PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

RESOLUCIÓN DEL MANUFACTURING CELL DESIGN PROBLEM MEDIANTE BAT ALGORITHM

ANDRÉS MARCEL ALARCÓN ALBORNOZ
CAROLINA BELÉN ZEC SOTO

INFORME FINAL DEL PROYECTO
PARA OPTAR AL TÍTULO PROFESIONAL DE
INGENIERO CIVIL EN INFORMÁTICA

Julio 2015

PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

RESOLUCIÓN DEL MANUFACTURING CELL DESIGN PROBLEM MEDIANTE BAT ALGORITHM

ANDRÉS MARCEL ALARCÓN ALBORNOZ
CAROLINA BELÉN ZEC SOTO

Profesor Guía: **Ricardo Soto de Giorgis**

Profesor Co-referente: **Ignacio Araya Zamorano**

Carrera: **Ingeniería Civil Informática**

Julio 2015

Dedicatoria

Andrés:

A mi hija, quien hace casi dos años llego a mi vida
a llenarla de alegría y felicidad.

Carolina:

A Vedran, quien con su risa, inocencia y ternura ha alegrado
cada momento en mi vida, desde el día en que llegó

Agradecimientos

Andres:

A mi madre, por darme la vida y la posibilidad de cumplir mis sueños, por estar a mi lado siempre y apoyarme en los momentos difíciles, sin ella nada hubiese sido posible.

A mi abuelita, por ser un pilar fundamental en mi vida, quien me educó y entregó los valores que hoy me hacen ser mejor persona.

A mi hermana, por ser un gran ejemplo de que los sueños se pueden cumplir y por todos esos momentos vividos juntos desde pequeños.

A mi padre, por siempre demostrarme y hacerme entender lo importante que es estudiar.

A Carolina, por ser una gran compañera de Universidad, con quien finalizo este largo proceso.

Carolina:

A mis padres, quienes me han dado mis valores, amor y comprensión frente a todas las cosas, y por apoyarme en cada momento, porque han luchado incansablemente por verme feliz, sin sus palabras jamás hubiera entendido muchas cosas.

A Loreto y Yerko, quienes a lo largo de este proceso fueron mis grandes compañeros, que no cambiaría, porque de todo se aprende y me regalaron lindos recuerdos.

A mis hermanos y familia, quienes siempre tienen un abrazo para mí, también por su apoyo incondicional.

A mis profesores, quienes me entregaron las herramientas para llegar al final.

A mis amigos y compañeros de la universidad, con todos guardo grandes momentos. Las risas y buenos momentos siempre se quedan en el corazón, gracias por su apoyo.

A Andrés, por su compañerismo y atención durante este tiempo.

A Dios, quien me tomo de su mano y es mi gran guía.

Índice

Resumen

Manufacturing Cell Design es un problema destinado a la distribución de diferentes máquinas de un centro de producción en celdas con el objetivo de minimizar el movimiento interceldario de las partes que componen el producto final. Bat Algorithm es una metaheurística inspirada en el comportamiento de ecolocalización de los murciélagos. Una característica interesante de este algoritmo es que el balance entre la exploración y explotación se controla por las tasa de volumen y de emisión de pulsos de los murciélagos. El presente trabajo presenta la resolución del Manufacturing Cell Design Problem por medio de esta metaheurística. Ilustramos resultados prometedores en los que el algoritmo implementado es capaz de alcanzar el óptimo global para varias instancias del problema.

Palabras Claves: Metaheurística, Manufacturing Cell Design, Bat Algorithm.

Abstract

Manufacturing Cell Design is a problem aimed at distributing the different machines of a production center in cells, so that inter-cell movement of parts of final products is minimized. Bat Algorithm is a metaheuristic inspired by the echolocation behavior of bats. An interesting feature of this algorithm is that the balance between exploration and exploitation is controlled by the rate of volume and emission pulses of bats. The present work presents the resolution of the Manufacturing Cell Design Problem by means of this metaheuristic. We illustrate promising results where the implemented bat algorithm is able to reach the global optimum for several instances.

Keywords: Manufacturing Cell Design, Bat Algorithm.

Índice de Tablas

Índice de Figuras

1 Introducción

La búsqueda de una solución a un determinado problema, se hace cada vez más compleja cuando este tiene un espacio de búsqueda muy grande. Existen los llamados métodos aproximados, y dentro de este grupo se clasifican las denominadas Metaheurísticas, que buscan una solución cercana al óptimo global en un tiempo acotado y con una buena utilización de recursos. Estos métodos recorren el espacio de búsqueda alcanzando una solución aproximada del problema.

El Manufacturing Cell Design Problem (MCDP) es un problema organizacional basado en Group Technology (GT), que es una metodología común para el incremento de la productividad dentro de una empresa manufacturera. La formación de celdas es un paso clave en GT y es el problema que se intentará resolver. La solución consiste en disponer las máquinas de una fábrica para la elaboración de un producto, dentro de una cantidad limitada de celdas, que se encargarán de dividir el centro en diferentes grupos, de tal forma que los movimientos de las partes entre las celdas sean minimizados. Para ello se requiere identificar las familias de partes contenidas en el proceso de fabricación del producto.

Para resolver el problema que se plantea, se utilizará Bat Algorithm (BA), que es un algoritmo de optimización basado en el comportamiento natural de los micro-murciélagos, ecolocalización, el cual determina la distancia a la que se encuentra de su presa o alimento.

Los bats se encargarán de recorrer el espacio de búsqueda intentando alcanzar la solución óptima en la que se realicen la menor cantidad de movimientos de las partes entre las distintas celdas, dado que se desea disponer las máquinas de tal manera de reducir los costos de transporte de un lugar a otro.

Dentro del grupo de problemas utilizados para realizar los análisis del comportamiento del algoritmo frente al problema, se tiene un grupo de instancias cuyos resultados son obtenidos por medio de métodos exactos, en los cuales se alcanzó un resultado óptimo en gran parte de los casos. En el segundo grupo, si bien, el método de resolución del problema no era el mismo, se alcanzaron buenos resultados en gran parte de estos.

En la sección 2 se definen los objetivos que se desean conseguir con este trabajo. En la sección 3 se destacan los trabajos que se han realizados para resolver el MCDP junto con sus autores y años de publicación. En la sección 4 se describe y analiza en profundidad el problema. En la sección 5 se presenta el algoritmo bat junto con todas sus características y funciones principales que ayudarán a encontrar la mejor solución posible al problema. Finalmente en la sección 6 se presentan los resultados obtenidos con un análisis detallado de las pruebas llevadas a cabo.

2 Objetivos

A continuación se define el objetivo general de este trabajo junto con los objetivos específicos que se quieren alcanzar a lo largo de su desarrollo.

2.1 Objetivo General

- Resolución del Manufacturing Cell Design Problem (MCDP) mediante Bat Algorithm (BA).

2.2 Objetivos Específicos

- Comprender el Manufacturing Cell Design Problem, a fin de plantear el problema para su resolución.
- Comprender el Algoritmo Bat.
- Implementar el Algoritmo Bat para la resolución del Manufacturing Cell Design Problem.
- Realizar los experimentos correspondientes, de forma de poder observar el comportamiento del trabajo desarrollado.

3 Estado del Arte

Los trabajos de investigación que se han realizado para resolver el problema de formación de celdas han seguido dos líneas complementarias, que se pueden organizar en dos grupos diferentes: los métodos aproximados y los métodos exactos. Los métodos aproximados son en su mayoría centrados en la búsqueda de una solución óptima en un tiempo limitado, sin embargo, no garantizan encontrar un óptimo global. Por el contrario los métodos exactos, tienen como objetivo analizar por completo el espacio de búsqueda para garantizar un óptimo global, aunque el costo es mucho mayor tanto en memoria como en tiempo si se compara con los métodos aproximados.

Dentro del grupo de métodos aproximados, se han utilizado diferentes Metaheurísticas para la formación de celdas. Aljaber, Baek y Chen [1] utilizaron Tabú Search. Wu, Chang y Chung [2] presentaron un enfoque Simulated Annealing (SA), inspirado en el proceso de recocido del acero y cerámicas, una técnica que consiste en calentar y luego enfriar lentamente el material para variar sus propiedades físicas. El calor causa que los átomos aumenten su energía y que puedan así desplazarse de sus posiciones iniciales (un mínimo local de energía), el enfriamiento lento les da mayores probabilidades de recristalizar en configuraciones con menor energía que la inicial (mínimo global). Durán, Rodríguez y Consalter [3] combinaron el Particle Swarm Optimization (PSO), que consiste en partículas que se mueven a través de un espacio de soluciones y que con el tiempo son aceleradas, con una técnica de minería de datos. Venugopal y Narendran [4] propusieron el uso de Genetic Algorithms (GA), que se basan en el proceso genético de los organismos vivos. Gupta, Kumar y Sundaram [5] utilizaron también GA pero con un enfoque de optimización multi-objetivo diferente, que consiste en la minimización simultanea del número total de movimientos entre celdas y variación de carga entre ellas. Técnicas híbridas también es posible encontrar en la resolución del problema. Es el caso de Wu et al. [6] que combinaron SA con GA. James, Brown and Keeling [7] presentaron una solución híbrida que combina la búsqueda local y GA. Nsakanda, Diabe y Prince [8] propusieron una metodología de solución basada en una combinación de GA y técnicas de optimización a gran escala. Soto et al. [9], utilizo Constraint Programming (CP) y Boolean Satisfiability Technology (SAT) para la resolución del problema, quien desarrollo el problema aplicando 5 solvers distintos. Dos solvers CP, dos SAT y un híbrido entre SAT y CP.

En el grupo de métodos exactos, experimentos preliminares para la formación de celdas se desarrollaron mediante el uso de programación lineal, como son los trabajos de Purcheck [10] y Olivia-López y Purcheck [11]. Kusiak y Chow [12] y Boctor [13] propusieron el uso de modelos cuadráticos lineales. Se destaca también el uso de otro paradigma para el uso de métodos exactos, como es Goal Programming (GP), el cual puede ser visto como una generalización de la programación lineal para la manipulación de múltiples funciones objetivos. Sankaran [14] y Shafer y Rogers [15] trabajaron en GP.

También se han hecho trabajos combinando ambos métodos. Es el caso de Boulif y Atif (2006) [16], quienes combinan técnicas de brach-and-bound con GA.

4 Manufacturing Cell Design Problem

A continuación se realizará una descripción del problema a resolver por medio del trabajo a desarrollar.

4.1 Descripción del Problema

El Manufacturing Cell Design Problem consiste en la organización de una planta o centro de producción en un conjunto de celdas, donde cada una de ellas estará compuesta por diferentes máquinas que se encargarán de procesar diferentes partes o piezas de un producto que posean características similares. El criterio utilizado para agrupar las máquinas en las celdas se basa en la minimización de movimientos entre ellas. De esta manera, el objetivo será agrupar las máquinas de la mejor manera posible, para así, reducir el costo de transporte de una parte o pieza desde un lugar a otro.

4.2 Planteamiento del Problema

Se modelará el problema de formación de celdas mediante el uso de una matriz inicial. La idea es representar el procesamiento de cada una de las partes en la matriz para mantener información de las máquinas por las que tiene que viajar cada una de ellas hasta terminar su proceso de producción. Esta matriz recibe también el nombre de matriz de incidencia y está compuesta por un dominio binario. Se denota A = $[a_{ij}]$, donde

$a_{ij} = 1$ Si la parte j_{th} visita la máquina i_{th}
$a_{ij} = 0$ Caso contrario

Una vez construida la matriz es difícil distinguir la distribución ideal de los grupos de máquinas o el conjunto de partes. Por lo tanto, el objetivo principal es la formación de un conjunto de máquinas en celdas de manera que el número de viajes entre celdas se reduzca al mínimo. La matriz de incidencia (fig. 1) tiene que ser transformada en una matriz ordenada (fig. 2), que tiene como característica una estructura diagonal. Este reordenamiento tiene por objeto la reducción al mínimo del total de movimientos entre celdas.

0	0	1	0	1	0	1	0	0	0	0	0
0	0	1	0	0	0	1	1	0	1	0	0
0	0	1	0	1	0	0	1	0	1	1	0
0	0	1	0	0	0	0	0	0	0	1	0
0	0	1	0	1	0	1	1	0	0	0	0
0	0	1	0	0	0	0	0	0	1	0	0
0	1	0	1	0	1	0	0	1	0	0	1
0	1	0	0	0	1	0	0	1	0	0	0
1	0	0	1	0	0	0	0	1	0	0	1
1	0	0	1	0	1	0	0	1	0	0	1
0	1	0	1	0	0	0	0	1	0	0	0
0	1	0	1	0	1	0	0	1	0	0	1

Figura 1: Matriz de Incidencia

1	0	1	1	0	0	0	0	0	0	0	0
1	1	1	0	1	0	0	0	0	0	0	0
1	1	0	1	1	1	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	1	1	0	1	1	1
0	0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	0	1	1	0	1	1
0	0	0	0	0	0	0	1	1	1	1	1
0	0	0	0	0	0	1	0	0	0	1	1
0	0	0	0	0	0	1	1	0	1	1	1

Figura 2: Matriz Ordenada

El modelo de optimización es el siguiente [13]:

M: Número de máquinas
P: Número de partes
C: Número de celdas
i: Índice de máquinas (i = 1,…, M)
j: Índice de partes (j = 1,…, P)
k: Índice de celdas (k = 1,…, C)
A = [a_{ij}] corresponde a la matriz de incidencia máquina parte (M x P)
M_{max}: Número máximo de máquinas por celda

Para obtener la matriz ordenada, es necesario utilizar otras tres matrices. La primera es la matriz inicial o de incidencia que corresponde a la instancia. La segunda es una matriz que se va a denominar intermedia y contiene valores que se obtienen mediante el algoritmo a utilizar. Y la tercera matriz se va a denominar inferida y contiene valores que se obtienen a partir de las dos matrices anteriormente nombradas.

La matriz intermedia (M x C) permitirá distribuir las máquinas en distintas celdas. Esta matriz se denota Y = [y_{ik}], donde

$y_{ik} = 1$ Si la máquina i pertenece a la celda k
$y_{ik} = 0$ Caso contrario

Sin embargo, para efectos del algoritmo que se utilizará para resolver el problema, la matriz intermedia será representada por un vector, que almacenará la celda a la que pertenece la máquina. Este vector se denota V = [v_i] donde

$v_i \in [1, C]$

1	4	5	4	4	3	2	1	1	3	5	2

Figura 3: Ejemplo del vector solución para 12 máquinas y 5 celdas

Por otro lado, la matriz inferida (P x C) se denota Z = [z_{jk}], donde

$z_{jk} = 1$ Si la parte j pertenece a la celda k
$z_{jk} = 0$ Caso contrario

El problema va a ser representado por el siguiente modelo matemático:

$$Minimizar \sum_{k=1}^{C} \sum_{i=1}^{M} \sum_{j=1}^{P} a_{ij} z_{jk} (1 - y_{ik}) \qquad (5.2.1)$$

Sujeto a las siguientes restricciones:

- Una máquina solo puede pertenecer a una celda.

$$\sum_{k=1}^{C} y_{ik} = 1 \quad \forall i \tag{5.2.2}$$

En el desarrollo del presente trabajo, dado que la representación de la solución es mediante un vector, esta restricción queda sin efecto, solo es necesario controlar que el valor de $v_i \in [1, C]$.

- Una parte solo puede pertenecer a una celda.

$$\sum_{k=1}^{C} z_{jk} = 1 \quad \forall j \tag{5.2.3}$$

Independiente de que una parte necesite de varias máquinas dispuestas en diferentes celdas, es necesario determinar a qué celda se va a asociar esa parte.

Para determinar la celda apropiada, lo que se hará es consultar las máquinas que necesita esa parte y en base a la celda que contenga el mayor número de esas máquinas es la que se seleccionará. En el caso de tener la misma cantidad, la selección de la celda es indiferente, cualquiera servirá sin afectar el resultado final.

- El número de máquinas por celda no puede superar el máximo establecido de máquinas por celda.

$$\sum_{i=1}^{M} y_{ik} \leq M_{max} \quad \forall k \tag{5.2.4}$$

En el caso de este trabajo, se debe cumplir que:

$$f(v_c) \leq M_{max} \quad \forall c \quad con \quad c = 1, .., M \tag{5.2.5}$$

(f: frecuencia)

5 Algoritmo Bat

El algoritmo bat se basa en las características de ecolocalización de micro-murciélagos. Una de las características de este algoritmo es la utilización de una técnica de frecuencia de ajuste que le permite aumentar la diversidad de las soluciones. Además utiliza zoom automático para equilibrar la exploración y explotación durante el proceso de búsqueda, técnica en la cual imita las variaciones de las tasas de pulso y volumen de los murciélagos cuando buscan a sus presas.

5.1 Ecolocalización

Existen alrededor de 1000 especies diferentes de murciélagos [17]. Sus tamaños pueden variar ampliamente, desde pequeños murciélagos que pesan entre 1,5 y 2 gramos hasta murciélagos gigantes que llegan a pesar 1 kilogramo. La mayoría de ellos utiliza la ecolocalización en cierto grado, sin embargo son los micro-murciélagos los que la utilizan de manera más amplia, a diferencia de los mega-murciélagos que simplemente no la utilizan.

Se entiende como ecolocalización a la capacidad de algunos animales de conocer su entorno por medio de la emisión de sonidos y la interpretación del eco que los objetos producen a su alrededor. Esto les permite detectar a sus presas, evitar obstáculos y localizar grietas en la oscuridad para poder dormir. Los pulsos que generan provocan un eco al rebotar en los objetos circundantes [18]. Varían en propiedades y se relacionan directamente a la estrategia de caza dependiendo de la espacie. Los micro-murciélagos pueden emitir entre 10 y 20 estallidos de sonido por segundo, mientras que su tasa de pulso se puede acelerar a 200 pulsos por segundo al acercarse a su presa [19].

5.2 Características Principales

En base a la descripción de ecolocalización y las características de los murciélagos, se desarrolla el algoritmo bat en base a las siguientes tres reglas [20]:

- Todos los murciélagos utilizan la ecolocalización para detectar la distancia, y también "saben" la diferencia entre alimentos o presas y barreras de fondo, de alguna manera mágica.
- Los murciélagos vuelan al azar con velocidad v_i, una posición x_i, y con una frecuencia f_{min}, variando su longitud de onda y volumen A_0 para buscar a su presa. Pueden ajustar automáticamente la longitud de onda (o frecuencia) de los pulsos emitidos y ajustar la tasa de emisión de pulso $r \in [0, 1]$ en la medida que se acerca a su objetivo.
- Aunque el volumen puede variar de muchas maneras, se supone que el volumen varía de entero (positivo) A_0 a un valor constante mínimo A_{min}.

5.3 Movimiento del Murciélago

Cada murciélago es asociado con una velocidad v_i^t, una ubicación x_i^t y una iteración *t* en un espacio de búsqueda. Entre todos los murciélagos existe una mejor solución actual x_*. Por lo tanto, las tres reglas anteriores mencionadas pueden ser traducidas a las siguientes ecuaciones [19]:

$$f_i = f_{min} + (f_{max} - f_{min})\beta \tag{6.3.1}$$

$$v_i^{t+1} = v_i^t + (x_i^t - x_*)f_i \tag{6.3.2}$$

$$x_i^{t+1} = x_i^t + v_i^t \tag{6.3.3}$$

Donde $\beta \in [0, 1]$ es un vector aleatorio extraído de una distribución uniforme.

Se utilizará para la implementación una f_{min} y f_{max}, que inicialmente tendrá un valor aleatorio que se extrae de manera uniforme a partir de $[f_{min}, f_{max}]$. Por esta razón el algoritmo bat puede ser considerado como un algoritmo de frecuencia de ajuste ya que proporciona una combinación equilibrada entre la exploración y explotación. El volumen y la tasa de pulso en tanto proporcionan un mecanismo de zoom automático en la región con soluciones prometedoras.

5.4 Variaciones de Volumen y Tasa de Pulso

Con el fin de proporcionar un mecanismo eficaz para controlar la exploración y explotación, se tiene que variar el volumen A_i y la tasa de emisión de pulso r_i durante las iteraciones. Dado que el volumen generalmente disminuye una vez que el murciélago ha encontrado a su presa, mientras que la tasa de emisión de pulso aumenta, el valor del volumen puede ser escogido como cualquier valor de conveniencia entre A_{min} y A_{max}. De esta manera, si $A_{min} = 0$ significa que un murciélago acaba de encontrar una presa y temporalmente detendrá la emisión de cualquier sonido [19].

Con estas premisas, se tiene que:

$$A_i^{t+1} = \alpha A_i^t, \quad r_i^{t+1} = r_i^0 \left[1 - \exp(-\gamma t)\right] \tag{6.4.4}$$

Donde α y γ son constantes. Para cualquier $0 < \alpha < 1$ y $\gamma > 0$, se tiene que:

$$A_i^t \to 0, \quad r_i^t \to r_i^0, como\ t \to \infty \tag{6.4.5}$$

En el caso más simple, se puede utilizar $\alpha = \gamma$.

5.5 Código

A continuación se presenta la función principal del código desarrollado:

```
1   public void run() {
2      boolean estado = false;
3
4      while (t < T) {
5         for (fila = 0; fila < n; fila++) {
6            # Se actualiza volumen y tasa de pulso
7            # Se actualiza mejor fitness global
8         }
9
10        # Se actualiza menor fitness y vector solución
11
12        for (fila = 0; fila < n; fila++) {
13           if (rand.nextFloat() > vectorPulso[fila]) {
14              for (columna = 0; columna < M; columna++) {
15                 do{
16                    epsilon = rand.nextFloat()*2 - 1;
17                    matrizSolucion[fila][columna] = aproximar(matrizSolucion[fila][columna] +
18                                                      (epsilon * obtenerPromedioVolumen()));
19                    estado = verificarMaxMaq(fila, columna);
20                 }while(estado == true);
21              }
22           }
23
24           if (rand.nextFloat() < vectorVolumen[fila] && vectorFitness[fila] > mejorFitness) {
25              do{
26                 beta = rand.nextFloat();
27                 vectorFrecuencia[fila] = fMin + (fMax - fMin) * beta;
28                 for (columna = 0; columna < M; columna++) {
29                       matrizVelocidad[fila][columna] = matrizVelocidad[fila][columna] +
30                                                      (vectorSolucionActual[columna] –
31                                                      matrizSolucion[fila][columna]) *
32                                                      vectorFrecuencia[fila];
33                       matrizSolucion[fila][columna] = aproximar(matrizSolucion[fila][columna]+
34                                                      matrizVelocidad[fila][columna]);
35                       estado = verificarMaxMaq(fila, columna);
36                       if (estado == true) break;
37                 }
38              }while(estado == true);
39           }
40
41           crearMatrizInferida(fila);
42           eFitness = obtenerFitness(fila);
43
44           # Se actualiza menor fitness y vector solución
45        }
46
47        vectorFitnessIteraciones[t] = mejorFitness;
48        t++;
49     }
50  }
```

Figura 4: Bat Algorithm implementado para la resolución del MCDP

Entre las líneas 1 – 50 se ejecuta la función principal del algoritmo, es decir, los movimientos que van efectuar los bats sobre las distintas iteraciones.

Entre las línea 6, se asigna un volumen y una tasa de pulso si el valor random obtenido es mejor al que ya posee el bat.

En la línea 7 se actualiza el mejor fitness global si el bat obtuvo uno mejor.

Entre las líneas 10 y 44 se actualiza el menor fitness hasta el momento y se guarda el vector solución actual.

Entre las líneas 13 – 21 se crea una nueva solución si su tasa de pulso es menor al valor random obtenido. Además se verifica que se cumpla la restricción de mMax.

Entre las líneas 21 – 39 se crea una nueva solución si su volumen es mayor al valor random obtenido y su fitness mayor al mejor fitness global. En el proceso de asignación se toman en cuenta también otros factores como frecuencias y velocidades del bat. También se verifica que se cumpla la restricción del mMax.

En la línea 41 se crea la matriz inferida del bat, en la 42 se obtiene su fitness y en la 47 se guarda en un vector de iteraciones los mejores fitness de cada iteración.

Cabe destacar que con el fin de mejorar la variabilidad de las posibles soluciones, se ha propuesto emplear caminos aleatorios [22]. Una vez que se selecciona una solución de entre las mejores soluciones actuales, se aplica el llamado camino aleatorio, que consiste en cambiar la posición mediante la siguiente formula:

$$x_i^j = x_i^j + \epsilon \bar{A} \qquad (6.5.1)$$

Donde $\bar{A}(t)$ es el promedio de volumen de todos los bat en el tiempo t y $\varepsilon \in [-1, 1]$.

Además, dentro del algoritmo, se utilizan constantes ad-hoc como son α y γ que adquieren un valor pequeño que para efectos del problema serán igual a ε [21].

A continuación se definen otras funciones de importancia utilizadas en la ejecución del algoritmo:

- **initial ( ):** Función que inicializa todas las variables a utilizar. Además realiza la primera iteración del algoritmo asignando volumen y tasa de pulso a cada bat.

- **verificarMaxMaq ( ):** Función que comprueba que el número de máquinas de una celda este dentro del máximo permitido. Cada vez que se asigna una celda a una máquina, se verifica que no vulnere la restricción del máximo de máquinas, de lo contrario, se repara solicitando una nueva celda.

- **obtenerBinario ( ):** Función que traduce el valor entero de la celda a un valor binario necesario en la fórmula para calcular el valor de fitness.

- **aproximar ( ):** Función para concretizar un valor decimal a un número entero, que corresponderá al número de celda. Los valores decimales se obtienen al aplicar variables de frecuencia y velocidad.

- **obtenerPromedioVolumen ( ):** Función que obtiene el volumen promedio de los bats. Valor necesario para obtener caminos aleatorios según el autor [21].

- **crearMatrizInferida ( ):** Función que a partir de la matriz inicial y el vector solución de máquinas crea una nueva matriz binaria que identifica a que celda fue asignada cada parte.

- **obtenerFitness ( ):** Función que retorna el valor de fitness, el cual se calcula utilizando la matriz inicial, el vector solución o matriz intermedia y la matriz inferida, aplicado a un modelo matemático a minimizar.

- **verificarMatrizInferida ( ):** Función que verifica cuantas celdas han sido asignadas a una parte. En el caso de ser mayor o igual a 2, se escoge una sola celda dependiendo de la que contenga el mayor número de máquinas.

- **buscarMaquinas ( ):** Función utilizada para obtener la cantidad de máquinas que utiliza una parte en una determinada celda.

6 Resultados

En esta sección se discuten los experimentos llevados a cabo en la ejecución del algoritmo. Los parámetros utilizados se establecieron en base a otros problemas ya resueltos mediante el uso de bats que han tenido buenos resultados.

Para la ejecución del algoritmo los parámetros que se consideraron fueron los siguientes:

- Frecuencia mínima (fMin): 0.25
- Frecuencia máxima (fMax): 0.50
- Constante alpha (alpha): 0.99
- Constante gamma (gamma): 0.99
- Constante epsilon (epsilon): 0.99
- Nº de máquinas (M): 16
- Nº de partes (P): 30
- Nº de iteraciones (T): 300
- Nº de bats (n): 50

Las pruebas se realizaron en base a 10 problemas de prueba, los cuales podrían comprender el uso de 2 o 3 celdas. En el caso de utilizar 2 celdas, el máximo de máquinas (mMax) en cada una de ellas puede tomar valores entre 8 y 12. En el caso de utilizar 3 celdas, el mMax puede variar entre 6 y 9 máquinas por celda. En ambos casos el valor de mMax se mantiene constante durante toda la ejecución del algoritmo.

Los 10 problemas utilizados se encuentran en el anexo A.

Según el número de celdas y el número máximo máquinas presentes en cada una de ellas, las soluciones óptimas para los 10 problemas son las siguientes [13]:

Tabla 1: Soluciones óptimas para los 10 problemas de prueba

	N = 2					N = 3			
Problema	m = 8	9	10	11	12	m = 6	7	8	9
1	11	11	11	11	11	27	18	11	11
2	7	6	4	3	3	7	6	6	6
3	4	4	4	3	1	9	4	4	4
4	14	13	13	13	13	27	18	14	13
5	9	6	6	5	4	11	8	8	6
6	5	3	3	3	2	6	4	4	3
7	7	4	4	4	4	10	5	5	4
8	13	10	8	5	5	14	11	11	10
9	8	8	8	5	5	12	12	8	8
10	8	5	5	5	5	10	8	8	5

En la ejecución del algoritmo se utilizó un computador con las siguientes características:

- Procesador Intel Core i7-3632QM 2.20 GHz
- Memoria 8 GB
- Sistema Operativo Windows 8 64 bits

Los resultados obtenidos en la resolución del problema del MCDP, mediante el algoritmo bat, bajo todas las condiciones antes mencionadas son los siguientes:

R: Resultado obtenido
O: Valor Óptimo
V: Variación (%)

Tabla 2: Soluciones obtenidas para los problemas con 2 celdas

N = 2															
Problema	m = 8			9			10			11			12		
	R	O	V	R	O	V	R	O	V	R	O	V	R	O	V
1	11	11	0%	11	11	0%	11	11	0%	11	11	0%	11	11	0%
2	7	7	0%	6	6	0%	5	4	25%	4	3	33%	4	3	33%
3	4	4	0%	4	4	0%	4	4	0%	3	3	0%	1	1	0%
4	14	14	0%	13	13	0%	13	13	0%	13	13	0%	13	13	0%
5	9	9	0%	6	6	0%	6	6	0%	5	5	0%	4	4	0%
6	5	5	0%	3	3	0%	3	3	0%	3	3	0%	2	2	0%
7	7	7	0%	4	4	0%	4	4	0%	4	4	0%	4	4	0%
8	13	13	0%	10	10	0%	8	8	0%	5	5	0%	5	5	0%
9	8	8	0%	8	8	0%	8	8	0%	5	5	0%	5	5	0%
10	8	8	0%	5	5	0%	5	5	0%	5	5	0%	5	5	0%

Para los problemas compuestos por 2 celdas, se llegó al óptimo en un 94% de los casos, es decir 47 de las 50 instancias analizadas. El problema 2 fue el único que no obtuvo un rendimiento del 100% en los 5 casos, sin embargo, la variación resulto ser mínima, solo por un movimiento no se alcanzó el óptimo buscado para los mMax de 10, 11 y 12 del problema.

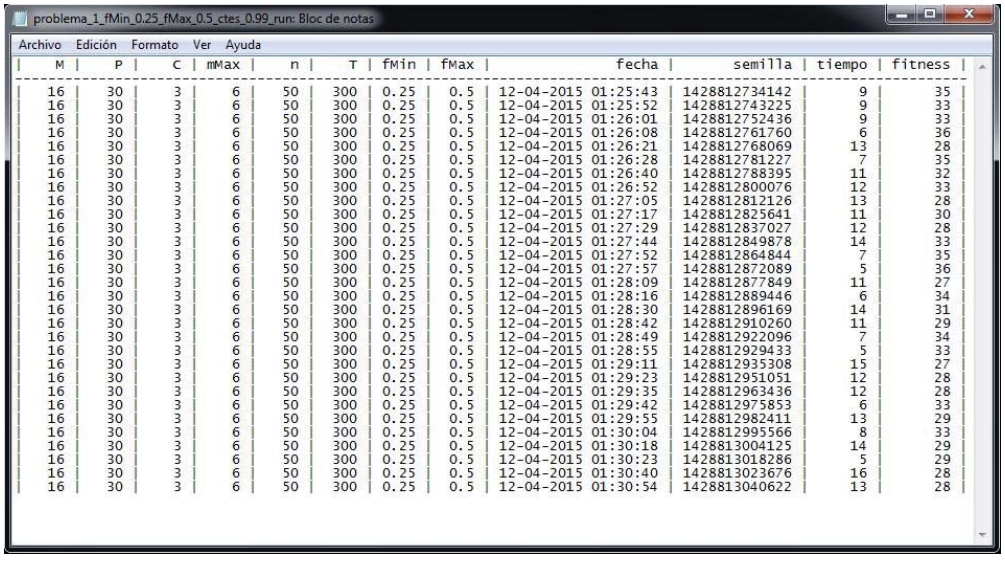
Tabla 3: Soluciones obtenidas para los problemas con 3 celdas

N = 3												
Problema	m = 6			7			8			9		
	R	O	V	R	O	V	R	O	V	R	O	V
1	27	27	0%	18	18	0%	11	11	0%	11	11	0%
2	8	7	14%	7	6	16%	7	6	16%	6	6	0%
3	9	9	0%	4	4	0%	4	4	0%	4	4	0%
4	27	27	0%	18	18	0%	14	14	0%	13	13	0%
5	11	11	0%	8	8	0%	9	8	12%	6	6	0%
6	6	6	0%	4	4	0%	4	4	0%	3	3	0%
7	11	10	10%	5	5	0%	5	5	0%	4	4	0%
8	14	14	0%	11	11	0%	11	11	0%	10	10	0%
9	12	12	0%	12	12	0%	8	8	0%	8	8	0%
10	10	10	0%	8	8	0%	8	8	0%	5	5	0%

En el caso de los problemas con 3 celdas, la tendencia fue similar, obteniendo buenos resultados. Se logró alcanzar el óptimo en un 87.5% de los casos, es decir, 35 de las 40 instancias analizadas. Al igual que en los problemas con 2 celdas, fue el problema 2 el que obtuvo el peor rendimiento, obteniendo una mínima variación del fitness para los mMax de 6, 7 y 8, siendo superado por solo un movimiento del valor optimo a alcanzar. Lo mismo ocurrió para los problemas 5 y 7, con un mMax de 8 y 6 respectivamente.

En general, de las 90 instancias puestas a prueba, se llegó al óptimo en 82 ocasiones lo que equivale a más del 91% de los casos.

Cada problema se ejecutó 30 veces de manera de tener una muestra más representativa del comportamiento del algoritmo en las distintas ejecuciones. Se generó también un archivo de texto con los resultados obtenidos.


problema_1_fMin_0.25_fMax_0.5_ctes_0.99_run: Bloc de notas

Archivo Edición Formato Ver Ayuda

M	P	C	mMax	n	T	fMin	fMax	fecha	semilla	tiempo	fitness
16	30	3	6	50	300	0.25	0.5	12-04-2015 01:25:43	1428812734142	9	35
16	30	3	6	50	300	0.25	0.5	12-04-2015 01:25:52	1428812743225	9	33
16	30	3	6	50	300	0.25	0.5	12-04-2015 01:26:01	1428812752436	9	33
16	30	3	6	50	300	0.25	0.5	12-04-2015 01:26:08	1428812761760	6	36
16	30	3	6	50	300	0.25	0.5	12-04-2015 01:26:21	1428812768069	13	28
16	30	3	6	50	300	0.25	0.5	12-04-2015 01:26:28	1428812781227	7	35
16	30	3	6	50	300	0.25	0.5	12-04-2015 01:26:40	1428812788395	11	32
16	30	3	6	50	300	0.25	0.5	12-04-2015 01:26:52	1428812800076	12	33
16	30	3	6	50	300	0.25	0.5	12-04-2015 01:27:05	1428812812126	13	28
16	30	3	6	50	300	0.25	0.5	12-04-2015 01:27:17	1428812825641	11	30
16	30	3	6	50	300	0.25	0.5	12-04-2015 01:27:29	1428812837027	12	28
16	30	3	6	50	300	0.25	0.5	12-04-2015 01:27:44	1428812849878	14	33
16	30	3	6	50	300	0.25	0.5	12-04-2015 01:27:52	1428812864844	7	35
16	30	3	6	50	300	0.25	0.5	12-04-2015 01:27:57	1428812872089	5	36
16	30	3	6	50	300	0.25	0.5	12-04-2015 01:28:09	1428812877849	11	27
16	30	3	6	50	300	0.25	0.5	12-04-2015 01:28:16	1428812889446	6	34
16	30	3	6	50	300	0.25	0.5	12-04-2015 01:28:30	1428812896169	14	31
16	30	3	6	50	300	0.25	0.5	12-04-2015 01:28:42	1428812910260	11	29
16	30	3	6	50	300	0.25	0.5	12-04-2015 01:28:49	1428812922096	7	34
16	30	3	6	50	300	0.25	0.5	12-04-2015 01:28:55	1428812929433	5	33
16	30	3	6	50	300	0.25	0.5	12-04-2015 01:29:11	1428812935308	15	27
16	30	3	6	50	300	0.25	0.5	12-04-2015 01:29:23	1428812951051	12	28
16	30	3	6	50	300	0.25	0.5	12-04-2015 01:29:35	1428812963436	12	28
16	30	3	6	50	300	0.25	0.5	12-04-2015 01:29:42	1428812975853	6	33
16	30	3	6	50	300	0.25	0.5	12-04-2015 01:29:55	1428812982411	13	29
16	30	3	6	50	300	0.25	0.5	12-04-2015 01:30:04	1428812995566	8	33
16	30	3	6	50	300	0.25	0.5	12-04-2015 01:30:18	1428813004125	14	29
16	30	3	6	50	300	0.25	0.5	12-04-2015 01:30:23	1428813018286	5	29
16	30	3	6	50	300	0.25	0.5	12-04-2015 01:30:40	1428813023676	16	28
16	30	3	6	50	300	0.25	0.5	12-04-2015 01:30:54	1428813040622	13	28

Figura 5: Archivo de salida con los datos de la ejecución del algoritmo

6.1 Análisis Problema 4

A modo de ejemplo se analizará a continuación el comportamiento del problema 4, bajo 3 celdas y un mMax de 7, el cual logro alcanzar el valor óptimo de 18, lo que se traduce en 18 movimientos interceldarios de las partes entre las celdas.

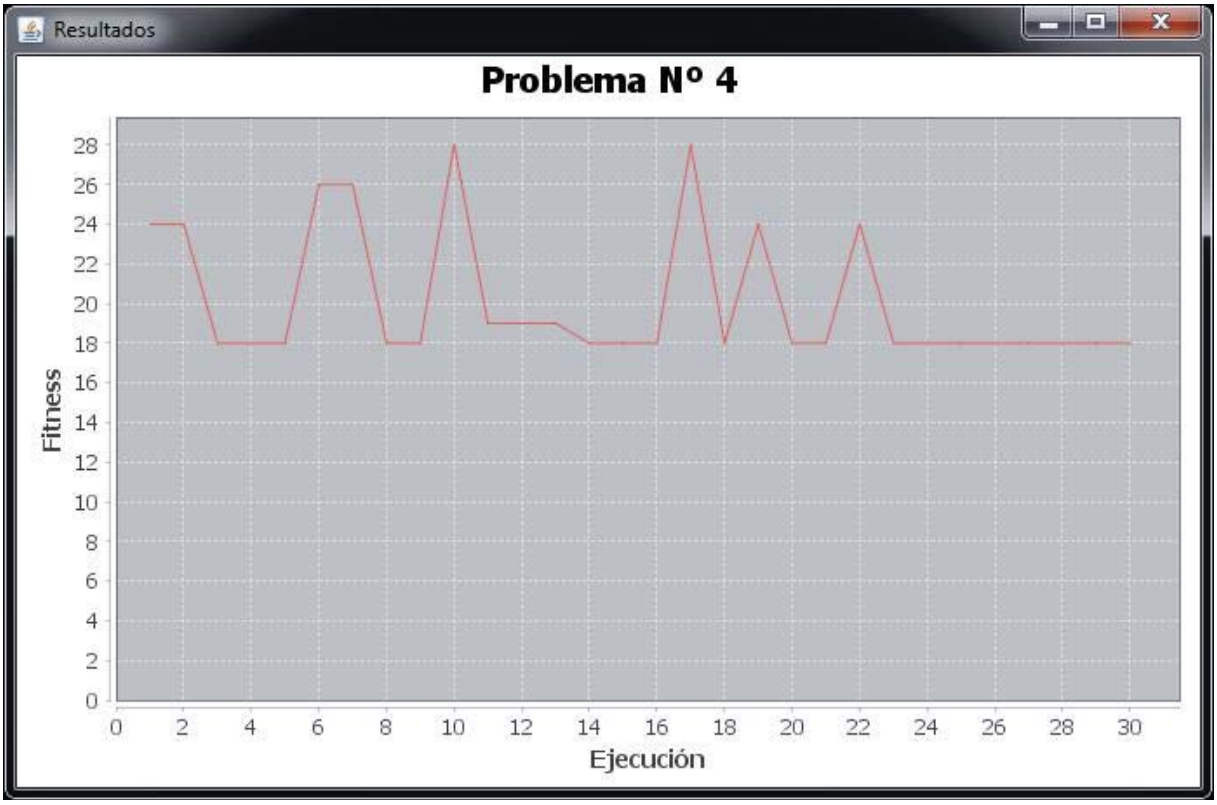


Figura 6: Gráfico de fitness v/s ejecución del problema 4

El promedio de fitness entre las 30 ejecuciones que se realizaron en el problema 4 fue de 20, alcanzando el valor óptimo en 19 ejecuciones (3, 4, 5, 8, 9, 14, 15, 16, 18, 20, 21, 23, 24, 25, 26, 27, 28, 29, 30), mientras que el peor resultado se obtuvo en la ejecución número 10 y 17 con un fitness de 28 en ambos casos. De esta manera, en un 63.3% de las ejecuciones se obtuvo éxito.

Analizando el resultado obtenido en la ejecución número 3 que logro alcanzar el óptimo, se puede definir su comportamiento mediante el siguiente gráfico:

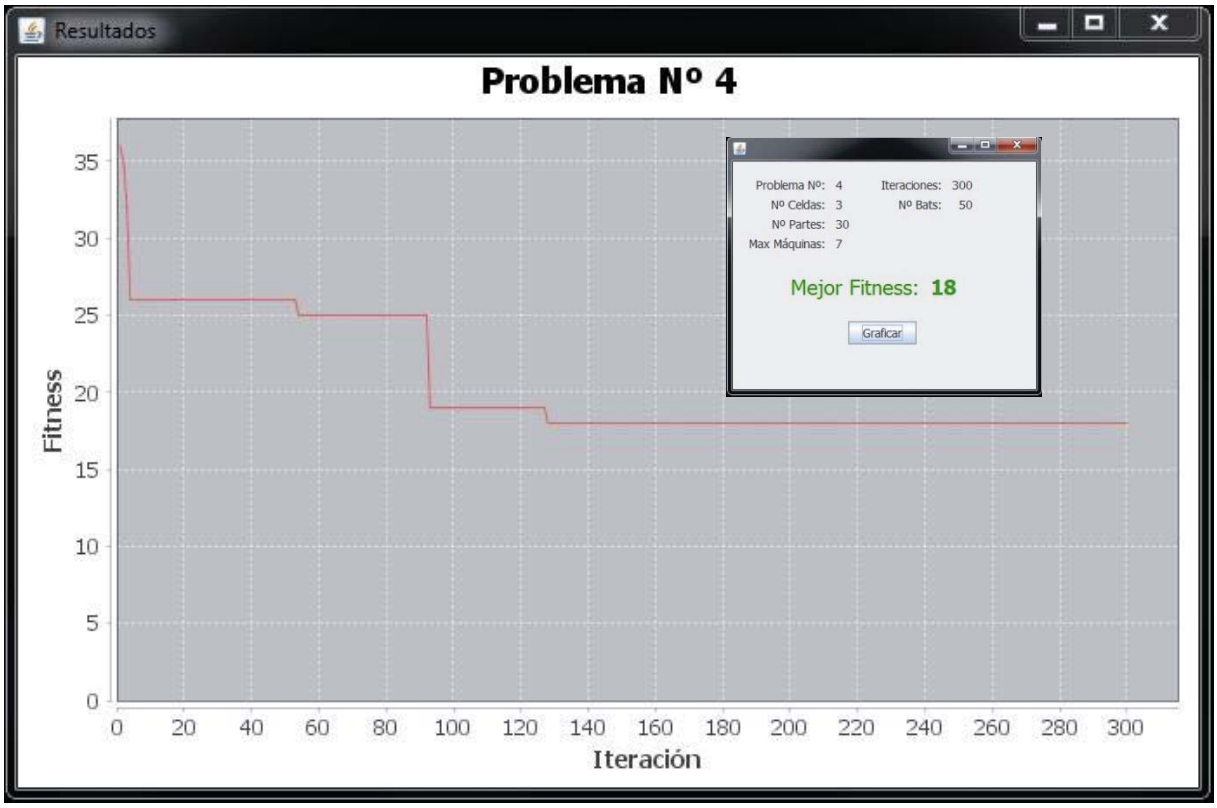


Figura 7: Gráfico de fitness v/s iteración de la ejecución 3 del problema 4

Se distingue claramente que en las primeras interacciones, con los valores random de volumen y tasa de pulso, se logró dar un salto importante desde un fitness de 36 a uno de 26, que se mantuvo constante por más de 50 iteraciones, hasta dar tres nuevos saltos, alcanzando en el primero de ellos un valor fitness de 25, en el segundo un valor de 19 y en el tercero un valor de 18. En el resto de interacciones no hubo cambios y el valor se mantuvo constante.

Cabe destacar que en la mayoría de los problemas, el último cambio de fitness ocurre antes de las 150 iteraciones, más allá, el resultado se mantiene igual. Esto es de gran importancia debido a que garantiza encontrar el mejor fitness dentro de las primeras 150 iteraciones, lo cual se traduce en un ahorro en tiempos de ejecución ya que el resto de iteraciones se pierden.

A continuación se analizará el comportamiento de los bats en la iteración 126 que obtuvo el mejor resultado.

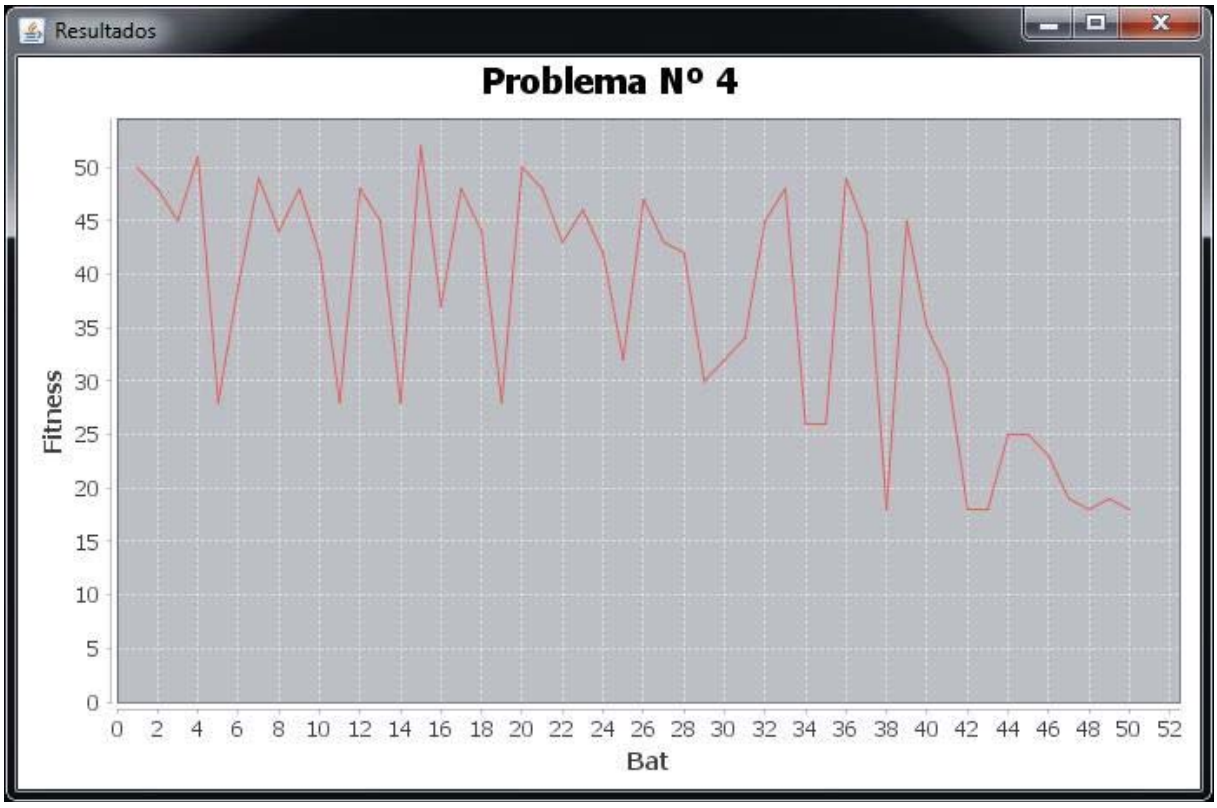


Figura 8: Gráfico de fitness v/s bats de la iteración 126 del problema 4

Se puede apreciar que la mayoría de los bats tuvo una tendencia a arrojar resultados por un valor de fitness superior a 25, solo 9 bats de los 50 obtuvieron un fitness menor o igual. El peor resultado lo obtuvo el bat 15 con un fitness de 52, mientras que la solución óptima se alcanzó en cinco oportunidades, en los bats 38, 42, 43, 48 y 50.

El vector solución de los cinco bats que alcanzaron el óptimo resulto ser el mismo y los valores de celdas asignadas a cada máquina fueron los siguientes.

3	1	3	1	3	3	3	2	3	1	1	3	2	1	1	1

Figura 9: Vector solución del problema 4

Analizando en profundidad el vector, es posible determinar la distribución de las diferentes máquinas en las celdas, en la denominada matriz ordenada, reflejando que máquinas utiliza cada parte.

	2	5	7	8	9	12	13	14	15	18	20	21	22	25	26	1	3	4	6	10	11	16	17	19	23	24	27	28	29	30
2	1				1		1	1		1		1	1		1												1			
4	1				1					1																				
10			1		1							1	1		1														1	
11	1	1		1			1			1				1		1								1						
14						1			1	1		1		1																1
15		1		1		1				1	1		1	1	1							1								
16		1						1		1					1				1											
8							1			1																				
13				1				1							1															
1			1																1			1		1		1	1	1	1	
3														1		1		1	1		1	1	1					1	1	
5																1		1	1		1	1		1	1	1			1	
6						1										1		1		1	1			1	1			1	1	
7								1									1		1	1		1	1				1		1	1
9	1															1			1	1					1			1		1
12						1											1	1				1	1		1	1		1		1
	1	0	1	1	0	2	1	2	0	1	0	0	0	1	1	1	0	0	1	0	0	1	0	1	0	0	1	0	1	1

Figura 10: Matriz ordenada del problema 4 con 3 celdas y un mMax de 7

En la figura se pueden apreciar las máquinas que necesita cada una de las 30 partes. Los rectángulos celeste y verde, representan una determinada celda a la cual se le han asignado determinadas partes. Esto no excluye que esa parte no pueda ocupar máquinas ubicadas en otras celdas. Los números de color morado indican las partes que necesitaron de máquinas ubicadas fuera de su celda, cuya suma corresponde al fitness encontrado del problema, 18, es decir, el total de movimientos interceldarios que realizaron las partes que así lo requerían.

En la figura, además, es posible ver las máquinas dispuestas en las celdas 1, 2 y 3, con 7, 2 y 7 máquinas respectivamente. Cumpliendo la condición de un mMax de 7, es decir, un límite de 7 máquinas por celda.

6.2 Aplicación del Bat a otras instancias

Para analizar la efectividad del algoritmo implementado, en un rango mayor de problemas, se investigaron nuevas instancias de diferentes autores. Los tamaños de las matrices variaban desde 5 a 40 máquinas y desde 7 a 100 partes. El principal problema que se presento fue que la solución era calculada de forma diferente al modelo utilizado en este trabajo, lo mismo ocurrió con los mMax, debiéndose buscar otra forma de poder comparar la real variación de los valores obtenidos en la ejecución del algoritmo. Se tomó la decisión de utilizar el paper de los autores Tabitha L. James, Evelyn C. Brown y Kellie B. Keeling, A Hibrid Grouping Genetic Algorithm (HGGA) for the Cell Formation Problem, el cual utiliza un algoritmo híbrido genético para la resolución de los nuevos problemas. El motivo de seleccionar este paper fue que en su anexo contiene las matrices ordenadas, que muestran la solución final del problema. Esto permitió visualmente obtener los mMax y los movimientos interceldarios encontrados por el autor. [23]

A continuación se muestran las instancias utilizadas, junto con los mMax encontrados en el paper:

Tabla 4: Nuevas instancias

N°	Autor	Máquinas	Partes	Celdas	mMax
1	King and Nakornchai	5	7	2	3
2	Waghodekar and Sahu	5	7	2	4
3	Seifoddini	5	18	2	3
4	Kusiak	6	8	2	3
5	Kusiak and Chow	7	11	5	2
6	Boctor	7	11	4	2
7	Seifoddini andWolfe	8	11	4	3
8	Chandrasekharan and Rajagopalan	8	20	3	4
9	Chandrasekharan and Rajagopalan	8	20	2	5
10	Mosier and Taube	10	10	5	4
11	Chan and Milner	10	15	5	4
12	Askin and Subramanian	14	24	7	6
13	Stanfel	14	24	7	3
14	McCormick et al.	16	24	8	5
15	Srinivasan et al.	16	30	6	6
16	King	16	43	8	4
17	Carrie	18	24	9	4
18	Mosier and Taube	20	20	6	7
19	Kumar et al.	23	20	7	6
20	Carrie	20	35	5	5
21	Boe and Cheng	20	35	5	5
22	Chandrasekharan and Rajagopalan	24	40	7	5
23	Chandrasekharan and Rajagopalan	24	40	7	5
24	Chandrasekharan and Rajagopalan	24	40	7	5
25	Chandrasekharan and Rajagopalan	24	40	11	5
26	Chandrasekharan and Rajagopalan	24	40	12	3

27	Chandrasekharan and Rajagopalan	24	40	12	3
28	McCormicket al.	27	27	6	11
29	Carrie	28	46	10	5
30	Kumar and Vannelli	30	41	14	4
31	Stanfel	30	50	13	3
32	Stanfel	30	50	14	4
33	King and Nakornchai	36	90	17	6
34	McCormicket al.	37	53	3	15
35	Chandrasekharan and Rajagopalan	40	100	10	6

Para el problema 34 no fue posible encontrar la matriz de incidencia, por lo tanto se considera como total de problemas 34 y no 35. En el anexo B se encuentran las matrices ordenadas obtenidas desde el paper HGGA.

Los resultados obtenidos al someter cada problema al algoritmo bat son los que se muestran a continuación:

Tabla 5: Soluciones obtenidas para las nuevas instancias

N°	Autor	Solución HGGA	Solución Bat	Variación %
1	King and Nakornchai	0	0	0,00
2	Waghodekar and Sahu	4	3	-25,00
3	Seifoddini	7	5	-28,57
4	Kusiak	2	2	0,00
5	Kusiak and Chow	9	8	-11,11
6	Boctor	4	4	0,00
7	Seifoddini andWolfe	10	7	-30,00
8	Chandrasekharan and Rajagopalan	27	28	3,70
9	Chandrasekharan and Rajagopalan	9	7	-22,22
10	Mosier and Taube	3	0	-100,00
11	Chan and Milner	0	0	0,00
12	Askin and Subramanian	9	2	-77,78
13	Stanfel	10	0	-100,00
14	McCormick et al.	37	25	-32,43
15	Srinivasan et al.	28	17	-39,29
16	King	42	24	-42,86
17	Carrie	33	29	-12,12
18	Mosier and Taube	54	42	-22,22
19	Kumar et al.	50	41	-18,00
20	Carrie	9	12	33,33
21	Boe and Cheng	44	12	-72,73
22	Chandrasekharan and Rajagopalan	1	16	1500,00
23	Chandrasekharan and Rajagopalan	10	24	140,00
24	Chandrasekharan and Rajagopalan	20	37	85,00
25	Chandrasekharan and Rajagopalan	51	33	-35,29
26	Chandrasekharan and Rajagopalan	61	61	0,00

27	Chandrasekharan and Rajagopalan	63	61	-3,17
28	McCormicket al.	78	64	-17,95
29	Carrie	10	95	850,00
30	Kumar and Vannelli	41	50	21,95
31	Stanfel	50	61	22,00
32	Stanfel	76	78	2,63
33	King and Nakornchai	140	124	-11,43
34	McCormicket al.	315	-	-
35	Chandrasekharan and Rajagopalan	37	138	272,97

- Coincide con el paper
- Mejor que el paper
- Peor que el paper (Variación menor al 100%)
- Peor que el paper (Variación mayor al 100%)

De los 35 problemas puestos a prueba, se llegó al mismo resultado que el paper HGGA en 5 ocasiones, mientras que se mejoró el resultado 19 veces. Por lo tanto son 24 los problemas en que se obtuvo éxito, equivalente al 70.58% del total analizados. Mientras que en 10 problemas no se alcanzó la solución del autor, de los cuales 6 tuvieron una leve variación menor al 100% y 4 una variación mayor que superó el 100%, llegando en un caso, en el problema 22, superar en un 1500% el valor obtenido por el autor. Estos 4 casos se desvían de la tendencia del algoritmo a tener leves variaciones en problemas que no se alcanza el valor buscado. Se estima que la razón de estos 4 malos resultados pudo deberse a que la matriz de incidencia utilizada no correspondía al problema o el mMax determinado no fue el correcto.

Otro dato importante, es que los problemas 31 y 35 tomaron un tiempo mucho mayor en la obtención de los resultados que el resto de problemas que han sido puestos a prueba en este trabajo, superando los 30 minutos en cada uno de ellos para las 30 ejecuciones realizadas.

Aunque las soluciones del autor no eran los óptimos, si eran valores cercanos que permitieron tener un acercamiento concreto al valor que se quería alcanzar.

7 Conclusión

El Manufacturing Cell Design es un problema muy frecuente en empresas o industrias manufactureras de hoy en día, quienes no regulan de la mejor manera los tiempos y costos que conlleva la fabricación de un producto. El manejo adecuado y la buena distribución de las máquinas que componen un centro de producción resulta clave en la rapidez de fabricación, que tiene un fuerte impacto a nivel económico y volumen de producción.

Poder resolver el Manufacturing Cell Design Problem mediante la aplicación del algoritmo bat resulta ser novedoso ya que no se había realizado antes. Entre muchos algoritmos metaheurísticos, este algoritmo se destaca por su sencillez y flexibilidad, junto con lo fácil que resulta su implementación.

Permite tener una tasa de convergencia muy rápida, al menos en las primeras etapas de la búsqueda, en comparación a otros algoritmos donde el tiempo para obtener buenos resultados puede ser considerable. El zoom automático que posee permite explotar intensamente una región donde se han encontrado soluciones prometedoras, y el control de parámetros le da cierta ventaja sobre otros algoritmos metaheurísticos, donde la mayoría de los parámetros son fijados antes de la ejecución, en tanto, el algoritmo bat puede variar los valores de volumen y tasa de pulso en cada iteración. Esto permitirá que cambie automáticamente desde la exploración a la explotación cuando la solución óptima se acerca.

Por medio de la implementación del algoritmo, se pudo identificar los componentes que inciden en el movimiento del bat, como es el caso de la frecuencia mínima y máxima que se han definido en un intervalo mayor a 0 y menor a 1. Se determinó una frecuencia mínima 0.25 y una frecuencia máxima de 0.50 las que han permitido obtener resultados satisfactorios.

De las 90 instancias en total, se logró alcanzar el óptimo en 82 ocasiones, más del 91% de los casos. Este resultado demuestra la eficiencia y robustez del algoritmo, donde se pudo determinar que una de las cualidades principales que posee, es dar saltos importantes en las primeras interacciones, que permiten comenzar con una buena solución base.

Se destaca también la rapidez en que se obtienen los resultados. En el caso de la utilización de 2 celdas fue mucho más frecuente que las 30 ejecuciones alcanzaran el óptimo, o al menos la mayoría de ellas.

En los 35 nuevos problemas los resultados fueron bastante buenos logrando más del 70% de éxito, la única complicación fue determinar los mMax y una solución cercana para comparar los resultados, debido a que ningún paper poseía estos valores de acuerdo al modelo utilizado en este trabajo.

8 Referencias

[1] N. B. W. &. C. C. Aljaber, "A tabu search approach to the cell formation," *Computers and Industrial Engineering,* vol. 1, no. 32, pp. 169-185, 1997.

[2] S. D. A. E. I. &. O. L. Lozano, "A one-step tabu search algorithm for manufacturing cell design," *Journal of the Operational Research Society,* vol. 5, no. 50, 1999.

[3] O. R. N. &. C. L. Durán, "Collaborative particle swarm optimization with a data mining technique for manufacturing cell design," *Expert Systems with Applications,* vol. 2, no. 37, pp. 1563-1567, 2010.

[4] V. &. N. T. T. Venugopal, "A genetic algorithm approach to the machine-component grouping problem with multiple objectives," *Computers and Industrial Engineering,* vol. 4, no. 22, pp. 169-480, 1992.

[5] Y. G. M. K. A. &. S. C. Gupta, "A genetic algorithm-based approach to cell composition and layout design problems," *international Journal of Production Research,* vol. 2, no. 34, pp. 447-482, 1996.

[6] T. C. C. &. C. S. Wu, "A simulated annealing algorithm for manufacturing cell formation problems," *Expert Systems with Applications,* vol. 3, no. 34, pp. 1609-1617, 2008.

[7] T. B. E. &. K. K. James, "A hybrid grouping genetic algorithm for the cell formation problem," *Computers and Operations Research,* vol. 7, no. 34, pp. 2059-2079, 2007.

[8] A. D. M. &. P. W. Nsakanda, "Hybrid genetic approach for solving large-scale capacitated cell formation problems with multiple routings," *European Journal of Operational Research,* vol. 3, no. 171, pp. 1051-1070, 2006.

[9] H. K. O. D. B. C. E. M. F. P. Ricardo Soto, "Cell Formation in group technology using constraint programming and boolean satisfiability," *Expert Systems with Applications,* vol. 39, pp. 11423-11427, 2012.

[10] G. Purcheck, "A linear-programming method for the combinatorial grouping of an incomplete set," *Journal of Cybernetics,* no. 5, pp. 51-58, 1975.

[11] E. &. P. G. Olivia-Lopez, "Load balancing for group technology planning and control," *International Journal of MTDR,* no. 19, pp. 259-268, 1979.

[12] A. &. C. W. Kusiak, "Efficient solving of the group technology problem," *Journal of Manufacturing Systems,* no. 6, pp. 117-124, 1987.

[13] F. Boctor, "A linear formulation of the machine-part cell formation problem," *International Journal Production Research,* vol. 2, no. 29, pp. 343-356, 1991.

[14] S. Sankaran, "Multiple objective decision making approach to cell formation: A goal programming model," *Mathematical Computer Modeling,* no. 13, pp. 71-77, 1990.

[15] S. &. R. D. Shafer, "A goal programming approach to cell formation problems," *Journal of Operations Management,* no. 10, pp. 28-34, 1991.

[16] M. &. A. K. Boulif, "A new branch-&-bound-enhanced genetic algorithm for the manufacturing cell formation problem," *Computers and Operations Research,* no. 33, pp. 2219-2245, 2006.

[17] T. Colin, "The Varienty of Life," Oxford, 2000.

[18] P. Richardson, "Bats," London, 2008.

[19] X.-S. Yang, "Bat algorithm: literature review and applications," *Int. J. Bio-Inspired Computation,* vol. 5, no. 3, pp. 141-149, 2013.

[20] X. S. Yang, "A New Metaheuristic Bat-Inspired Algorithm, in: Nature Inspired Cooperative Strategies for Optimization (NISCO 2010) (Eds. Cruz, C. González, J. R.; Pelta, D. A.; Terrazas, G)," *Studies in Computational Intelligence,* vol. 284, pp. 65-74, 2010.

[21] X.-S. Yang., "Bat algorithm for multi-objective optimisation," *International Journal of Bio-Inspired Computation,* vol. 3, no. 5, pp. 267-274, 2011.

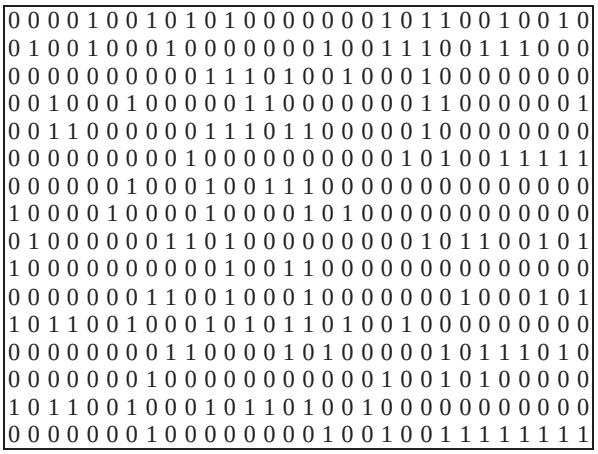
[22] L. A. M. P. K. A. C. D. R. J. P. P. R. Y. M. Nakamura, "BBA: A Binary Bat Algorithm for Feature Selection," Bauru.

[23] E. C. B. K. B. K. Tabitha L. James, "A hybrid grouping genetic algorithm for the cell formation problem," *Computers & Operations Research,* vol. 34, no. 2007, p. 2059–2079, 2005.

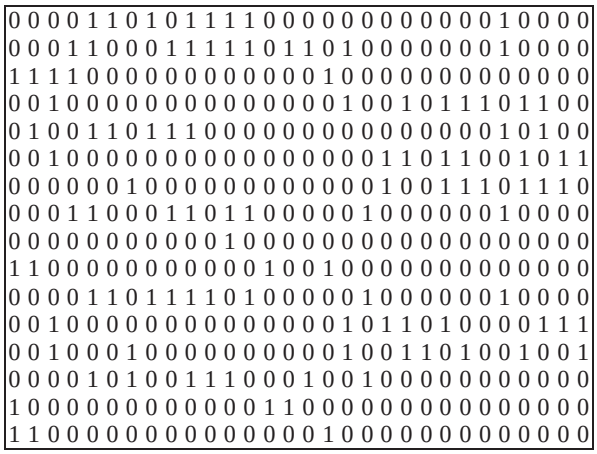
[24] H. K. O. D. B. C. E. M. F. P. Ricardo Soto, "Cell formation in group technology using constraint programming and boolean satisfiability," *Experts Systems and Applications,* no. 39, pp. 11423-11427, 2012.

Anexos

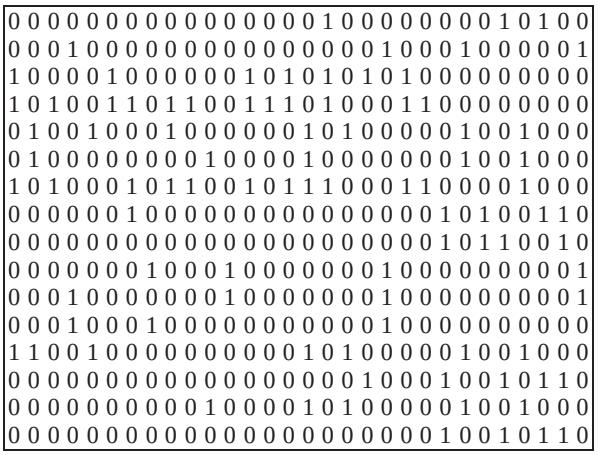
A. Matrices de Incidencia Boctor



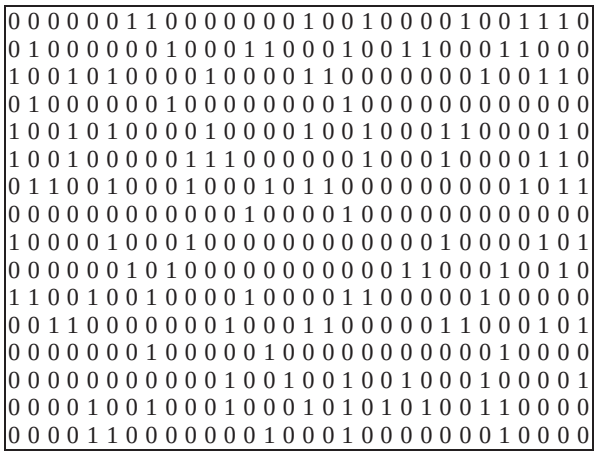
Problema 1



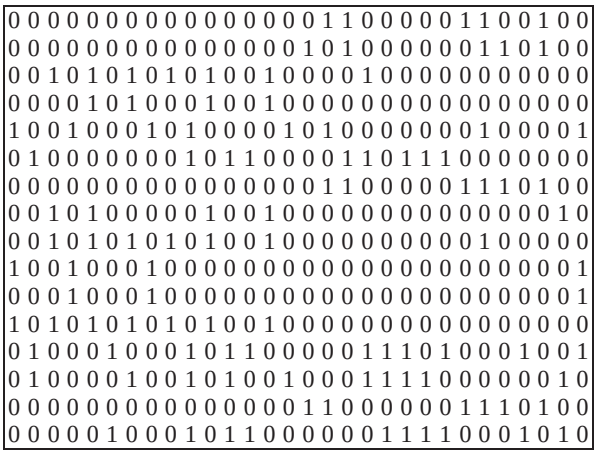
Problema 2



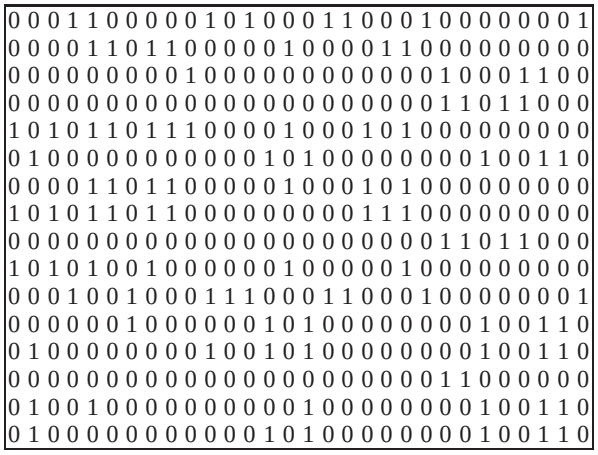
Problema 3



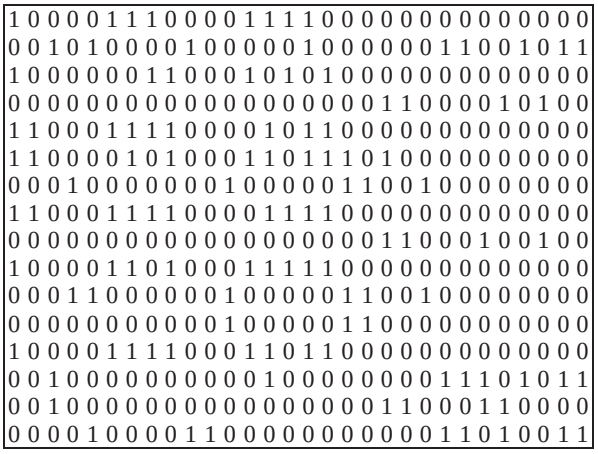
Problema 4



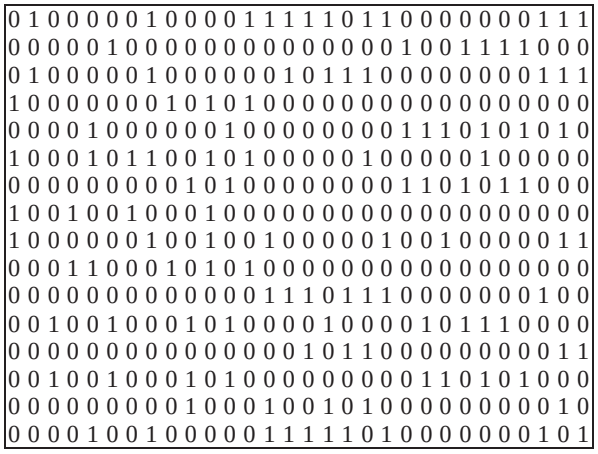
Problema 5



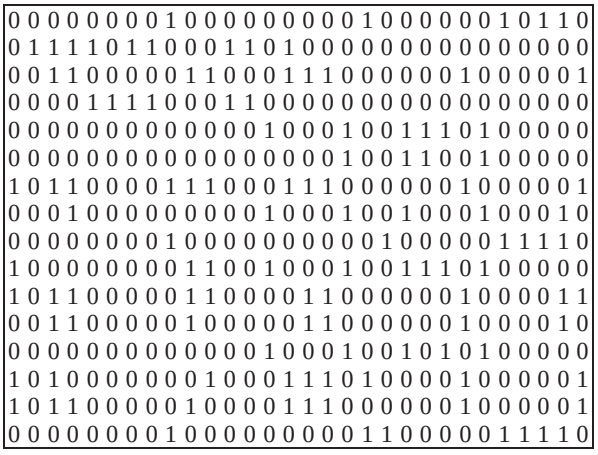
Problema 6



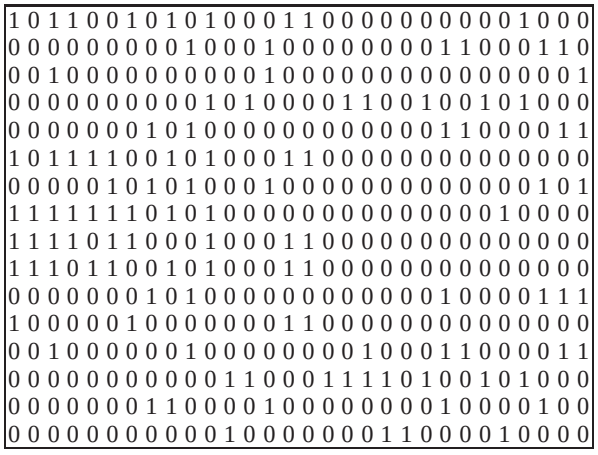
Problema 7



Problema 8

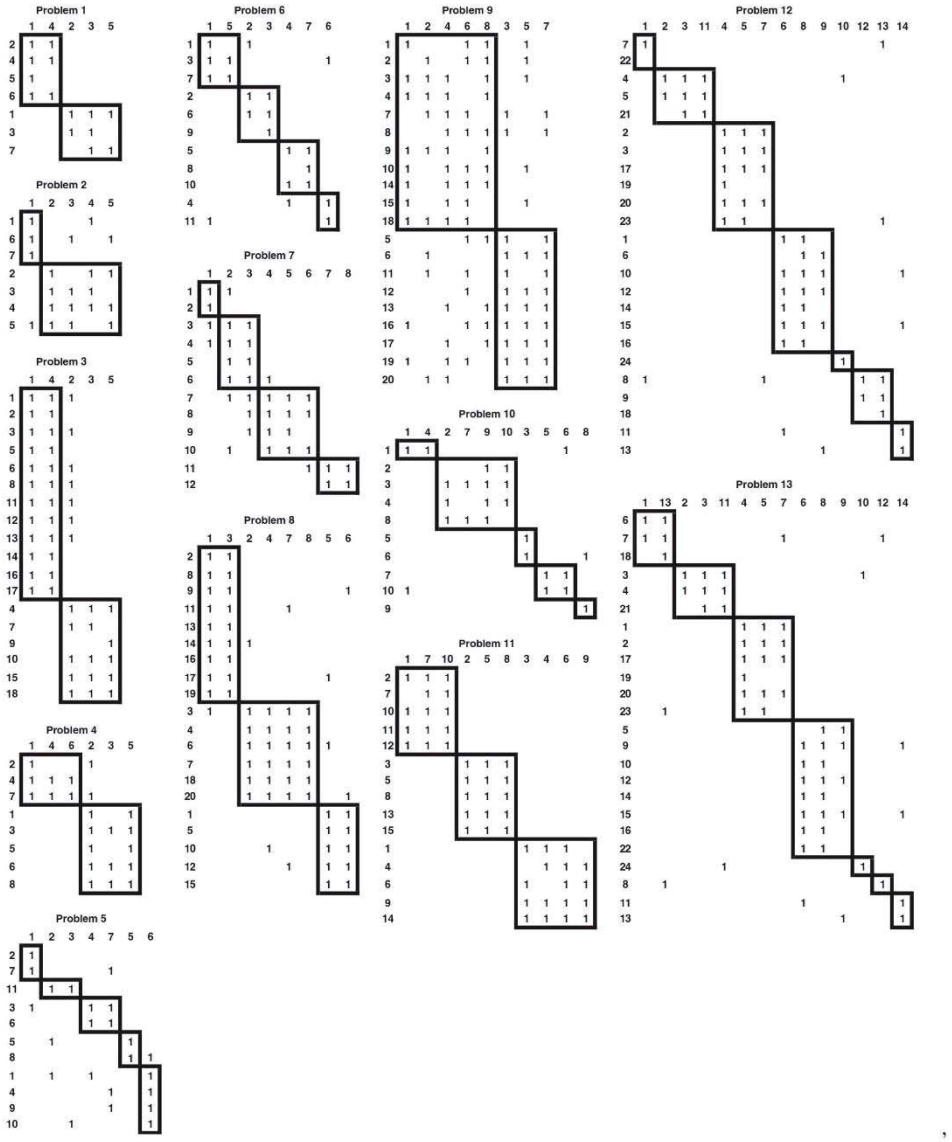


Problema 9

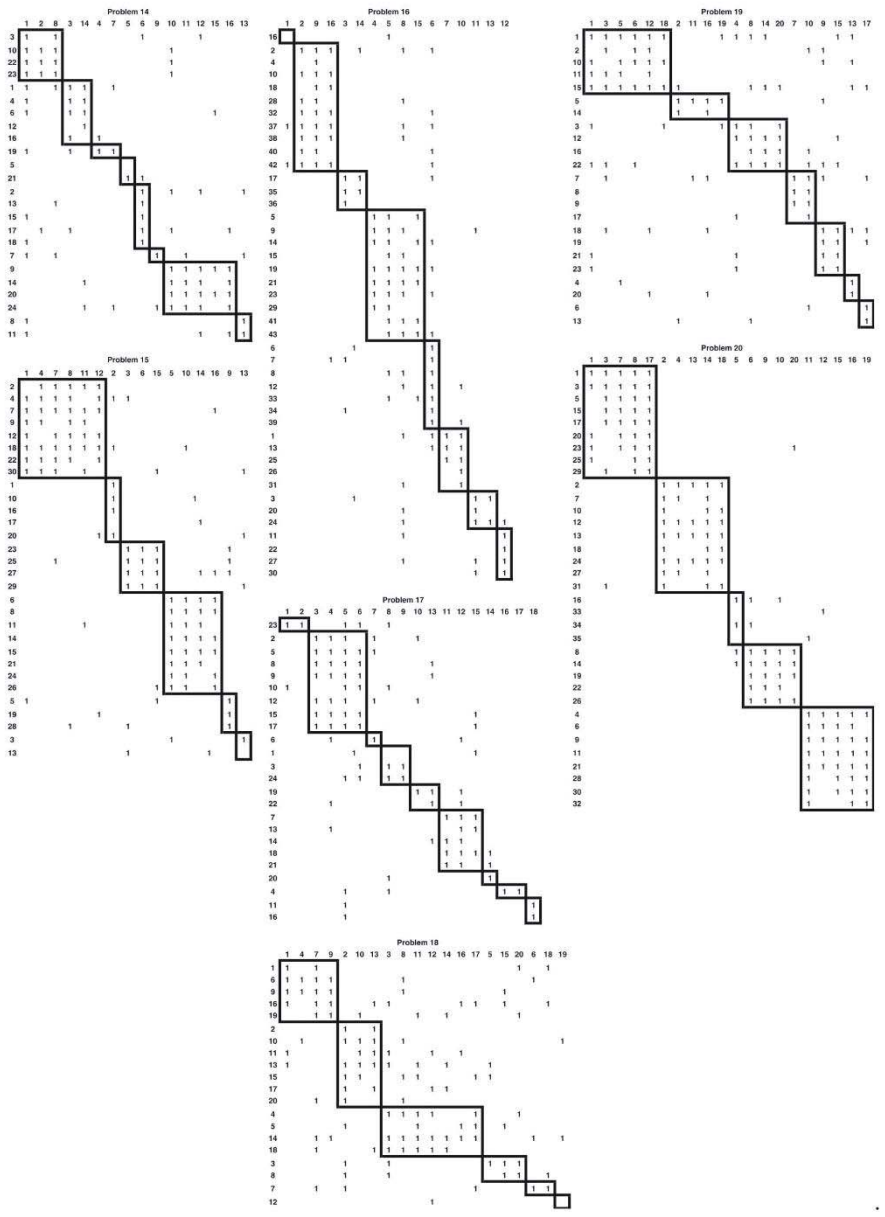


Problema 10

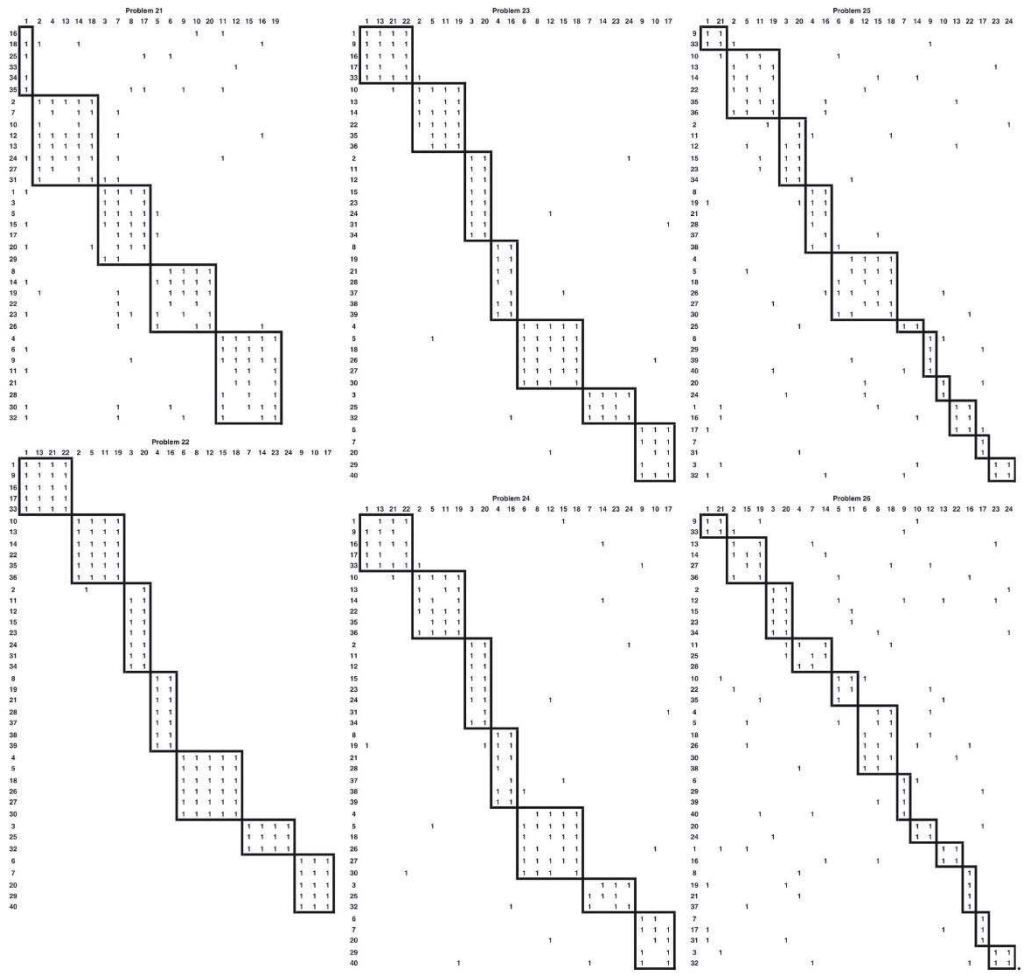
B. Matrices Ordenadas 35 nuevos problemas



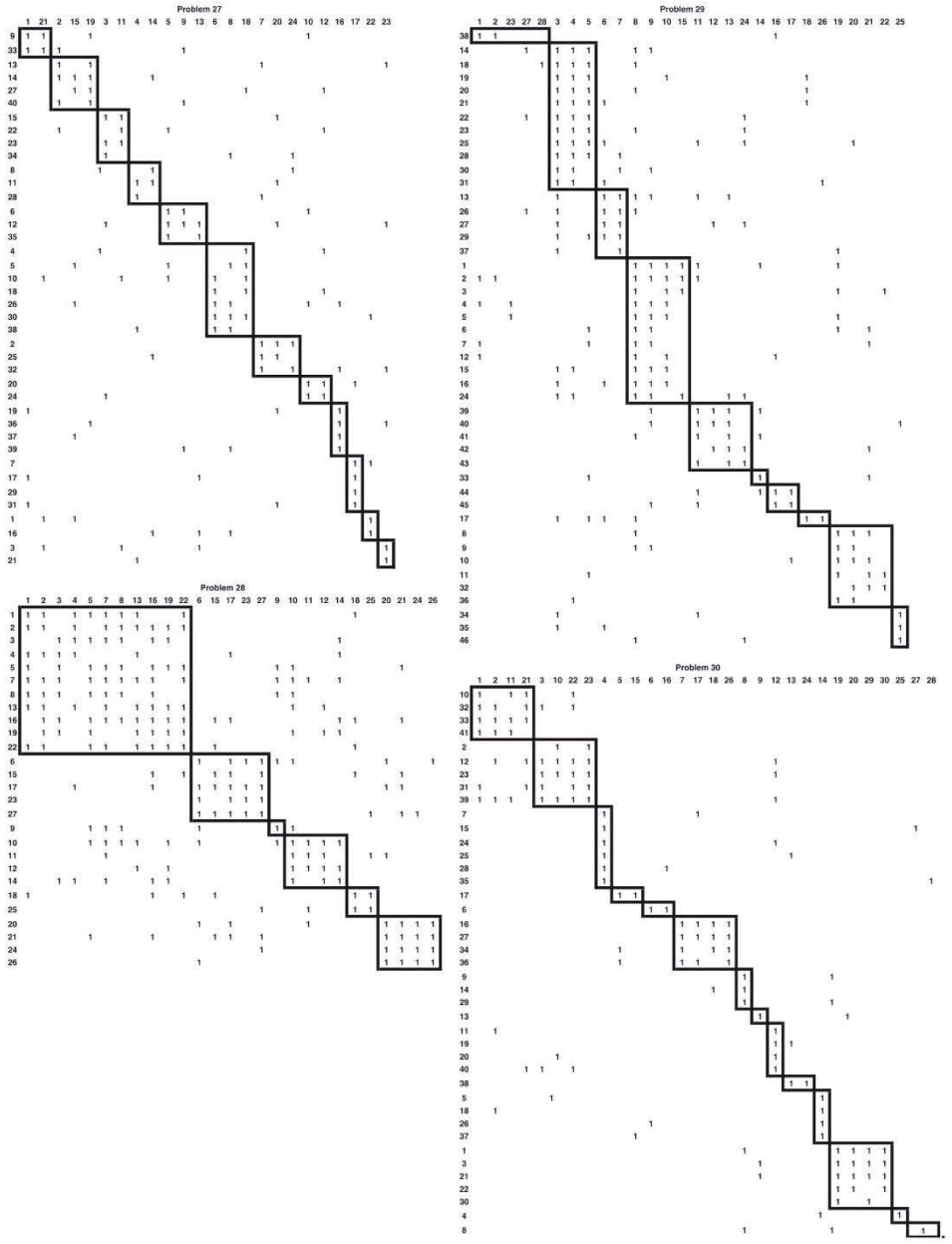
Problemas del 1 al 13



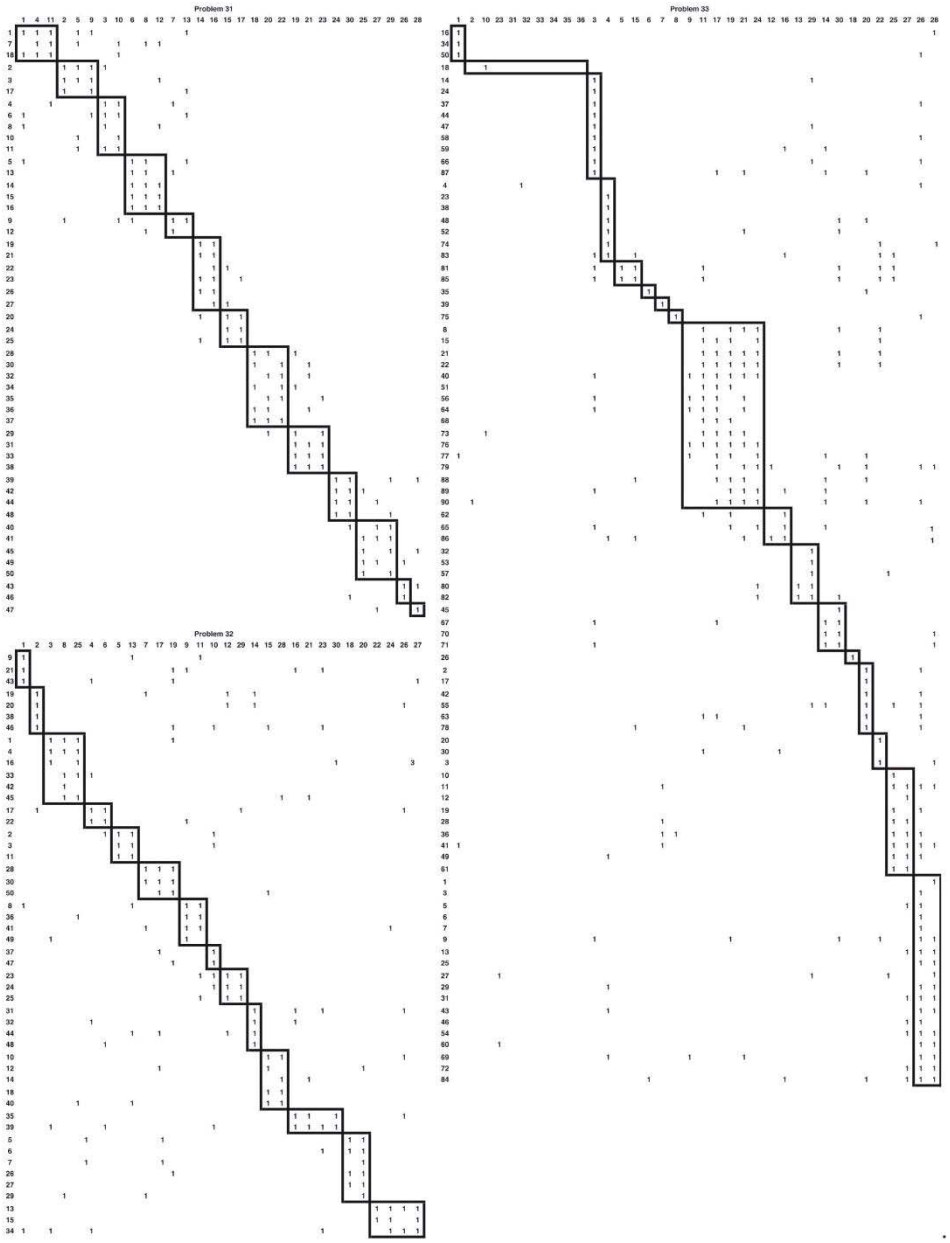
Problemas del 14 al 20



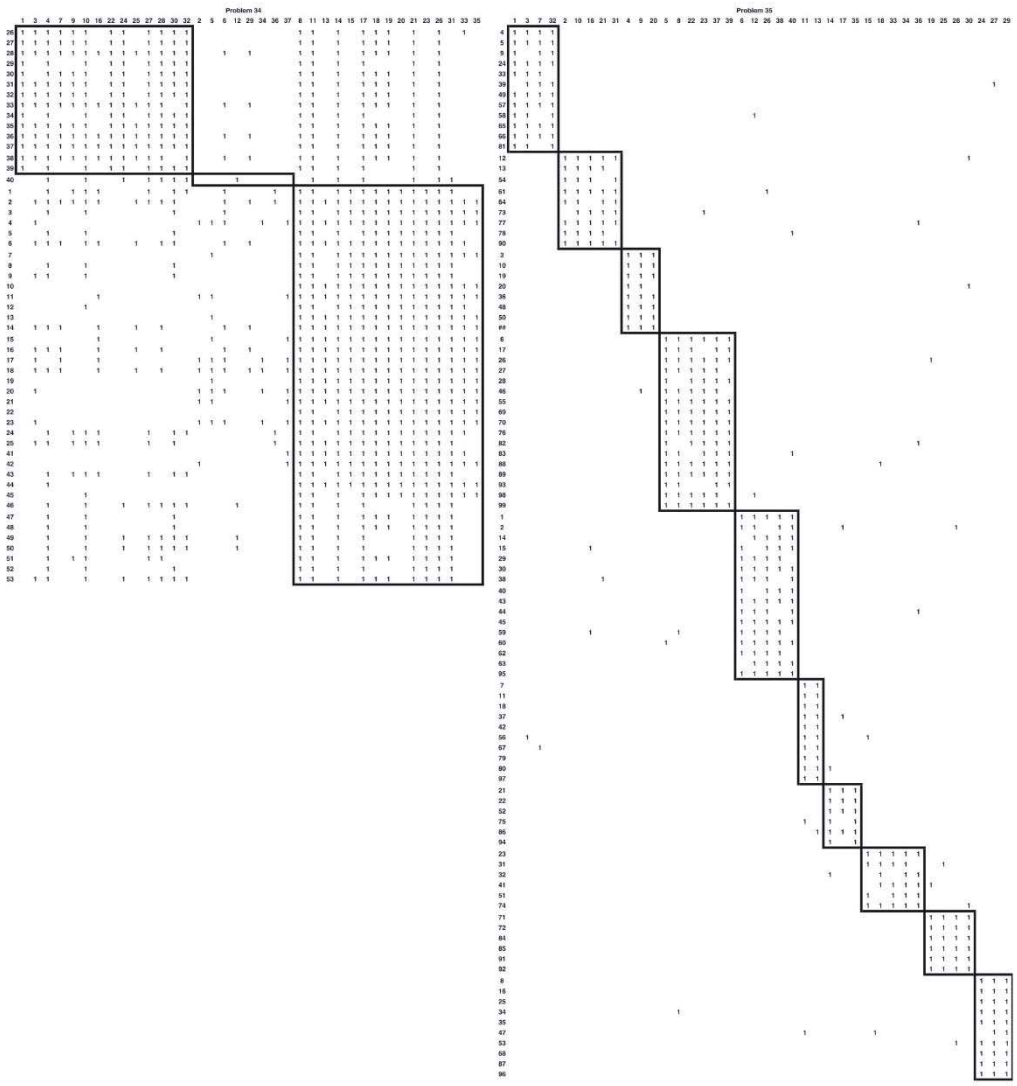
Problemas del 21 al 26



Problemas del 27 al 30



Problemas del 31 al 33



Problemas del 34 al 35