considera que una solución *s* cumple el *criterio de aspiración* si $f(s) < f(s^*)$. Este movimiento facilita una nueva dirección de búsqueda y garantiza que no se produzca ciclos.

Cualquier movimiento vecinal de este tipo supone la incorporación de una conjunto de arcos y la eliminación de otros (según se ilustró en los apartado 2 y 3). Para que en las iteraciones siguientes no se vuelva a la soluciones anteriores se impedirán los movimientos que supongan la incorporación en la solución actual de arcos eliminados en iteraciones recientes (*atributos tabú activos*).

Al sustituir en *Búsqueda_Tabú_Básico N* por N^3_1 se obtiene el procedimiento *Búsqueda_Tabú_Básico_Or*. Análogamente al sustituir *N* por N_2 y añadir tras (*) la sentencia *'Ejecutar Búsqueda_Local_Or* (*k, r'*) y *Búsqueda_Local_Or* (*k, r''*) donde *r'* y *r''* son las dos rutas de s_0 modificadas para dar lugar a *s'* se obtiene *Búsqueda_Tabú_Básico_Ge*. En las siguientes secciones se comparan estos procedimeintos.

5. BÚSQUEDA LOCAL CON INSTANCIAS SIMULADAS

Se van a comparar los procedimientos Búsqueda_Local_Or con $k = 3$ y $k = ¥$, y Búsqueda_Local_Ge con $k = 3$. Para ello se han considerado dos tipos de instancias, el primero con descarga pura ($n2 = 0$), y el segundo con igual puntos de carga y descarga ($n1 = n2$); para cada tipo se consideran 10 tamaños deferentes: 10 puntos, 20 puntos, ..., hasta 100 puntos. Para cada tipo y tamaño se generan 10 instancias. Los datos de cada instancia se definen de la forma siguiente:

- Se asigna a cada punto del problema dos coordenadas x e y, cuyos valores son generados aleatoriamente con distribución uniforme entre 0 y 100. La distancia entre cada par de puntos se define como la distancia euclídea correspondiente. Los tiempos (en minutos) de trayecto se consideran igual a la distancia (en Kms.). (Es decir consideramos una velocidad de 60 kms/hora). El coste por Km. es de 1 u.m.

- A e_i y l_i, para $i = 2, ..., n1+n2$, se les asigna respectivamente dos valores enteros aleatorios generados uniformemente: entre 600 y 720 (minutos) en el primer caso y entre 960 y 1.080 en el segundo. Se hace e_1 igual a 480 y l_1 igual a 1.200.

- A cada $q(i)$, $i = 2, ..., n1+n2$, se le asigna un valor entero generado uniformemente de forma aleatoria entre 1 y 12.

- En todos los casos se supone dos tipos de vehículos, con capacidades 10 y 20 y con costes 1.000 y 1.200 u.m. respectivamente.

Los 3 procedimientos de Búsqueda local utilizan la misma solución inicial obtenida por una adaptación a este modelo del algoritmo de *inserción más lejana*. En las tablas siguientes se muestran los resultados medios de los costes y los tiempos de computación según el tipo de instancia y tamaño.

TIPO I ($n2 = 0$)

Tamaño	Solución Inicial		B. Local_Or (k=3)		B. Local_Or (k=¥)		B. Local_Ge (k=3)	
10	3885.3	0.091	3709.4	0.000	3606.2	0.06	3589.1	0.09
20	7742.1	0.240	7517.4	0.015	7449.3	0.701	7299.8	0.462
30	11194.3	0.570	10962.1	0.420	10850.5	2.555	10829.0	0.890
40	16057.6	1.171	15680.3	0.893	15458.2	8.973	15303.5	2.475
50	20278.8	2.023	19731.4	1.802	19288.1	26.899	19397.1	4.346
60	24140.1	3.277	23653.4	2.693	23210.8	55.809	23080.0	6.358
70	27691.1	5.008	27199.1	3.957	26523.5	92.652	26418.5	10.504
80	31425.8	7.290	31119.2	3.818	30289.5	163.276	30154.6	15.891
90	34298.4	9.924	33724.9	7.700	32758.0	304.917	32678.9	22.421
100	39609.5	13.428	39066.0	10.306	38103.0	458.801	37866.6	30.425

TIPO II ($n1 = n2$)

Tamaño	Solución Inicial		B. Local_Or (k=3)		B. Local_Or (k=¥)		B. Local_Ge (k=3)	
10	2615.4	0.090	2581.8	0.030	2558.6	0.060	2567.9	0.080
20	5054.4	0.310	4940.7	0.190	4828.7	0.481	4687.7	0.311
30	6780.8	0.811	6665.1	0.390	6610.3	1.674	6592.3	0.632
40	8700.0	1.722	8530.4	0.904	8369.1	5.488	8397.3	1.350
50	11708.2	3.145	11193.0	1.843	11052.3	17.114	11072.4	3.074
60	13628.5	5.335	13186.9	2.846	12887.0	25.306	12926.7	5.068
70	15628.8	8.381	15267.3	3.997	15014.1	56.522	14927.8	7.792
80	18115.1	12.258	17845.8	4.735	17317.1	80.485	17370.0	11.544
90	19656.5	17.327	18983.5	8.563	18344.8	146.381	18358.7	17.614
100	21997.7	23.624	21546.6	11.756	21097.2	258.811	20992.3	25.015

6. BÚSQUEDA TABÚ CON TSPLIB

Para complementar las pruebas realizados, probamos los vecindarios y movimientos vecinales propuestos insertándolos en el procedimientos de Búsqueda Tabú Básico, es decir, comparamos *Búsqueda_Tabú_Básico_Or* y *Búsqueda_Tabú_Básico_Ge* usando las instancias del VRP de la famosa librería de problemas TSPLIB (1.997), mantenida por Gerard Reinelt. A continuación se indican los resultados correspondientes a cada uno de los dos algoritmos, indicando el coste de la solución y el tiempo de computación empleado para obtener dicha solución. Se añaden dos columnas: una en la que se indica la mejor solución conocida obtenida hasta este momento (según información que muestra la propia pagina donde está TSPLIB), y otra en la que se muestran los resultados obtenidos por un nuevo algoritmo consistente en añadir a *Búsqueda_Tabú_Básico_Ge* una serie de elementos que se explicarán en trabajos posteriores.

Problema	***Bús.Tabú_Básico_Or***		***Bús.Tabú_Básico_Ge***		**Mejor Solución**	**Nuevo Algoritmo**
att48	42113	0.63	40101	3.38	12649,00	40003,00
eil7	104	0.00	104	0.01	104,00	104,00
eil13	250	0.41	247	0.21	247,00	247,00
eil22	379	0.81	375	1.81	375,00	375,00
eil23	570	0.57	569	1.00	569,00	569,00
eil30	509	6.24	505	0.36	534,00	503,00
eil31	474	1.64	383	36.63	1212,00	379,00
eil33	935	0.09	835	17.89	835,00	835,00
eil51	573	5.08	522	15.37	521,00	521,00
eilA76	931	7.60	843	33.48	832,00	832,00
eilB76	1094	21.17	1039	21.06	1031,00	1023,00
eilC76	835	12.81	738	164.54	735,00	735,00
eilD76	747	59.90	688	109.9	683,00	685,00
eilA101	1074	15.44	827	114.13	817,00	815,00
eilB101	1359	15.82	1076	78.05	1077,00	1076,00

7. CONCLUSIONES

A la vista de los resultados las conclusiones son muy claras de cara a indicarnos que tipo de vecindario es más eficaz:

- *Búsqueda_Local_Ge (k=3)* da, tanto en resultados medios como instancia a instancia, resultados claramente mejores que *Búsqueda_Local_Or (k=3)*. Por contra, el tiempo de computación que emplea *Búsqueda_Local_Ge (k=3)* es mayor que *Búsqueda_Local_Or(k=3)*, siendo entre 2 y 3 veces más lento en la mayoría de los casos. La complejidad en ambos procedimientos es de $\Theta(n^2)$.

- Solamente *Búsqueda_Local_Or (k=∞)* parece que aporta soluciones que "compiten" en algunos casos, especialmente en el segundo tipo de instancias, con *Búsqueda_Local_Ge (k=3)*; pero emplea un tiempo de computación de orden $\Theta(n^3)$ que puede resultar excesivo.

- Las pruebas realizadas con Búsqueda Tabú y TSPLIB confirman lo anterior: clara incapacidad de *Búsqueda_Tabú_Básico_Or* de converger a soluciones de calidad, especialmente a medida que aumenta el tamaño del problema. Por contra *Búsqueda_Tabú_Básico_Ge* ofrece resultados claramente esperanzadores: soluciones cercanas a las mejores conocidas y empleando solamente el procedimiento básico. Al añadir otros elementos (Intensificación, Oscilación) se dan resultados muy interesantes superando en algún caso las mejores soluciones conocidas.

8. REFERENCIAS Y BIBLIOGRAFÍA

BODIN L. D. and GOLDEN B. L. (1.981): "Classification in Vehicle Routing and Scheduling". Networks, vol.11, nº 2, 97-108.

CAMPOS V. y MOTA E. (1.995): "Metaheurísticos para el CVRP". XXII Congreso Nacional de Estadística e Investigación Operativa. Sevilla, Noviembre 1.995.

CLARKE G. and WRIGHT J. W. (1.964): "Scheduling of Vehicles from a Central Depot to a Number of Delevery Points". Oper. Res. 12, 568-581.

DESROCHERS M., LENSTRA J. K., SAVELSBERGH M. W. P. and SOUMIS F. (1.988): "Vehicle Routing with Time Windows: Optimization and Approximation". In Vehicle Routing: Methods and Studies, (Studies in Management Sciences and Systems, vol.16), eds: GOLDEN B. L. and ASSAD A. A., Nort-Holland, 65-84.

GENDREU M., HERTZ A. and LAPORTE G. (1.991): "A Tabu Search Heuristic for Vehicle Routing Problem". Report CRT-777. Centre de Recherche sur les Transports. Univ. Montréal.

GLOVER F. (1.989). Tabú Search: Part I. ORSA Journal on Computing, Vol 1, pp. 190-206.

GLOVER F. (1.990). Tabú Search: Part II. ORSA Journal on Computing, Vol 2, pp. 4-32.

HAOUARI M., DEJAX P. et DESROCHERS M. (1.990): "Les Problems de Tournées avec Contraintes des Fenênetres de Temps: L'Etat de l'Art". Recherche Operationnelle/Operations Research, vol. 24, nº 3, 217-244.

KONTORAVDIS G. and BARD J. F. (1.995): "A GRASP for the Vehicle Routimg Problem with Time Windows". ORSA Journal on Computing, 7: 10-23.

LAPORTE G. (1.992): "The Vehicle Routing Problem: An overview of exact and approximate algorithms". European Journal of Operations Research, 59, 345-358.

LIN S. (1.965): "Computer Solutions to the Traveling Salesman Problem". Bell Syst.Tech.Jou., vol 44, 2245-2269.

LIN S. y KERNIGHAN B. W. (1.973): "An Effective Heuristic Algorithm for the Traveling Salesman Problem". Operations Researsch, vol.20, 498-516.

NURMI K. (1.991): "Traveling Salesman Problem Tools for Microcomputers". Computers & Ops.Res. Vol. 18, nº 8, 741-749.

OR I. (1.976). Traveling Salesman Type Combinatorial Problems y their Relations to the Logistics of Blood Banking. Ph.Thesis, Dpt. of Industrial Enginnering y Management Sciences, Northwestern Univ.

OSMAN I. H. (1.993): "Metastrategy Simulated Annealing and Tabu Search Algorithms for the Vehicle Routing Problem". Annals of Operations Research, 41, pg. 421-451.

PACHECO J. y DELGADO C. (1.996). Adaptación del Algoritmo de Or al VRPTW con Carga y Descarga simultánea. X Reunión ASEPELT-ESPAÑA , Albacete, Junio 1.996.

PACHECO J. y DELGADO C. (1.997b). "Problemas de Rutas con Ventanas de tiempo y carga y Descarga simultánea: Diseño de Filtros para algoritmos de intercambio (caso de un sólo vehículo)". Estudios de Economía Aplicada, nº 7 , pgs. 79-100.

REGO C. (1.998): "A Subpath Ejection Method for the Vehicle Routing Problem". Management Science, vol.44, nº 10, pg. 1447-1459.

TSPLIB: "http://w.w.w.iwr.uni-heildelberg.de/iwr/comopt/soft/TSPLIB95/CVRP.html"