Algoritmos Celulares con Operadores Específicos para Resolver un Problema de Ruteo de Vehículos

Carlos Bermudez[1], Carolina Salto, Hugo Alfonso[2]

Laboratorio de Investigación en Sistemas Inteligentes
Facultad de Ingeniería – Universidad Nacional de La Pampa
Calle 9 esq. 110, (6360) General Pico – La Pampa – Argentina.
[1]bermudezc@yahoo.com.ar, [2]{saltoc, alfonsoh}@ing.unlpam.edu.ar

Abstract. El Problema de Ruteo de Vehículos con Capacidad limitada consiste en encontrar la mejor asignación de órdenes de transporte a una determinada flota de vehículos para cumplir con todas las órdenes de entrega, minimizando el costo de transporte sin dejar de considerar la capacidad máxima de cada una de las unidades. En un trabajo previo diseñamos un operador de recombinación específico para este problema con el que se obtuvieron mejores resultados que con operadores tradicionales utilizados en la literatura para resolver este problema. El objetivo de este trabajo es analizar la incorporación de nuestro operador de recombinación en un algoritmo evolutivo celular, que mostró un buen desempeño para resolver este problema. Los resultados obtenidos en este trabajo permiten suponer que la incorporación de este operador en el proceso evolutivo de un algoritmo celular logra obtener buenas soluciones al problema en estudio, y para algunas instancias analizadas mejorar el rendimiento de otras propuestas.

Keywords: Capacitated Vehicle Routing Problem, Recombinación, Algoritmo Evolutivo Celular

1. INTRODUCCIÓN

Los Algoritmos Evolutivos (*Eas – Evolutionary Algorithms*) son técnicas de optimización que trabajan sobre un conjunto (población) de potenciales soluciones (individuos) para que mediante la aplicación de un conjunto de operadores estocásticos puedan progresar en la búsqueda de la solución óptima a un dado problema. El modelo más difundido en sus orígenes es el denominado panmíctico, donde se trabaja en el proceso evolutivo con una única población. Desde hace poco más de una década surgieron otros modelos que se caracterizan por hacer un *tratamiento descentralizado* de la población única que maneja el modelo panmíctico. Los modelos más difundidos son los *Algoritmos Evolutivos Distribuidos (dEA –distributed Evolutionary Algorithms)* [24] y *Algoritmos Evolutivos Celulares (cEA – cellular Evolutionary Algorithms)* [23]. Generalmente se clasifican a los EA descentralizados como de *grano grueso* o de *grano fino* según la relación que presenten entre computación y comunicación. Se desprende de ello, que los *EA descentralizados* trabajan sobre distintos *gránulos*, pudiendo cada uno de ellos llevar adelante su propio EA, lo que implica definir sus propias estructuras y su propio plan de evolución.
Está demostrado [3,8] que las plantillas de un sistema adaptativo granulado, como lo es un EA descentralizado, es lo suficientemente potente y general como para realizar un cómputo arbitrario. Dicha demostración se basa en el hecho de que cualquier computación puede ser representada como una máquina de Turing no determinista [20] (asumiendo modelos computacionales discretos) y mostrando que un sistema adaptativo puede simularla.

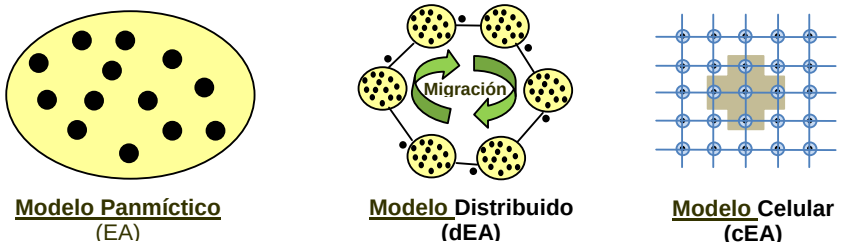

Figura 1.Tipos de Algoritmos Evolutivos según manejo de la población

La diferencia principal entre los dEA y cEA estriba en que para los dEA cada gránulo contiene una subpoblación independiente (isla) de tamaño arbitrario sobre la que se seleccionará el conjunto de soluciones padres a recombinar

para la generación de las soluciones hijas, que podrán competir o no con los padres para pasar a la próxima generación. En el caso de los cEA, cada gránulo tiene definido un vecindario propio sobre el que se seleccionarán las soluciones padres a recombinar para la generación de las soluciones hijas, entre las cuales se definirá la solución que quedará en ese gránulo. Este algoritmo sincroniza los gránulos al final de cada ciclo reproductor, en cambio en los dEA la sincronización se hace con menor frecuencia. La Figura 1 intenta resaltar la diferencia de tratamiento de las poblaciones de soluciones tanto para el enfoque centralizado (de única población o panmíctico) y los enfoques que distribuyen la población (modelos distribuido y celular).

El problema que abordaremos en este trabajo es el conocido Problema de Ruteo de Vehículos (*VRP – Vehicle Routing Problem*), el objetivo de este problema es encontrar el mejor plan de ruta que permita asignar a cada uno de los vehículos de la flota un conjunto de clientes a los que debe entregar la mercadería. Este sencillo problema se torna irresoluble en la medida que se agreguen más clientes a visitar o se incorporen más consideraciones a contemplar en su resolución, por ello reciben la atención de la comunidad científica. Algunas soluciones o ideas que permiten avanzar en la resolución están relacionadas al uso de metaheurísticas. Una de las últimas, que mejores resultados reporta, es a través del uso de un cEA pero con la particularidad de que aborda el problema utilizando operadores de recombinación y mutación genéricos para individuos representados como permutaciones de números enteros.

Nuestro último aporte a la resolución de este problema fue utilizando un AG (Algoritmo Genético) para el cual diseñamos un operador de recombinación específico para el problema denominado BRBAX (Best Route Better Adjustment recombination) [15]. Éste transmite la mejor ruta (grupo de clientes) de un padre al hijo, se considera una buena ruta aquella que hace un mejor uso de la capacidad del vehículo y también minimiza la distancia total del viaje. El uso de este operador mejoró la calidad de los resultados respecto a los AG que usaban operadores genéricos.

Este trabajo se centra en el análisis del rendimiento de un Algoritmo Evolutivo Celular al que se le incorporó nuestro operador de recombinación BRBAX para resolver el CVRP (Capacitated Vehicle Routing Problem), variante del VRP que agrega la restricción de capacidad de cada vehículo. La idea central de nuestro estudio empírico es evaluar el nivel de mejoras alcanzadas con la utilización de un algoritmo evolutivo celular respecto del algoritmo genético usado previamente para incorporar el operador BRBAX. Y además cotejar nuestros resultados contra los resultados de otras opciones existentes en la literatura que resuelven la misma clase de problema de ruteo.

Este trabajo se organiza de la siguiente forma, inicialmente en la Sección 2 se describe el problema a abordar, luego en la Sección 3 se caracteriza el algoritmo celular a utilizar. En la cuarta sección se brindan detalles de la implementación y pruebas a realizar para en las dos últimas secciones detallar resultados alcanzados y conclusiones a las que se arribaron.

2. CVRP

El Problema de Ruteo de Vehículos (VRP - Vehicle Routing Problem)[9] consiste en encontrar un plan de entrega de mercancía a un conjunto de clientes que realizaron pedidos, minimizando el costo de recorrido de los vehículos, que deben comenzar y finalizar su recorrido en el depósito. El VRP está clasificado como un problema NP-duro [18] y tiene muchas aplicaciones industriales siendo abordadas desde un punto de vista teórico como así también desde uno práctico. Este problema dio origen a una serie de subproblemas que agregan distintos tipos de restricciones al original, una de ellas es el CVRP (Capacitated Vehicle Routing Problem) en el cual se debe considerar en la planificación la restricción de la capacidad de los vehículos de transporte. Esta variante será la utilizada en este trabajo.

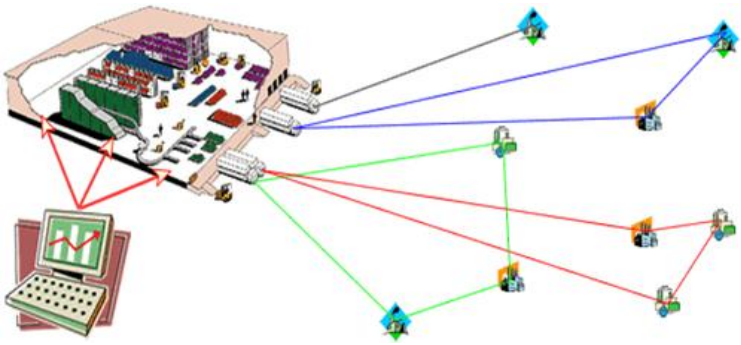

Figura 2 **.** Caracterización de un problema de ruteo

El CVRP puede ser definido como el grafo completo no dirigido $G = (V, E)$ consistente de $c + 1$ nodos (V), y un conjunto de arcos (E) con pesos no negativos que referencian los tiempos de traslado entre ambos. Los nodos representan c clientes y un nodo adicional que representa al depósito. Cada clientes i ($i \in (1,2,..,c)$) tiene asociada una determinada demanda q_i. A su vez se cuenta con k vehículos idénticos de capacidad limitada Q, que serán utilizados para hacer los repartos de mercancías a los clientes. Cada ruta comienza y finaliza en el depósito y cada cliente debe ser

visitado por sólo un vehículo, no pudiendo fraccionar el pedido. El problema es minimizar la distancia total de un determinado plan de ruta y no excediendo la capacidad *Q* del vehículo con la mercancía en el tramo de la ruta a recorrer y la duración de cada una de las rutas no exceda un límite máximo *L* (límite de duración de la ruta). Cabe aclarar que los problemas de CVRP que consideran esta última restricción frecuentemente se los identifican con la sigla DVRP (Distance-constrained VRP)
Por ser VRP un problema de gran relevancia práctica se propusieron diversos métodos heurísticos y metaheurísticos para resolverlo, por ejemplo Búsqueda Tabú [13], Recocido Simulado [19], Colonia de Hormigas[6], Algoritmo Evolutivo[4, 25, 15], entre otros.

3. ALGORITMOS CELULARES PARA EL CVRP

Los *Algoritmos Evolutivos Celulares(cEA)* que trabajan sobre una estructura espacial teniendo un fuerte avance en la última década, ya que ellos proveen algunas características interesantes respecto al mantenimiento de la diversidad poblacional y el ajuste flexible de la presión selectiva y mayor eficacia [10, 22]. Además, existe un campo de trabajo interesante si consideramos su paralelización en máquinas SIMD en las que se asigna un individuo por procesador, pero dejando la posibilidad también de hacer la implementación sobre una única máquina en forma secuencial, aprovechando así las propiedades algorítmicas y no las físicas.

De todos los posibles tipos de estructuras espaciales, se seleccionó para hacer este trabajo la topología que responde a una forma de rejilla, y dentro de todos los posibles tipos de vecindarios particularmente se eligió el *NEWS (North-East-West-South)*. Esto no supone perdida alguna de generalidad ya que es posible con este esquema simular la presión selectiva de cualquier otro algoritmo, incluido el panmíctico [14].

La literatura [21] distingue entre vecindarios:

- *Lineales (**Lr**)*. Los vecinos de un punto dado incluyen a las r-1 estructuras más cercanas elegidas sobre los ejes horizontal y vertical.
- *Compactos (**Cr**)*. Los vecinos de un punto dado incluyen a las r-1 estructuras más cercanas, pudiendo describir formas compactas de cuadrados o rombos.

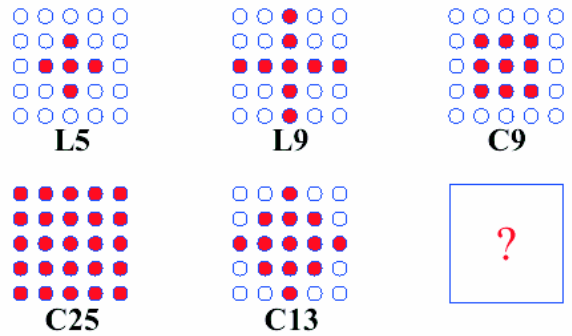

Figura 3**.** Tipos de vecindarios: Lineales y Compactos

Puede observarse en la Figura 3 distintos tipos de vecindarios, de los cuales se puede intuir un diferente costo computacional a la hora de trabajar con cada uno de ellos.
Considerando esta clasificación de los vecindarios, el vecindario NEWS se corresponde al L5 de esa figura. En el caso particular del C25 mostrado en la figura se observa que el vecindario incluye a todos los elementos de la rejilla, lo que implica que el efecto alcanzado es el mismo al considerar una rejilla única (panmixia).

El cEA típico parte de las celdas (individuos), cargados en la rejilla en forma aleatoria, y procede en el Ciclo Evolutivo a actualizarlas sucesivamente, a través del uso de operadores evolutivos, hasta que una determinada condición de terminación se cumpla. Actualizar las celdas en un cEA significa seleccionar dos padres en el vecindario propio del individuo en consideración (incluyendo en la selección al mismo), aplicar los operadores genéticos de recombinación y mutación y finalmente reemplazar la celdas por el nuevo individuo generado, el reemplazo se puede realizar siempre o sólo si es mejor.

El pseudocódigo mostrado en Algoritmo 1 describe, con un alto nivel de abstracción, este mecanismo para una rejilla de dos dimensiones de tamaño (ANCHO x ALTO) y para actualizar simultáneamente todas las celdas utiliza una estructura

auxiliar aux_pop en la que se guardan todos los individuos que pasarían a conformar la nueva estructura, de esta forma pospone el reemplazo por la actual en el momento que toda la estructura fue recorrida. Dicha estructura auxiliar sería innecesaria en el caso de que se realice la implementación en máquinas paralelas.

Algoritmo 1 Pseudocódigo de EA Celular.

```
 1: proc Ciclo_Evolutivo(cea)       //Parámetros del Algoritmo en  `cea'
 2: while no Criterio_Terminación() do
 3:   for x ← 1 to ANCHO do
 4:         for y ← 1 to ALTO do
 5:               n_list ← Calcular_Vecinos(cea,posición(x,y));
 6:               padres← Selección_Entre_Vecinos(n_list);
 7:               aux_indiv ← Recombinar(cea.Pc,padres);
 8:               aux_indiv ← Mutar(cea.Pm,aux_indiv);
 9:               Evaluar_Aptitud(aux_indiv);
10:               Inserta_Nuevo_Indiv(pos(x,y),aux_indiv,cea.{si_mejor|siempre},
                                      aux_pop);
11:         end for
12:   end for
13:   cea.pop ← aux_pop;
14:   Actualizar_Estadísticas(cea);
15: end while
16: end proc Ciclo_Evolutivo;
```

Dependiendo del ALTO y ANCHO definido para la rejilla se pueden obtener distintas formas, que pueden a su vez describir distinto comportamiento por su estrecha relación con la diversidad de la población y la interacción propia del vecindario de cada celda con la de sus vecinos. Las formas que describen, por lo general, se denominan: *Cuadrada*, *Rectangular* o *Alargada*.

La forma de la rejilla puede permanecer inalterable durante la evolución o bien cambiar, ello da origen a dos enfoques distintos:

- **cEA Estático** - sin modificar el tamaño de la rejilla durante la evolución.
- **cEA Dinámico** - Modifica el tamaño de la rejilla durante la evolución atendiendo distintos criterios que contemplan la evolución de las características fenotípica o genotípicas.

Por otra parte, las celdas pueden ser modificadas en forma:

- **Síncrona** - Responde a un procesamiento *paralelo* en la que todas las celdas cambian sus estados en forma simultánea.
- **Asíncrona** - Responde a un procesamiento *secuencial* en la que la actualización de las celdas se hace una por vez y en un determinado orden.

El cEA crea una población inicial de soluciones generadas aleatoriamente, las cuales distribuye en la grilla toroidal y calcula la aptitud de cada una de tales soluciones. Luego comienza el proceso evolutivo tal como se describe en el Algoritmo 1. Recorrerá cada posición de la grilla toroidal y determina el vecindario de cada uno de los individuos dependiendo de la posición, luego aplica los operadores de recombinación y mutación a los individuos seleccionados de tal vecindario. La solución obtenida pasará a la nueva población en idéntica posición, se puede configurar para que ese pasaje se haga siempre o sólo cuando mejora la solución actual. Este proceso evolutivo se realizará hasta cumplir un determinado criterio de terminación. Esta metaheurística al ser poblacional tiene la particularidad de generar un conjunto de soluciones posibles de buena calidad.

La búsqueda es guiada mediante el valor calculado de función objetivo para cada una de las potenciales soluciones, hasta alcanzar un valor óptimo o una solución aceptable sea alcanzada. La función de aptitud calcula un valor de aptitud a cada solución potencial $f(S)$, la cual se calcula de la siguiente forma [14, 15]:

$$f(S) = f_{\text{RoutePlanCost}}(S) + \lambda * \text{overcap}(S)$$

La función $f(S)$ se calcula sumando el costo de todas las rutas realizadas por cada uno de los vehículos de acuerdo al plan de ruta descripto por el individuo ($f_{\text{RoutePlanCost}}(S)$), con la particularidad de que penaliza ese valor de aptitud cuando la capacidad de algunos de los vehículos ha sido excedida respecto al máximo establecido (overcap(S)). Overcap mide

el exceso de carga de cada uno de los vehículos en todas las subrutas. Este valor se ajusta multiplicando un factor λ, en este trabajo se usó $\lambda = 1000$ [11].

La representación utilizada de las soluciones candidatas (individuos) es una permutación de números enteros (siguiendo la idea de Alba y Dorronsoro [1]). Cada número representa un cliente o será un separador de ruta, por ello los números de la permutación deben estar en el rango [1…*n*] donde $n = c + k - 1$ para poder representar una solución para un CVRP con *c* clientes y *k* -1 separadores de rutas. Cada ruta del plan de ruta estará descripta por los números de clientes a visitar entre dos separadores de ruta. Los identificadores de los clientes serán números en el rango [1…*c*], mientras que los *k - 1* separadores de ruta pertenecerán al rango [*c*+1…*n*].
El número de vehículos *k* se calcula de la siguiente manera:

$$k = \text{Demanda\_total\_de\_las\_rutas} / \text{capacidad\_del\_vehículo} * 1.3$$

En este trabajo la longitud de la ruta es minimizada independientemente del número de vehículos utilizados. Pueden obtenerse rutas vacías ubicando dos separadores de ruta en forma contigua.

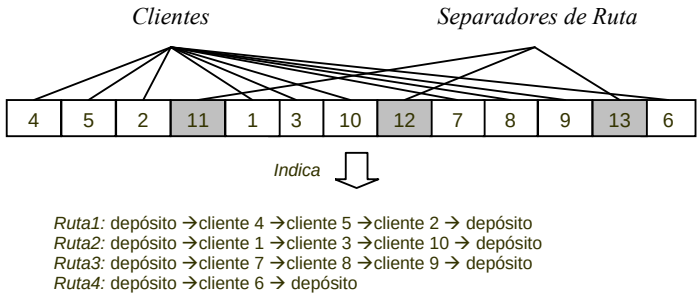

Figura 4. Individuo que representa una solución con 10 clientes y 4 vehículos

Por ejemplo en la Figura 4 se muestra un individuo que representa una posible solución a un hipotético problema CVRP con 10 clientes usando al menos 4 vehículos. Los valores [1, … ,10] representan a los clientes, mientras que los números [11, …, 13] representan a los separadores de ruta. Tal representación indica que el vehículo que recorre la Ruta 1 comienza en el depósito y visita los clientes 4, 5 y 2 (en ese orden), y regresa al depósito, etc.

El operador de recombinación utilizado para la experimentación es el denominado *Best Route Better Adjustment recombination* (BRBAX). Este operador lo diseñamos especialmente para este problema y ha obtenido buena calidad de soluciones en [15]. Éste transmite la mejor ruta (grupo de clientes) de un padre al hijo, considerando una buena ruta aquella que hace un mejor uso de la capacidad del vehículo y también minimiza la distancia total del viaje.

Algoritmo 2 Pseudocódigo de BRBAX

```
 1:   proc Best_Routes (ind1,ind2 )
 2:   i1 ← ind1
 3:   i2 ← ind2
 4:   ordeno_por_aptitud ( i1, i2 ) //i1 es el padre con mejor aptitud
 5:   rutas ← k / 2          //cantidad de rutas a transmitir del 1° padre
 6:    l  ←  obtener_listado_de_rutas (  i1 )
 7:   for  j ← 1 to rutas do              // lleno con el 1° padre
 8:         rn ← Seleccionar_mejor_ruta ( l )
 9:         newind ← copiar_ruta ( rn )
10:         remover_ruta_de_lista ( rn, l )
11:   end for
12:   for i ← ultima_posicion to n do     // lleno con el 2° padre
13:         aux ← retira_primer_elemento_distinto ( newind, i2 )
14:         newind ← copiar_elemento ( aux, i )
15:   end for
16:   return newind
17:   end proc
```

El Algoritmo 2 muestra en pseudocódigo del operador BRBAX, el cual trabaja sobre dos individuos padres (***i*1 e *i*2**), a los que ordena de acuerdo a la aptitud global (**ordeno_por_aptitud**). Del individuo que presente mejor aptitud (***i*1**) se extrae en el vector *l* la mitad de las rutas (**obtener_listado_de_rutas**), **se seleccionarán** ordenadamente aquellas que realicen un mejor aprovechamiento de la capacidad del vehículo. Las rutas extraídas pasan a conformar el nuevo individuo *newind*. Luego se completa el nuevo individuo con los identificadores de clientes o separadores de ruta que aún no se hayan incorporado al nuevo individuo (**retira_primer_elemento_distinto**), de acuerdo al orden de ocurrencia en el segundo padre (***i*2**).

El operador de mutación utilizado combina tres operadores bien conocidos para los problemas de ruteo que son el operador *Insertion* [12], el *Swap* [5] y el *Inversion*[16], que se denomina *"Combined"*[1]. El operador *Combined* aleatoriamente con igual probabilidad aplica uno de los tres operadores tradicionales. El operador *Insertion* selecciona un gen (cliente o separador de ruta) y lo inserta en otro lugar del individuo elegido aleatoriamente, el operador *Swap* elige dos genes en una solución en forma aleatoria y las intercambia. Y el operador *Inversion* elige aleatoriamente dos posiciones de la permutación e invierte entre ambas la secuencia de genes.

4. IMPLEMENTACIÓN

A continuación se describe la experimentación realizada para asegurar su replicación en caso de que sea necesario. El cEA utilizado fue desarrollado por el Grupo Neo y corresponde al paquete Jcell [17] implementado en Java al que se incorporó el operador de recombinación BRBAX. El Algoritmo celular fue configurado para que trabaje en forma estática y la actualización se realice en forma síncrona. Se utilizó una población de 100 individuos distribuidos en una grilla cuadrada, la que se hace evolucionar por 300 ciclos. La población inicial fue generada en forma aleatoria. La selección de los padres se realiza utilizando torneo binario dentro de los individuos que conforman el vecindario de acuerdo el modelo L5. El operador de recombinación BRBAX se aplica en función de la probabilidad Pc seteada en 0.65, mientras que la probabilidad de mutación Pm fue seteada en 0.85.
Además, se incorporó un operador de optimización local que se aplica a todos los individuos que se obtienen luego de la aplicación de los operadores de recombinación y mutación. La optimización local consiste en aplicar los procedimientos λ-Interchange y 2-Opt al mismo individuo mediante 20 movimientos, seleccionar la solución que presente mejor aptitud la que será insertada en la nueva población sólo si es mejor a la solución que se encuentra en esa posición.
Los valores paramétricos se fijaron en estos valores para poder realizar las correspondientes comparaciones a las experimentaciones ya realizadas.

Las instancias utilizadas para la prueba pertenecen al benchmark estándar para CVRP de Christofides, Mingozzi y Toth[7]. Este benchmark consta de catorce problemas de diferente complejidad cubriendo las necesidades de 50 a 200 clientes (**n***xx*) y utilizando entre 5 y 18 vehículos (**k***yy*). Por ejemplo la instancia CMT-n51-k5 referencia que pertenece al benchmark de estos autores, atiende a 51 clientes y utiliza 5 vehículos. Las siete instancias denominadas CMTd, indican además que deben respetar la restricción de que la duración de cada una de las rutas no exceda un límite máximo L (límite de duración de la ruta).

Los algoritmos fueron ejecutados en un AMD Phenom 8450 Triple-core Processor a 2GHz con 2 GB, bajo SLACKWARE 12 con un una versión de kernel 2.6.27.

5. RESULTADOS

En esta sección presentamos los resultados obtenidos durante la experimentación para cada una de las catorce instancias del benchmark. Se realizaron 50 corridas independientes del algoritmo para cada una de las instancias.
La Tabla 1 muestra los resultados alcanzados para cada una de las instancias. En la primer columna siguiente a la denominación de la instancia, se indica el valor de aptitud del **mejor conocido** para el problema. En las restantes columnas se muestran los resultados obtenidos en nuestra experimentación: **mejor encontrado** –indica la mejor aptitud encontrada entre las 50 corridas- , **error** -indica el porcentaje de error de la mejor solución encontrada en la experimentación respecto al mejor valor conocido-, **hits(%)** – indica el porcentaje de veces que se alcanzó el mejor valor conocido, **promedio** – indica la aptitud promedio alcanzada-, **desvio** –indica cuán amplia es la distribución de la aptitud de la mejor solución encontrada respecto del promedio- y la última columna **evaluaciones** – indica el promedio de la cantidad de evaluaciones necesarias para alcanzar la mejor solución.

Tabla 1. Resultados de la experimentación usando el cEA+BRBAX

Instancia	mejor conocido	mejor encontrado	error	hits(%)	promedio	desvio	evaluaciones
CMT-n51-k5	524.61	**524.61**	0.00	84.00	525.22	1.57	106700.00
CMT-n76-k10	835.26	**835.26**	0.00	7.07	840.56	4.71	440994.95
CMT-n101-k8	826.14	**826.14**	0.00	8.00	830.68	3.33	623956.00
CMT-n151-k12	1028.42	1029.87	0.14	0.00	1044.93	4.46	1372206.00
CMT-n200-k17	1291.45	1303.20	0.91	0.00	1324.01	7.11	2175806.00
CMTd-n51-k6	555.43	**555.43**	0.00	18.00	558.64	2.19	131464.00
CMTd-n76-k11	909.68	**909.68**	0.00	34.00	912.84	2.94	222566.00
CMTd-n101-k9	865.94	**865.94**	0.00	44.00	869.85	4.82	402146.00
CMTd-n151-k14	1162.55	1164.12	0.14	0.00	1173.16	7.86	1216652.00
CMTd-n200-k18	1395.85	1406.46	0.76	0.00	1422.05	9.71	1965968.00
CMT-n121-k7	1042.11	1043.89	0.17	0.00	1135.53	44.46	1426162.00
CMT-n101-k10	819.56	**819.56**	0.00	78.00	823.74	7.96	219122.00
CMTd-n121-k11	1541.14	1544.22	0.20	0.00	1563.66	14.04	842158.00
CMTd-n101-k11	866.37	**866.37**	0.00	88.00	867.19	2.40	301860.00

Del análisis de los resultados se puede observar que en 8 de las 14 instancias el algoritmo alcanzó el mejor valor conocido (están resaltados en negrita) y en los casos que no, el error promedio es menor al 1%, lo que evidencia una buena calidad de resultados. El mayor error se observa para las dos instancias que consideran 200 clientes.
Para aquellas instancias que se obtuvo el mejor valor, el porcentaje de veces que se alcanzó el óptimo varía en el rango de 7 a 88%. Las instancias para las que no se ha logrado el mejor valor son todas aquellas que tienen 120 o más clientes y son las que necesitaron realizar una mayor cantidad de evaluaciones, lo que implica un costo computacional mayor.

Para mostrar la mejora en los resultados obtenidos respecto del trabajo anterior, donde se trataba de determinar si el operador propuesto permitía obtener mejores resultados en un algoritmo genético, se muestra la Tabla 2.

Tabla 2. Mejores resultados obtenidos usando BRBAX con nuestros dos algoritmos evolutivos

Instancias	**AG** [15]	cEA	Diferencias
CMT-n76-k10	906,48	**835,26**	71,22
CMT-n101-k8	945,14	**826,14**	119,00
CMT-n151-k12	1377,53	1029,87	347,66
CMT-n200-k17	1964,08	1303,20	660,88
CMT-n121-k7	1737,77	1043,89	693,88
CMT-n101-k10	1062,66	**819,56**	243,10

El análisis comparativo se puede realizar con sólo seis de las instancias utilizadas en este trabajo, debido a que en [15] se utilizaron sólo las instancias puras del CVRP, es decir aquellas que no introducen restricciones de tiempo. Como se puede observar la nueva versión algorítmica, que además de utilizar el operador específico para este problema, usa un procedimiento de optimización local nos permitió mejorar la calidad de todos los resultados, lo que se desprende al leer la columna **diferencia,** que calcula la diferencia entre las aptitudes de las mejores soluciones encontradas con cada uno de los algoritmos. Particularmente, el algoritmo celular en tres de las seis instancias consideradas se obtuvo el mejor valor conocido hasta el momento.

Para poder evaluar el rendimiento del algoritmo celular utilizado respecto a los mejores resultados reportados por Alba y Dorronsoro [2] se muestra la Tabla 3.

Tabla 3. Cuadro comparativo de resultados alcanzados por un cEA+ERX reportados en [2]

y nuestro cEA + BRBAX

Instancia	mejor conocido	mejor		diferencia	error	
		cEA+ERX	cEA+BRBAX		cEA+ERX	cEA+BRBAX
CMT-n51-k5	524.61	524.61	524.61	0.00	0.00	0.00
CMT-n76-k10	835.26	835.26	835.26	0.00	0.00	0.00
CMT-n101-k8	826.14	826.14	826.14	0.00	0.00	0.00
CMT-n101-k10	819.56	819.56	819.56	0.00	0.00	0.00
CMT-n121-k7	1042.11	1042.12	1043.89	-1.77	0.00	0.17
CMT-n151-k12	1028.42	1035.22	1029.87	5.35	0.66	0.14
CMT-n200-k17	1291.45	1315.67	1303.20	12.47	1.88	0.91
CMTd-n51-k6	555.43	555.43	555.43	0.00	0.00	0.00
CMTd-n76-k11	909.68	909.68	909.68	0.00	0.00	0.00
CMTd-n101-k9	865.94	865.94	865.94	0.00	0.00	0.00
CMTd-n101-k11	866.37	866.37	866.37	0.00	0.00	0.00
CMTd-n121-k11	1541.14	1543.63	1544.22	-0.59	0.16	0.20
CMTd-n151-k14	1162.55	1165.1	1164.12	0.98	0.22	0.14
CMTd-n200-k18	1395.85	1410.92	1406.46	4.46	1.08	0.76

El análisis comparativo se puede realizar sobre las catorce instancias que fueron evaluadas en nuestro trabajo aunque el completo trabajo de Alba y Dorronsoro se consideran las instancias de casi todos los benchmarks conocidos sobre este problema.

En este caso estamos básicamente evaluando el aporte del operador de recombinación BRBAX, pues la configuración y parametrización del algoritmo cEA reportada por Alba et al. ha tratado de ser replicada solo cambiando el operador de recombinación ya que en ese trabajo se utilizó el operador ERX (Edge Recombination Operator). En la Tabla 3 para cada instancia se muestra el mejor valor conocido, seguido por los mejores valores obtenidos usando el operador ERX y el BRBAX. La diferencia del rendimiento resultante se observa en la columna **diferencia**, en 8 instancias la calidad es coincidente (distancia =0.00) y de los 6 casos restantes el algoritmo de Alba et al. alcanza una solución con mejor aptitud en 2 instancias (CMT-n121-k7 y CMTd-n121-k11), ambas con 121 ciudades, e inclusive una de ellas alcanza el mejor valor conocido (instancia CMT-n121-k7). En las restantes 4 instancias la incorporación de este operador que considera información propia del problema encuentra mejores soluciones aunque no llega a alcanzar el mejor valor conocido, en consecuencia el porcentaje de error se disminuye. En todos los casos este porcentaje de error es menor al 1%.

En la Figura 5 se muestra un gráfico comparativo del porcentaje de éxitos alcanzados por el cEA que usa el operador de recombinación ERX y nuestra versión que usa el BRBAX. Se observa que en 6 de las instancias esta nueva versión, aquí reportada, alcanza una tasa de éxito mayor. El algoritmo original obtiene un mayor porcentaje en las instancias CMT-n101-k10 y CMTd-n51-k6.

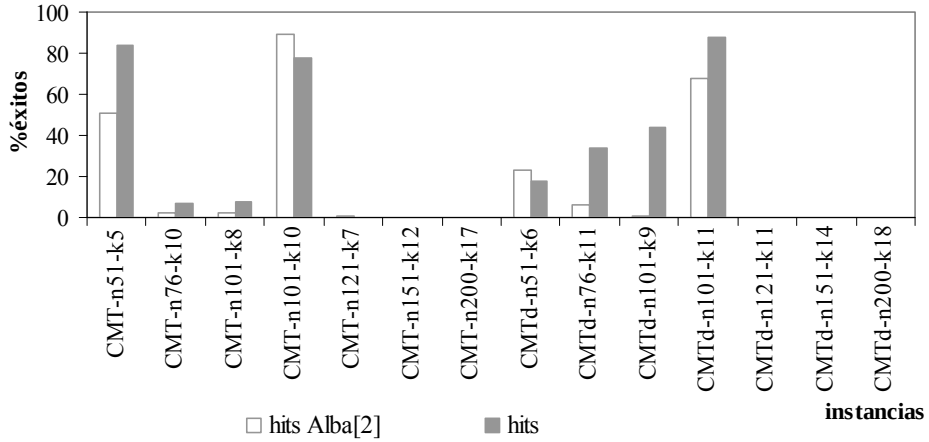

Figura 5. Porcentaje de éxitos alcanzados por los cEA que reportaron los mejores valores [2] y nuestro cEA usando BRBAX

6. CONCLUSION

En este trabajo se analiza el impacto en el comportamiento de un algoritmo evolutivo celular al incorporar un operador con conocimiento específico del problema CVRP. El operador aludido es un operador de recombinación denominado BRBAX, que toma las mejores rutas de sus padres para trasladarlas a la solución hija resultante, el cual mostró un buen rendimiento frente a operadores tradicionales usados en la literatura.
En este caso se incorporó nuestro operador a un Algoritmo Evolutivo Celular que recientemente obtuvo los mejores resultados reportados en la literatura [2]. Podemos considerar que, si bien el salto cualitativo logrado en la resolución de este problema con la utilización de un algoritmo celular, que además utiliza un operador de optimización local en el proceso evolutivo, existen posibilidades de mejorar esos resultados a través de la incorporación de operadores que cuenten con mayor nivel de conocimiento del problema, tal como sucede con el operador BRBAX.

Nuestras ideas a seguir, para dar continuidad a este trabajo, están relacionadas a la adecuación del diseño del BRBAX para considerar las particularidades del problema que cuenta con mayor cantidad de clientes. Por otra parte se puede trabajar con el ajuste del tipo de malla toroidal sobre el que se dispone la población en el cEA para que guíe la búsqueda en cuanto a si se desea explorar o explotar el espacio de soluciones.

RECONOCIMIENTOS

Este trabajo fue desarrollado gracias a los aportes realizados por la Universidad Nacional de La Pampa y la ANPCYT en Argentina, y fundamentalmente por la apertura de la comunidad científica que, además de escribir reportes con detalles técnicos completos, ponen a disposición de la comunidad paquetes algorítmicos que pueden ser modificados y adecuados al mismo o distinto problema tal como el Grupo NEO de la Universidad de Málaga.

REFERENCIAS

[1] E. Alba, B. Dorronsoro, *Computing nine new best-so-far solutions for capacitated VRP with a cellular genetic algorithm*, Information Processing Letters 98, pages 225–230. 2006.

[2] E. Alba, B. Dorronsoro, *A Hybrid Cellular Genetic Algorithm for the Capacitated Vehicle Routing Problem*, in: Engineering Evolutionary Intelligent Systems, Chapter 14, Springer-Verlag, Studies in Computational Intelligence Series, Vol. 82, 2008. ISBN: 978-3-540-75395-7.

[3] T. Bäck. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, Informatik Centrum Dortmund, 1996.

[4] B. M. Bäker, M. A. Ayechew. *A genetic algorithm for the vehicle routing problem*. Computers & Operations Research, pages 787-800. 2003.

[5] W. Banzhaf. *The "molecular" traveling salesman*. Biological Cybernetics 64, pages 7–14. 1990.

[6] J. Bell, P. McMullen. *Ant colony optimization techniques for the vehicle routing problem*. Advanced Engineering Informatics. pages 41-48. 2004.

[7] N. Christofides, A. Mingozzi , P. Toth, *The vehicle routing problem*, Combinatorial Optimization, pages 315–338. 1979.

[8] C. Cotta, E. Alba, J.M. Troya. *Un estudio de la potencia computacional y de la robustez de los algoritmos evolutivos paralelos*. Inteligencia Artificial, 5:6--13, 1998.

[9] Dantzing, G., Ramster, R.: *The truck dispatching problem*. Management Science 6, pages 80-91,1959.

[10] K. De Jong, J. Sarma. *On decentralizing selection algorithms*. In Genetic Algorithms: Proceedings of the Sixth International Conference (ICGA95), pages 17--23, Pittsburgh, PA, July 1995. Morgan Kaufmann.

[11] T. Duncan: *Experiments in the use of neighbourhood search techniques for vehicle routing*. Technical Report AIAI-TR-176, Artificial Intelligence Applications Institute, University of Edinburgh, Edinburgh, 1995.

[12] D. Fogel. *An evolutionary approach to the travelling salesman problem*. Biological Cybernetics 60, pages 139–144. 1988.

[13] M. Gendreau, A. Hertz, G. Laporte. *A tabu search heuristic for the vehicle routing problem.* Manage. Sci. Institute for Operations Research and the Management Sciences (INFORMS), pages 1276-1290. 1994.

[14] V. S. Gordon, D. Whitley. *Serial and parallel genetic algorithms as function optimizers*. Procs. of the 5th ICGA, pages 177--183, 1993.

[15] Graglia P., Stark N., Salto C., Alfonso H., A Comparison of Recombination Operators for Capacitate Vehicle Routing Problem. Proceedings of CACIC 2008.

[16] Holland, J. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI (1975)

[17] *Jcell*, Software provisto por Grupo NEO, Universidad de Málaga, España. Software disponible en http://neo.lcc.uma.es/software/jcell/index.php

[18] J. K. Lenstra, A. H. G. Rinnooy Kan. *Complexity of vehicle routing and scheduling problems*. Networks 11, pages 221-227. 1981.

[19] I. H. Osman. *Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem*. Ann. Operations Research 40(1), pages 421-451. 1993.

[20] C.H. Papadimitriou, I. C. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, Englewood cliffs edition, 1982.

[21] J. Sarma, K.A. De Jong. *An analysis of the effects of neighborhood size and shape on local selection algorithms*. In Proceedings of the Fourth International Conference on Parallel Problem Solving from Nature (PPSN96), pages 22--26, Sept. 1996. Berlin, Germany.

[22] J. Sarma, K.A. De Jong. *An analysis of local selection algorithms in a spatially structured evolutionary algorithm*. In T. Bäck, editor, ICGA97, pages 181--186. Morgan Kaufmann, 1997.

[23] P. Spiessens, B. Manderick. *A massively parallel genetic algorithms: implementation and first analysis*. In Proceedings of the 4th International Conference on Genetic Algorithms, pages 279--285, 1991. Morgan Kaufmann Publishers.

[24] Reiko Tanese. *Distributed genetic algorithms*. In J. D. Schaffer, editor, Proceedings of the Third International Conference on Genetic Algorithms, pages 434--439, San Mateo, CA, 1989. Morgan Kaufman.

[25] Y L Xu, M. H. Lim, M. J. Er. *Investigation on Genetic Representations for Vehicle Routing Problem*, IEEE International Conference on System, Man and Cybernetics, pages 3083- 3088 2005.