CAPÍTULO 7: UN ALGORITMO EVOLUTIVO AVANZADO PARA TSP

7.1. INTRODUCCIÓN

En función a los problemas presentados por los algoritmos evolutivos para resolver eficiente y eficazmente el problema del viajante de comercio, Michalewicz en [109, 139] propone un algoritmo evolutivo basado en un operador especial denominado *inver-over*. Este operador incorpora, en una solución, el conocimiento tomado desde otros individuos de la población. El cual fusiona la inversión y la recombinación. La inversión se aplica a una parte de un único individuo, pero la selección del segmento a invertir depende de otros individuos pertenecientes a la población. El método provee soluciones óptimas o muy cercanas a las mismas velozmente, superando así a otros operadores evolutivos propuestos en el pasado (ver en el capítulo 5). Se explica este método en la sección 7.2.

Como las opciones multirecombinativas [35, 44, 47], que permiten la múltiple aplicación de crossovers sobre los padres seleccionados, han mostrado una mejor performance en problemas de optimización altamente combinatoriales en [110] se muestra una variante del operador inver-over llamada *multi-inver-over*. La cual permite explotar las características de los contribuyentes intervinientes de una nueva solución en el espacio de búsqueda. En la sección 3 se desarrolla esta variante.

En las secciones 4 y 5 se presenta una comparación de los resultados obtenidos con ambas alternativas.

7.2. DESCRIPCIÓN DEL ALGORITMO EVOLUTIVO CON EL OPERADOR *INVER-OVER*

En el algoritmo evolutivo donde se aplica el operador inver-over, cada individuo compite con su hijo. Aquí se usa únicamente este operador, adaptativo, que toma como guía a la población actual. Mientras que sus componentes adaptativos son: el número de inversiones promedio aplicadas a un único

individuo y el segmento a ser invertido (determinado por otro individuo elegido al azar). Además, también es variable el número de veces que el operador se aplica a un individuo.

De lo anterior es posible deducir las siguientes características del algoritmo evolutivo con el operador inver-over [109]:

- Existe una población P con μ individuos, donde $P = \{S_1, S_2, \ldots\ldots\ldots\ldots, S_\mu\}$.
- La representación cromosómatica usada es la representación por permutación.
- Cada individuo compite sólo con su hijo.
- Se utiliza un solo operador de inversión, que no es aleatorio sino adaptativo. Este operador toma como guía a la población actual.
- El número de veces que el operador se aplica a un individuo durante una única generación, es variable. Los componentes adaptativos son: el número de inversiones promedio aplicadas a un único individuo y el segmento a ser invertido (determinado por otro individuo elegido al azar).

La figura 7.1 detalla el algoritmo que describe el uso del operador adaptativo inver-over. Donde *p* es la probabilidad de que la segunda ciudad sea seleccionada aleatoriamente, *rand( )* es una función que genera un número (perteneciente a los reales positivos) en el rango [0, 1], y *eval(S)* es una función que evalúa al individuo *S* calculando la suma de las distancias euclideanas entre las ciudades que forman el ciclo hamiltoniano.

Este algoritmo puede verse como *m*-procedimientos hill-climbing en paralelo. En este procedimiento, el número de inversiones y el segmento a invertirse dependen de la población actual. Este es un algoritmo evolutivo, con un operador adaptativo, el cual es una combinación de las operaciones de inversión y crossover, esto es, cuando la segunda ciudad es seleccionada en base a otro individuo de la población.

Si *rand( ) > p* un segundo individuo, elegido en forma aleatoria desde la población *P*, provee una guía para el segundo marcador de la inversión. Si el operador de inversión es similar al crossover, parte del patrón del segundo individuo aparece en el hijo.

```
Inicialización aleatoria de la población P
while (la condición la condición de terminación no sea satisfecha) do
        for cada individuo Si ∈ P do{
            S' = Si
        selecciona (aleatoriamente) una ciudad c desde S'
            repeat{
            if (rand( ) ≤ p)
                    selecciona la ciudad c' desde las ciudades restantes en S'
                else{
                    select (randomly) an individual from P
                    asigna a c' la "siguiente" ciudad a la ciudad c en el individuo seleccionado
                }
                if (la ciudad siguiente o la ciudad previa de la ciudad c a la ciudad c' en S'
                  exit desde el loop repeat
            invertir la la sección desde la ciudad siguiente de la ciudad c a la ciudad c' en S'
                c = c'
             }
             if (eval(S') ≤ eval(Si))
                Si = S'
        }
}
```

Figura 7.1. El Algoritmo Evolutivo usando el operador *inver-over*

Para ilustrar esta operación se presenta el siguiente ejemplo. Se supone un total de 8 ciudades, es decir, $m = 8$, la siguiente representación (o individuo) describe un tour:

$$S' = (2,8,7,6,5,4,3,1)$$

y que la actual ciudad, c, es 8. Si $rand( ) \leq p$, otra ciudad, c', del mismo individuo S' es seleccionada al azar; para este ejemplo c' es 3, y el segmento correspondiente es invertido creando el siguiente vástago:

$$S' = (2,8,3,4,5,6,7,1)$$

En cambio si *rand( ) > p* se selecciona otro individuo desde la población, por ejemplo:

I = (6,4,7,2,8,5,1,3)

Entonces en este último individuo, *I*, se busca la ciudad $c´$ siguiente a c = 8, es decir que $c´$ = 5, por lo tanto la sección a invertirse en $S´$ comienza después de la ciudad 8 y termina en la 5. Por ende el hijo generado es el que se muestra a continuación:

$S´$= (2,8,5,6,7,4,3,1).

Por último si $eval(S´) \leq eval(S_i)$ entonces el individuo S_i es reemplazado por $S´$ en la población actual. Es por esto, que el padre compite con su hijo.

7.3. EL OPERADOR *INVER-OVER* Y LA MULTIRECOMBINACIÓN

Las opciones de multirecombinación, presentadas en el capítulo 3, forman parte de una familia más amplia de algoritmos evolutivos. Los cuales incluyen múltiples contribuyentes y operadores para intercambiar material genético y son denominados en inglés como "*Multiplicity Feature Evolutionary Algorithms*" (MFEA).

Con una nueva perspectiva a la naturaleza del operador, MFEAs puede también ser divisado por el operador inver-over [110]. La idea es continuar aplicando sobre el mismo individuo $S´$, un número predeterminado n_1 de veces, esperando encontrar mejores soluciones. Esto es, al comparar las evaluaciones de $S´$ y S_i , si $S´$ no supera a S_i entonces la operación de inver-over es repetida sobre S_i; siendo n_1 la máxima cantidad de veces que esta operación se puede repetir. Esta reiteración permite que otros individuos de la población compitan como donadores hasta que eventualmente un mejor vástago sea creado y reemplace a S_i en la población actual.

7.4. EXPERIMENTACIÓN Y ANÁLISIS DE RESULTADOS

Se han realizado cinco diferentes experimentos aplicando los métodos descriptos en las secciones 7.2 y 7.3, *IO-1* a *IO-5*. Donde *IO-1* representa la aplicación del operador inver-over tradicional. Mientras que *IO-2* a *IO-5* representan al algoritmo de multirecombinación donde el valor numérico indica la cantidad n_1 de veces que fue aplicado el operador. Todos ellos se han probado sobre siete instancias para el TSP extraídas de TSPLIB95 [116]. La tabla 7.1 presenta estas instancias:

IDENTIFICADOR	*INSTANCIA*	*MEJOR VALOR CONOCIDO*
Eil51	Eil51 (51 ciudades)	426
Eil76	Eil76 (76 ciudades)	538
KroA100	KroA100 (100 ciudades)	21282
KroD100	KroD100 (100 ciudades)	21294
Pr76	Pr76 (76 ciudades)	108159
St70	St70 (70 ciudades)	675
Bier127	Bier127 (127 ciudades)	118282

Tabla 7.1. Instancias para el TSP

Para cada instancia se han llevado a cabo una serie de diez ejecuciones. En la tabla 7.2 se describen la configuración de los parámetros que se usaron en todos los algoritmos evolutivos con operador inver-over:

Tamaño de la población:	100, 200
Probabilidad *p*:	0.02
Nº máximo de generaciones:	4000
Elitismo:	Sí

Tabla 7.2. Parámetros

Como criterio de terminación se ha decidido detener al algoritmo si el mejor individuo se mantiene sin cambios después de 50 generaciones consecutivas a partir de la generación 200. Como una indicación de la performance del algoritmo se evalúan las siguientes variables:

- ***Ebest*** = (abs (*opt_val* – mejor valor) / *opt_val*)100. Esto es el error porcentual del mejor individuo encontrado cuando es comparado con el valor conocido, estimado u óptimo *opt_val*. Esto da una medida de cuán alejado se está del *opt_val.*
- ***Epop*** = (abs (*opt_val* – fitness promedios poblacional) / *opt_val*)100. Esto es el error porcentual del fitness promedio poblacional cuando es comparado *opt_val*. Esto da una medida de cuán alejado está el individuo promedio del *opt_val.*
- ***Gbest:*** Identifica la generación donde el mejor individuo (retenido por elitismo) fue encontrado.

La tabla 7.3 muestra resultados detallados para la instancia Eil51 bajo cada opción. Los valores promedios para las variables antes mencionadas se muestran en las columnas 2, 3 y 4. La columna 5 muestra el valor de Ebest correspondiente al mejor individuo encontrado en los experimentos.

Instancia Eil51				
Opción	*Gbest Promedio*	*Ebest Promedio*	*Epop Promedio*	*Ebest Mínimo*
IO-1	531.5	7.034321455	19.82355847	4.739703521
IO-2	411	2.877604296	10.69841458	1.14099061
IO-3	313.1	2.495894155	9.628347113	1.543312676
IO-4	271.2	1.805229531	8.646420681	0.674121127
IO-5	276	1.552542559	7.297843404	0.82870493

Tabla 7.3. Valores de las variables de performance para la instancia Eil51.

Es posible observar que todas las opciones que implementan multirecombinación tienen un comportamiento mejor que el presentado por la opción *IO-1*. Considerando la calidad de los individuos mejores y promedios, los valores medios de *Ebest* y *Epop* son mayores sobre *IO-1* para esta instancia. Esto se repite para el *Ebest* mínimo.

Las siguientes figuras muestran un comportamiento promedio para todas las instancias del TSP bajo cada alternativa evolutiva, aquí presentada (*IO-1* a *IO-5*).

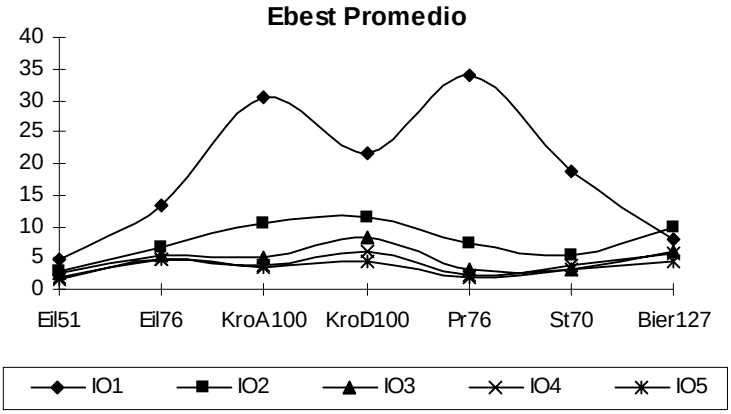

Figura 7.2. Promedio de las medias de los valores de Ebest para todas las instancias bajo cada alternativa.

La figura 7.2, indica la media de los valores de Ebest promedio para todas las instancias bajo cada alternativa. También aquí, se observa un mejor comportamiento promedio bajo las opciones de multirecombinación. Los mejores valores se obtienen con valores de $3 \leq n_1 \leq 5$, destacándose los resultados obtenidos bajo *IO-5*. El cual presenta valores en el rango de 1.6% para Eil51 a 4.7% para Eil76. Los peores resultados se obtienen bajo *IO-1*, en el rango de 7.0% para Eil51 a 34.1% para Pr76.

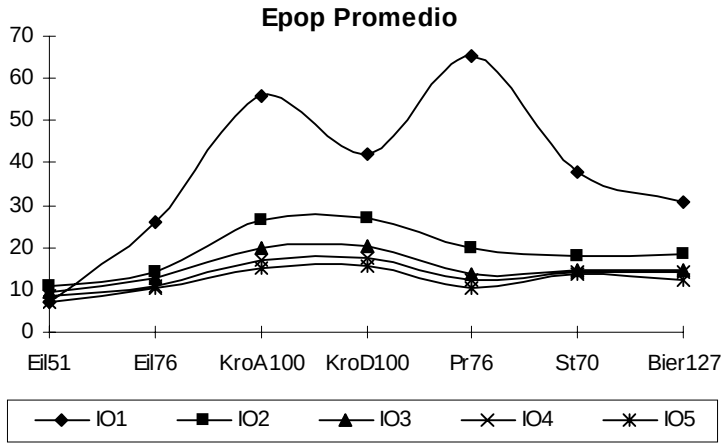

Figura 7.3. Promedio de las medias de los valores de Epop para todas las instancias bajo cada alternativa.

La figura 7.3, muestra la media de los valores de *Epop* promedio para todas las instancias bajo cada opción. En la cual se observa un comportamiento promedio similar. Donde además, los mejores valores se obtienen con $3 \le n_1 \le 5$ e *IO-5* y revelan la mejor conducta, con valores en un rango de 7.3% para Eil51 a 15.6% para KroD100. Nuevamente los peores valores son obtenidos bajo *IO-1* en el rango de 19.8% para Eil51 a 36.51% para Pr76.

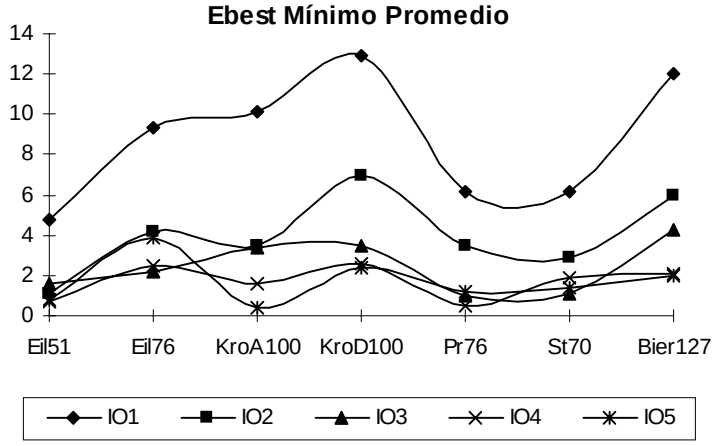

Figura 7.4. Promedio de valores de Ebest mínimo para todas las instancias bajo cada alternativa.

La figura 7.4, indica el promedio de los valores de Ebest mínimo para todas las instancias bajo cada alternativa. Los mejores valores se obtienen nuevamente con $3 \le n_1 \le 5$ en el rango de 0.41% (*IO-5* en KroA100) a 4.2% (*IO-3* en Bier127). *IO-5* brinda los mejores resultados en el rango de 0.41% para Kroa100 hasta 3.84% para Eil76. Mientras que *IO-1* obtiene los valores más desdeñables en un rango que va desde el 4.74% para eil51 a un 12.94% para KroD100.

La figura 7.5, indica el promedio general sobre todos los experimentos de la generación donde se encuentran los individuos mejor adaptados bajo cada alternativa. Como es de suponer, el mayor número de ciudades incrementa el número de generaciones necesarias para encontrar al mejor individuo en todos los algoritmos implementados. Nuevamente se puede detectar una mejor performance con $3 \le n_1 \le 5$. *IO-1*, es la opción que logra la performance más baja, con valores en el rango de las 532 generaciones en Eil51 a 1644 en Bier127 para encontrar la mejor solución. Mientras que todas las alternativas con

multirecombinación realizan una mejor tarea. *IO-4* e *IO-5* presentan los mejores desempeños; ellos necesitan desde 271 y 276 generaciones (en Eil51) hasta un total de 828 y 796 generaciones (en Bier127), respectivamente.

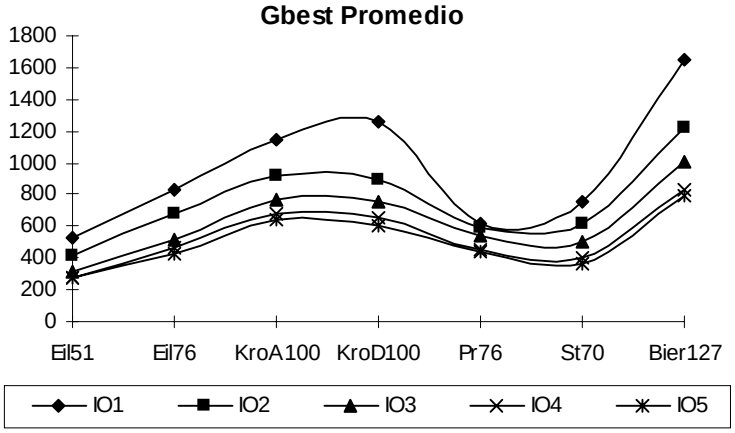

Figura 7.5. Promedio de los valores de Gbest para todas las instancias bajo cada alternativa.

7.5. CONCLUSIONES DE LOS EXPERIMENTOS

Las cuatro variantes con multirecombinación introducidas a la opción de inver-over, aplicadas al TSP: *IO-2 – IO-5*, permiten la múltiple aplicación del operador inver-over a cada individuo en la población. Lo cual se contrasta con el método original *IO-1*.

Todas los algoritmos evaluados siempre generan hijos factibles y se prueban para un conjunto de instancias seleccionadas del TSP.

A la luz de los resultados se puede concluir que:

- Considerando la calidad de los resultados (en los mejores individuos y en los individuos promedios) y la velocidad para encontrar soluciones muy cercanas a la óptima, todos los métodos que incluyen la multirecombinación superan al método original *IO-1*.
- Teniendo en cuenta todas las variables de performance Ebest, Epop, y Gbest antes analizadas, los valores son mejorados al incrementar el número n_1 de las operaciones de inver-over.