ESTRUCTURA HIBRIDA DE VECINDAD PARA EL PROBLEMA DEL EMPAREJAMIENTO DE PESO MAXIMO

Marco Antonio Cruz-Chávez, Yessica Yazmin Calderón-Segura
Universidad Autónoma del Estado de Morelos, CIICAp
Avenida Universidad 1001. Col. Chamilpa.
C.P. 62210. Cuernavaca, Morelos.
{mcruz, ycaderons}@uaem.mx

Resumen:
En este artículo se presenta el procedimiento para encontrar el emparejamiento peso máximo de un grafo no dirigido mediante un análisis experimental de cuatro estructuras de vecindad incluyendo una estructura de vecindad híbrida, la cual se conforma de cuatro estructuras que llevan por nombre el siguiente: estructura de vecindad con un par aleatorio, estructura de vecindad con dos pares aleatorios, estructura de vecindad con un tres pares aleatorios, estructura de vecindad con cuatro parares aleatorio. Cada estructura de vecindad fue analizada detenidamente para probar la eficacia y la eficiencia dentro del problema de emparejamiento de peso máximo. Este análisis demostró cuál de las cinco estructuras de vecindad llevaba a cabo una mejor exploración y explotación del espacio de soluciones para este problema de optimización que es el de emparejamiento de peso máximo.

La estructura de vecindad que demostró ser la más eficiente y eficaz en comparación con las estructuras de vecindad antes mencionadas es la estructura de vecindad con un par aleatorio esta será descrita más adelante.

Palabras Claves: Optimización, emparejamiento máximo, grafo simple, grafo bipartido,grafo dirigido,grafo no dirigido, emparejamiento de peso máximo, problema Np-Completo.

1. Introducción

En la área de optimización combinatoria muchos problemas de interés práctico pueden ser intratables mediante técnicas de cálculo .Ante esta dificultad, los algoritmos heurísticos y meta-heurísticos (o de aproximación) resultan ser muy útiles para encontrar soluciones óptimas para la solución de estos problemas. Por ejemplo una colonia de hormigas, algoritmos genéticos, algoritmo de recosido simulado [23] etc.
En este trabajo se hace uso del problema de emparejamiento de peso máximo para encontrar la solución inicial, de la búsqueda local que se describe en la sección tres , la aplicación de la búsqueda local iterada que se describe en la sección cuatro y el desarrollo de la aplicación de las estructuras de vecindad.

El problema de emparejamiento de peso máximo que requiere de la búsqueda de soluciones aproximadas[6], debido a su complejidad del mismo se encuentra clasificado como un problema Np-Completo por ello se ha empleado técnicas heurísticas de búsqueda aproximada[14], ya que este problema encuentra un nueva solución en cada iteración. Entonces, para problemas grandes puede ser conveniente hallar el emparejamiento de peso máximo en forma aproximada mediante alguna heurística, la heurística propuesta en este artículo es la búsqueda por vecindad hibrida, las cuales han demostrado ser métodos eficientes en búsqueda de soluciones aproximadas para diversos problemas. Como es el problema de Neighborhood Hybrid Structure for Discrete Optimization Problems [4], Hybrid Variable Neighbourhood Search Algorithm for Attribute Reduction in Rough Set Theory., Data Mining and Optimisation Research Group[20] , Hybrid Variable Neighbourhood Search Algorithm for Attribute Reduction in Rough Set Theory[21].

La búsqueda de vecindad propuesta en este artículo es la hibrida [4], para desarrollar esta estructura, se realizó un revisión en la literatura, para buscar las estructuras de vecindad que contaran con baja complejidad computacional y que hayan tenido buenos resultados en su aplicación.

Las pruebas experimentales se realizaron para el problema de emparejamiento de peso máximo, debido a la complejidad de este problema en el espacio de soluciones, las estructuras de vecindad permiten manejar los resultados con más facilidad.

La literatura demuestra que la búsqueda por vecindad hibrida siempre ha dado buenos resultado para los siguientes problemas:

- Optimización de colonia de hormigas(AntRSAR)por Jensen and Shen[3]and [5].
- Optimización de colonia de hormigas (ACOAR) by LiangJun Ke ET[21]
- Algoritmo de recosido simulado(SimRSAR) por Jensen and Shen[22]

- Algoritmos genéticos(GenRSAR) por Jensen and Shen[22]
- Búsqueda Tabú(TSAR) by Hedaret[22].
- Depresión de las búsqueda(SSAR) by Wang et al[7] and [23].
- Teoría de conjuntos en bruto (HVNSAART) by Yahyaz and Salwani[20].
- Problema de Optimización (NHSDOP) by Marco Antonio Cruz and Alina Martínez[4].

  El presente trabajo de muestra con resultado que para el caso de un problema de emparejamiento de peso máximo la estructura hibrida aleatoria propuesta se encuentra arroja un promedio en eficiencia y eficacia y la estructura con mejor eficiencia y eficacia es la estructura con un par aleatorio los resultados se muestran en la tabla 2.

Esta estructura de vecindad híbrida permite la incorporación de las ventajas de las cuatro estructuras de vecindad que serán explicadas más adelante de modo que los movimientos realizados por la estructura de vecindad hibrida interactúen de forma aleatoria para la obtención de mejores resultados. De esta forma que se logra exploración y explotación del espacio de soluciones para alcanzar algún optimo local.

El presente artículo se divide en las siguientes secciones:

La sección número uno, contiene la introducción, La sección número dos, define el problema de emparejamiento de peso máximo, el cuál es utilizado para nos da la primera solución para poder aplicar las estructuras de vecindad. La sección número tres, presenta las estructuras de vecindad, la búsqueda local y la búsqueda local iterada que se utilizan en este trabajo. La sección cuatro detalla las pruebas experimentales, así como el funcionamiento de cada estructura de vecindad empleada. La sección cinco presenta las conclusiones obtenidas en esta investigación.

2. Emparejamiento de peso máximo

El presente trabajo de investigación trata al problema de emparejamiento de peso máximo en un grafo no dirigido; En la actualidad muchos problemas de la vida real han sido esquematizados a través de grafos, para poder hallarles una solución a través de la optimización, dado a sus diversas aplicaciones en la vida real, este puede ser adaptado a problemas que lo requieran, un ejemplo es aplicado en el área laboral .Que en diversos estudios sobre del comportamiento de las relaciones entre la educación superior y el empleo constituye siempre una cuestión de máximo interés en los egresados[11].

Podemos encontrar una gran cantidad de problemas de en donde podemos aplicar el emparejamiento de peso máximo, tanto en la industria como en la ciencia. Desde los clásicos problemas de diseño de redes de telecomunicación u organización de la producción hasta los más actuales en ingeniería y re-ingeniería de software, existe una infinidad de problemas teóricos y prácticos en donde el problema de emparejamiento de peso máximo puede ser aplicado de forma teórica.

Hallar un emparejamiento de peso máximo en un grafo puede resolverse en tiempo polinomial en la cantidad de nodos del grafo [17],[18]. Pero, por un lado si aplicamos el problema de emparejamiento de peso máximo en el área laboral, Sea un grafo G = (V, E),[19] ; donde V define los vértices o trabajos a desarrollar, V = {1, …, n} y E representa las aristas o eficiencia de cada trabajo desarrollado el vértice i a uno j, Se supone que cada individuo puede realizar algunas de las tareas. Se pretende asignar a cada individuo una única tarea de modo que se realicen el mayor número posible de estas. Si se supone que cada individuo sólo va a realizar una tarea, las demandas son todas 1 y las disponibilidades 1. Los costos de cada celda se definirán con 1 ó 0. Un 1 significa que el individuo puede hacer esa tarea y un 0 que no puede hacerla[15]. por lo que E = {(i, j) : i, j € V}, y sea cij el costo asociado al vértice (i, j) sea xij el emparejamiento que solo contiene una sola arista, de acuerdo a esto se muestra la formulación matemática para el emparejamiento máximo de un grafo se presenta en la figura 2.1.1

Esta es la matriz de los costos en donde (C_{ij}=1 si el individuo *i* puede realizar la tarea *j y es 0* en caso contrario).Realizar el mayor número de tareas es equivalente a maximizar $\sum c_{ij}$

2.1 Modelo matemático

La figura 2.1.1 muestra el modelo matemático utilizado para la solución inicial

$$Max\ \mathrm{f} = \sum_{\mathrm{i}=1}^{2\mathrm{n}-1} \sum_{\mathrm{j}=\mathrm{i}+1}^{2\mathrm{n}} \mathrm{c}_{\mathrm{ij}} x_{ij}$$

$$s.a.$$

$$(1) \quad \sum_{\mathrm{k}=1}^{\mathrm{i}-1} x_{ki} + \sum_{j=i+1}^{2n} x_{ij} = 1, i = 1..2n$$

$$(2) \quad i < j$$

$$(3) \quad x_{ij} \in \{0,1\}$$

Figura 2.1.1 Modelo matemático para el emparejamiento de peso máximo

Las restricciones básicas del problema se enlistan a continuación:

1. Cada trabajador i debe ser evaluado solo una vez.
2. Cada individuo j debe desempeñar un solo trabajo i
3. Indica si un individuo j fue asignado o no a la i trabajo.

De acuerdo al modelo matemático presentado en la figura 2.1.1, tenemos que las restricciones representadas por (1), (2) y (3) que especifican que cada trabajador debe desarrollar un trabajo sólo una vez, con la finalidad de cumplir la función objetivo que se presenta en la ecuación (3) y que maximizar el costo total de la eficiencia de cada trabajo.

3 Búsqueda por Vecindad

En esta sección, se presenta la idea básica de la búsqueda por vecindad y la aplicación a la estructura de vecindad.

Se utiliza la búsqueda por vecindad por que es una técnica utiliza en los problemas de optimización para mejorar una solución inicial en donde el tamaño del espacio de soluciones permite obtener el óptimo local [8],[10] y [12].

Para aplicar la búsqueda local a un problema particular, sólo se necesita para especificar la estructura de vecindad y una solución factible inicial [3].

La búsqueda por vecindad trata de encontrar la mejor solución evaluando la función objetivo del problema de emparejamiento para encontrar su valor máximo.

Para determinar la estructura de una vecindad, se debe definir un vecindario y el criterio de selección de un vecino. Es decir, si se cumple el criterio de selección, se lleva a cabo el movimiento, el proceso se repite hasta que la solución encontrada no pueda ser mejorada, por lo que se dice que hemos llegado a un óptimo local. El tipo de movimiento a realizar para seleccionar un vecino se define en la estructura de la vecindad propuesta .De lo contrario, la solución inicial se mantiene como el óptimo local con respecto a la zona de la estructura[3],[4].

3.1 Estructura de Vecindad Par Aleatorio

En esta sección, se presenta la estructura de vecindad con un par aleatorio y se da una breve explicación del procedimiento a seguir.

Para la estructura de búsqueda por vecindad con un par aleatorio se genera una solución *s* Inicial factible a partir de la cuál se elige un número aleatorio Gv (Generar vértice aleatorio), mismo que corresponde a una posición de una matriz en donde se encuentra almacenada la solución inicial.

Una vez que se tiene el vértice **Gv**, se verifica que este corresponda a la solución inicial (*figura 3.1.1*) se toma en cuenta el número de arcos del vértice elegido y se elige uno nuevo vértice **V1** de los posibles vértices a conectar, se examina si este es factible si este es factible se realiza el cambio y se obtiene la nueva solución vecina *s´* en caso contrario se mantiene la solución si modificaciones .

La figura 3.1.1.2 representa el primer movimiento de un par aleatorio.

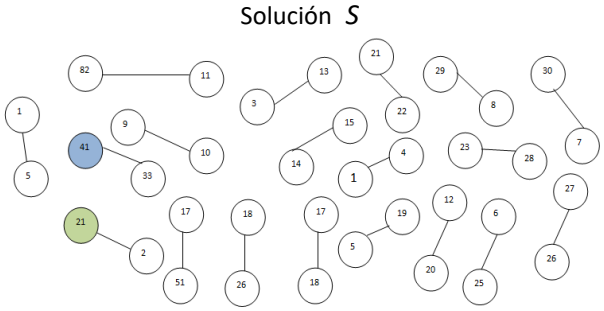


Figura 3.1.1 Solución Inicial

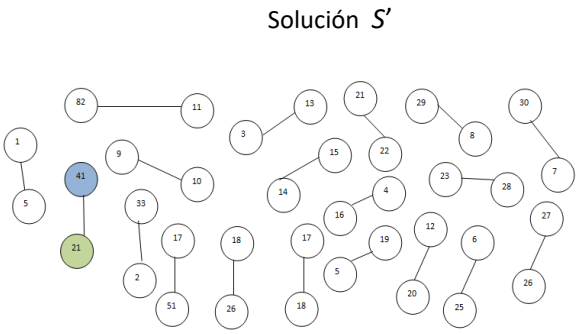
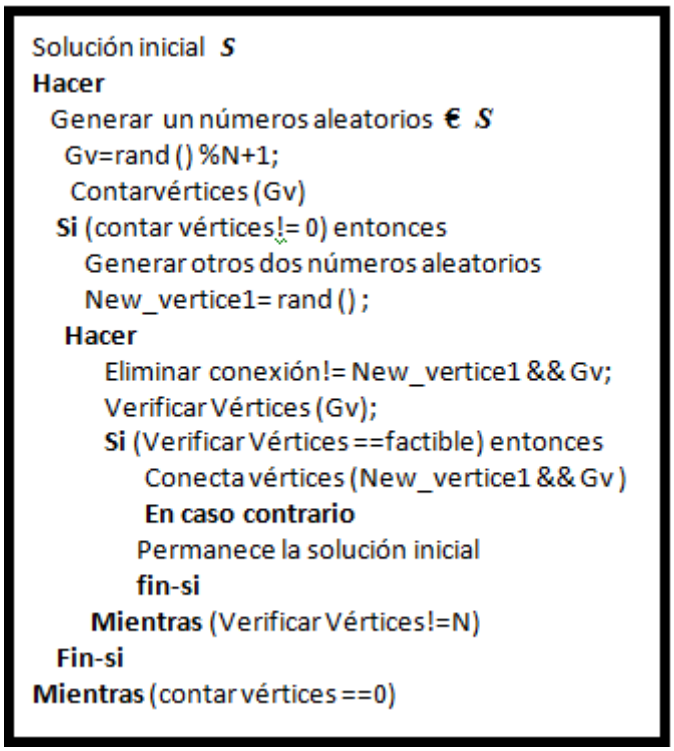


Figura 3.1.1.2.Estructura de vecindad con un par aleatorio

La figura 3.1.1.3 muestra el pseudocódigo utilizado para la estructura de datos con un par aleatorio.



```
Solución inicial S
Hacer
  Generar un números aleatorios € S
   Gv=rand () %N+1;
   Contarvértices (Gv)
  Si (contar vértices!= 0) entonces
     Generar otros dos números aleatorios
     New_vertice1= rand () ;
   Hacer
       Eliminar conexión!= New_vertice1 && Gv;
       Verificar Vértices (Gv);
       Si (Verificar Vértices ==factible) entonces
            Conecta vértices (New_vertice1 && Gv )
            En caso contrario
           Permanece la solución inicial
           fin-si
      Mientras (Verificar Vértices!=N)
  Fin-si
Mientras (contar vértices ==0)
```

Figura 3.1.1.3 Estructura de vecindad con un par aleatorio

3.2 Estructura de vecindad con dos pares aleatorios

La estructura de vecindad con dos pares aleatorios, se desarrolla mediante dos vértices que sean generados de forma aleatoria, que cumplan con la condición presentada en la figura (*figura 3.1.1.3*), y el segundo vértice debe ser distinto al primer vértice .En la figura 3.2.1.2 se muestra el movimiento desarrollado en la estructura de vecindad. Y la figura 3.2.1.3 muestra el pseudocódigo utilizado.

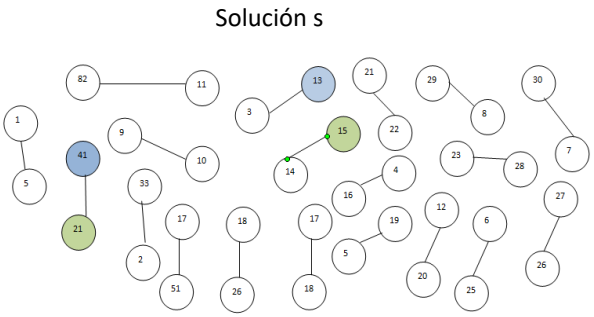


Figura 3.2.1 Solución inicial

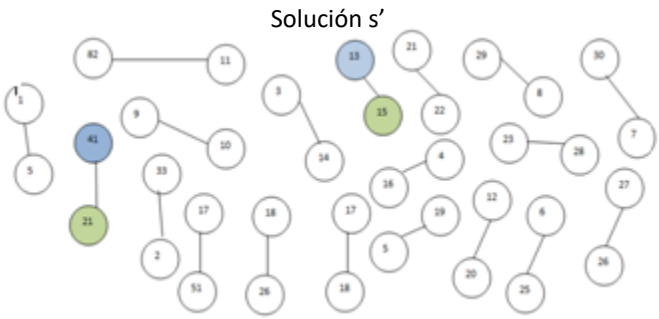


Figura 3.2.1.2 Estructura de vecindad con dos par aleatorio

```
Solución inicial S
Hacer
  Generar dos números aleatorios € S
    Gv=rand () %N+1;
    Gv 2=rand () %N+1;
    Mientras (Gv2== Gv)
      Contarvértices (Gv &&Gv2)
  Si (contar vértices!= 0) entonces
     Generar otros dos números aleatorios
     New_vertice1= rand ();
     New_vertice2= rand ();
  Hacer
       Eliminar conexión!= New_vertice1 && Gv;
       Eliminar conexión!= New_vertice2&&Gv2;
      Verificar Vértices (Gv);
       Si (Verificar Vértices ==factible) entonces
            Conecta vértices (New_vertice1 && Gv )
            Conecta vértices (New_vertice2 && Gv2)
          En caso contrario
           Permanece la solución inicial según
          se desarrolle cada movimiento
           fin-si
      Mientras (Verificar Vértices!=N)
  Fin-si
Mientras (contar vértices ==0)
```

Figura 3.2.1.3 Pseudocódigo de la estructura de vecindad con dos par aleatorio

3.3 Estructura de vecindad con tres pares aleatorios

La estructura de vecindad con tres pares aleatorios , es necesario el generar tres números aleatorios que cumplan el mismo procedimiento mostrado en la *figura 3.1.1.3* ,y agregando al código la siguiente condición : el tercer vértice generado no debe ser igual al segundo vértice y el primer vértice debe ser distinto al segundo y tercero y se cumple la condición se desarrolla el movimiento en caso contrario no se desarrolla ningún movimiento.En la figura 3.2.1.2 se muestra el movimiento desarrollado en la estructura de vecindad con tres pares aleatorios.

Solución *S*

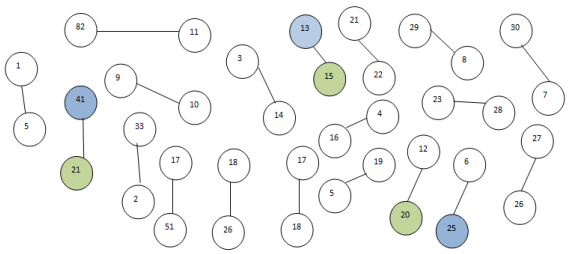


Figura 3.2.1.1 Solución inicial

Solución *S'*

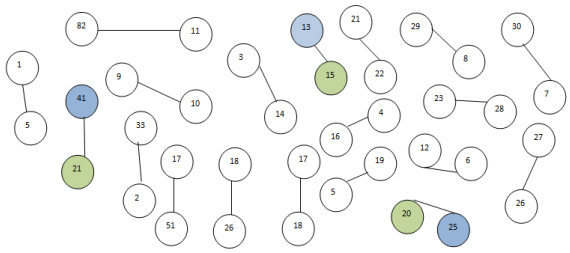


Figura 3.2.1.2 .Estructura de vecindad con tres pares aleatorios

```
Solución inicial S
Hacer
  Generar tres números aleatorios € S
    Gv=rand () %N+1;
    Gv 2=rand () %N+1;
    Gv 3=rand () %N+1 ;
  Mientras (Gv3== Gv | | Gv3== Gv2)
      Contarvértices (Gv &&Gv 1&& Gv3)
  Si (contar vértices!= 0) entonces
      Generar otros dos números aleatorios
      New_vertice1= rand () ;
      New_vertice2= rand () ;
      New_vertice3= rand () ;
   Hacer
       Eliminar conexión!= New_vertice1 && Gv;
       Eliminar conexión!= New_vertice2&&Gv2;
       Eliminar conexión!= New_vertice3&&Gv3;
      Verificar Vértices (Gv);
       Si (Verificar Vértices ==factible) entonces
           Conecta vértices (New_vertice1 && Gv )
           Conecta vértices (New_vertice2 && Gv2)
           Conecta vértices (New_vertice3&& Gv3)
         En caso contrario
          Permanece la solución inicial según
         se desarrolle cada movimiento
          fin-si
      Mientras (Verificar Vértices!=N)
  Fin-si
Mientras (contar vértices ==0)
```

Figura 3.2.1.3 Pseudocódigo de la estructura de vecindad con tres pares aleatorios

3.4 E

3.4 Estructura de vecindad con cuatro pares aleatorios

La estructura de vecindad con cuatro pares aleatorios , se generan cuatro vértices aleatorios que cumplan el mismo procedimiento mostrado en la *figura 3.4.1.3* ,y se coloca la siguiente condición al código :el cuarto vértice aleatorio debe ser distinto al tercer vértice generado y no debe ser igual al segundo vértice ni al primer vértice. Los cuatro deben ser distinto y si se cumple la condición se desarrolla el movimiento de la nueva estructura de vecindad en caso contrario no se desarrolla ningún movimiento.

En la figura 3.4.1.2 se muestra el movimiento desarrollado en la estructura de vecindad con cuatro pares aleatorios.

Solución s

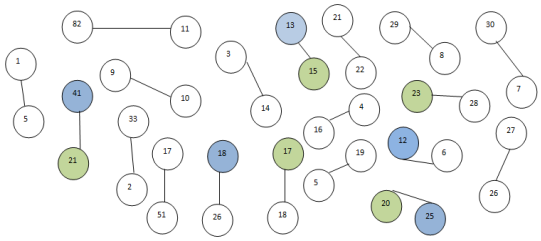


Figura 3.4.1.1 Solución inicial

Solución s'

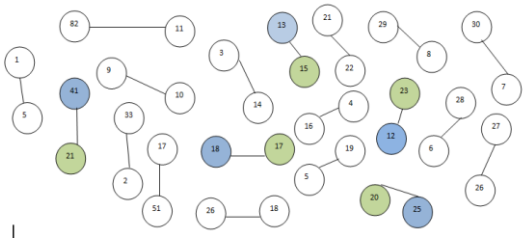


Figura 3.4.1.2 .Estructura de vecindad con tres par aleatorio

```
Solución inicial S
Hacer
  Generar cuatro números aleatorios € S
   Gv=rand () %N+1;
    Gv 2=rand () %N+1;
    Gv 3=rand () %N+1;
    Gv 4=rand () %N+1;

  Mientras (Gv4== Gv|| Gv4== Gv2 || Gv4== Gv3)
      Contarvértices (Gv &&Gv 1&& Gv3&& Gv4)
  Si (contar vértices!= 0) entonces
     Generar otros dos números aleatorios
     New_vertice1= rand () ;
     New_vertice2= rand () ;
     New_vertice3= rand () ;
    New_vertice4= rand () ;
   Hacer
       Eliminar conexión!= New_vertice1 && Gv;
       Eliminar conexión!= New_vertice2&&Gv2;
       Eliminar conexión!= New_vertice3&&Gv3;
       Eliminar conexión!= New_vertice4&&Gv4;
      Verificar Vértices (Gv);
       Si (Verificar Vértices ==factible) entonces
            Conecta vértices (New_vertice1 && Gv )
            Conecta vértices (New_vertice2 && Gv2)
            Conecta vértices (New_vertice3&& Gv3)
          Conecta vértices (New_vertice 4&& Gv4)
         En caso contrario
           Permanece la solución inicial según
           se desarrolle cada movimiento
           fin-si
      Mientras (Verificar Vértices!=N)
   Fin-si
Mientras (contar vértices ==0)
```

Figura 3.4.1.3 Pseudocódigo de la estructura de vecindad con cuatro tres pares aleatorios

3.5 Estructura de vecindad hibrida con pares aleatorios

Esta estructura parte de una solución inicial factible que de forma aleatoria elige que tipo de estructura de vecindad que va a ejecutar el algoritmo mostrado en la figura 3.6 .Este cuenta con un menú de 4 opciones aleatorias, cada opción contiene una estructura lista para ejecutarse .Dependiendo del número aleatorio generado será la estructura que lleve a cabo el movimiento para obtener la nueva solución vecina. El procedimiento a seguir es el siguiente:

La estructura hibrida es una mezcla de las cuatro estructuras mencionadas anteriormente y en las figura mostrada a continuación son los nuevos movimiento aplicados siempre y cuando cumplan con las condiciones de cada estructura elegida.

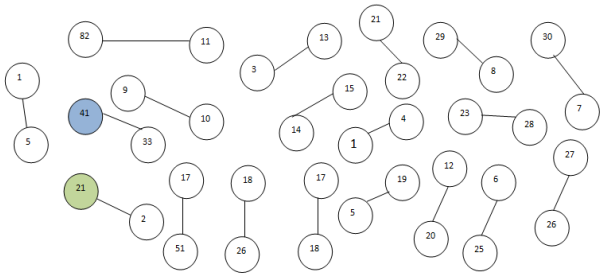


Figura 3.5.0 Solución inicial

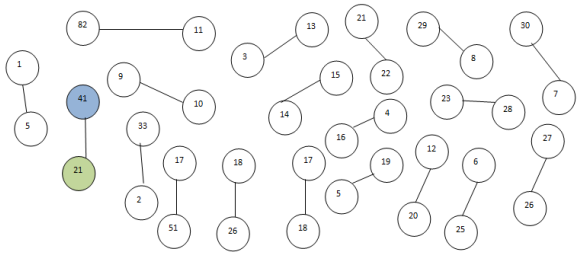


Figura 3.5.1 .Estructura de vecindad con un par aleatorio

Figura 3.5.1 .Estructura de vecindad con dos par aleatorio

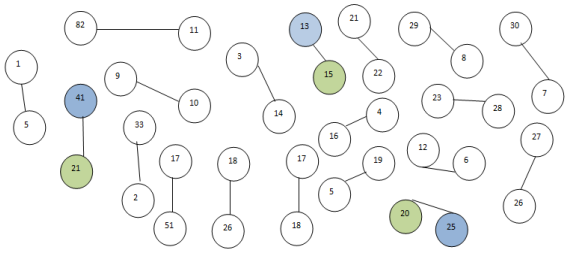


Figura 3.5.1 .Estructura de vecindad con tres par aleatorio

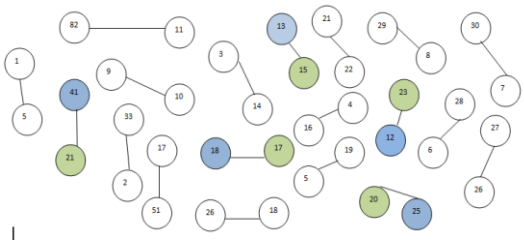


Figura 3.5.1 .Estructura de vecindad Hibrida

3.6 Diagrama de flujo de la estructura de vecindad hibrida

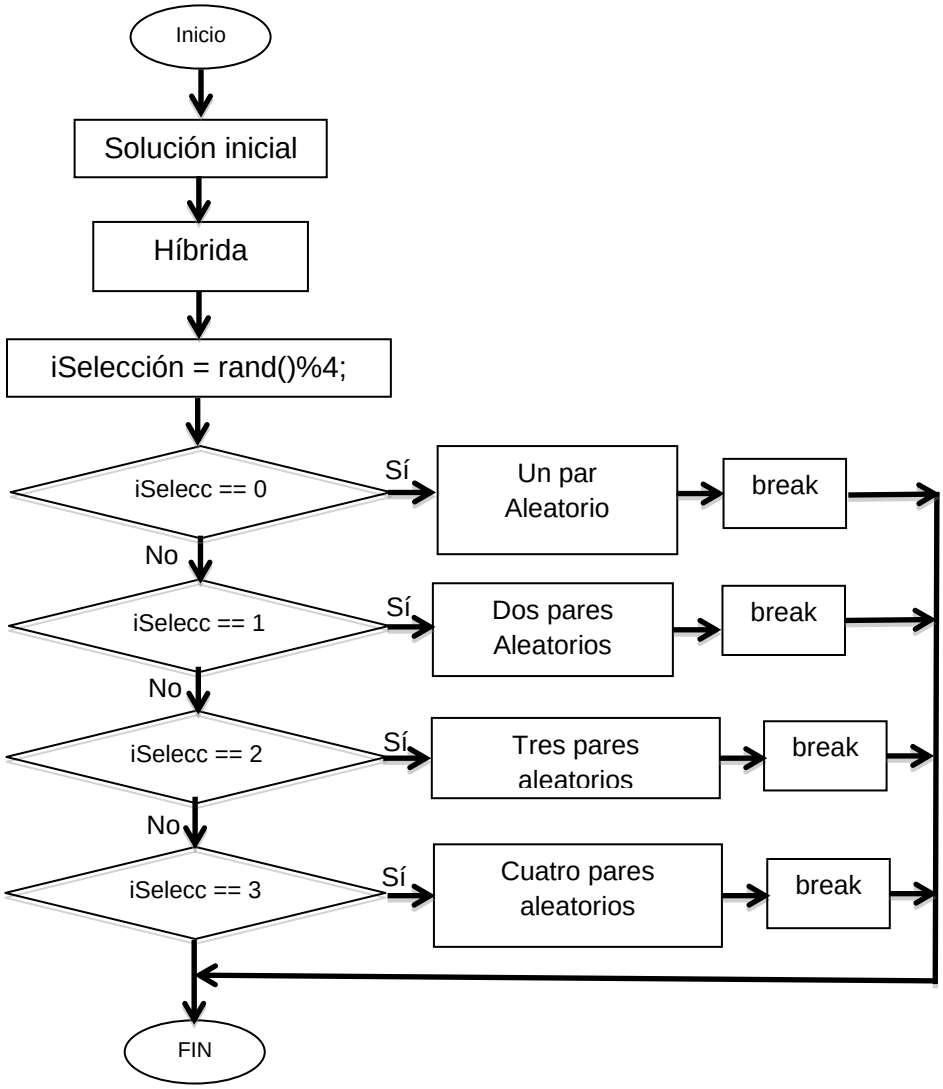


Figura 3.6 Diagrama de flujo de la estructura de vecindad Hibrida

4. Búsqueda local iterada

La Búsqueda Local Iterada (ILS, Iterated Local Search) [], es una meta heurística que propone una estrategia exploratoria de mejora de las soluciones obtenidas por una determinada heurística mediante la iteración. Un esquema en el se incluye una heurística base que mejora los resultados mediante la repetición de dicha heurística. Esta idea ha sido propuesta en la literatura con distintas denominaciones, como descenso iterado, grandes pasos con cadenas de Markov, Lin-Kerningan iterado, búsqueda perturbada o ruidosa o la búsqueda de entorno variable con agitación donde la solución aportada por una heurística de búsqueda por entornos es agitada para producir una solución de partida para la heurística de búsqueda.

La estrategia ILS actúa de la siguiente forma:

La búsqueda local iterada se empieza con una solución inicial s , y a partir de esta,se genera un proceso de búsqueda local dentro de un subespacio definido por los óptimos relativos(Lourenco et al.,2001).Un algoritmo de búsqueda transforma una solución cualquiera s en otra s' que es el óptimo local. Una vez que el proceso de solución encuentra un óptimo local en la que se estanca este óptimo se perturba hacia una solución diferente que no incluya las soluciones de la última búsqueda local este procedimiento se desarrollo mediante las estructuras de vecindad. La nueva solución, después de haber sido perturbada o hecho un movimiento, sirve de punto de partida para generar un segundo grupo de soluciones con un nuevo óptimo local s' ,a la cual se vuelve a aplicar el algoritmo de búsqueda local iterada para alcanzar el nuevo optimo local s* esto se repite hasta alcanzar el criterio de paro de la búsqueda local iterada.

En la figura 4.0, se muestra el algoritmo general de búsqueda local iterada.

```
Mientras no se cumpla la condición de paro de do

f(CS_ILS)=Dato;

Hacer
Evaluar costo Cs_actual =solución inicial s
   Hacer
      Evaluar costo Cs'_LS =solución movimiento σ/s
      Si (f (Cs'_LS) >= f (Cs_actual)) entonces
         Cs_LS'= mejor solución encontrada
         Cs_actual = Cs'_LS
      Fin-si
   Mientras Criterio de Paro de Búsqueda Local
   Si (f(Cs'_LS)>= f(CS_ILS)) entonces
      CS_ILS= Cs'_LS;
   Fin-si
Mientras Criterio de Paro de Búsqueda Local Iterada
```

Figura 4.0 Pseudocódigo de la búsqueda local iterada

4.1 Resultados Experimentales

Las pruebas experimentales realizadas para las estructuras de vecindad para el emparejamiento de peso máximo fueron realizadas en una computadora portátil con procesador Intel Core i7-740QM quad processor (1.73GHz) whith Turbo Boost up to 2.93 Ghz con sistema operativo Genuine Windows 7 Home Premium 64-bit . El compilador utilizado fué Visual C 2008 se ejecuto la estructura de vecindad con un par aleatorio ,la estructura de vecindad con dos pares aleatorios, la estructura de vecindad con tres pares aleatorios ,la estructura de vecindad con cuatro pares aleatorios y la estructura hibrida aleatoria .

Los tamaños de instancia probadas fueron de 100 vértices fueron generados de forma aleatoria. Las estructuras de vecindad fueron ejecutadas 30 veces para cada tamaño de instancia, teniendo un número total de 100 iteraciones realizadas durante la ejecución y registrando los resultados obtenidos de la mejor solución, en cuanto su función de costo. Cada algoritmo fue ejecutado con la respectiva estructura de vecindad para poder realizar una comparación de los resultados obtenidos y hacer un análisis y demostrar la eficiencia y la eficacia de la mejor estructura de vecindad que en este caso es la estructura de vecindad con un par aleatorio.

4.2 Pruebas de Eficiencia

Se realizó el cálculo del tiempo para encontrar 100 soluciones para cada instancia 100 nodos

La tabla 2 presenta los resultados obtenidos para cada estructura de vecindad. Desarrolladas en este articulo para encontrar el emparejamiento de peso máximo de las cuáles se hizo una evaluación de la mejor solución, peor solución, la función de costo promedio, así como su desviación estándar σ de las 30 pruebas realizadas con 100 iteraciones para cada estructura de vecindad. Se puede observar que la estructura de vecindad de par aleatorio en este trabajo es la más eficiente y eficaz.

La segunda mejor estructura en eficacia es la que realiza dos movimientos aleatorios para la instancia de 100 vértices.

La estructura promedio en eficiencia es la estructura hibrida aleatoria con un tiempo de 8.93 minutos aleatorios para la instancia de 100 vértices.

El comportamiento de la desviación estándar para cada estructura muestra que la dispersión de las soluciones en las 30 pruebas, es diferente dependiendo del permutación que se aplique, mientras mayor sea la desviación estándar, mayor será la variabilidad, es decir mayor explotación del espacio de soluciones.

Tabla 2 Resultados para 100 Vértices. 30 ejecuciones con la búsqueda local iterada para cada estructura.

100 VÉRTICES					
TIPO DE ESTRUCTURA	**TIEMPO**	**MEJOR**	**PEOR**	**PROMEDIO**	**σ**
Par Aleatorio	**2 min**	**4285**	**3502**	**4137.23**	**19.369**
Dos Pares Aleatorios	6.1053 min	4052	3239	3921.3	69.89
Tres Pares Aleatorios	40.38 min	3971	3168	3812	80.33
Cuatro Pares Aleatorios	558.516 min	3826	2901	72.62	3693.4
Híbrida	8.93 min	4030	3271	3881.7	69.59

La tabla 2 muestra el análisis completo de cada una de las 30 ejecuciones de cada estructura para 100 vértices y se resalta la estructura de vecindad con un par aleatorio que demostró ser la más eficiente con un tiempo de 2 minutos por ejecución y la más eficaz con un costo de 4285 que cumple con la función objetivo del emparejamiento de peso máximo.

Grafica de la estructura de vecindad con un par aleatorio

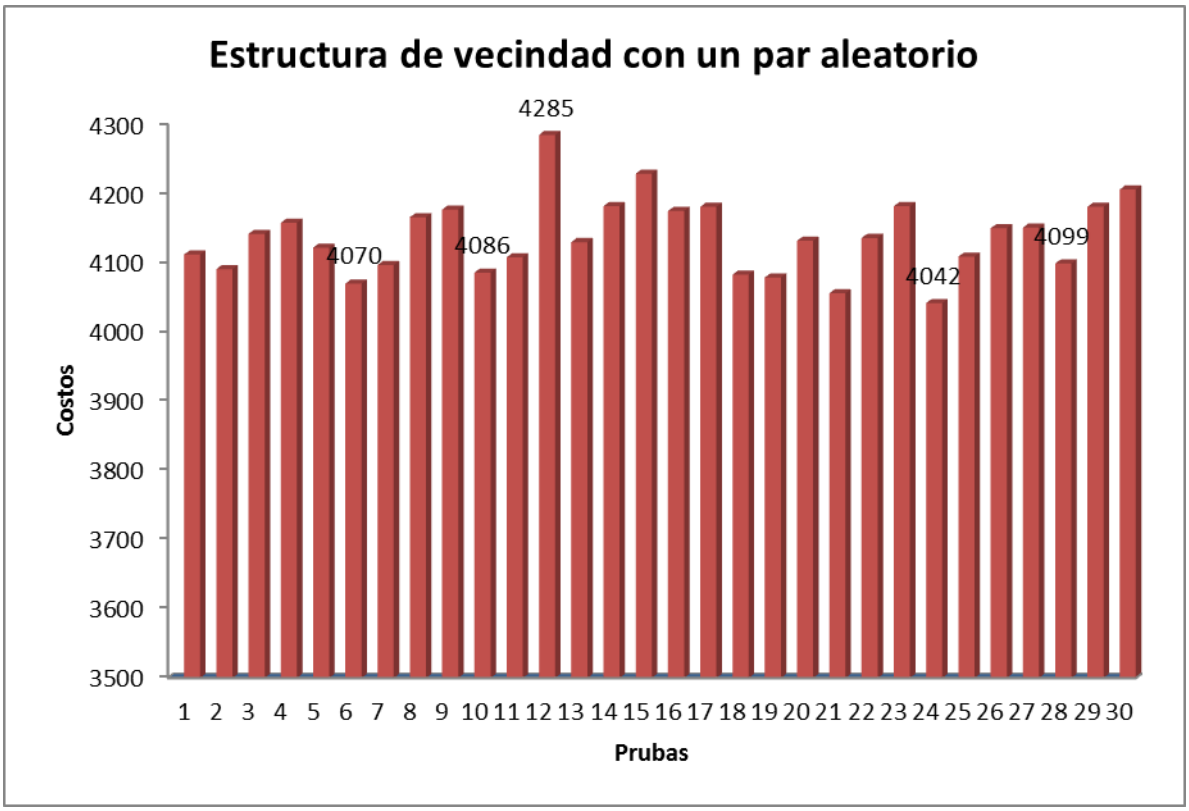


Figura 4.2.1 Pruebas de la estructura de vecindad con un par aleatorio

La figura 4.2.1 muestra los resultados obtenidos de 30 ejecuciones realizadas para la estructura de vecindad con un par aleatorio la función de costo muestra que para esta estructura la mejor solución de las 30 ejecuciones es la de 4285 y la peor es con un costo de 4042.

Grafica de la estructura de vecindad con dos pares aleatorio

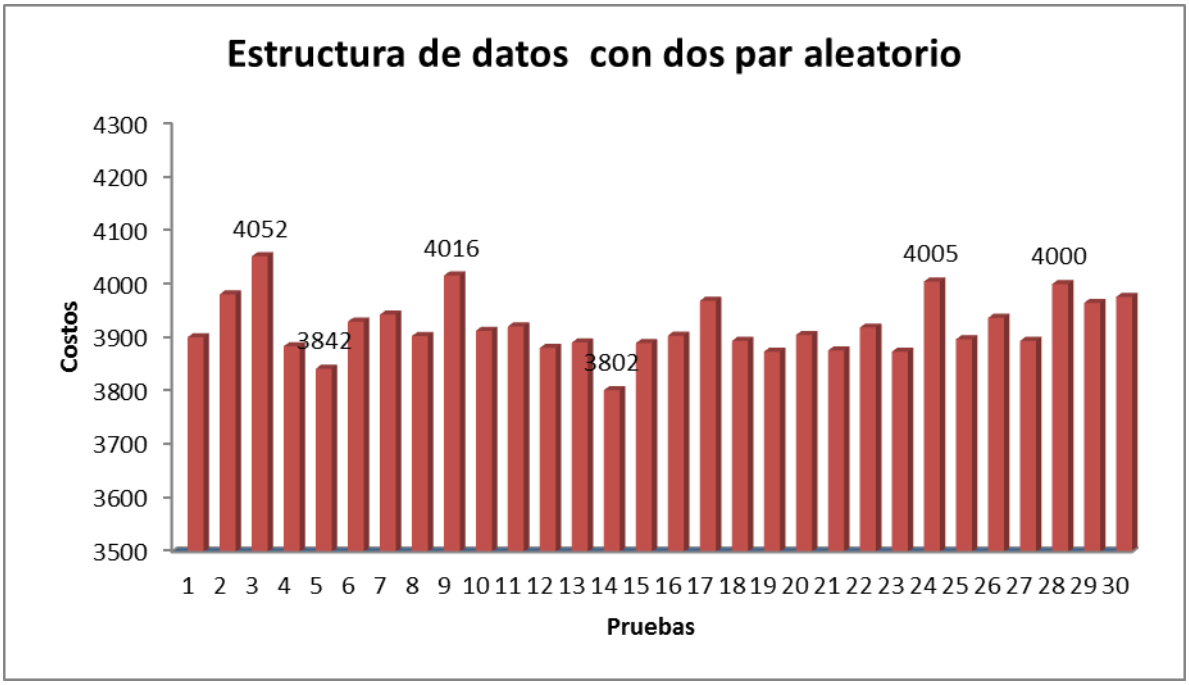


Figura 4.2.2 Pruebas de la estructura de vecindad con dos pares aleatorios

La figura 4.2.2 muestra los resultados obtenidos de las mejores soluciones de la función de costo para la estructura de vecindad con dos pares aleatorios. Se puede observar que para esta estructura la mejor solución encontrada de las 30 ejecuciones es de 4052.El peor costo de esta estructura de vecindad con dos pares aleatorios es de 3802.

Grafica de la estructura de vecindad con tres pares aleatorio

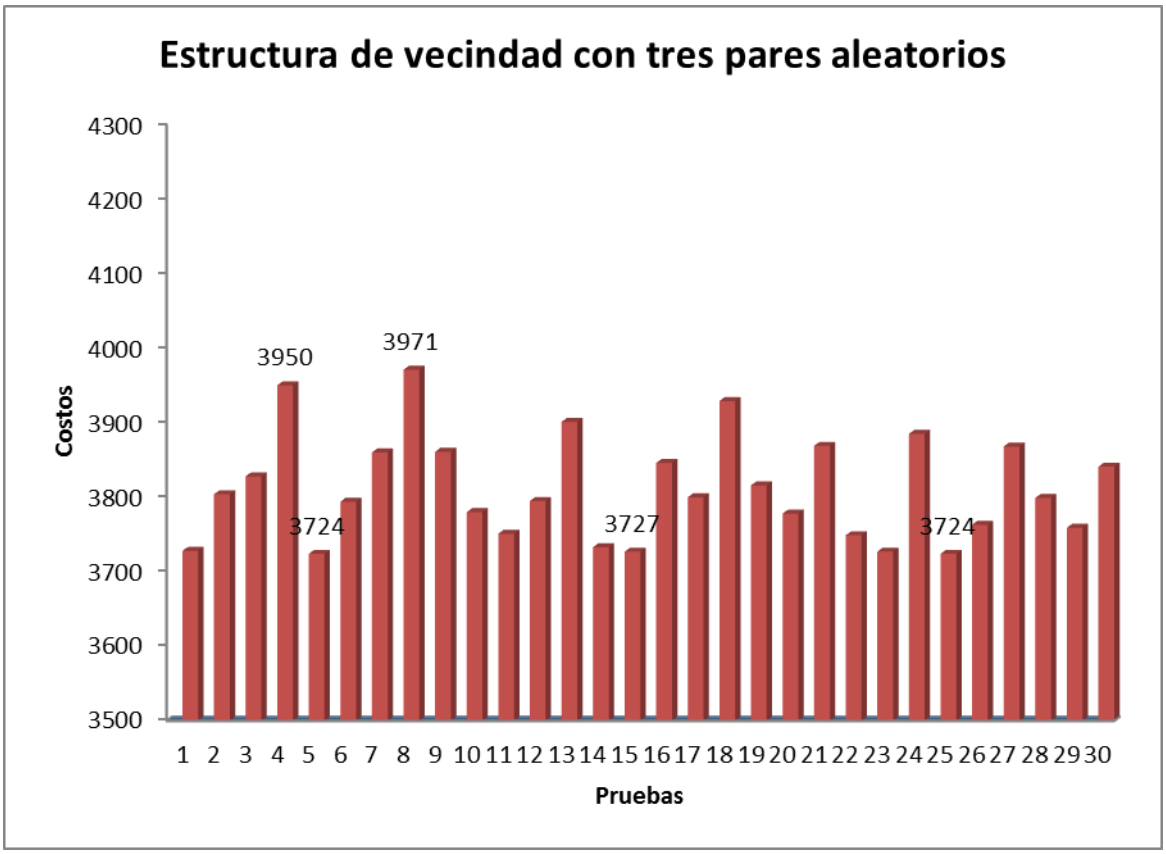


Figura 4.2.3 Pruebas de la estructura de vecindad con tres pares aleatorios

La figura 4.2.3 muestra los resultados obtenidos de las mejores soluciones de la función de costo para la estructura de vecindad con tres pares aleatorios. Se puede observar que para esta estructura la mejor solución encontrada de las 30 ejecuciones es de 3971.El peor costo de esta estructura de vecindad con tres pares aleatorios es de 3724.

Grafica de la estructura de vecindad con cuatro pares aleatorio

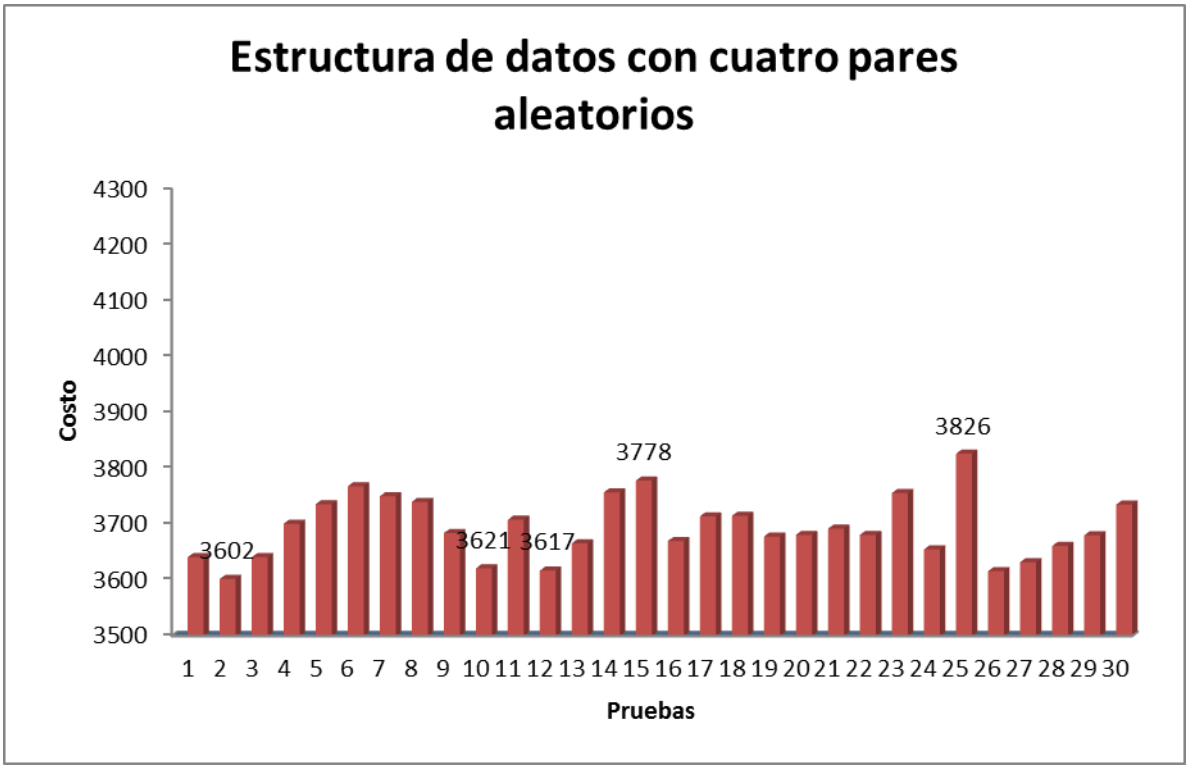


Figura 4.2.4 Pruebas de la estructura de vecindad con cuatro pares aleatorios

La figura 4.2.4 muestra los resultados obtenidos de las mejores soluciones de la función de costo para la estructura de vecindad con cuatro pares aleatorios. Se puede observar que para esta estructura la mejor solución encontrada de las 30 ejecuciones es de 3826.El peor costo de esta estructura de vecindad con tres pares aleatorios es de 3602.

Grafica de la estructura de vecindad hibrida aleatoria

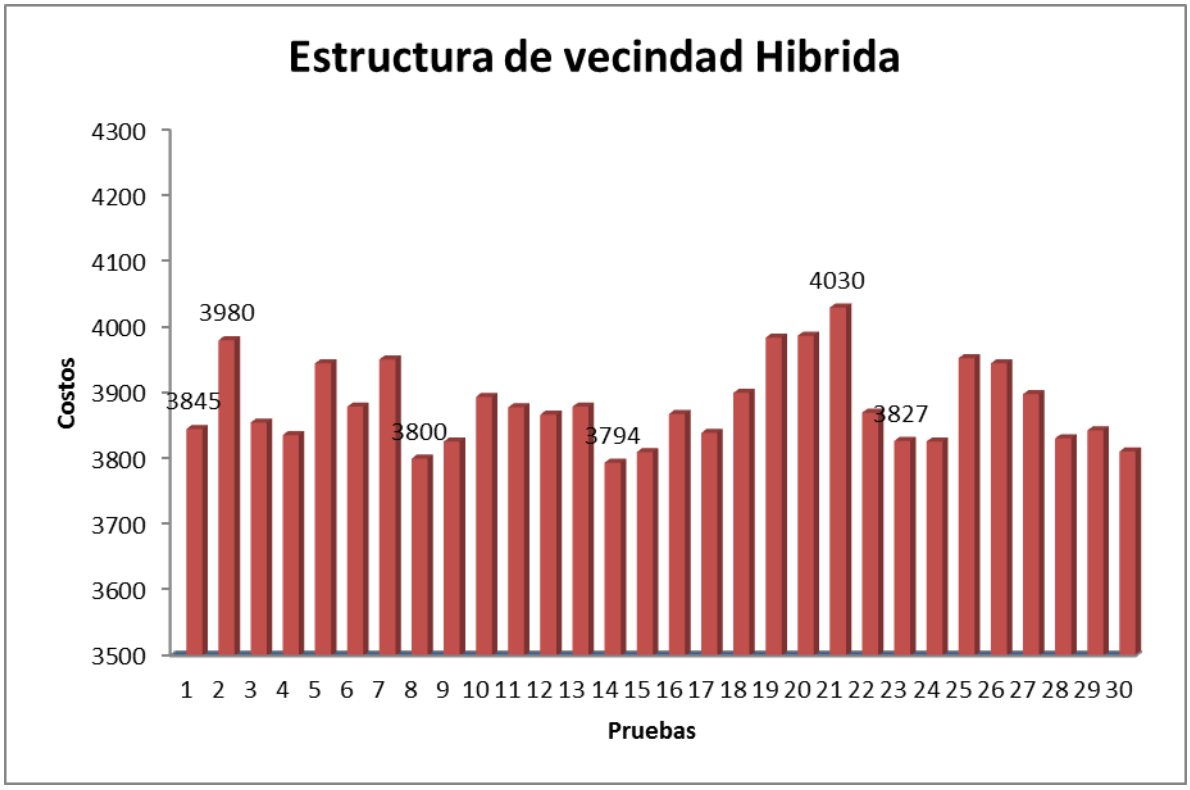


Figura 4.2.5 Pruebas de la estructura de vecindad hibrida aleatoria

La figura 4.2.5 muestra los resultados obtenidos de las mejores soluciones de la función de costo para la estructura de vecindad hibrida aleatoria. Se puede observar que para esta estructura la mejor solución encontrada de las 30 ejecuciones es de 4030.El peor costo de esta estructura de vecindad hibrida aleatoria es de 3794.

4.3 Trabajo Futuro

Se Aplicara la estructura de vecindad con un par aleatorio y la estructura hibrida aleatorio al Algoritmo de aceptación por umbral, para evaluar la eficiencia y eficacia de dicho algoritmo.

5 Conclusiones

Como se menciona en un principio el problema de emparejamiento de peso máximo se ha clasificado dentro de la teoría de complejidad como NP-completo, debido a que es un problema difícil de resolver, por ello se hace uso de las técnicas de búsqueda por vecindad y encontrar el óptimo local.

Este trabajo de investigación demuestra que la estructura con un par aleatorio en cuanto a eficacia es la mejor a comparación con las otras cuatro estructuras. En eficiencia la estructura híbrida esta en un rango intermedio a comparación con las otras estructuras, es decir no es la de mejor tiempo de ejecución, pero tampoco es la de peor rendimiento. Podemos decir que este tipo de estructura puede ser aplicada a una meta heurística.

Se concluye que la estructura hibrida propuesta no es la mas eficiente para el problema de emparejamiento máximo.

6 Referencias

[1] Cruz-Chávez M. A., Díaz-Parra O., Juárez Romero D., Barreto Sedeño E., Zavala Díaz C, Martínez Rangel M. G.,Un Mecanismo de Vecindad con Búsqueda Local y Algoritmo Genético para Problemas de Transporte con Ventanas de Tiempo, CICos 2008, 6to Congreso Internacional de Cómputo en Optimización y Software, ACD, ISBN(e) 978607 00-0165-9, ISBN(i) 978-970-9750-26-3, pp 23-32, 25-27 Junio, México, 2008.

[2] T. StÄutzle. Local search algorithms for combinatorial problems analysis, algorithms and news applications. DISKI Dissertationen zur KÄunstliken Intelligenz., 1999.

[3] [3] R. Jensen and Q. Shen, "Finding Rough Set Reducts with Ant Colony Optimization, " *Proc. 2003 UK Workshop Computational Intelligence*, pp. 15-22, 2003.

[4] Cruz-Chávez M.A, Martínez-Oropeza A., Serna Barquera S. A, Neighborhood Hybrid Structure for Discrete Optimization Problems, Electronics, Robotics and Automotive Mechanics Conference, CERMA2010, IEEE

Computer Society, ISBN 978-0-7695-4204-1, pp , September 28 - October 1, México, 2010.

[5] R. Jensen and Q. Shen, "Fuzzy-rough sets for descriptive dimensionality reduction, " in Fuzzy Systems, 2002. FUZZIEEE' 02. Proceedings of the 2002 IEEE International Conference on, 2002, pp. 29-34.

[6] Papadimitriou, C.H., Steiglitz, K. Combinatorial Optimization, Algorithms and Complexity. Dover Publications, Inc. Mineola, New York. USA ed. 1998.

[7] Michiels Wil, Aarts Emile, Korst Jan Theoretical Aspects of Local Search, Philips Research High Tech Campus 345656 AE Eindhoven The Netherlands and Eindhoven University of Technology P.O. Box 513 5600MB Eindhoven The Netherlands, 1998.

[8]. Wetzel. A. Evaluation of the Effectiveness of Genetic Algorithms in Combinatorial Optimization. University of Pittsburgh, Pittsburgh (unpublished). 1983.

[9] Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., Protasi, M.: Complexity and approximation: Combinatorial Optimization Problems and their Aproximability Properties. Springer-Verlag. 1999.

[10] HOOS, H.H.; STÜTZLE, T. Stochastic local search: foundations and applications, Morgan Kaufmann Publishers, San Francisco, CA, 2005.

[11] Cristina Borra, Francisco Gómez .:Los Factores determinantes del emparejamiento Educación-Empleo., Universidad de sivilla ., may, 2006., p.p. 1-11., IXV jornada de la Economía de la educación.

[12] Cruz-Chávez M. A, Rivera-López R., A Local Search Algorithm for a SAT Representation of Scheduling Problems, Lecture Notes in Computer Science, Springer-Verlag Pub., Berlin Heidelberg, ISSN: 0302-9743, Vol.4707, No. 3, pp. 697-709, 2007.

[13] Cruz-Chávez, M.A., Frausto-Solís, J., Zavala-Díaz, J.C., Sanvicente-Sánchez, H., Cruz-Rosales, M.H.: A Simulated Annealing Algorithm with Cooperative Processes for Scheduling Problems. LNCS. Springer, Heidelberg, ISSN: 0302-9743, 2006.

[14] Garey M. R., D. Johnson. "Computers and Intractability" a Guide to the Theory of NP-Completeness.Freeman. New York NY. ISBN 0-7167-1044-7. 1979.

[15] Gabow, H.: An efficient implementation of Edmonds' algorithm for maximum matching on graphs. Journal of the ACM 23 (1976) 221–234

[16] Daisuke Takafuji. ,Satoshi Taoka., Toshimasa Watanabe. :Efficent Approximation Algorithms for the Maximum Weinght Matching Problem,Graduate school of engineering. , Hiroshima University ., 1-4-1, Kagamiyama , Higashi-hiroshima, 739-8527 Japan, Feb. 2002.; ISSN:0-7803-7448-7 pp. 457-460, IEEE Computer Society Press.

[17]Yanfeng Zheng., Shutao Sun.:Parallelized Scheduling Algorithm for Imput Queued Switches Using Local Search Technique., Institute of computing technology Chinese academy of sciences Beijing 100080,china,May.2005 ,pp .43-48, ISSN:0-7803-8991-3., IEEE Computer Society Press.

[18]] Cruz-Chávez, M.A., Fredy Juárez-Pérez ., Erika Yesenia Ávila-Melgar.,Alina Martínez-Oropeza.: Simulated Annealing Algorithm for the weighted Unrelated Parallel Matching . CIICAp,Universidad Autónoma del Estado de Morelos Avenida Universidad 1001.Col.Chamilpa,C.P.62609.Cuernavaca,Morelos,Mexico.Sep,2009.,ISSN: 978-0-7695-3799-3 ,pp .94-99, IEEE ,Cerma.

[19] Thanh Minh Hoang.:On the Matching Problem for Special Graph Classes.,Institute for Theoretical Computer Science., University of Ul,,Germany., Jun,2010., ISSN: 1093-0559/10,pp .139-150, IEEE , Computer Society Press.

[20] Yahya Z.Aray., Salwani Abdullah.: Hybrid Variable Neighbourhood Search Algorithm for Attribute Reduction in Rough Set Theory.,Data Mining and Optimisation Research Group(DMO),Centre for Artificial Intelligence Technology University Kebangsaan Malaysia,43600 Bangi Selangor,Malaysia.,oct,2010., ISSN:978-1-4244-8136-1 , pp .110-120, IEEE,International Conference on Intelligent Systems Design and Application.

[21] L. Ke, Z. Feng, and Z. Ren, "An efficient ant colony optimization approach to attribute reduction in rough set theory," Pattern Recognition Letters, vol. 29, pp. 1351-1357, 2008.

[22] [8] A. Hedar, J. Wang, and M. Fukushima, "Tabu search for attribute reduction in rough set theory," Soft Computing – A Fusion of Foundations, Methodologies and Applications, vol. 12, pp. 909-918, 2008.

[23] J. Wang, A. Hedar, G. Zheng, S. Wang. "Scatter Search for Rough Set Attribute Reduction," in Computational Sciences and Optimization, 2009. CSO 2009. International Joint Conference on, 2009, pp. 531-535.