Estudio del Espacio de Soluciones del Problema del Cajero Viajante

Osvaldo Gómez
Universidad Nacional de Asunción
Centro Nacional de Computación
San Lorenzo, Paraguay
ogomez@cnc.una.py

Pedro Esteban Gardel Sotomayor
Universidad Nacional de Asunción
Centro Nacional de Computación
San Lorenzo, Paraguay
pgardel@cnc.una.py

Benjamín Barán
Universidad Nacional de Asunción
Centro Nacional de Computación
San Lorenzo, Paraguay
bbaran@cnc.una.py

Resumen

El presente trabajo estudia el espacio de soluciones del *Traveling Salesman Problem (TSP*). Debido al enorme tamaño del espacio de soluciones de los problemas estudiados, se ha decidido tomar muestras de los mismos con el objetivo de tener una visión general de su estructura. Para este fin, se utilizaron dos políticas en la obtención de las muestras, una basada en estudios anteriores y otra aquí propuesta por los autores. Se han tomado veinte instancias del *TSP* de la *TSPLIB*. El análisis de los resultados obtenidos es congruente en todos los casos con la conjetura de un espacio de soluciones globalmente convexo del *TSP*. Esto es, un espacio con características de "*Gran Valle*" como fuera sugerido por Boese, y verificado experimentalmente en el presente trabajo.

Palabras Claves: *Traveling Salesman Problem*, Optimización local, Espacio globalmente convexo.

Abstract

The present paper studies the space of solutions of the Traveling Salesman Problem (TSP). Due to the enormous size of the solutions space of the studied problems, it has been decided to take samples with the objective of having a general vision of the problem structure. For this purpose, two policies were used to obtain the samples, one already used in previous studies and the other designed by the authors. For this study, 20 instances of the *TSP* have been taken from the *TSPLIB*. The analysis of the results was coherent with the conjecture of a globally convex structure of the *TSP*'s solutions space. That is, a space that has characteristics of "*Great Valley*" as it was suggested by Boese and experimentally proved in the present work.

Keywords: Traveling Salesman Problem, Local optimization, Globally convex space.

1 Introducción

El problema del cajero viajante, más conocido por sus siglas en inglés *TSP (Traveling Salesman Problem),* ha sido utilizado como *"Problema de Prueba"* (o problema *paradigma*) para estudiar el comportamiento de muchos algoritmos de optimización [1,2,4,5] por lo que resulta interesante un estudio más extenso del espacio de soluciones del mismo con el fin de conocer su estructura. Este trabajo se inspira en las propuestas de Boese [1,2] y otros autores como Stützle et al. [5] que han realizado algunos estudios similares para instancias específicas del *TSP*, pero no de manera sistemática, como se propone en el presente trabajo. Estos estudios consistieron en general en hallar óptimos locales mediante distintos algoritmos de búsqueda local para luego analizar los resultados obtenidos e intentar entender la estructura del *TSP* para las instancias estudiadas.
En contrapartida a los trabajos citados, el presente trabajo analiza un conjunto de 20 problemas de solución óptima conocida, extraídos de la *TSPLIB* [6], una fuente de problemas referenciales del *TSP*, muy utilizada por diversos investigadores del área. El presente trabajo propone entonces obtener soluciones factibles del espacio de búsqueda (o muestras), compararlas con la solución óptima y con las demás soluciones factibles, para finalmente intentar conocer la estructura del *TSP*. Para obtener estas muestras, se utilizaron básicamente 2 políticas de muestreo:

1. Siguiendo los trabajos de Boese [1,2], se utilizó una heurística de optimización local, conocida como *3-opt*, con iniciación aleatoria.
2. Alternativamente, se generaron muestras de soluciones uniformemente distribuidas en el espacio de soluciones, para cada problema en estudio.

El trabajo fue organizado de la siguiente manera: a continuación se describe el *TSP* y se formaliza la notación a ser utilizada. Seguidamente, se describen los métodos utilizados para obtener las muestras de soluciones del *TSP*. En la cuarta sección se presentan los resultados experimentales, mientras que en la sección 5 se describe la estructura del *TSP* como un espacio globalmente convexo. Finalmente se presentan las conclusiones del estudio.

2 Problema del Cajero Viajante.

El *TSP* puede ser representado por un conjunto de ciudades o nodos *V={N1,N2,…..Nn}* con *n* elementos (o ciudades) y un conjunto *A* de arcos que asocia a cada par de ciudades o nodos *{Ni, Nj}* una distancia *d(i,j)*. La solución del *TSP* es conocida como *tour* (*r*) y está dada por un conjunto ordenado de las *n* ciudades o nodos *{Ni}* que contiene a todas las ciudades del problema exactamente una vez, esto es:

$$r = \{N(r(1)), N(r(2)), ..., N(r(n))\}; \qquad N(r(i)) \in V, \quad N(r(i)) \neq N(r(j)) \text{ si } i \neq j \qquad (1)$$

Cada *tour* tiene asociado una longitud definida como:

$$l(r) = \sum_{i=1}^{n-1} d(N(r(i)), N(r(i+1))) + d(N(r(n), N(r(1))). \qquad (2)$$

El objetivo del *TSP* es hallar al *tour* de menor longitud, también conocido como *tour* óptimo (*r**).

El presente trabajo sólo considera problemas en los que los arcos representen distancias euclidianas entre los nodos *i* y *j*, por lo tanto, *d(i,j) = d(j,i)*. Este tipo de problemas es conocido como *TSP simétrico*. En este contexto, dados dos *tours r1* y *r2*, la distancia $\delta(r1, r2)$ se define como *n* menos la cantidad de arcos comunes a ambos *tours*. Lógicamente, la distancia de un *tour r* a la solución óptima *r** estará denotada como $\delta(r, r^*)$.

Una población *P* es un conjunto de *m* soluciones o *tours*, esto es:

$$\text{Población:} \quad P = \{r1, r2, \ldots, rm\} \qquad (3)$$

La distancia media de un *tour r* a una población *P* se define como

$$\delta(P, r) = \frac{1}{m} \sum_{i=1}^{m} \delta(ri, r) \qquad (4)$$

y da una idea de cuan cerca esta un *tour* a una población *P*. Corresponde hacer notar que dada una población *P* uniformemente distribuida en un sub-espacio, los *tours* que se encuentren en su centro tendrán una menor distancia media a la población que los que se encuentren en su periferia, por lo que $\delta(P,r)$ podrá ser utilizado para estimar la posición de un punto respecto a una población.

3 Métodos de muestreo utilizados para el análisis

Inspirado en los trabajos de Boese [1,2], el primer método utilizado intenta crear una población de soluciones localmente óptimas, para lo que se utilizará un algoritmo que se dará en llamar "*3-opt*". El mismo utiliza una heurística de búsqueda local mostrada en el *Pseudocódigo 1,* para hallar *m* óptimos locales y conformar así una población de estudio *P*. Este acercamiento ya fue abordado por otros autores en [1,3,5] demostrándose que las

soluciones obtenidas se concentran en una zona determinada del espacio de soluciones y no se distribuyen uniformemente en todo el espacio.
A raíz de la distribución no uniforme de la población de óptimos locales encontrada en publicaciones anteriores, este trabajo propone adicionalmente el estudio de una población más uniforme, con soluciones distribuidas a lo largo de un mayor espacio de soluciones, para lo cual se propone un segundo método de selección de muestras. Este método, que damos en llamar "*Dist*", obtiene un conjunto de soluciones con la característica de tener un subconjunto de cardinalidad constante, perteneciente a cada posible distancia del óptimo r^*.

3.1 Descripción del algoritmo *3-opt*

Al inicio, esta heurística guarda en una variable r_{opt} un *tour* hallado aleatoriamente de la siguiente manera: el *tour* comienza en una ciudad cualquiera del problema, luego se elige aleatoriamente otra ciudad de *Vc*, siendo *Vc* el conjunto de ciudades todavía no visitas, y se elimina ésta de *Vc*. El proceso se repite iterativamente hasta que el conjunto *Vc* quede vacío y se hayan visitado todas las ciudades del problema, por lo que solo queda regresar a la ciudad de origen.
Una vez encontrado un *tour* inicial, se realiza iterativamente el siguiente proceso: Se cambian tres arcos elegidos al azar de r_{opt}, dando origen a un nuevo *tour* válido que se guarda en r_{new}. Se comparan las longitudes de r_{opt} y r_{new}, si $l(r_{new})$ es menor, se guarda r_{new} en r_{opt}. Se realiza este proceso hasta que r_{opt} no varíe durante una cantidad de iteraciones (*CI*) igual a 1000 para los resultados experimentales presentados en este trabajo.
El proceso mencionado se repite hasta obtener el Número de Tours (*NT*) deseado para generar la población *P*. Para este trabajo, se escogió $NT = 1000$ tours y se calcularon los correspondientes valores de longitud $l(r)$, distancia al óptimo $\delta(r, r^*)$ y distancia media a la población $\delta(P,r)$, para cada tour $r \in P$.

Pseudocódigo 1: *3-opt*

Parámetros de entrada: matriz de distancias $D = d(i,j)$, número de tours *NT*, cantidad de iteraciones *CI*
Parámetros de salida: población *P*, con los correspondiente valores de $l(r)$, $\delta(r, r^*)$ y $\delta(P,r)$
Inicializar $k = 0$
Repetir mientras $k < NT$
    r_{opt} = *Nuevo tour ( )*
    Hallar $l(r_{opt})$
    Inicializar $c = 0$
    Repetir mientras $c < CI$
        r_{new} = *Cambios* (r_{opt})
        Hallar $l(r_{new})$
        Si $l(r_{new}) < l(r_{opt})$
            $r_{opt} = r_{new}$
            $l(r_{opt}) = l(r_{new})$
            c=0
        Fin si
        c=c+1
    Fin mientras
P(k) = r_{opt}
Calcular $\delta(r_k, r^*)$
Calcular $l(r_k)$
$k = k + 1$
Fin mientras
Calcular $\delta(P,r)$

Función r_{opt} = *Nuevo tour ( )*
r_{opt} = conjunto vacío
Mientras *Vc* no sea vacío
    Se escoge al azar una ciudad (*Ni*) de *Vc*
    Se elimina *Ni* de *Vc*
    r_{opt} = (r_{opt}, *Ni*)
Fin mientras

Función r_{new} = *Cambios* (r_{opt})
Seleccionar tres arcos al azar de r_{opt}
Cortar r_{opt} en los arcos seleccionados
Reordenar los segmentos de r_{opt} para construir un nuevo tour (r_{new}) con $\delta(r_{new}, r_{opt}) = 3$

3.2 Descripción del Algoritmo *Dist*

Este algoritmo tiene como objetivo hallar un numero de *tours* (*NT'*) a todas las distancias posibles del óptimo *r**. Las distancias posibles al óptimo van de 2 hasta *n*, siendo *n* la cantidad de ciudades en el problema. Para cada una de estas distancias se genera entonces un subconjunto de *NT'* soluciones, por lo que el número total de tours estudiados es de: *NT'(n*-2). Por ejemplo, para un problema de 280 ciudades (como el *a280* mostrado en la tabla 1), con *NT'*=24, se analizaron 6672 tours componentes de la población *P*, mientras que para el problema *pr1002* con *NT'=6* (ver Tabla 1) la población *P* contenía solo 6000 tours.
Este algoritmo selecciona arcos al azar en el *tour* óptimo *r** y los reordena formando nuevo *tour* válido r_{new}. Seguidamente se calcula $\delta(r^*, r_{new})$ y se comprueba si r_{new} está a la distancia deseada, de no ser así, se continua el proceso de cambio de arcos. Una vez que se han obtenido la cantidad *NT'* de *tours* deseados a una distancia dada del óptimo, se procede a hallar las siguientes muestras, posiblemente a la distancia siguiente. En consecuencia, el algoritmo también produce una población *P* de *tours*, con sus correspondientes valores de $l(r)$, $\delta(r, r^*)$ y $\delta(P,r)$. A continuación se describe el pseudocódigo correspondiente.

Pseudocódigo 2: *Dist*

Parámetros de entrada: *r**, *NT'*
Parámetros de salida: población *P*, con los correspondiente valores de $l(r)$, $\delta(r, r^*)$ y $\delta(P,r)$
NC = cantidad de ciudades en *r**
Inicializar *P* como conjuntos vacíos
Para *nc* = 2 hasta *NC*
  Para *k* = 1 hasta *NT'*
    r_{new} = *Cambios (r*, nc)*
    Hallar $l(r_{new})$
    $P = [P, r_{new}]$
    $l(r_k) = l(r_{new})$
    Calcular $\delta(r_k, r^*)$]
  Fin para
Fin para
Calcular $\delta(P,r)$

Función *Cambios (r*, nc)*
$r_{new} = r^*$
Mientras $nc > \delta(r_{new}, r^*)$
  Seleccionar arcos al azar de r_{new} y cortar el *tour*
  Reconstruir r_{new} con los segmentos y obtener un nuevo r_{new}
  Hallar $\delta(r_{new}, r^*)$
  Si $\delta(r_{new}, r^*)$ tiene distancia buscada, retornar r_{new}
Fin mientras.

4 Análisis de la estructura del espacio de soluciones del TSP

4.1 Problemas Estudiados

Se utilizaron un total de 20 problemas con soluciones óptimas ya conocidas. Estos problemas ampliamente difundidos [6], se detallan a continuación en la Tabla 1 que muestra el nombre del problema, el número *n* de ciudades, los parámetros *NT* y *NT'*, así como la cantidad de tours analizados.

La cantidad *NT'* = 24 fue elegida para permitir obtener una cantidad apreciable de soluciones en los problemas pequeños. En los problemas de mayor tamaño se disminuyó *NT'* debido al enorme tiempo de procesamiento que requiere poblaciones muy grandes.

Por falta de suficientes recursos computacionales, no se realizó la búsqueda de soluciones con *3-opt* en los problemas de más de 101 ciudades, lo que no invalida las conclusiones del trabajo, dada la uniformidad de los resultados experimentales que serán mostrados en la siguiente sección.

Nombre del Problema en TSPLIB	Número de ciudades (*n*)	*Tours* en cada subconjunto (*NT'*) (usando *Dist*)	Tamaño de la Población *P* (usando *Dist*)	Cantidad de *tours NT* (usando *3-opt*)
fri26	26	24	600	1000
bayg29	29	24	672	1000
bays29	29	24	672	1000
att48	48	24	1128	1000
eil51	51	24	1200	1000
berlin52	52	24	1224	1000
st70	70	24	1656	1000
pr76	76	24	1800	1000
eil76	76	24	1800	1000
kroa100	100	24	2376	1000
kroc100	100	24	2376	1000
krod100	100	24	2376	1000
eil101	101	24	2400	1000
lin105	105	24	2496	
ch130	130	24	3096	
ch150	150	24	3576	
tsp225	225	24	5376	
a280	280	24	6696	
pcb442	442	9	3969	
pr1002	1002	6	6006	

Tabla 1: Resumen de problemas en estudio.

4.2 Presentación de los resultados

En las poblaciones halladas con el algoritmo *Dist* no se consideraron individuos repetidos, es decir, las poblaciones solo contenían una vez a cada solución encontrada. En cambio, en las poblaciones halladas con la heurística *3-opt* se hallaron 1000 individuos diferentes en la mayoría de los problemas, excepto en los siguientes casos:

- En *fri26*, donde se obtuvieron 687 individuos diferentes y se alcanzo el óptimo en 87 ocasiones.
- En *bayg29*, donde se encontraron 877 individuos y se llego al óptimo 57 veces, y
- en *bays29*, donde se obtuvieron 897 *tours* no repetidos y se encontró el óptimo en 35 ocasiones.

Cabe destacar que en ninguno de los demás problemas considerados se halló el óptimo. Para cada población *P* se calcularon los siguientes valores de correlación, mostrados en la Tabla 2:

- La correlación entre la distancia al óptimo $\delta(r, r^*)$ y la longitud del *tour l(r)*, denotado como $\rho(\delta(r, r^*); l(r))$.
- La correlación entre la distancia media a la población $\delta(P,r)$ y la longitud de los *tours l(r)* denotado como $\rho(\delta(P,r); l(r))$.
- La correlación entre la distancia al óptimo $\delta(r, r^*)$ y la distancia media a la población $\delta(P,r)$ denotado como $\rho(\delta(r, r^*); \delta(P,r))$.

Como puede observase en la Tabla 2, las correlaciones calculadas son muy significativas en todos los casos, siendo en general más altas para las poblaciones uniformes generadas con el algoritmo *Dist* que para las poblaciones de óptimos locales generados con la heurística *3-opt*. Cabe notar además que el valor de estas correlaciones crece en general con el tamaño *n* del problema, por lo que se puede presumir que en general, estas correlaciones aumentan con la complejidad del problema, por lo que resulta razonable intentar analizar estos resultados experimentales para intentar conocer la estructura del *TSP*, especialmente para el caso de problemas de creciente complejidad.

Antes de iniciar el referido estudio, se presentan las figuras 1 al 8, ilustrando para algunos de los problemas tipo estudiados las características resaltantes de las 3 variables estudiadas. Debido a la enorme cantidad de problemas estudiados, resulta imposible presentar figuras para cada problema analizado. Seguidamente se presentan los gráficos de los problemas *fri26*, *berlin52*, *kroa100*, *pcb442* y *pr1002*.
Para cada problema se muestran tres gráficos. En las figuras 1, 2 y 3 se presentan los resultados usando la heurística *3-opt* y en las figuras 4 al 8 los resultados usando el algoritmo *Dist*.

Correlaciones Experimentales para poblaciones halladas con el algoritmo *Dist*			
	$\rho(\delta(r,r^*);\ l(r))$	$\rho(\delta(P,r);\ l(r))$	$\rho(\delta(r,r^*);\ \delta(P,r))$
fri26	0.95448091033952	0.95619980040881	0.99968988875798
bayg29	0.96133911986948	0.96009781141200	0.99979852774148
bays29	0.96070250670296	0.95803519904316	0.99984148510148
att48	0.97170200570656	0.97164083616104	0.99996217348502
eil51	0.97995087068297	0.97993644346292	0.99997253792214
berlin52	0.97834824516068	0.97817268025183	0.99996432970952
st70	0.98645264864984	0.98642983813238	0.99998855603249
pr76	0.98678670617501	0.98680411134174	0.99999198845579
eil76	0.98703033452627	0.98703837221474	0.99999132039835
kroa100	0.98807255595606	0.98806085294992	0.99999613193151
kroc100	0.98799045322066	0.98799244346144	0.99999496117053
krod100	0.98889865432748	0.98886823772830	0.99999600926837
eil101	0.99012944939611	0.99011129306537	0.99999623319082
lin105	0.98824266471120	0.98823017753611	0.99999675465118
ch130	0.99274096381607	0.99275294041509	0.99999754184527
ch150	0.99428751931525	0.99428768016066	0.99999847528659
tsp225	0.99561774098757	0.99561979503179	0.99999931367740
a280	0.99645694386082	0.99645797223804	0.99999954320402
pcb442	0.99816600999945	0.99816520843327	0.99999958515832
pr1002	0.99913041235388	0.99913038639726	0.99999989057276
Correlaciones Experimentales para poblaciones halladas con heurística *3-opt*			
	$\rho(\delta(r,r^*);\ l(r))$	$\rho(\delta(P,r);\ l(r))$	$\rho(\delta(r,r^*);\ \delta(P,r))$
fri26	0.79740539145696	0.82356744995048	0.78618554174895
bayg29	0.75737847421895	0.87465290225778	0.88584646493907
bays29	0.69656272456600	0.86772165118454	0.77833781021983
att48	0.59475322499705	0.82072366195982	0.75013111753384
eil51	0.65725125076772	0.91529026136113	0.75356737848526
berlin52	0.71519426726462	0.86492285540173	0.84911820026034
st70	0.60717715417841	0.89953192544241	0.67846357101462
pr76	0.70820045789984	0.88077517172286	0.83383889710767
eil76	0.69559421901902	0.94123566483957	0.77263029989318
kroa100	0.69158036378025	0.90389926628806	0.84802935161287
kroc100	0.69957464878138	0.92368211552670	0.79578866223248
krod100	0.69873339821833	0.91241888905004	0.79223990205942
eil101	0.70788007788079	0.94446148831836	0.77358426643261

Tabla 2: Resumen de los resultados experimentales.

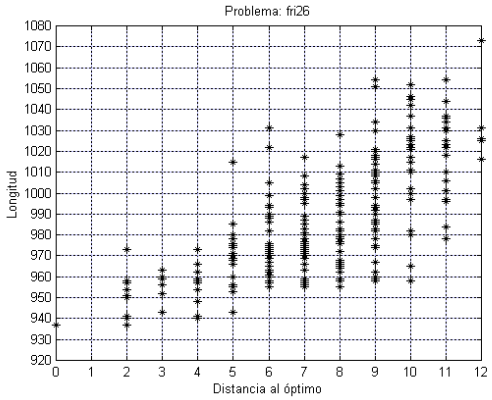

Figura 1.1: *l(r)* en función de *δ(r, r*)*

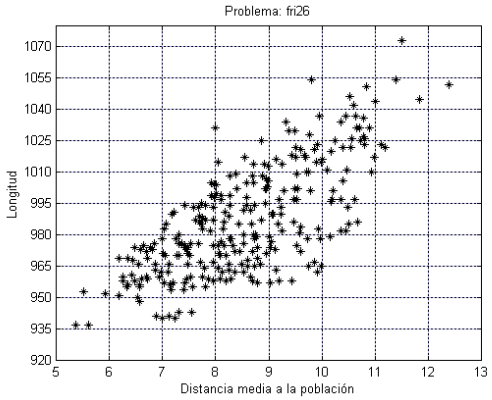

Figura 1.2: *l(r)* en función de *δ(P, r*)*

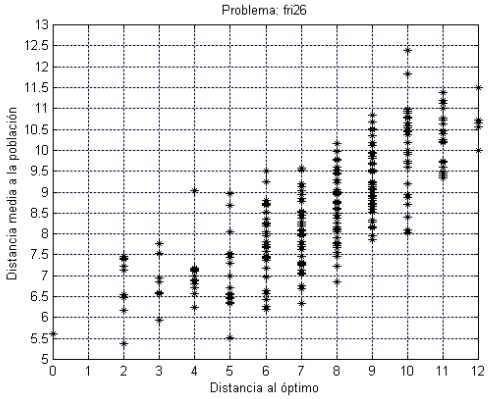

Figura 1.3: *δ(P, r*)* en función de *δ(r, r*)*

Figura 1: Problema *fri26* usando *3-opt*

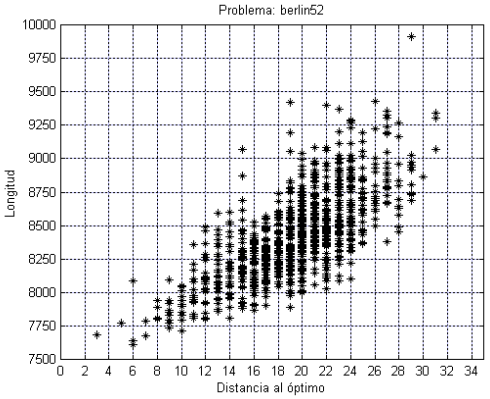

Figura 2.1: *l(r)* en función de *δ(r, r*)*

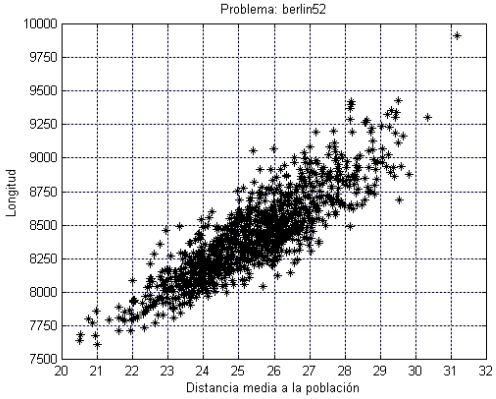

Figura 2.2 *l(r)* en función de *δ(P, r*)*

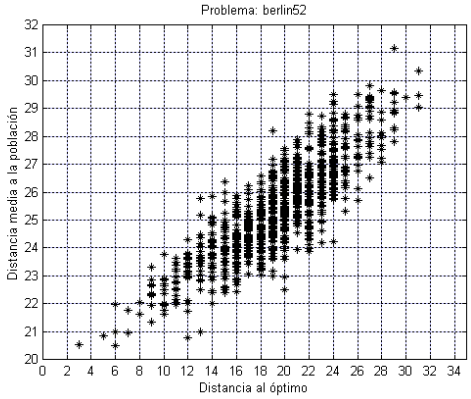

Figura 2.3: *δ(P, r*)* en función de *δ(r, r*)*

Figura 2: Problema *berlin52* usando *3-opt*

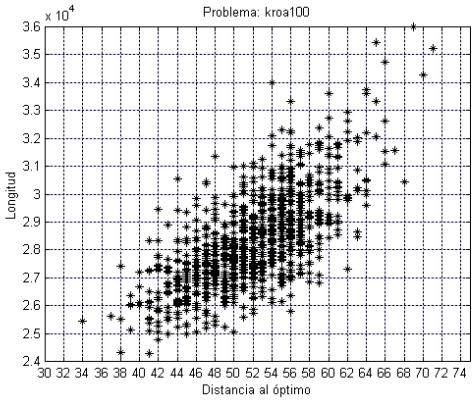

Figura 3.1: *l(r)* en función de *δ(r, r*)*

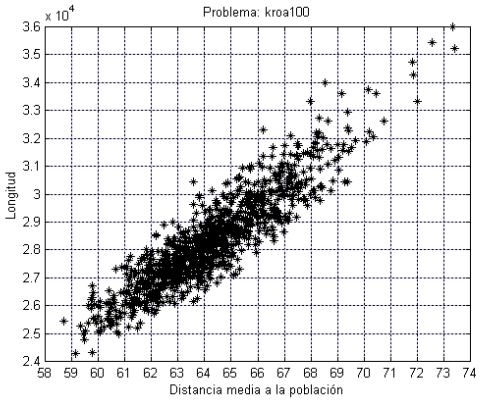

Figura 3.2: *l(r)* en función de *δ(P, r*)*

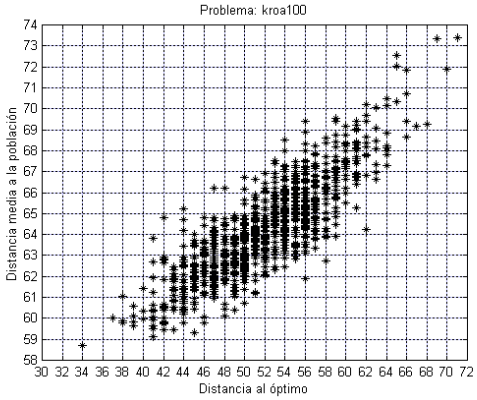

Figura 3.3: *δ(P, r*)* en función de *δ(r, r*)*

Figura 3: Problema *kroa100* usando *3-opt*

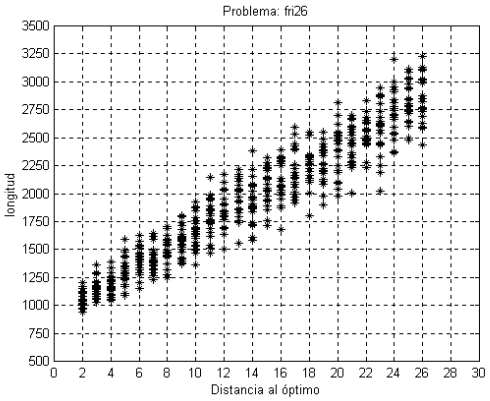

Figura 4.1: *l(r)* en función de *δ(r, r*)*

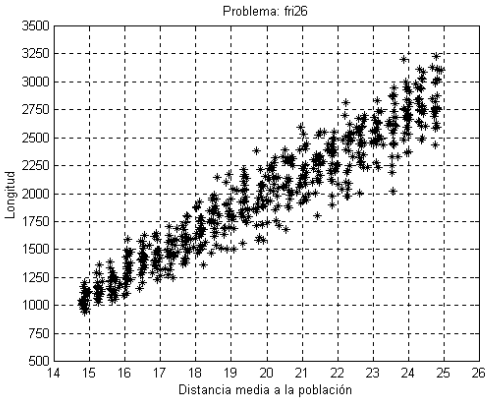

Figura 4.2: *l(r)* en función de *δ(P, r*)*

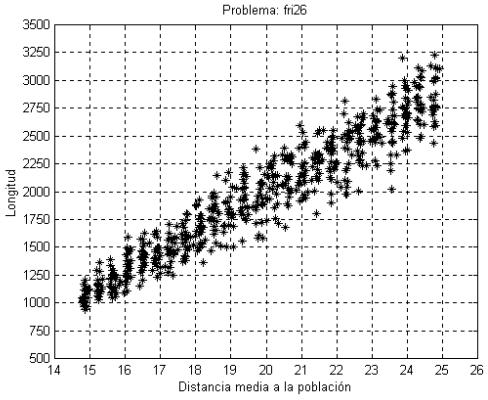

Figura 4.3 *δ(P, r*)* en función de *δ(r, r*)*

Figura 4: Problema *fri26* usando *Dist*

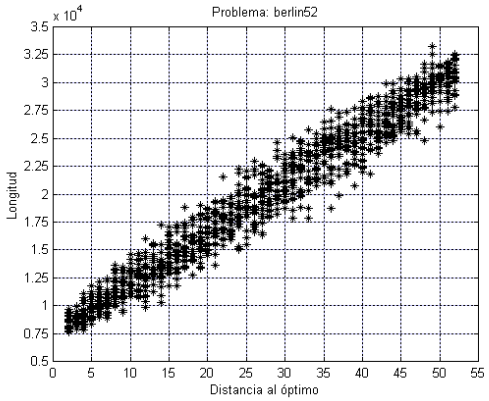

Figura 5.1: *l(r)* en función de *δ(r, r*)*

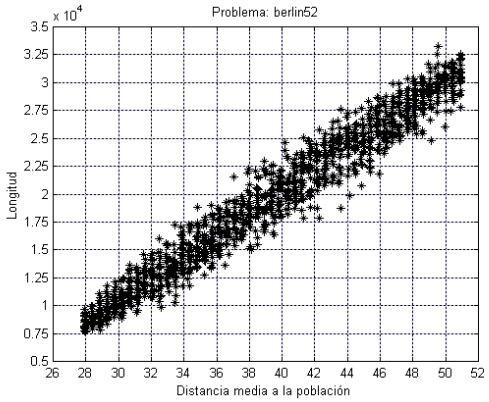

Figura 5.2 *l(r)* en función de *δ(P, r*)*

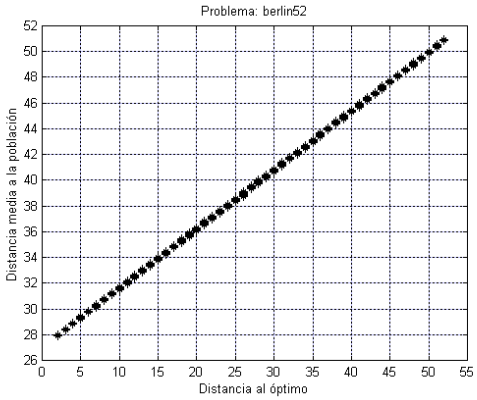

Figura 5.3: *δ(P, r*)* en función de *δ(r, r*)*

Figura 5: Problema *berlin52* usando *Dist*

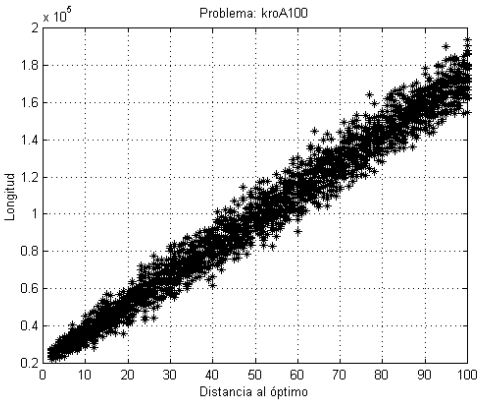

Figura 6.1: *l(r)* en función de *δ(r, r*)*

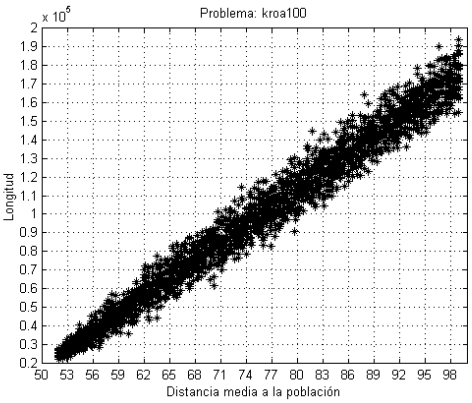

Figura 6.2: *l(r)* en función de *δ(P, r*)*

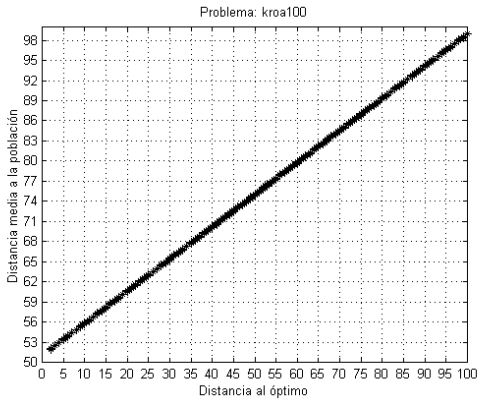

Figura 6.3: *δ(P, r*)* en función de *δ(r, r*)*

Figura 6: Problema *kroa100* usando *Dist*

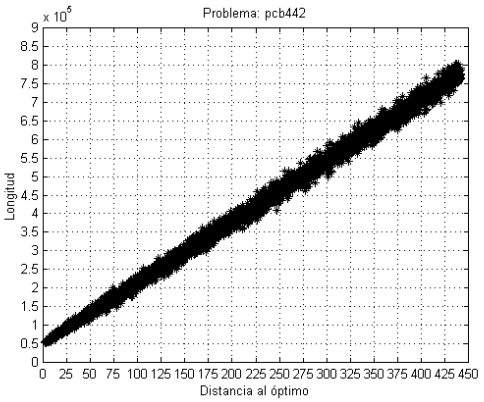

Figura 7.1: *l(r)* en función de *δ(r, r*)*

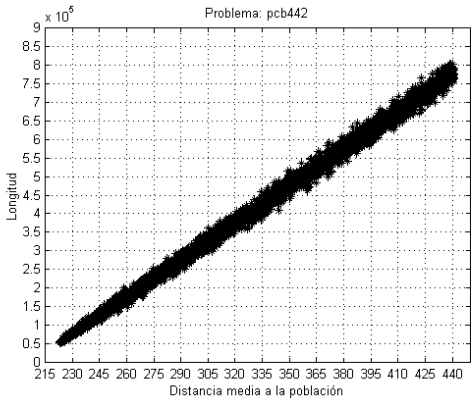

Figura 7.2: *l(r)* en función de *δ(P, r*)*

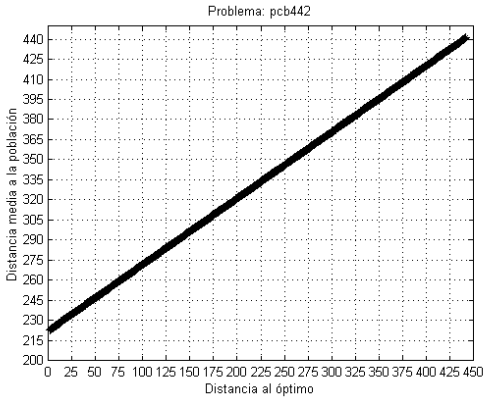

Figura 7.3: *δ(P, r*)* en función de *δ(r, r*)*

Figura 7: Problema *pcb442* usando *Dist*

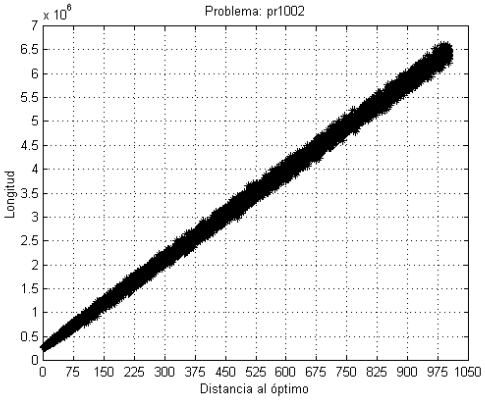

Figura 8.1: *l(r)* en función de *δ(r, r*)*

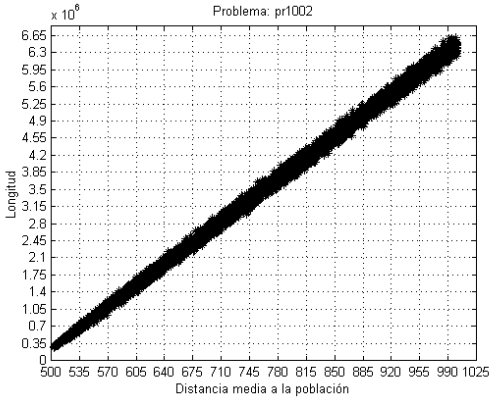

Figura 8.2: *l(r)* en función de *δ(P, r*)*

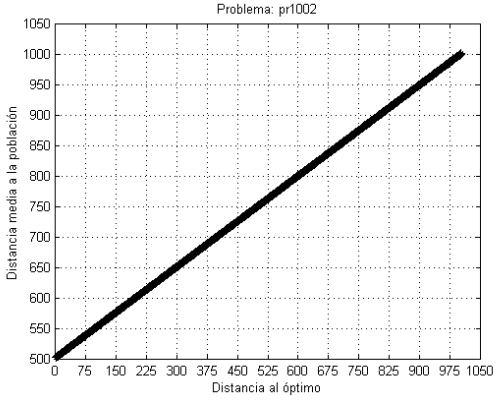

Figura 8.3: *δ(P, r*)* en función de *δ(r, r*)*

Figura 8: Problema *pr1002* usando *Dist*

5 Definición de espacio globalmente convexo

En una investigación previa, Boese [1] sugiere una interpretación intuitiva de un espacio globalmente convexo, haciendo una analogía con una estructura de *gran valle*, con un solo mínimo perceptible desde lejos, pero con picos y valles (mínimos locales) visto desde cerca. Una representación gráfica simplificada de esta interpretación puede observarse en la figura 9.

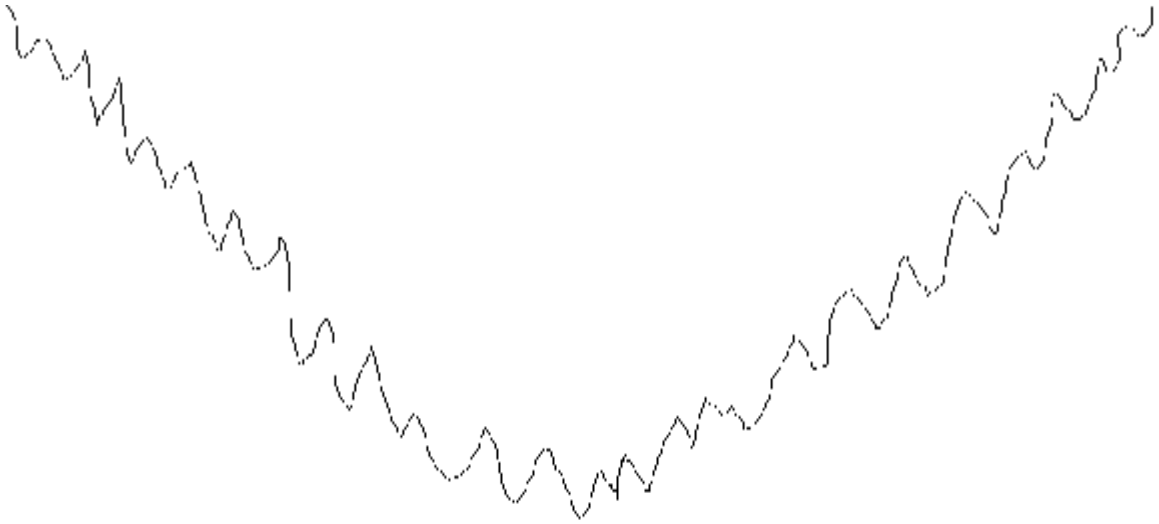

Figura 9: Representación gráfica de un espacio globalmente convexo.

Como puede notarse en la Figura 9, si tomamos una población uniformemente distribuida en el dominio de dicha función globalmente convexa, o entre los óptimos locales de esta, esta población tendría exactamente las mismas características que las encontradas en todos los problemas del TSP estudiados, es decir, se obtendrían valores muy significativos de las 3 correlaciones estudiadas y ampliamente graficadas en el presente trabajo:

1. $\rho(\delta(r,r^*); l(r))$ pues claramente las longitudes decrecen al acercarse a la solución óptima r^*.
2. $\rho(\delta(P,r); l(r))$ pues las mejores soluciones están en el centro del valle y las peores soluciones están en la periferia, conforme fuera explicado al final de la sección 2.
3. $\rho(\delta(r,r^*); \delta(P,r))$ como lógica consecuencia de las altas correlaciones anteriores, indicando además que el óptimo r^* estaría en una región central del espacio de búsqueda, rodeado de muy buenas soluciones.

En consecuencia, el presente trabajo presenta suficientes datos experimentales como para confirmar que la estructura del espacio de soluciones del *TSP* tiene en general una estructura globalmente convexa, conforme a lo sugerido sin demostración por Boese [1,2]. Lógicamente, no se pude afirmar que está estructura globalmente convexa se da necesariamente en todos los casos, pero definitivamente, los resultados experimentales demuestran que si se dan en la mayoría de los casos, y de hecho, los autores no lograron encontrar siquiera un problema del TSP que no sea globalmente convexo.

6 Conclusiones

Los resultados muestran unas muy altas correlaciones entre la $l(r)$ y su $\delta(r,r^*)$ en todas las poblaciones halladas con *Dist*. Esto claramente indica que a medida que un *tour* comparte un mayor número de arcos con el óptimo su longitud tiende a disminuir. Dicho de otra forma la posibilidad de hallar una buena solución es mayor mientras más cerca del óptimo r^* se busque. El estudio de la correlación entre $\delta(P,r)$ y $l(r)$ refuerza la idea de que las soluciones buenas se encuentran cerca unas de otras y en una zona central del espacio solución.
Esta propiedad resulta de especial importancia al intentar explicar el excelente desempeño de los algoritmos evolutivos basados en poblaciones, como las colonias de hormigas y los algoritmos genéticos, que al trabajar con una población que va optimizando el valor de $l(r)$, de acuerdo a lo aquí expuesto estaría concentrando su búsqueda en la zona central del espacio de búsqueda, donde en efecto se encuentran las mejores soluciones, y r^* en particular, dada la característica globalmente convexa del espacio de soluciones [3].
Cabe resaltar que las poblaciones halladas con la heurística de optimización local *3-opt* presentaron menores correlaciones que las halladas con *Dist*. Esto podría deberse a que son óptimos locales y no se hallan uniformemente distribuidas en el espacio de soluciones. Esto indica que la heurística al comenzar en puntos aleatorios fue

dirigiéndose automáticamente a la zona más cercana al óptimo. Es decir, que halló mejores soluciones a medida que encontraba arcos que coincidían con los del óptimo.
En resumen, el presente trabajo aporta por primera vez, amplios resultados experimentales que soportan la conjetura de un espacio de búsqueda globalmente convexo, en el caso del TSP, lo que explicaría el excelente desempeño de diversos algoritmos basados en poblaciones, a la hora de resolver el problema del cajero viajante.

Referencias

[1] Boese, K.D.: Cost versus Distance in the Traveling Salesman Problem. *Technical Report 950018*, University of California, Los Angeles, Computer Science Department, (1995).

[2] Boese, K.D, Kahng, A.B. y Muddu, S.: A new Adaptive Multi-Start Technique for Combinatorial Global Optimization, *Operations Research Center*. Vol 16, (1994). pp. 101-113

[3] Gómez O. y Barán B.: Arguments for ACO's Success, a ser publicado en "Genetic and Evolutionary Computation Conference – GECCO 2004". Seattle, Washington. Estados Unidos.

[4] Johnson D.S. y McGeoch L.A.: The traveling salesman problem: A case of study in local optimization, *Local Search in Comb. Optimization*, Eds. New York: Wiley: New York, (1997).

[5] Stützle, T. y Hoos, H.H.: Max-Min Ant System. *Future Generation Computer Systems.* Vol 16, (2000), pp. 889-914

[6] TSPLIB: http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/