Estudio comparativo entre metaheurísticas de trayectoria y metaheurísticas poblacionales aplicadas al problema de asignación cuadratico QAP

Antonio Alvarez

Abstract

Una descripción breve del paper.

1 Introducción

El problema de asignación cuadrática (QAP por sus siglas en inglés) es un problema combinatorio en el cual hay "n" localidades con distancias definidas entre ellas y almacenadas en una matriz D y "n" sucursales con flujos determinados en una matriz F. La idea es asignar cada sucursal a una localidad distinta tal que se minimice la siguiente ecuación:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} f_{ij}d_{p(i)p(j)}$$

Para esto hay que hallar una permutación "p" que lo haga posible [1]. Dicha permutación expresa el orden de asignación de una localidad con respecto a las sucursales o viceversa.

Este problema fue demostrado NP-completo por Sahni, S. y T. Gonzalez [2] lo cual hace que sea infactible una búsqeda exhaustiva en el espacio de soluciones para n de tamaño moderado y grande por ser n! combinaciones posibles. En su lugar se usan algoritmos de aproximación que consigan una respuesta dentro de un margen de error dado y en un tiempo de cómputo razonable.

Estos algoritmos de aproximación empiezan con una solución inicial y dependiendo de los criterios de selección presentes se va moviendo por el espacio de soluciones hasta que se cumple el criterio de parada y devuelve la mejor solución encontrada [5].

Este problema ha sido base para distintas investigaciones buscando proponer técnicas y optimizaciones con las cuales se pueda hallar los óptimos globales para las distintas instancias. Entre dichas investigaciones se encuentran la realizada por Li, Y. en "Heuristic and exact algorithms for the quadratic assignment problem" [12] donde plantea mejoras a partir de reducciones realizadas a las matrices de flujos y distacias, aporta nuevas cotas con la construcción de un algortimo "branch & bound" basado en la optimización de las matrices, muestra una nueva versión de búsqueda local con una nueva estructura de vecindad y diseña un algoritmo genético que incluye un nuevo operador de cruce que muestra ser efectivo al combinarse con "local hill-climbing".

Otro trabajo es el de Karisch, S. "Nonlinear approaches for the quadratic assignment and graph partition problems" [11]. Donde logra dar las cotas inferiores más bajas para muchos de los problemas pequeños de la librería a través de la utilización de enfoques no lineales. Por su parte Johnson, T. en "New linear programming-based solution procedures for the quadratic assignment problem" [13]. Plantea procedimientos basados en la programación lineal que van a servir como base a las formulaciones de otros investigadores para resolver QAP.

Este trabajo busca comparar el desempeño tanto en precisión como en tiempo de cómputo de 5 metaheurísticas distintas al buscar él óptimo global para instancias de este problema ; éstas son Recocido Simulado (SA), Búsqueda Tabú (TS), Búsqueda Local Iterada (ILS), Algoritmo genético (GA) y Colonia de Hormigas (AC). El objetivo es determinar cual de estos métodos es el más efectivo al momento de buscar una buena solución al problema de asignación cuadrática.

Al existir un extenso repertorio de metaheurísticas que está en constante expansión, se torna difícil elegir un método a utilizar para la resolución del problema planteado; más aún cuando las carcterísticas de dicho problema afectan el desempeño del método. Por lo tanto existe una constante necesidad de probar las distintas metaheurísticas con cada problema de optimización planteado para determinar su comportamiento ante los mismos. De esta manera, este trabajo espera probar las 5 mencionadas anteriormente y definir, en base a los resultados, cual de ellas se adapta mejor a QAP.

En lo que concierne al resto del informe se tiene la descripción de las 5 metaheurísticas utilizadas y la especificación de sus parámetros en la sección de metodología con el fin de dejar sentadas las bases con las cuales se realizó el experimeto. La sección de resultados presenta las tablas con los promedios de 10 corridas para cada método probado con distintas instancias y un boxplot por cada método para mostrar el desempeño general del mismo. Luego se presenta el análisis de los resultados, donde se destacan los puntos resaltantes del experimento que conlleven a la determinación del mejor método sobre los demás y finalmente se encuentran las conclusiónes donde se generaliza lo ya destacado en el

análisis de resultados y se determinan los aportes realizados.

2 Metodología

Las soluciones factibles se modelan como un arreglo de enteros de tamaño n con una permutación de los número del 1 al n (incluidos) donde n representa el número de localidades y sucursales. La operación de movimiento o vecindad se define como el intercambio de exactamente 2 elementos dentro del arreglo, por lo tanto la vecindad está formada por $n*(n+1)/2$ elementos distintos. La función objetivo es: $\sum_{i=1}^{n}\sum_{j=1}^{n} f_{ij}d_{p(i)p(j)}$ y una solución es considerada mejor que otra si al evaluarla con la función objetivo se obtiene un resultado menor.

Para este trabajo se utiliza la librería QAPLIB [4]. Las instancias seleccionadas son las de la serie Chr que representan valores para árboles y grafos completos; la serie Esc que se basan en la prueba de circuitos; Sko con matrices cuyos números son generados aleatoriamente; la serie a de Tai que son generados uniformemente y son simétricas y Wil cuyas distancias son rectangulares. Cada instancia es probada 10 veces y se reporta el promedio de la evaluación de la función objetivo y tiempo de cada corrida. Las pruebas realizadas fueron hechas con un procesador Intel Celeron N3160, 2GB de RAM usando una versión de 32-bit de Ubuntu 14.04.5.

Ahora se procede a la implementación de los distintos algoritmos utilizados:

2.1 Búsqueda Local

Búsqueda local se usa como la heurística base con la cual se comparan las otras metaheurísticas. La idea de búsqueda local es ir eligiendo soluciones cada vez mejores en cada iteración hasta que no se consiga ninguna mejoría en la vecindad de la solución actual, situación que da por culminada la búsqueda y da como resultado un óptimo local dentro del espacio de soluciones.

Para la solución inicial se optó por una solución aleatoria porque como explica Talbi, E. G. [6] Con una vecindad de grandes dimensiones se mitiga el impacto que genera la elección de la solución inicial como es el caso de las instancias con más de 100 localidades y para las instancias pequeñas (menos de 30 localidades) el aumento en iteraciones del movimiento por el espacio de soluciones no muestra un impacto importante en el tiempo de cómputo cuando en promedio la evaluación de una instancia de 30 toma 0.12 segundos.

Luego se decidió parametrizar búsqueda local de tal manera que pudiera operar bajo distintos criterios de selección de vecinos en cada iteración; estos criterios son: elegir el primer vecino encontrado que mejore la solución actual, elegir la mejor solución de la vecindad, elegir la mejor solución dentro de un porcentaje dado de la vecindad y elegir un vecino aleatorio.

En los procesos de búsqueda en la vecindad se empieza la permutación ordenada de cada par de elementos dentro del arreglo de la solución de tal manera que se verifiquen todas las combinaciones posibles (excepto para la búsqueda aleatoria).

En la figura 1 se presenta el pseudo-código para la búsqueda local propuesta. Nótese que f es la función objetivo

Algorithm 1: Busqueda Local

Input: matrices con las distancias y flujos y tipo de selección dentro de la búsqueda
Output: Arreglo con una permutación de las asignaciones de las localidades a las sucursales

```
1 S = S0    /* escogiendo S0 aleatoriamente */
2 repeat
3 |   Generar vecino S' según el tipo de búsqueda
4 |   if f(S') ≤ f(S) then
5 |   |   S = S'
6 until f(S) ≠ f(S')
7 return S
```

2.2 Recocido simulado

El recocido simulado es una metaheurística propuesta en 1983 por S. Kirkpatrick et al. en su artículo "Optimization by Simulated Annealing" [7]. Se toma como inspiración el proceso metalúrgico por el cual se calienta a altas temperaturas un metal y se enfría lentamente para formar estructuras cristalinas fuertes. En su trabajo, Kirkpatrick hace un simil y establece que el proceso de calentamiento y enfriamiento es aplicable a la busqueda de soluciones para los problemas de optimización.

Esta metaheurística establece que en una búsqueda de soluciones dentro del espacio se realiza una exploración en donde una solución es aceptada si mejora la actual o en caso contrario es aceptada con probabilidad $e^{\frac{-(f(s')-f(s))}{T}}$. Donde f es la función de costos, s' la solución que se está generando y evaluando, s la solución actual y T la temperatura actual.

Por lo tanto, la idea es empezar con una temperatura inicial con la cual se pueda aceptar cualquier movimiento dentro del espacio de soluciones y así promover la diversificación. Luego, pasado cierto tiempo, que puede ser un número de iteraciones en que no hay mejoría o un número fijo, se enfría la temperatura y se repite la búsqueda en la nueva vecindad hasta que se llegue a una temperatura final. Con este esquema, ya cuando se llega a temperaturas bajas la tasa de aceptación debe ser menor que el principio de tal manera

que se intensifique la búsqueda en una región específica, la cual debería contener el óptimo global.

Cabe destacar que el proceso de enfriamiento puede llevarse a cabo con distintas funciones, las más comunes son la función geométrica $T' = \alpha * T$, y logarítmica $Ti = \frac{T0}{log(i)}$. Donde mientras más lenta sea la tasa de decremento, más oportunidad se le da al algoritmo para explorar y poder conseguir el óptimo global.

En este sentido los parámetros que hay que ajustar a las necesidades del problema son: una temperatura inicial que permita aceptar movimientos a cualquier vecino, una temperatura final en la cual ya no se acepten soluciones que empeoren la evaluación de la función de costos, cuántas iteraciones han de evaluarse en cada temperatura y cual es la función con la cual se enfría la temperatura una vez se llegue a un estado de equilibrio.

En la figura 2 se muestra el pseudo código del Recocido Simulado para este trabajo. Se estableció como temperatura inicial T0 = 100000, temperatura final Tf = 1000, número de iteraciones para conseguir el equilibrio = 1000 y tasa de enfriamiento = 0.95*T para todas las instancias.

Algorithm 2: Recocido Simulado

Input: matrices con las distancias y flujos
Output: Arreglo con una permutación de las asignaciones de las localidades a las sucursales

```
1  S = S0    /* escogiendo S0 aleatoriamente */
2  T = 100000          /* temeratura inicial */
3  Tf = 1000           /* temperatura final */
4  while T > Tf do
5      while no pasen 1000 iteraciones sin mejorías
       do
6          Generar vecino S'
7          ΔE = f(S') − f(S)
8          if ΔE ≤ 0 then
9              S = S'
10         else
11             S = S' con probabilidad e^(−(f(s')−f(s))/T)
12     T = 0.95 * T
13 return S
```

2.3 Búsqueda Tabú

La búsqueda tabú es una metaheurística propuesta por F. Glover en su trabajo "Tabu search: Part I" [8]. Este método busca escapar de los óptimos locales permitiendo movimientos que empeoren la evaluación de la función de costos siempre que cumplan con un criterio de aspiración o que no esté en la lista tabú (lista de movimientos prohibidos).

En la versión más sencilla de búsqueda tabú, sólo se implementa la lista tabú; en ella se almacena los rasgos que caracterizan el movimiento que hace el algoritmo en cada iteración de tal manera que al momento de aceptar peores soluciones no se entre en un ciclo por el cual se repitan movimientos. La naturaleza de las estructuras almacenadas en la lista tabú depende directamente del problema a resolver; para problemas combinatorios dicha lista puede almacenar las arista intercambiadas en un problema de grafos o las posiciones intercambiada en un arreglo de elementos.

La búsqueda continúa hasta que se cumple un criterio de parada; entre ellos se encuentran parar la búsqueda luego una cantidad de iteraciones fijas o luego de que pase un número determinado sin mejorar la solución. Para este trabajo se utiliza la segunda.

Existen otras versiones de búsqueda tabú que usan memorias a mediano y largo plazo con información referente a la cantidad de veces que se repite un patrón dado de las mejores soluciones. Esta información luego se utiliza en fases de intensificación en donde se limita la búsqueda a los candidatos que presenten dichos patrones o en fases de diversificación en las cuales se aleja la búsqueda de los patrones. En este trabajo dichas memorias no son consideradas en favor a mantener una implementación sencilla con la cual se pueda ver el comportamiento de un algoritmo basado en una lista de restricciones.

En la figura 3 se muestra el pseudocódigo de la búsqueda tabú utilizada. Para todas las instancias se coloca como condición de parada el paso de 1000 iteraciones sin mejoría de los resultados y una lista tabú de tamaño 10.

Algorithm 3: Búsqueda Tabú

Input: matrices con las distancias y flujos
Output: Arreglo con una permutación de las asignaciones de las localidades a las sucursales

```
1  S = S0    /* escogiendo S0 aleatoriamente */
2  Inicializar la lista tabú
3  while no pasen 1000 iteraciones sin mejorías do
4      Conseguir el mejor vecino S' que no se
       encuentre en la lista tabú o que sea la mejor
       solución conseguida hasta el momento
       /* criterio de aspiración */
5      S = S'
6      Actualizar la lista tabú con el movimiento
       realizado
7  return S
```

2.4 Búsqueda Local Iterada

La búsqueda Local Iterada (ILS por sus siglas en inglés) parte de una idea sencilla; conseguir un óptimo local utilizando algún método, perturbar la solución obtenida y repetir el método utilizado para conseguir el óptimo local con el objetivo de conseguir una mejor solución. Este proceso se repite hasta que se cumpla un criterio de parada y se reporta la mejor solución encontrada.

ILS está conformado por 4 componentes que entonar para conseguir la configuración óptima que se adapte al problema en cuestión [9]. Estos son: la solución inicial de la búsqueda, la perturbación realizada sobre el óptimo local encontrado en cada iteración, el criterio de aceptación de los óptimos conseguidos periódicamente y el método utilizado dentro del ciclo para conseguir óptimos locales.

La solución inicial puede ser aleatoria o resultado de un algoritmo ávido; la ventaja de esta última opción involucran una menor cantidad de iteraciones utilizadas por ILS para converger a una solución (salvo en casos específicos en donde la topología juega en contra de obtener mejores resultados a partir de buenas soluciones iniciales) y en algunos problemas puede conllevar a una mejor respuesta final. Sin embargo, si obtener una respuesta por parte de un algoritmo ávido es especialmente costoso se tiende a optar por una solución aleatoria.

Con respecto a la perturbación realizada a la solución obtenida se debe realizar en una medida que le permita escapar del óptimo local en el que se encuentra pero no ser un cambio tan drástico que se pierda todas las características que conforman la respuesta que está perturbando. La idea detrás de esto es conseguir un óptimo local, el cual debe poseer algún rasgo que comparta con el óptimo global y al conservarlo se pueda explorar el espacio de soluciones de manera más efectiva que si se empezara una busqueda aleatoria en cada fase del algoritmo. Un ejemplo de una perturbación moderada para TSP es el cambio de 4 aristas por otras nuevas [9]. El cambio es díficil de deshacer como para regresar a la combinación original y además conserva el resto de la trayectoria original; por lo que si la solución original es buena, la nueva también lo será.

Por su parte, el criterio de aceptación determina cuál solución producida en cada iteración es aceptada como el punto de partida para la siguiente vuelta. Un criterio puramente elitista sólo acepta las soluciones que vayan mejorando la evaluación de la función de costos, este criterio favorece la intensificación. El otro extremo es la aceptación de todas las soluciones generadas, lo cual favorece la diversificación. También se encuentran puntos intermedios como la utilización de una temperatura como en Recocido Simulado de tal manera que se diversifique al principio de la corrida y se intensifique al final.

Por último se encuentra el método usado en cada iteración para la producción de un óptimo local. Dicho método puede ser un búsqueda local sencilla u otra metaheurística como recocido simulado o búsqueda tabú.

Es a partir de la combinación de los 4 módulos vistos anteriormente que se entona ILS a los problemas para los cuales tiene que generar soluciones satisfactorias y en tiempos razonables. Para este trabajo se usó como solución inicial una generada aleatoriamente, como perturbación la operación 2-opt, la cual invierte la porción de la lista que está delimitada por 2 elementos escogidos aleatoriamente, el método interno usado es una búsqueda local común que va eligiendo la primera mejor solución de la vecindad que consiga y el criterio de aceptación es tomado de Recocido Simulado aceptando siempre mejores soluciones y con probabilidad $e^{\frac{-(f(s')-f(s))}{T}}$. Además se baja la temperatura después de 20 búsquedas locales que no mejoren la solución y se baja la temperatura a una tasa de 0.8*T, la temperatura inicial es 100000 y la final es 1000. En la figura 4 el pseudocódigo.

Algorithm 4: Búsqueda Local Iterada

Input: matrices con las distancias y flujos
Output: Arreglo con una permutación de las asignaciones de las localidades a las sucursales

```
1  S = S0    /* escogiendo S0 aleatoriamente */
2  T = 100000          /* temperatura inicial */
3  Tf = 1000             /* temperatura final */
4  sinMejoria = 0
5  while T > Tf do
6      while sinMejoria < 20 do
7          S* = localSearch(S)
8          if f(S*) < f(S) then
9              S = S*
10             sinMejoria = 0
11         else
12             S = S* con probabilidad e^((-(f(s')-f(s)))/T)
13         sinMejoria = sinMejoria +1
14     T = T*0.8
15 return S
```

2.5 Algoritmo Genético

El concepto de algoritmo genético aparece por primera vez en "Adaptation in natural & artificial systems" gracias a Holland, H. [10]. La metaheurística está inspirada en la teoría evolutiva en la cual la descendencia que presente rasgos genotípicos que le permitan apatarse al

entorno tiende a proliferar más y así pasar dichos rasgos a la siguiente generación. Un algoritmo genético parte de una población de soluciones llamadas cromosomas, aplica un criterio de selección para elegir aquellos que van a reproducirse, cruza las soluciones con algún método, aplica un criterio de reemplazo para determinar cuales cromosomas sobreviven en cada ciclo y uno para las mutaciones que se puedan dar dentro de la población. Luego de que se cumpla un criterio de parada determinado se reporta la mejor solución encontrada.

La selección de la población inicial conlleva a elegir cuántos cromosomas van a conformarla; si son pocos los cromosomas podría no estarse explorando bien el espacio de soluciones con lo cual se restringe la búsqueda a pocos sectores, por otra parte, si son muchos los cromosomas se utiliza más tiempo de cómputo en cada iteración volviendo menos eficiente el algoritmo; por lo tanto una cantidad moderada es deseable y dicha cifra está sujeta a cada instancia con la que se prueba el algoritmo.

La población inicial suele elegirse aleatoriamente, sin embargo esta elección puede tener problemas al momento de intentar abarcar la mayor cantidad posible del espacio de soluciones; por lo tanto para algunos problemas se idea una forma más efectiva de elegir la población inicial con el objetivo de distanciar cada solución lo más posible; un ejemplo consiste en inciar la población para instancias de QAP con todas las permutaciones que contengan las distintas combinaciones de sucursales y localidades distintas.

La selección por su parte puede ser completamente elitista con un sistema de jerarquización en el cual se ordenan los cromosomas de peor a mejor resultado y la probabilidad de elegir un cromosoma en específico es $\alpha + \beta * k$ donde k es el índice del cromosoma en la lista ordenada; así se da mayor peso a los mejores resultados al momento de hacer la elección. Otro sistema utilizado es el torneo, en el cual se toma un número aleatorio de cromosomas y se elige el mejor de la muestra.

En lo concerniente al cruce se tiene que el método a emplear es dependiente del problema a resolver. Por ejemplo, para problemas con strings binarios se tienen los "X point crossover", los cuales fijan X puntos en los dos padres y se va copiando las porciones entre los puntos de los dos padres de manera intercalada en los hijos.

Otro ejemplo, utilizado en arreglos de permutaciones, es "Partially Mapped Crossover" (PMX) que consiste en elegir dos puntos con los cuales se establece una guía de intercambio entre los elementos que aparezcan entre dichos puntos para cada padre; esto es, si un gen X aparece entre los dos puntos en un padre y un gen Y aparece en la misma posición en el otro padre, entonces todas las apariciones de Y en el segundo padre son sustituidas por X y viceversa.

Aparte de PMX se encuentra OX, también usado para soluciones que representan permutaciones; éste consiste en utilizar una máscara de bits con el cual se fijan en los hijos los elementos de los padres que coincidan en posición con los 1s y se va rellenando los elementos faltantes en el orden de aparición que tengan en el otro padre.

El reemplazo puede ser generacional, en donde la progenie reemplaza a los padres excepto al mejor y así evitar que se pierda la posible solución final o puede ser bajo un criterio donde los hijos reemplazan a los padres si son mejores que éstos o a las peores soluciones de la población. Nótese que esta última opción es elitista y puede llegar a homogeneizar la población, convergiendo prematuramente a un óptimo local no deseado.

En lo referente a la mutación se debe determinar la frecuencia con la que ocurre, qué cromosomas mutan y cuáles son los genes que cambian. El cambio de los genes en el cromosoma que muta debe ser tal que produzca una solución "cercana" a la original, si el problema puede ordenar los valores de los genes es recomendable elegir valores próximos para asegurar dicha cercanía.

Finalmente la condición de parada puede ser un número estático de iteraciones, una cantidad de ciclos sin mejoría de la mejor solución o forzar la culminación cuando la diversidad de la población baje de cierto punto. Para este trabajo se utiliza una población inicial de 10 cromosomas para mantener los tiempos de cómputo competitivos con las metaheurísticas de trayctoria; como condición de parada 3000 iteraciones sin mejoría de la solución óptima encontrada; se usa el torneo de 3 cromosomas para la seleción de los individuos a reproducirse; el operador de cruce es OX eligiendo dos genes aleatoriamente; la mutación puede ocurrir en cada iteración con un 10% de probabilidad sobre un número aleatorio de cromosomas de la población y se eligen 2 genes aleatoriamente para su intercambio y los hijos reemplazan a los padres si son mejores que ellos. En la figura 5 se muestra el pseudocódigo.

3 Resultados y análisis

Las tablas 1 y 5 muestran los resultados de la evaluación de la función objetivo, los porcentajes de error y tiempo de cómputo para las distintas variantes de búsqueda local (elección de la mejor solución dentro de la vecindad, primera solución que mejore la evaluación de la función de costo actual o elección aleatoria de un vecino) respectivamente.

El error relativo muestra que la mayoría de las instancias seleccionadas para el experimento muestran una topología con pocas mesetas (puntos en los cuales se encuentran los óptimos locales), por lo que se puede aproximar a una buena solución (aquellas que están a no más de 5% de error) con una búsqueda local que se

Table 5: Porcentajes del error relativo y tiempo de cómputo en segundos de las distintas búsquedas locales

Instancia	Mejor posible	segundos	Primer mejor	segundos	Aleatorio	segundos
chr12a	34.35	0.003433	37.46	0.003792	257.5	0.000421
chr15a	61.04	0.00635	51.62	0.003551	340.75	0.000475
chr20a	47.9	0.026156	44.83	0.019761	295.41	0.000595
chr22a	12.83	0.05023	18.49	0.034667	91.91	0.000661
chr25a	51.96	0.081039	51.21	0.046922	392.8	0.000807
sko42	4.01	1.11221	1.61	1.39335	25.3	0.002022
wil50	1.07	3.26927	1.49	3.58704	14.5	0.002199
sko56	3.14	4.67231	1.89	6.42853	20.28	0.002522
sko72	2.81	28.7295	2.74	25.636	18.53	0.004021
sko81	2.38	67.1344	1.62	60.6198	19.07	0.004808
sko100a	2.35	132.649	1.83	123.815	17.52	0.074144
esc128	3.13	65.2117	18.75	20.6649	409.38	0.011112
Promedio	18.91		19.46		158.58	

Table 6: Porcentajes del error relativo y tiempo de cómputo en segundos de las distintas metaheurísticas

Instancia	Annealing	segundos	Tabu Search	segundos	ILS	segundos	GA	segundos
chr12a	4.87	1.44	8.66	0.7	23.98	0.36	20.77	0.72
chr15a	11.91	1.8	9.02	1.24	57.46	0.75	46.83	1.17
chr20a	98.39	3.08	8.75	3.08	43.52	1.94	53.79	2.33
chr22a	38.41	3.24	5.31	4.89	15.14	2.72	17.79	2.94
chr25a	180.8	4.09	29.48	5.52	61.67	4.35	86.49	3.01
sko42	17.87	10.23	1.4	10	2.95	10	8.18	10
wil50	6.85	14.2	0.82	15	1.17	15	4.22	15
sko56	16.9	17.85	1.61	18	2.48	18	7.47	18
sko72	15.22	28.69	2.53	29	2.04	29	7.41	29
sko81	14.76	35.67	3.8	36	1.47	36	7.81	36
sko100a	13.33	59.45	5.38	60	15.72	60	7.89	60
esc128	245.31	130.72	15.63	131	10.94	131	84.38	131
promedios	55.39		7.7		19.88		29.42	

Table 7: Porcentajes del error relativo y tiempo de cómputo en segundos de las distintas metaheurísticas para las instancias Tai

Instancia	Annealing	segundos	Tabu Search	segundos	ILS	segundos	GA	segundos
tai12a	0	1.42	0.84	0.84	8.55	0.37	7.47	1.63
tai15a	1.63	1.88	0.31	1.74	4.37	0.75	5.39	1.87
tai17a	2.38	2.12	0.82	1.8	6.14	1.1	6.93	1.88
tai20a	1.41	2.55	0.87	3.7	7.54	1.9	8.11	2.33
tai25a	2.94	3.83	1.16	4	4.63	4.33	7.71	4.8
tai30a	2.31	5.1	1.52	5	5.72	5	8.24	5
tai35a	1.74	6.61	1.99	7	5.02	7	9.00	7
tai40a	2.81	8.84	1.91	9	5.34	9	9.51	9
tai50a	3.41	14.34	2.34	15	4.15	15	10.19	15
tai60a	3.3	19.47	2.51	20	4.61	20	10.02	20
tai80a	3.28	33.57	2.41	34	3.79	34	9.18	34
tai100a	3.12	59.82	2.39	60	3.55	60	8.43	60
promedios	2.36		1.59		5.28		8.35	

Algorithm 5: Algoritmo genético

Input: matrices con las distancias y flujos
Output: Arreglo con una permutación de las asignaciones de las localidades a las sucursales

```
Se inicializa la población con 10 soluciones aleatorias
Se ordenan los cromosomas de mejor a peor solución
sinMejoria = 0
while sinMejoria <1000 do
    Elegir 2 padres realizando un torneo de 3 cromosomas aleatorios por cada uno
    Cruzar los padres con el operador OX
    Elegir los 2 mejores entre los hijos y los padres y colocarlos en la población en la posición de los padres
    if número aleatorio entre 0 y 1 <0.10 then
        Elegir aleatoriamente los cromosomas que van a mutar e intercambiar dos genes aleatoriamente
    Reordenar la población con quicksort
    sinMejoria = sinMejoria + 1
return S
```

Table 1: Resultados de la evaluación de la función objetivo de las distintas búsquedas locales

Instancia	Mejor posible	Primer mejor	Aleatorio	Optimo
chr12a	12833	13130	34148	9552
chr15a	15936	15004	43616	9896
chr20a	3398	3328	9086	2298
chr22a	6945	7294	11814	6156
chr25a	5768	5740	18706	3796
sko42	16445	16066	19812	15812
wil50	49338	49544	55894	48816
sko56	35539	35108	41446	34458
sko72	68120	68072	78536	66256
sko81	93160	92474	108354	90998
sko100a	155578	154780	178636	152002
esc128	66	76	326	64

Table 2: Resultados de la evaluación de la función objetivo de las distintas metaheurísticas

Instancia	Annealing	Tabu Search	ILS	GA	Optimo
chr12a	10017	10379	11843	11536	9552
chr15a	11075	10789	15582	14530	9896
chr20a	4559	2499	3298	3534	2298
chr22a	8520	6483	7088	7251	6156
chr25a	10659	4915	6137	7079	3796
sko42	18637	16034	16278	17105	15812
wil50	52162	49215	49389	50878	48816
sko56	40281	35014	35312	37033	34458
sko72	76345	67935	67607	71165	66256
sko81	104436	94452	92340	98105	90998
sko100a	172266	160177	175902	163991	152002
esc128	221	74	71	118	64

Table 3: Resultados de la evaluación de la función objetivo de las distintas metaheurísticas para las instancias Tai

Instancia	Annealing	Tabu Search	ILS	GA	Optimo
tai12a	225512	226302	243605	241171	224416
tai15a	394562	389423	405193	409139	388214
tai17a	503560	495850	522003	525896	491812
tai20a	713464	709579	756490	760509	703482
tai25a	1201602	1180783	1221276	1257268	1167256
tai30a	1860162	1845799	1922108	1968014	1818146
tai35a	2464222	2470123	2543633	2640086	2422002
tai40a	3227670	3199238	3307058	3437817	3139370
tai50a	5110196	5057192	5146534	5444702	4941410
tai60a	7446890	7389530	7540892	7930669	7208572
tai80a	14002730	13885116	14071478	14801907	13557864
tai100a	21784623	21630624	21875477	22906747	21125314

Table 4: Resultados de la evaluación de la función objetivo de las distintas metaheurísticas para las instancias Esc

Instancia	Annealing	Tabu Search	ILS	GA	Optimo
esc16a	74	68	70	68	68
esc16b	292	292	292	292	292
esc16c	167	160	161	160	160
esc16d	18	16	18	17	16
esc16e	31	28	31	28	28
esc32a	283	147	155	187	130
esc32b	344	198	221	250	168
esc32c	744	642	644	643	642
esc32d	272	204	213	218	200
esc32e	2	2	2	2	2
esc64a	202	118	119	124	116

enfoque en el mejor de la vecindad o el primer vecino que mejore. Sólo las instancias chr presentan errores mayores al 20%.

Con respecto al desempeño de los distintos tipos de búsqueda local, se muestra una notoria diferencia entre una búsqueda moviéndose por el mejor de la vecindad y por el primero que mejore la solución con respecto a una búsqueda aleatoria. Esta última presenta errores superiores al 15% en la mayoria de sus instancias y con desempeños mediocres en las instancias chr y esc128, los cuales superan el 100%.

Comparando una búsqueda local que se mueve al mejor de la vecindad con una que se mueve al primero que mejore la solución, se muestra un mejor desempeño por parte de la segunda sobre la primera; en general se aprecia una diferencia de aproximadamente 2% de error entre una y la otra, casos como chr15a muestran una acentuada diferencia con 10% de error a favor de moverse al primer vecino que mejore la solución. Sin embargo hay casos como chr22a y esc128 que se inclinan a favor de una búsqueda basada en el mejor de la vecindad.

Al momento de comparar tiempos de cómputo, la búsqueda aleatoria muestra el mejor desempeño entre los 3 con tiempos que nunca llegan al segundo inclusive para la instancia más grande de 256 localidades.

Por su parte, los dos otros métodos muestran un desempeño similar en tiempo para instancias pequeñas y medianas y se evidencia una diferencia es a partir de instancias grandes como sko81, donde se torna a favor de la búsqueda local moviéndose por el primer mejor vecino. Por lo tanto, se evidencia que a partir de instancias de 81 localidades la búsqueda considerando el primer mejor vecino empieza a mostrar ventaja.

La marcada diferencia en tiempos de ejecución entre la búsqueda aleatoria y el resto se debe a que es menos costoso ir generando un par de números aleatorios en cada iteración a tener que recorrer el arreglo de la solución cambiando cada par posible de posiciones. Por su parte, la diferencia entre la búsqueda del mejor de la vecindad y el primer mejor viene dado porque en la primera se debe examinar toda la vecindad, mientras que en la segunda se examina parcialmente; esta diferencia es menos notoria cuando las instancias son pequeñas y medianas (menor a 70 localidades) ya que una vecindad, como se reseñó anteriormente tiene $n * (n + 1)/2$ candidatos.

Tomando en consideración los errores relativos y los tiempos de cómputo se nota un mejor desempeño por la búsqueda local con el primer mejor sobre el resto al considerar que en las intancias grandes tarda menos en computar que una busqueda local considerando el mejor de la vecindad y que el error relativo es parecido y hasta mejor.

Ahora pasando a los resultados de la evaluación de las distintas metaheurísticas se presentan los resultados de la evaluación de la función de costos, el porcentaje de error relativo y los tiempos de ejecución en las tablas 2, 3, 4, 6, 7, 8.

Empezando con las metaheurísticas de trayectoria en general TS tuvo el mejor desempeño de las 3. Al revisar la tabla 6 se observa como en promedio TS tiene 7.7% de error, mientras que ILS posee 19.88% y SA 55.39 %. Al desglosar estos resultados evaluando las instancias individuales se observa que TS tuvo errores relativos consistentes por debajo del 10% en la mayoría de las instancias a excepción de che25a y esc128, los cuales presentan 29.48% y 15.63% respectivamente. Más aún, si se baja la cota a 5% como margen de soluciones aceptables TS tiene 5 soluciones de 12 en esta categoría.

Por su parte ILS presenta en promedio 19.88% de error; al igual que TS tiene 5 soluciones con errores por debajo del 5%, pero de resto presenta errores superiores al 10%, siendo las instancias chr las que presentan un peor desempeño con chr25a como la peor con 61.67% cuando TS devuelve resultados para esta instancia con 29.48% de error en promedio. Finalmente SA presenta en promedio 55.39% de error que es atribuible a los resultados de las instancias chr20a, chr25a y esc128; los cuales muestran errores de 98.39%, 180.8% y 245.31%. Además el resto de las instancias muestran resultados superiores al 10% con sólo la instancia chr12a por debajo del 5%, posicionando así a SA como el peor de las 3 metaheurísticas para evaluar las instancias chr y sko en general.

Cabe destacar que los tiempos de cómputo a partir de la instancia sko42 se restringen a los tiempos obtenidos por SA de tal manera que se pueda comparar resultados obtenidos por los 3 métodos en tiempos similares.

Pasando a los resultados de la instancias de la serie Tai se encuentra un comportamiento similar por parte de TS donde sigue siendo el que presenta mejores resultados de los 3 con 1.59% de error promedio, con una cota superior de 2.51% obtenida de tai60a. Por su parte SA presenta un error promedio de 2.36%, mostrando un buen desempeño en todas las instancias y siendo el peor resultado 3.3% en tai60a. ILS en esta oportunidad figura de último con un error promedio de 5.28%, presentando una diferencia notoria en general con respecto a los otros 2 métodos al observar que aparecen resultados por encima de 5% de error y con un máximo de 8.55% en chr12a.

En esta oportunidad se vuelve a restringir los tiempos de cómputo de Tabu Search e ILS a partir de la instancia tai25a para comparar el desempeño de los algoritmos fijado a un tiempo; este tiempo es el que demora SA en culminar con las especificaciones dadas.

Finalmente, al revisar los resultados de la serie Esc Tabu Search es de nuevo el mejor de los 3 con 3.15% de error promedio, destacando que se obtuvo el óptimo en 7 instancias de 11 y la presencia de 2 instancias que superan el 10%, éstas son esc32a y esc23b con 13.08% y 17.86%. Si se observa los resultados de ILS se tiene un 7.9% de error promedio que es producto de los elevados errores obtenidos en esc16d, esc16e, esc32a y esc32b que se encuentran sobre 10% y con esc32b presentando 31.54%. De último se encuentra SA con 34.99% de error promedio con especial énfasis en esc32a, esc32b y esc64a que presentan 117.69%, 104.76% y 74.14% respectivamente; contrastanto fuertemente con los resultdados de instancias como esc16b y esc32e donde se consigue el óptimo global o esc16c que está por debajo del 5%.

En esta oportunidad se fijó el tiempo de cómputo para los algoritmos al obtenido por TS debido a que consiguió los óptimos globales y buenas solucioes para la mayoria de las instancias en menor tiempo que las otras dos metaheurísticas. Todas los esc16 y esc32 se limitaron a corridas de 2 segundos y esc64 a 10 segundos.

Los resultados de Simulated Annealing se pueden explicar porque no se le dió suficiente tiempo para mejorar las soluciones encontradas. Como se demuestra en [7] si se deja computando SA en un tiempo que tienda al infinito la solución converge al óptimo global. Otro factor que puede estar influyendo es la solución inicial que siendo aleatoria puede influir en el tiempo necesario para converger a una buena solución.

Pasando a las metaheurísticas poblacionales...

Conclusiones

Aquí concluyen.

References

[1] Burkard, R. et al. QAPLIB-A Quadratic Assignment Problem Library. 2002

[2] Sahni, S., and Gonzalez, T. P-complete approximation problems. *J. of ACM*, 23, 555–565. 1976

[3] Misevičius, A. A modified simulated annealing algorithm for the quadratic assignment problem *Informatica*, 14(4), 497-514. 2003

[4] Burkard, R. et al. Quadratic Assignment Problem. http://anjos.mgi.polymtl.ca/qaplib/ . 2002

[5] Talbi, E. G. Metaheuristics: from design to implementation *John Wiley* 2009

[6] Ibid. 101

[7] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[8] Glover, F. Tabu search: Part I. *ORSA Journal on Computing*, 1(3):190–206, 1989.

[9] Lourenço, H. Iterated Local Search *Kluwer's International Series*, 340-352, 2003.

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

[11] Karisch, S. Nonlinear approaches for the quadratic assignment and graph partition problems. *Graz University of Technology*, Graz, Austria, 1995.

[12] Li, Y. Heuristic and exact algorithms for the quadratic assignment problem. *The Pennsylvania State University*, USA, 1992.

[13] Johnson, T. New linear programming-based solution procedures for the quadratic assignment problem *Clemson University*, Clemson, USA, 1992.

Table 8: Porcentajes del error relativo y tiempo de cómputo en segundos de las distintas metaheurísticas para las instancias Esc

Instancia	Annealing	segundos	Tabu Search	segundo	ILS	segundos	GA	segundos
esc16a	8.82	2	0	1.12	2.94	2	0	2
esc16b	0	2	0	1.12	0	2	0	2
esc16c	4.38	2	0	1.13	0.62	2	0	2
esc16d	12.5	2	0	1.13	12.5	2	6.25	2
esc16e	10.71	2	0	1.13	10.71	2	0	2
esc32a	117.69	2	13.08	2	19.23	2	43.85	2
esc32b	104.76	2	17.86	2	31.54	2	48.81	2
esc32c	15.89	2	0	2	0.31	2	0.16	2
esc32d	36	2	2	2	6.5	2	9	2
esc32e	0	2	0	2	0	2	0	2
esc64a	74.14	10	1.72	10	2.58	10	6.9	10
promedios	34.99		3.15		7.9		10.45	