Proyecto Fin de Grado
Grado en Ingeniería de las Tecnologías Industriales

Análisis comparativo de algoritmos para la optimización de formación de células de producción

Autor: Paloma Navarro Delgado
Tutor: José Manuel Framiñán Torres

Dpto. de Organización Industrial y Gestión de Empresas I
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2020

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías Industriales

Análisis comparativo de algoritmos para la optimización de formación de células de producción

Autor:
Paloma Navarro Delgado

Tutor:
José Manuel Framiñán Torres

Dpto. de Organización Industrial y Gestión de Empresas I
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2020

Proyecto Fin de Carrera: Análisis comparativo de algoritmos para la optimización de formación de células de producción

Autor: Paloma Navarro Delgado

Tutor: José Manuel Framiñán Torres

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2020

El Secretario del Tribunal

A mi familia

A mis amigos

Agradecimientos

Llega el final de una etapa muy importante de mi vida y ahora sólo me queda dar las gracias a todas las personas que me han formado parte de ella y que han estado apoyándome durante todo este tiempo.

Gracias a mi familia por el apoyo incondicional, por comprenderme y animarme en los peores momentos. Por educarme en valores tan importantes como el esfuerzo, que me han ayudado a llegar hasta esta meta.

Gracias a mis amigos, los que siempre estarán en lo bueno y en lo malo, que han creído en mi cuando yo no lo hacía y han celebrado mis victorias como si fueran suyas.

Gracias a los profesores que me han inculcado las ganas de seguir interesándome cada vez más por el campo de la ingeniería.

Resumen

En este documento se aborda un estudio y comparación de algunos algoritmos para la optimización de la organización de procesos en forma de células de fabricación, es decir, a partir de los datos de entrada que son las máquinas disponibles y los trabajos a realizar se optimiza su distribución para un mejor rendimiento del proceso productivo. Para ello se han seleccionado dos tipos de metaheurísticas, en primer lugar, el Algoritmo de Selección Clonal, basado en el comportamiento de clonación y memoria de los anticuerpos del sistema inmunológico, del cual se han realizado varias modificaciones para la observación de los cambios en los resultados. Por otra parte, se ha escogido el algoritmo Iterated Greedy, que está formado principalmente por dos fases y es utilizado normalmente para problemas de programación de operaciones.

Para llevar a cabo el análisis, se procederá a la resolución de una batería de problemas a través de los códigos realizados en el software *Codeblocks*. Posteriormente se pasará a una exposición de los resultados con su correspondiente comparación entre algoritmos. Al final del documento se explicarán las conclusiones identificadas tras el análisis y la observación de los cálculos obtenidos.

Abstract

This paper discusses a study and comparison of some algorithms for the optimization of process organisation in the form of manufacturing cells, that is to say, from the input data that are the available machines and the tasks to be performed, its distribution is optimized for a better performance of the production process. For this purpose, two types of metaheuristics have been selected, first, the Clonal Selection Algorithm, based on the cloning and memory behaviour of antibodies in the immune system, of which several modifications have been made for the observation of changes in results. On the other hand, the iterated greedy algorithm has been chosen, which consists mainly of two phases and is normally used for operations programming problems.

To carry out the analysis, a battery of problems will be resolved through the codes made in the Codeblocks software. The results will then be presented with a corresponding comparison between algorithms. The conclusions identified after the analysis and observation of the calculations obtained shall be explained at the end of the document.

Índice

Agradecimientos **ix**

Resumen **xi**

Abstract **xiii**

Índice **xv**

Índice de Tablas **xviii**

Índice de Figuras **xx**

1 Introducción **1**
1.1 Justificación *2*
1.2 Estructura del documento *3*

2 Fabricación celular **5**
2.1 Tecnología de grupos *5*
2.2 Fabricación celular *5*
2.2.1 Ventajas e inconvenientes de la fabricación celular 8
2.3 Descripción del problema *8*
2.4 Planteamiento del problema *10*

3 Metodologías de resolución **12**
3.1 Problemas de optimización *12*
3.2 Algoritmo de Selección Clonal *13*
3.2.1 Algoritmo Selección Clonal 1 14
3.2.2 Algoritmo Selección Clonal 2 15
3.3 Algoritmo Iterated Greedy *16*

4 Resultados **19**
4.1 Herramienta utilizada *19*
4.2 Batería de problemas *20*
4.3 Exposición de los resultados *22*
4.4 Comparación *37*

5 Conclusión **45**

Referencias **47**

Anexo A. Matrices resultado Data A **51**

Anexo B. Matrices resultado Data B **63**

Anexo C. Desarrollo del código **73**

C.1 Selección clonal 1 *73*

C.2 Selección clonal 2 *80*

C.3 Iterated Greedy d=N *86*

ÍNDICE DE TABLAS

Tabla 1. Data A 20
Tabla 2. Data B 21
Tabla 3. Resultados de Algoritmo Selección Clonal 1 (pob=100). Data A 23
Tabla 4. Resultados de Algoritmo Selección Clonal 1 (pob=100). Data B 24
Tabla 5. Resultados de Algoritmo Selección Clonal 2 (pob=100). Data A 25
Tabla 6. Resultados de Algoritmo Selección Clonal 2 (pob=100). Data B 26
Tabla 7. Resultados de Algoritmo Selección Clonal 1 (pob=1,000). Data A 27
Tabla 8. Resultados de Algoritmo Selección Clonal 1 (pob=1,000). Data B 28
Tabla 9. Resultados de Algoritmo Selección Clonal 2 (pob=1,000). Data A 29
Tabla 10. Resultados de Algoritmo Selección Clonal 2 (pob=1,000). Data B 30
Tabla 11. Resultados de Algoritmo Iterated Greedy (d=0.7 x num. células). Data A 31
Tabla 12. Resultados de Algoritmo Iterated Greedy (d=0.7 x num. células). Data B 32
Tabla 13. Resultados de Algoritmo Iterated Greedy (d=1 x num. células). Data A 33
Tabla 14. Resultados de Algoritmo Iterated Greedy (d=1 x num. células). Data B 34
Tabla 15. Resultados de Algoritmo Iterated Greedy (d=2 x num. células). Data A 35
Tabla 16. Resultados de Algoritmo Iterated Greedy (d=2 x num. células). Data B 36
Tabla 17. Comparativo de resultados entre algoritmos. Data A 38
Tabla 18. Comparativo de resultados entre algoritmos. Data B 39
Tabla 19. Ratio mejor resultado para cada algoritmo 40
Tabla 20. Variabilidad entre las pruebas realizadas por problema y algoritmo. Data A 42
Tabla 21. Variabilidad entre las pruebas realizadas por problema y algoritmo. Data B 43

ÍNDICE DE FIGURAS

Figura 1. Variedad y evolución de paradigmas de sistemas de fabricación. (Popovic, 2018) 2
Figura 2. Célula tipo lineal. [Elaboración propia] 6
Figura 3. Célula tipo serpentina. [Elaboración propia] 7
Figura 4. Célula tipo "U". [Elaboración propia] 7
Figura 5. Célula tipo "U" invertida. [Elaboración propia] 7
Figura 6. Matriz de incidencia pieza-máquina. (Ulutas, 2019) 9
Figura 7. Matriz diagonalizada. (Ulutas, 2019) 9
Figura 8. Matriz con penalización. [Elaboración propia] 11
Figura 9. Selección clonal en sistemas inmunitarios. (Talbi, 2009) 14
Figura 10. Gráfica de tiempos de ejecución por algoritmo y problema. Data A. 40
Figura 11. Gráfica de tiempos de ejecución por algoritmo y problema. Data B. 41

1 INTRODUCCIÓN

Insanity is doing the same thing over and over again and expecting different results.

- Albert Einstein -

El objeto de este Trabajo de Fin de Grado consiste en la comparación de varios métodos para la optimización de la formación de células de fabricación, más bien llamado problema de formación celular, conocido como CFP (Cell Formation Problem). Este problema consiste en que, para un sistema de producción dado, se agrupen una serie de productos, pertenecientes a la misma familia o no, y un número de máquinas para realizar esos trabajos. Una familia consiste en un conjunto de productos que comparten elementos en su diseño, fabricación e implementación (Riba *et al.*, 2006). Estos productos y las máquinas correspondientes para llevar a cabo su fabricación se agrupan en una unidad llamada célula.

La etapa de diseño del sistema productivo llamado fabricación celular está formada por tres fases principales:

1. Agrupar las máquinas en células, más bien llamado problema de formación celular.
2. La organización de las células en el *layout* de la fábrica.
3. El *layout* de las máquinas dentro de cada célula.

El anteriormente mencionado CFP al igual que muchos otros problemas relacionados con fabricación de productos, tales como el proceso de planificación o la asignación de recursos, se trata de un problema demostrado como *NP* completo (Dimopoulos and Zalzala, 2000) , es decir, no hay un algoritmo que lo pueda resolver en tiempo polinomial. Por tanto, normalmente se emplean heurísticas, metaheurísticas, metaheurísticas híbridas e inteligencia artificial para su resolución.

1.1 Justificación

Los sistemas productivos siempre han estado estrechamente relacionados con el comportamiento y las necesidades de los consumidores. A principios del siglo XX comenzó a implantarse la producción en masa, impulsada por la industria automovilística liderada por Henry Ford. Este cambio en la forma de producir redujo drásticamente los costes de producción y permitió que cada vez una mayor parte de la población pudiera acceder a productos y servicios que antes solo podían permitirse las clases más adineradas. (Cuatrecasas Arbós, 2012).

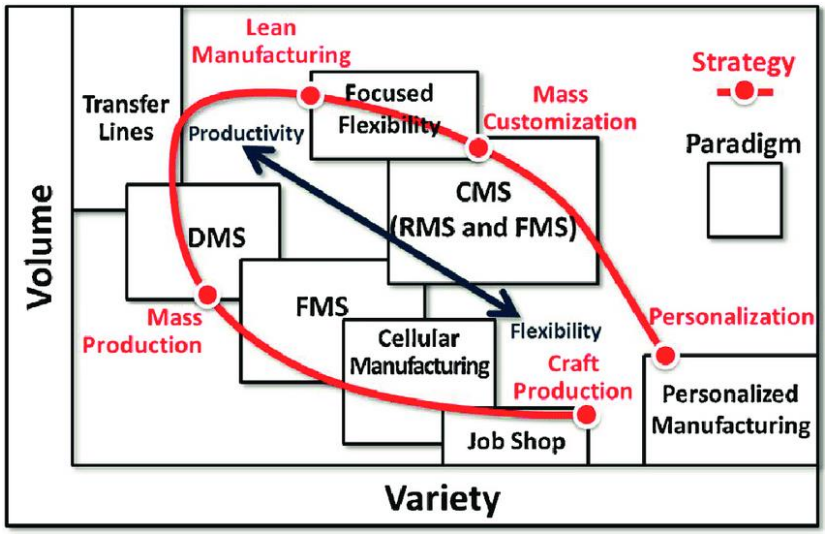


Figura 1. Variedad y evolución de paradigmas de sistemas de fabricación. (Popovic, 2018)

Sin embargo, lentamente se produce una evolución hacia nuevos enfoques en los que los tamaños de los lotes iban siendo más pequeños y la variedad de las piezas aumentaba. Ahora los consumidores van más allá y buscan diferenciarse con productos personalizados.

Esto tiene un impacto directo en el *layout* de las fábricas, es decir, en la distribución de la planta, ya que las clásicas configuraciones en línea no permiten la flexibilidad necesaria para la satisfacción de los requerimientos del cliente. Los talleres se enfrentaron a una gran diversidad de piezas, producidas en lotes comparativamente pequeños, y como consecuencia, el tiempo dedicado a la preparación para la producción aumentó exponencialmente.

Una solución a este problema es la formación de células de fabricación, es decir, organizar el sistema productivo en compartimentos individuales, que incluyen un conjunto de operaciones que se asignan a un grupo de trabajo con unos recursos y/o máquinas correspondientes. Se trata de un *layout* de máquinas con diversas funciones para el procesamiento de varias piezas o trabajos. Normalmente se dispone en forma de U, para facilitar el flujo y el movimiento de los trabajadores. Cada célula es independiente y permite adaptarse fácilmente a los cambios.

Idealmente, todas las operaciones que corresponden a un producto deberían de estar en la misma célula. Sin embargo, en la práctica los movimientos entre células están presentes. La fabricación celular aporta muchas ventajas para líneas de producción con variedad y volumen medios, por ejemplo, reduce los costes de *set-up* y de transporte, minimiza el inventario necesario y aprovecha mejor el espacio en la planta.

Ante este cambio, surge la necesidad de la optimización del proceso de diseño de las células, es decir, organizar los trabajos y las máquinas en diferentes unidades de la manera más eficaz posible.

1.2 Estructura del documento

Para la resolución del problema y alcanzar el objetivo del proyecto, el texto se ha estructurado de la siguiente forma:

- En primer lugar, en el capítulo 2 se introduce un marco teórico acerca de la tecnología de grupos y como a partir de ella deriva la fabricación celular. Además, se incluye una definición de los diferentes tipos de células y las ventajas e inconvenientes sobre esta forma de producción.
- A continuación, en el capítulo 3, se comienza con unos primeros conceptos sobre los problemas de optimización y una posible clasificación. Posteriormente, se pasa a una descripción exhaustiva de los diferentes métodos utilizados para resolver los problemas planteados más adelante. Incluye un pseudocódigo de cada uno de ellos para facilitar la comprensión de la codificación utilizada.
- Seguidamente, se encuentra el capítulo 4, donde se analizan y comparan los resultados obtenidos de dos baterías de problemas. Esto se ha realizado mediante cada una de las variaciones de los algoritmos codificados. La exposición de los cálculos se realiza mediante tablas y gráficas comparativas. Además, a parte del resultado numérico de la función objetivo, se incluyen datos de computación como el tiempo o el número de iteraciones.
- Para finalizar, en el capítulo 5 se enuncian las conclusiones que derivan de los resultados obtenidos en este proyecto, y cuáles serían buenas opciones para investigaciones futuras.
- Por último, se incluyen como apoyo para el trabajo tres anexos. El Anexo A y B dedicados a la representación gráfica de las agrupaciones celulares obtenidas como resultado y el Anexo C que refleja el código de los algoritmos utilizados para llevar acabo la resolución de los problemas.

2 FABRICACIÓN CELULAR

The Toyota style is not to create results by working hard. It is a system that says there is no limit to people's creativity. People don't go to Toyota to 'work' they go there to 'think'

- Taiichi Ohno -

Tras la introducción del objeto de este trabajo y su correspondiente justificación se expondrá un marco teórico sobre el tipo de sistema productivo llamado fabricación celular, los tipos de células que pueden existir y las ventajas e inconvenientes de la implementación de este sistema. Al inicio del capítulo se comenzará con una breve descripción de su predecesora, la tecnología de grupos.

2.1 Tecnología de grupos

La tecnología de grupos (García Díaz, 2002) está formada por el conjunto de métodos para organizar la fabricación de piezas. Estos procedimientos consisten en la agrupación e identificación de componentes similares para aprovecharse de sus similitudes en el diseño y producción para incrementar la eficiencia y la productividad.
Su principal objetivo es reducir tiempos de montaje, de *set-up*, de transporte o la cantidad de inventario entre otros. Normalmente, cuando la producción está organizada para gestionar familias de piezas, la disposición se describe como fabricación celular.

2.2 Fabricación celular

La fabricación celular es una aplicación del concepto de la tecnología de grupos que ayuda a las empresas a construir una variedad de productos para sus clientes con el menor desperdicio posible. Este sistema se organiza en una secuencia que soporta un flujo suave, de material y componentes, a través del proceso con un transporte o retraso mínimo.

La fabricación celular se basa en agrupar, normalmente, productos similares en familias que pueden procesarse con los mismos equipos y en la misma secuencia. La fabricación celular tiene grandes beneficios, entre los cuales se incluyen la reducción de WIP, mejor utilización del espacio, reducción del tiempo de entrega, mejor productividad y calidad, aumenta la flexibilidad y la visibilidad. (Kareem Sakran, Majeed Mahbuba and Saleh Jafer, 2016)

El hecho histórico más relevante que define el inicio de la expansión del método de Fabricación Celular viene marcado por los trabajos de Taiichi Ohno y Shingeo Shingo en la década de los 70 en Japón.

En su búsqueda en lo que posteriormente catalogaron como "el sistema de producción perfecto" se desarrollaría este método, que llegaría a ser una parte esencial en los modelos Just In Time y Flujo de una sola pieza de Toyota (Quesada Campos, 2019).

Dentro de las cualidades de la metodología Ohno-Shingo, se pueden destacar los siguientes puntos:

- Estaciones de trabajo adyacentes.
- Estaciones balanceadas y sincronizadas.
- No hay inventario entre estaciones.
- Se produce solamente lo que el cliente desea.
- Se demanda al proveedor solo el material que se necesita.

Este concepto fue evolucionando en los noventa hasta llegar a empresas de todos los sectores buscando la eficiencia.

Una célula de fabricación es, por tanto, un grupo de máquinas o equipos que se organizan para la fabricación de una serie de productos. Además, han de ser capaces de reconfigurarse rápidamente porque, como se ha mencionado en capítulos anteriores, es necesario un sistema productivo flexible para poder cumplir con las demandas sobre el producto (Quesada Campos, 2019).

Dependiendo del tipo de producto, la cantidad del mismo, su proceso de producción o el sector en el que se incluya la fábrica se pueden necesitar diferentes tipos de células como (Quesada Campos, 2019):

- Célula Tipo Lineal: La línea de producción es en línea recta, en este modelo, el tiempo de transporte aumenta y dificulta una comunicación efectiva entre operarios. Como consecuencia, es más difícil que un solo operario esté encargado de todas las operaciones. Por otro lado, con este modelo se pueden resolver problemas de flujo de materiales, ya que el producto se mueve en línea recta, de un extremo a otro.

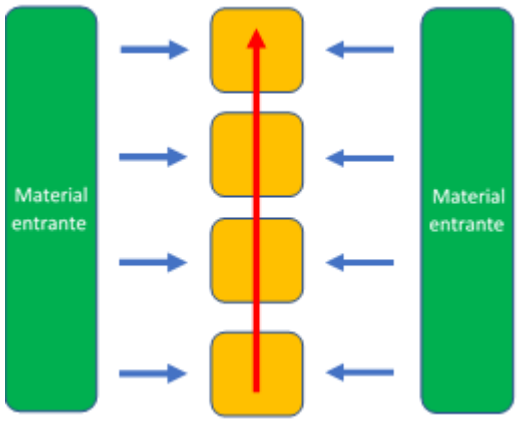


Figura 2. Célula tipo lineal.
[Elaboración propia]

- Célula Tipo Serpentina: Esta configuración aumenta la implicación de los operadores en la línea de producción, este modelo es más compacto que una celda en línea, consiguiendo de esta manera reducir el espacio y mejorar la comunicación entre diversos puntos de la célula.

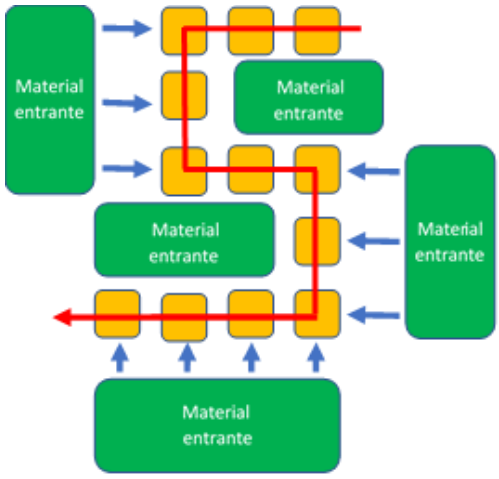


Figura 3. Célula tipo serpentina. [Elaboración propia]

- Célula Tipo "U": Es uno de los modelos más utilizados y flexibles ya que al reducir la distancia entre operaciones permite que un operario realice múltiples tareas reduciendo el tiempo de desplazamiento entre ellas. Por otra parte, no funciona bien cuando se tiene mucho almacenaje inicial ya que solo tiene una entrada de material.Si se hacen demasiado grandes se puede llegar a eliminar la ventaja sobre la comunicación.

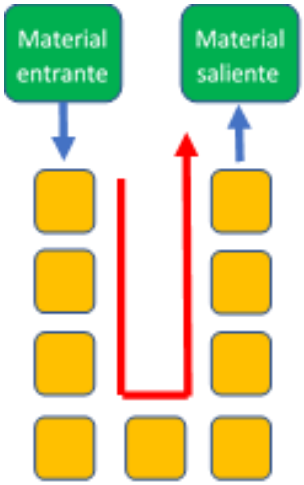


Figura 4. Célula tipo "U". [Elaboración propia]

- Célula Tipo "U" Invertida: Es una variación del tipo de "U", pero permite un poco de más de inventario inicial al poseer muchos posibles puntos de entrada de material. Aunque, es de mayor tamaño que la anterior.

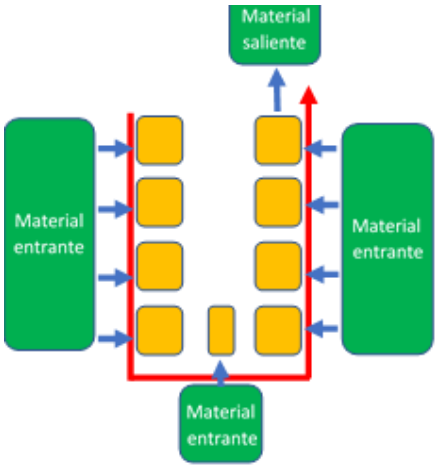


Figura 5. Célula tipo "U" invertida. [Elaboración propia]

2.2.1 Ventajas e inconvenientes de la fabricación celular

Algunas de las ventajas de la instalación de este sistema de fabricación son (García Díaz, 2002):

- Disminuye el stock entre procesos, puesto que cada pieza se realiza en una célula.
- Disminuyen los desplazamientos de material. Los transportes se reducen a el traslado de la materia prima desde el almacén a la célula y después el traslado del producto terminado. Por tanto, los tiempos de transporte, comparados con los del sistema *job shop*, donde las estaciones y los procesos no tienen por qué estar consecutivos, serán menores.
- Se reducen los tiempos de *set-up*. Como para formar las células se intentan coger productos de la misma familia, los procesos de producción son muy parecidos y no hay que hacer tantos cambios en las máquinas. Además, como los *set-up* son siempre los mismos, son más conocidos por los operarios.
- Se mejoran las relaciones humanas. En las células suele haber de dos a quince operarios, que se sienten como un equipo que trabajan para un resultado global que se trata de tener el producto finalizado. El trabajo en equipo favorece el compromiso, proporciona más seguridad y facilita la implementación de nuevas ideas y una gestión del conocimiento y del aprendizaje compartido.
- Especialización de los operarios en un tipo de piezas similares.
- Se aminora la complejidad de la gestión de las herramientas y las máquinas, ya que las piezas poseen similares características.

Por otro lado, algunas de las desventajas del sistema de fabricación en células son:

- Incremento de la inversión de capital en activos fijos, puesto que cada célula ha de contener todas las máquinas e instalaciones necesarias para producir las piezas, puesto que una máquina no puede ser utilizada por más de una célula, porque se perderían muchas de sus ventajas. Aun así, muchas veces las piezas salen de su célula sin estar acabadas para llevarlas a alguna máquina necesaria.
- Como se aumenta el número de máquinas disponibles, el nivel de utilización de las máquinas disminuye. Además, al estar las células muy especializadas en un tipo de productos, en caso de que la demanda de estos productos aumentara considerablemente, se tendrían unas células saturadas y por el contrario otras con mucho tiempo ocioso.

2.3 Descripción del problema

Antes de pasar a la definición de los métodos paso a paso, se van a introducir unos conceptos y consideraciones para la definición del problema y la complementación de la información posterior.

El dato de partida sobre el que se va a crear la distribución de las células es una matriz de productos y máquinas.

Esta matriz contiene todos los trabajos o productos existentes en las filas y por otro lado las máquinas necesarias en las columnas.

Los datos que se pueden obtener a través de la matriz son dos. Si en una casilla aparece un 0 significa que para ese producto (fila) no será necesario el paso por esa máquina (columna) y si aparece un 1 significa que si será necesaria su utilización. Esta matriz no presenta información referente a la secuencia de las operaciones o al número de máquinas existentes de cada tipo en la planta.

(a)	M1	M2	M3	M4
P1	0	1	0	1
P2	1	0	1	0
P3	0	1	0	1
P4	1	0	1	0
P5	1	0	0	0

Figura 6. Matriz de incidencia pieza-máquina. (Ulutas, 2019)

Para manejar mejor estos datos se codifica esta matriz en un vector (Ulutas, 2019). Para los trabajos se define el índice *i* que toma valores desde 1 hasta *p* y para las máquinas el índice *j* que toma de 1 hasta *m.* Las células *k* pueden ir desde 1 a c = min (*p, m*) porque cada célula debe contener al menos un trabajo y una máquina, por lo tanto, el máximo número de células posible es el mínimo entre el número de trabajos y el número de máquinas. El número de células en el vector está representado por 0, el número de ceros que aparece en el vector es igual al máximo de células posibles menos 1. Por tanto, el tamaño del vector sería nº trabajos + nº máquinas + nº "0".

Por ejemplo, para la matriz que se tiene en la imagen, el número de productos es 5, el de máquinas 4, entonces, habría 4 posibles células y 3 "0" en el vector. El vector quedaría de la siguiente forma:

{1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 0 ,0}

Trabajos Máquinas

De tal forma que, si desordenamos el vector aleatoriamente, podemos ver un posible resultado de una organización de células:

{1, 8, 9, 0, 2, 3, 5, 6, 0, 4, 7, 0}

En este caso, la primera célula estaría formada por el trabajo 1 y las máquinas 8 y 9, la segunda por los trabajos 2, 3, 5 y la máquina 6 y la tercera célula por el trabajo 4 y la máquina 7. No habría más células en este ejemplo.

Para poder evaluar este resultado se utilizan los conceptos de eficacia y eficiencia. Para poder explicar mejor estas definiciones primero se pasará a explicar algunas ideas.

Teniendo un ejemplo de resultado cualquiera del vector, si se representa en la matriz, quedaría de la siguiente forma:

(a)	M1	M2	M3	M4
P5	1	0	0	0
P1	0	1	**0**	1
P2	1	**0**	1	**0**
P3	0	1	**0**	1
P4	1	**0**	1	**0**

Figura 7. Matriz diagonalizada. (Ulutas, 2019)

En este ejemplo se tienen dos células una formada por la máquina 1 y la pieza 5, que forma una llamada célula unitaria, ya que solo contiene una pieza y una máquina. Y otra formada por las máquinas 2, 3, 4 y las piezas 1, 2, 3, 4.

Para medir a eficacia son necesarios dos conceptos, vacíos y excepciones. Los vacíos de la matriz corresponden a los "0" que se encuentran dentro de una célula, representan que esa pieza no necesita el uso de esa máquina, mientras más vacíos haya, menor es la utilización de la célula. En esta imagen hay un total de 6 vacíos. Por la otra parte, las excepciones se identifican como los "1" que quedan fuera de las células, siguiendo la definición hay 2 excepciones en la matriz.

2.4 Planteamiento del problema

El problema a tratar en este trabajo se centra en la maximización de la función objetivo del problema de formación de células que se corresponde con la eficacia de la respectiva agrupación de máquinas y trabajos en diferentes células (Suresh Kumar and Chandrasekharan, 1990):

$$\Gamma = \frac{1 - \Psi}{1 + \Phi} \tag{3-1}$$

Donde las variables, que irán cambiando dependiendo de la configuración, se corresponden con el número de excepciones y el número de vacíos de ese conjunto de células:

$$\Psi = \frac{nº\ de\ excepciones}{nº\ total\ de\ operaciones} \tag{3-2}$$

$$\Phi = \frac{nº\ de\ vacíos}{nº\ total\ de\ operaciones} \tag{3-3}$$

El número total de operaciones se define como la suma de las veces que cualquiera de los trabajos hace uso de una máquina, es decir, coincide el número de "1" que se incluyen en toda la matriz.

La eficacia tiene valores entre 0 y 1, siendo los mejores resultados de agrupaciones los que más se acercan a 1. Si se toma el ejemplo anterior, se obtendría una eficacia de 0.4666. Por otro lado, si una célula no incluye ninguna máquina o ningún trabajo se llama célula residual. Para la resolución de los problemas tomando el ejemplo de (Ulutas, 2019), este tipo de células se han tenido en cuenta para penalizar la eficacia, por tanto, si la solución incluye alguna célula de este tipo la eficacia tendrá menor valor, en concreto:

$$penalty = 0.5 * falta\ trabajo + 0.5 * falta\ máquina \tag{3-4}$$

La variable “falta trabajo” toma valores 1 ó 0, 1 en el caso de que existan una o más células en las que no haya ningún trabajo y 0 si no existe ningún caso. Lo mismo para “falta máquina”, que toma el valor 1 si hay una o más células en las que falta alguna máquina. Así, por tanto, si existen células residuales la eficacia se verá afectada de manera negativa.

	7	9	10	8	11	6	12
1	1	1	1	0	1	0	0
4	1	1	0	0	1	0	0
2	0	0	0	1	0	1	0
3	0	0	0	1	0	1	1
5	0	0	0	0	0	1	1

Figura 8. Matriz con penalización.
[Elaboración propia]

Por ejemplo, en la imagen adjunta la columna 10 no se encuentra incluida en ninguna célula, por lo tanto, se activaría falta máquina con valor 1, es decir, estaría penalizada.

Teniendo en cuenta el valor de la penalización la eficacia quedaría como:

$$Eficacia = \Gamma - penalty \tag{3-5}$$

Todas estas ecuaciones serán empleadas en los algoritmos que se definirán en las secciones siguientes.

Una vez bien definidos los conceptos previos y planteamiento del problema, además, del contexto acerca de la fabricación celular y sus orígenes, con sus principales características y sus diferencias respecto otros sistemas productivos se continuará con una descripción en profundidad de las metodologías seleccionadas para una posterior resolución de los problemas.

3 METODOLOGÍAS DE RESOLUCIÓN

Failure is simply the opportunity to begin again,
this time more intelligently.

- Henry Ford -

En este capítulo se mostrarán los algoritmos metaheurísticos que se han desarrollado para resolver el problema de formación de células de producción. Se partirá de unos datos de entrada que se transformarán para facilitar la resolución con los algoritmos, para, a continuación, desarrollar cada uno de los métodos propuestos. Primero con una descripción general del método y posteriormente un desarrollo más detallado de los pasos a seguir para poder reproducirlo de una forma clara.

Estos métodos nos permitirán alcanzar el objetivo de este documento, la comparación de diferentes metodologías para la resolución del problema CFP.

3.1 Problemas de optimización

En los problemas de optimización se busca encontrar el valor de unas variables para conseguir que una función objetivo alcance su máximo o su mínimo. En ocasiones el valor de los parámetros se encuentra supeditado al cumplimiento de unas restricciones.(Martí, 2001)

Existen numerosos tipos de problemas de optimización, pero, se hará una división en dos principalmente:

- Problemas lineales: Se consideran los problemas en los que la función a optimizar y las restricciones son lineales. Este tipo de problemas pueden ser resueltos con los llamados métodos exactos, por ejemplo, método Simplex. Este tipo de procedimientos nos proporcionan la solución óptima de nuestro problema. (Martí, 2001)
- Problemas *NP-hard*: Estos problemas poseen una complejidad algorítmica alta, por tanto, la utilización de métodos exactos para encontrar la solución puede que invierta demasiado tiempo. En este caso se utilizan otro tipo de planteamientos (Martí, 2001):
    - Método heurístico: Se trata de un procedimiento para resolver un problema de

optimización bien definido mediante una aproximación intuitiva, en la que la configuración del problema se utiliza para obtener una buena solución, no necesariamente óptima.(D´ıaz, A., Glover, F., Ghaziri, H.M., Gonzalez, J.L., Laguna, M, Moscato and F.T., 1996)

- Procedimientos metaheurísticos: Cuando los métodos heurísticos no son efectivos, aparecen una clase de métodos aproximados que proporcionan un nuevo conjunto de algoritmos híbridos combinando diferentes conceptos procedentes de la inteligencia artificial, la evolución biológica y los métodos estadísticos.(Osman, I.H. and Kelly, 1996)

Los métodos expuestos a continuación se incluyen en el grupo de las metaheurísticas. Dentro de este conjunto se han seleccionado dos procedimientos de diferentes orígenes: uno basado en aquellos métodos poblaciones bioinspirados, que será en donde se incluye el algoritmo de Selección Clonal y un segundo basado en vecindades, al que pertenece el algoritmo Iterated Greedy.

3.2 Algoritmo de Selección Clonal

Algunos aspectos de la biología siempre han servido de inspiración para desarrollar modelos computacionales y métodos de resolución de problemas. El sistema inmunológico es uno de ellos, a partir del cual se ha desarrollado el Sistema Inmunológico Artificial (AIS) (Dasgupta, Ji and Gonzalez, 2003). Las poderosas habilidades del sistema inmunológico para procesar información como extracción de características, reconocimiento de patrones, memoria y su carácter distributivo proveen ricas analogías para su homólogo artificial.

Existen diversas analogías sobre este sistema, para este algoritmo se va a utilizar el Principio de Selección Clonal (Talbi, 2009).

La selección clonal es una teoría ampliamente aceptada para modelar las respuestas del sistema inmunitario ante una infección. La teoría de la selección clonal fue propuesta por (Burnet, 1959). Está basada en los conceptos de clones y maduración por afinidad. Los linfocitos B y T son seleccionados para destruir antígenos específicos que invaden el cuerpo. Cuando el cuerpo está expuesto a un antígeno exógeno, aquellas células B que mejor se adhieran al antígeno proliferarán mediante clones. Las células B están programadas para clonar un tipo específico de anticuerpos Ab que se combinarán con un antígeno Ag. La unión entre un antígeno Ag y un anticuerpo Ab se rige por lo bien que un paratopo del Ab coincide con un epítopo del Ag. Se llama afinidad a lo fuerte que sería esta unión. Algunas de estas células clonadas son diferenciadas como células de plasma que son las que producen anticuerpos. El resto de los clones pasan a las células de memoria. Se aplica una mutación somática (hipermutación) a las células clonadas para favorecer la variedad genética. Además, se realiza una selección que implica la supervivencia de las células con mayor afinidad. Esto notifica que la selección clonal es un tipo de proceso evolutivo. Los algoritmos de selección clonal pueden ser vistos como una clase de algoritmos evolutivos que están inspirados en el sistema inmunitario.

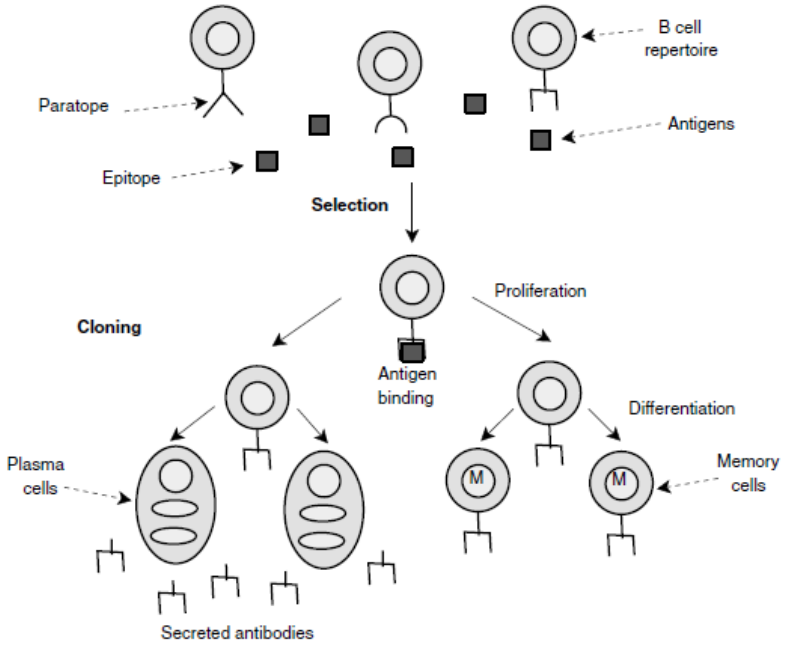


Figura 9. Selección clonal en sistemas inmunitarios. (Talbi, 2009)

Uno de los principales algoritmos basado en el principio de selección clonal es el CLONALG (Zuben, 2001). Las principales propuestas para este algoritmo son el mantenimiento de un set de memoria, selección y clones de los mejores anticuerpos, eliminar los peores, maduración por afinidad y selección de los clones proporcionalmente a su afinidad con el antígeno, y generación y mantenimiento de la diversidad (De Castro and Von Zuben, 2002). Un anticuerpo Ab representa una solución y un antígeno Ag representa el valor de la función objetivo a optimizar. En este caso, el anticuerpo estará representado por el vector, que incluye una disposición de células, y el antígeno será la función de eficacia.
A partir de este algoritmo se han realizado dos variaciones de la metaheurística original, algoritmo de Selección Clonal 1 y 2, que pasarán a ser definidas a continuación.

3.2.1 Algoritmo Selección Clonal 1

Tras la explicación en profundidad del contexto y el funcionamiento de la metaheurística elegida, en este apartado se pasará a una definición detallada, a través de un pseudocódigo, de cómo funciona el que se ha llamado algoritmo SC1. Para más información el código se encuentra en el Anexo C de este documento.

Pseudocódigo

1. Leer el fichero que contiene la matriz inicial.
2. Generar una población de anticuerpos aleatoria de tamaño N.
3. Mientras el criterio de parada no se satisfaga:
    3.1. Evaluar la población, cada uno de los individuos, y guardar su eficacia.
    3.2. Seleccionar los B% con mejor eficacia.
    3.3. Clonar los anticuerpos seleccionados.
    3.4. Madurar los clones y evaluarlos.
    3.5. Conservar los N mejores.
    3.6. Eliminar un 10% de los peores.
    3.7. Meter un 10% de nuevos anticuerpos aleatorios.

4. Resultado una matriz con la mejor población encontrada.

Para la población el tamaño N es determinado según el tamaño del problema a estudiar, N=100 para problemas pequeños y N=1000 para problemas grandes. La diferencia entre problemas grandes y pequeños se verá más adelante cuando se presente el set de problemas elegido. Seguidamente se pasará a una definición más exhaustiva de algunos de los pasos.

- 3: El criterio de parada está definido como 1.000 iteraciones sin ninguna mejora en la mejor eficacia obtenida de todos los individuos.
- 3.1: **Evaluar** significa obtener el valor de la función objetivo para cada uno de los vectores de la matriz. El objetivo del problema de formación de células es organizar los trabajos y las máquinas de la mejor manera posible, para esto como se explicó anteriormente se utiliza la medida de eficacia.
- 3.2: En este método se utiliza una **selección** por ruleta (Goldberg, 1989), es decir, cada anticuerpo de la población tiene un trozo de la ruleta proporcional a el valor de su función objetivo. El porcentaje de cada vector se calcula de la siguiente forma. En primer lugar, se suman los resultados de la función objetivo, la eficacia, de toda la población y después se dividen los valores de cada uno de los vectores entre el valor total calculado, por tanto, mientras mejor sea el valor de la eficacia, mayor porcentaje tendrá. Por cada iteración se elige aleatoriamente un número, entre 0% y 100%, por ejemplo 70%. Las soluciones seleccionadas serían aquellas que su porcentaje es mayor o igual a 70%, por tanto, el número de anticuerpos seleccionados en cada vuelta es variable.
- 3.3: En esta variación del método se ha elegido que se realicen 2 **clones** por cada anticuerpo seleccionado, como resultado tendríamos tres copias de un mismo vector.
- 3.4: **Madurar** consiste en mutar los clones de los anticuerpos seleccionados, se utiliza una mutación por pares, esto significa que dos localizaciones del vector se eligen aleatoriamente y los valores de estas localizaciones se intercambian.

Todos estos serían los pasos detallados de este primer algoritmo.

3.2.2 Algoritmo Selección Clonal 2

Se continúa con una segunda variación del anteriormente detallado algoritmo de Selección Clonal, nombrada SC2. Se pasará a una definición detallada, a través de un pseudocódigo, de cómo funciona y en qué se diferencia esta variación. En el Anexo C del documento se puede encontrar el código completo para más detalle.

Pseudocódigo

1. Leer el fichero que contiene la matriz inicial.
2. Generar una población de anticuerpos aleatoria de tamaño N.
3. Mientras el criterio de parada no se satisfaga:
    3.1. Evaluar la población, cada uno de los individuos, y guardar su eficacia.
    3.2. Seleccionar los B% con mejor eficacia.
    3.3. Clonar los anticuerpos seleccionados.
    3.4. Madurar los clones, evaluarlos y conservar los mejorados.
    3.5. Eliminar un 10% de los peores.
    3.6. Meter un 10% de nuevos anticuerpos aleatorios.
4. Resultado una matriz con la mejor población encontrada.

Para la población el tamaño N es determinado de la misma manera que en la primera modificación, según el tamaño del problema a estudiar explicado en la página anterior.

En esta variación la evaluación y la selección son exactamente de la misma forma que se ha explicado

en el algoritmo anterior. Los cambios se producen a partir del paso 3.3 donde se clonan los anticuerpos.

- 3.3: Esta variación solo incluye un **clon** por cada anticuerpo elegido, esta copia se realiza para más tarde mutarla.
- 3.4: **Madurar** consiste en mutar los clones de los anticuerpos seleccionados, se utiliza una mutación por pares, esto significa que dos localizaciones del vector se eligen aleatoriamente y los valores de estas localizaciones se intercambian. En esta opción mutamos el clon creado anteriormente y evaluamos su función objetivo. De la evaluación de estos clones surgen dos casuísticas:
    - En primer lugar, la eficacia puede ser mayor que la que tenía el anticuerpo antes de realizar la mutación. Como consecuencia, el anticuerpo pasaría a formar parte de la población total de anticuerpos. Ahora bien, como el número de la población es fijo, sustituiría al peor de la población.
    - En segundo lugar, si la eficacia es menor a la del anticuerpo anterior, se desecha esta mutación del anticuerpo.
- 3.5: Una vez que tenemos la población de 100 (o 1000) con los anticuerpos mejorados introducidos por la mutación, eliminamos un 10% de los peores anticuerpos, es decir, a los que dan peor resultado y se reemplazan por nuevos anticuerpos aleatorios.

3.3 Algoritmo Iterated Greedy

Este se trata de un algoritmo famoso para los problemas de permutación de *flowshop* pero, en este caso, se ha realizado algunos cambios para adaptarlo al problema de la formación de células.

Este algoritmo se remonta a 1954 con Johnson (Johnson, 1954), que propuso una regla para obtener secuencias óptimas para el problema de *flowshop* con dos máquinas. Desde entonces muchas heurísticas se han desarrollado como el ejemplo de la heurística NEH (Nawaz, Enscore and Ham, 1983) que es considerado uno de los mejores métodos. Mucho tiempo más tarde llegaron las metaheurísticas para este tipo de problemas. Los primeros intentos fueron el algoritmo Simulated Annealing (Osman and Potts, 1989) que usa temperatura constante y una inserción de vecindades con una simple regla de despacho, como que el primero que entra es el primero que sale. Más tarde la búsqueda tabú de Widmer y Hertz (Widmer and Hertz, 1989) más conocida como SPIRIT, consiste en dos fases: la primera que se trata de una solución inicial conseguida a través del problema Open Traveling Salesman Problem (OTSP) y la segunda que se trata de una búsqueda tabú, que consiste en intercambios entre vecindades. Otro algoritmo de búsqueda tabú surgió con Taillar (Taillard, 1990) y Colin (Colin, 1993) en los años noventa. Estos algoritmos tienen algo diferente a los algoritmos genéticos que también aparecieron por la época y es que, en vez de partir de una población aleatoria, como ocurre en el caso de los algoritmos genéticos, estos parten de una solución inicial no aleatoria, sino que se corresponde con la solución de alguna heurística como, por ejemplo, el método NEH.
Todos estos trabajos tienen en común que son fáciles de codificar y, por tanto, se pueden reproducir las soluciones fácilmente. En este apartado se va a desarrollar un algoritmo llamado Iterated Greedy (Ruiz and Stützle, 2007), se trata de un algoritmo simple y fácil de adaptar y que a su vez da unos resultados muy buenos a los problemas.
A grandes rasgos el método de Iterated Greedy genera una secuencia de soluciones mediante la iteración de dos fases principales, la fase de destrucción y la fase de construcción. La fase de destrucción se basa en eliminar algunos elementos de la solución inicial con la que empezamos la iteración, y la fase de construcción procede a reconstruir una nueva solución. Una vez acabada esta fase, se evalúa si se debe conservar la solución o no.
El número de iteraciones lo marca el criterio de parada que se le quiera dar dependiendo del problema.

Para el problema de permutación de *flowshop* la solución inicial que se utiliza es el resultado de la heurística NEH, pero para el problema que aborda este documento se va a utilizar como primera solución el vector resultante de la transformación de la matriz inicial.
Se pasará a una descripción más profunda de cada una de las fases:

- Fase de destrucción: Se realiza una extracción aleatoria de algunos de los elementos del vector sin repetición y se guardan en otro vector. Como resultado, al final de esta fase obtenemos dos vectores, el primero con el número de trabajos que tenía inicialmente menos los que se le han quitado y el segundo con los trabajos que faltan en el primer vector.
- Fase de construcción: Consiste en introducir cada uno de los valores anteriormente extraídos en cada una de las posiciones del vector e ir comprobando en cuál de las posiciones da mejor resultado. Esta fase para el problema de formación de células se realiza de una forma relativamente diferente puesto que para poder evaluar la función objetivo se necesita un vector con todos sus componentes para evaluar esa formación. Por lo tanto la solución es unir ambos vectores, pero en diferentes posiciones, de esta forma conseguimos que siempre tenga el tamaño inicial. Se realiza una inserción por todas las posiciones posibles del primer vector del que se extrajeron previamente. La posición que mejor función objetivo obtenga será en la que se quedará el elemento, por tanto, ahora el vector primero aumenta en tamaño y el segundo se reduce. Así se prosigue hasta que ya no queda ningún elemento en el segundo vector y todos están en sus mejores posiciones del primero.
- Después de estas fases se comprueba si la solución obtenida es mejor que la anterior, en ese caso se sustituye.

Este algoritmo cuenta con un parámetro que se introduce como dato *d,* con este parámetro se pueden realizar numerosas pruebas y comprobar diferentes resultados, como se realizará más adelante.

Para una mejor comprensión del algoritmo se adjunta un pseudocódigo (Pérez González, 2014).

Pseudocódigo

$\Pi := (\Pi_1, ….., \Pi_n)$ secuencia inicial; en nuestro caso el vector de la matriz inicial.
obj := Obj (Π); Calculamos la función objetivo de este vector.
$\Pi_b := \Pi$;
$obj_b := obj$;
$\Pi^R := \emptyset$, $\Pi^D := \emptyset$;
While stopping criterion is not satisfied do
    $\Pi' := \Pi$;
    For k=1 to *d* do
        π_j = *d* do Randomly selected job from Π';
        Π^D := Remove job π_j from Π';
        $\Pi^R := \Pi^R \cup \pi_j$;
    For k=1 to *d* do
        Π':= Insert π_j^R in the best position of Π^D ;
        $\Pi^D := \Pi'$;
    obj':= Obj (Π')
    if obj' > obj then
        $\Pi := \Pi'$;
        If obj > obj_b then
            $\Pi_b := \Pi$;
            obj_b := obj;
Return Π_b, obj_b

La secuencia inicial en este caso se corresponde con el vector de la matriz inicial, seguidamente se calcula la función objetivo que corresponde a este vector. A continuación, tras crear algunos vectores y variables, se entra en el bucle. La condición de parada de este bucle consiste en que no realice más

de 1000 iteraciones. Este número se ha obtenido en base a varias pruebas, y se ha determinado que es suficiente para conseguir buenas soluciones.

El bucle comienza con una primera selección aleatoria de elementos de este vector inicial, el número de selecciones está condicionado por el parámetro *d*, en este documento se han cogido varios valores para ver como afecta a las soluciones. Los elementos elegidos se guardan en otro vector. En el siguiente bucle lo que ocurre es que los elegidos anteriormente se van introduciendo entre los que han quedado previamente en el vector, viendo en que posiciones generan mejores resultados. Se van guardando las soluciones buenas en una variable.

Teniendo en cuenta el tamaño del vector inicial del que se dispone, los valores que ha tomado el valor delta son múltiplos del número máximo de células posibles para ese problema. Este número es igual al mínimo entre el número de trabajos y de máquinas. El motivo de esta referencia ha sido que este número está vinculado con el tamaño del vector para poder utilizar un mismo parámetro para todos los problemas, por tanto, tiene que ser dependiente del tamaño del problema.

En resumen, en este capítulo se han desarrollado en profundidad cada una de los procedimientos que se han aplicado para la resolución de las baterías de problemas que se exponen en el próximo capítulo, para poder obtener unas tablas de resultados con las que poder comparar y sacar las respectivas conclusiones.

4 RESULTADOS

Si todavía no lo has encontrado, sigue buscando.

- Steve Jobs -

Tras tener una idea general de todo el contexto y de los métodos y herramientas a utilizar para desarrollar este estudio, se procede a la exposición de los resultados adquiridos. Este apartado contiene la exposición y comparación de los resultados de los diferentes algoritmos como respuesta a los problemas de formación de células propuestos en las baterías.

Para la realización de dicha comparación se pasará a una presentación de los problemas utilizados.

4.1 Herramienta utilizada

Para la resolución de los problemas se ha utilizado el lenguaje C, por ser un lenguaje básico y universal, con la herramienta de compilación Codeblocks 17.12, que se trata de un entorno de desarrollo integrado de código abierto. La decisión de utilizar este programa ha sido por la facilidad y el conocimiento previo acerca del mismo. Para un mejor funcionamiento del programa se ha dispuesto de las librerías Schedule, utilizada en Programación de Operaciones, y Time, para medir la duración de los cálculos de cada problema.

En primer lugar, se han transformado los problemas de las baterías elegidas a archivos de texto para que el código pudiera leer los datos iniciales con facilidad. Para cada algoritmo se ha creado una carpeta que incluye: el archivo con el código programado, las librerías que se han utilizado, el archivo que Codeblocks llama proyecto. Además, en esta carpeta también se deben incluir los archivos de texto, de los problemas que queremos analizar con el algoritmo elegido.

El archivo de proyecto debe tener el mismo nombre que la carpeta en la que se guardan todos los archivos mencionados anteriormente.

Una vez que ejecutamos el código del algoritmo el programa nos proporciona estos datos de cada una de las pruebas: el vector con mejor resultado, la eficacia de ese vector, el tiempo que se ha tardado en realizar todos los cálculos y el número de iteraciones que han pasado hasta encontrar la mejor solución. Más tarde se ha utilizado Microsoft Excel para transformar y organizar estas soluciones para poder presentarlas de una forma ordenada y realizar algunas comparaciones.

4.2 Batería de problemas

Los problemas para analizar en este trabajo son los utilizados por Berna Ulutas (Ulutas, 2019) y están divididos en dos bloques, primero el set recogido por Gonçalves and Resende (Gonçalves and Resende, 2004) y más tarde fueron nombrados Data A y corregidos por Bychkov and Batsyn (Bychkov and Batsyn, 2017), gracias a los cuales se han obtenido los datos iniciales para poder realizar los estudios sobre los problemas. Este paquete incluye 35 problemas con un intervalo de 5 a 40 máquinas y de 7 a 100 piezas.

Tabla 1. Data A

Problem no	Data set source	Machine	Part
A1	(King, J. R., & Nakornchai, 1982)	5	7
A2	(Waghodekar, P. H., & Sahu, 1984)	5	7
A3	(Seifoddini, 1989)	5	18
A4	(Kusiak, A., & Cho, 1992)	6	8
A5	(Kusiak, A., & Chow, 1987a)	7	11
A6	(Boctor, 1991)	7	11
A7	(Seifoddini, H., & Wolfe, 1986)	8	12
A8	(Chandrasekaran, M. P., & Rajagopalan, 1986a)	8	20
A9	(Chandrasekaran, M. P., & Rajagopalan, 1986b)	8	20
A10	(Mosier, C. T., & Taube, 1985)	10	10
A11	(Chan, H. M., & Milner, 1982)	15	10
A12	(Askin, R. G., & Subramanian, 1987)	14	24
A13	(Stanfel, 1985)	14	24
A14	(McCormick, W. T., Schweitzer, P. J., & White, 1972)	16	24
A15	(Srinivasan, G., Narendran, T., &Mahadevan, 1990)	16	30
A16	(King, 1980)	16	43
A17	(Carrie, 1973)	18	24
A18	(Mosier, C. T.,&Taube, 1985)	20	20
A19	(Kumar, K. R., Kusiak, A., & Vannelli, 1986)	23	20
A20	(Carrie, 1973)	20	35
A21	(Boe, W., & Cheng, 1991)	20	35
A22	(Chandrasekharan, M. P.,&Rajagopalan, 1989) —Dataset 1	24	40
A23	(Chandrasekharan, M. P.,&Rajagopalan, 1989) —Dataset 2	24	40
A24	(Chandrasekharan, M. P.,&Rajagopalan, 1989) —Dataset 3	24	40
A25	(Chandrasekharan, M. P.,&Rajagopalan, 1989) —Dataset 5	24	40
A26	(Chandrasekharan, M. P.,&Rajagopalan, 1989) —Dataset 6	24	40
A27	(Chandrasekharan, M. P.,&Rajagopalan, 1989) —Dataset 7	24	40
A28	(McCormick, W. T., Schweitzer, P. J., & White, 1972)	27	27
A29	(Carrie, 1973)	28	46
A30	(Kumar, K. R., 1987)	30	41
A31	(Stanfel, 1985)—Figure 5	30	50
A32	(Stanfel, 1985)—Figure 6	30	50

A33	(King, J. R., & Nakornchai, 1982)	30	90
A34	(McCormick, W. T., Schweitzer, P. J., & White, 1972)	37	53
A35	(Chandrasekharan, M. P.,&Rajagopalan, 1987)	40	100

El segundo bloque está compuesto por problemas menos conocidos en artículos de investigación, también recogidos por Bychkov and Batsyn (Bychkov and Batsyn, 2017), llamado Data B. Este conjunto está formado por 32 problemas con el número de máquinas comprendido entre 6 y 50 y el número de piezas o trabajos entre 6 y 150.

Tabla 2. Data B

Problem no	Data set source	Machine	Part
B1	(Adil, G. K., Rajamani, D., & Strong, 1996)	6	6
B2	(Pa Rkin, R. E., & Li, 1997)	6	7
B3	(Brown, E., & Sumichrast, 2001)	6	11
B4	(Chan, H. M., & Milner, 1982)	7	5
B5	(Kusiak, A., & Chow, 1987b)	7	8
B6	(Zolfaghari, S., & Liang, 2002)	7	8
B7	(Won, Y., & Kim, 1997)	7	10
B8	(Sarker, B. R., & Khan, 2001)	8	8
B9	(Nair, 1999)	8	10
B10	(Islam, K. M. S.,&Sarker, 2000)	8	10
B11	(Kumar, K. R., Kusiak, A., & Vannelli, 1986)	9	15
B12	(HamI., Hitomi K., 1985)	10	8
B13	(Viswanathan, 1996)	10	12
B14	(Shargal, M., Shekhar, S., & Irani, 1995)	10	38
B15	(Won, Y., & Kim, 1997)	11	10
B16	(Seifoddini, 1989)	11	22
B17	(Moon, Y. B., & Chi, 1992)	12	19
B18	(Li, 2003)	14	14
B19	(Chan, H. M., & Milner, 1982)	15	10
B20	(Yang, M. S.,&Yang, 2008)	15	15
B21	(Yang, M. S.,&Yang, 2008)	15	15
B22	(Yang, M. S.,&Yang, 2008)	15	15
B23	(Harhalakis, G., Ioannou, G., Minis, I., & Nagi, no date)	17	20
B24	(Seifoddini, H.,&Djassemi, 1991)	18	24
B25	(Sandbothe, 1998)	20	10
B26	(Nagi, R., Harhalakis, G., & Proth, 1990)	20	51
B27	(Won, Y., & Kim, 1997)	26	28
B28	(Yang, M. S.,&Yang, 2008)	28	35
B29	(Seifoddini, H., & Djassemi, 1996)	35	15
B30	(Seifoddini, H., & Djassemi, 1996)	41	50
B31	(Yang, M. S.,&Yang, 2008)	46	105
B32	(Zolfaghari, S., & Liang, 1997)	50	150

4.3 Exposición de los resultados

A continuación, se pasará a una exposición de los resultados a la batería de problemas de cada uno de los algoritmos expuestos anteriormente.

En primer lugar, las modificaciones del algoritmo de selección clonal se han realizado pruebas para todos los problemas con dos tamaños de población, población pequeña (100 anticuerpos) y población grande (1.000 anticuerpos), a pesar de que se ha mencionado en este documento que la población grande sería solo para los problemas mayores, se ha decidido realizar una comparativa para ver cómo afectaría este cambio.

En segundo lugar, se mostrarán los resultados del algoritmo Iterated Greedy, para este se han realizado varias pruebas cambiando el parámetro delta, *d*, como se explica en el capítulo referente a las metodologías este parámetro, en este caso, es dependiente del número máximo de células posibles en un problema concreto. Para variar este parámetro, el número de células se ha multiplicado por diversos números. En particular se han recogido los resultados para:

- $d = 0.7$ x número de células posibles.
- $d = 1$ x número de células posibles.
- $d = 2$ x número de células posibles.

Para números mayores o menores el algoritmo no mostraba resultados admisibles para algunos problemas.

Se han ejecutado 10 pruebas para cada problema y la tabla de exposición de los resultados incluye:

-El máximo, el mínimo y el promedio de las eficacias obtenidas.

-El promedio del número de iteraciones en las que se encuentra la mejor solución.

-El promedio del tiempo de computación, en milisegundos, que se tarda en realizar cada una de las pruebas.

Se comenzará con la exposición de los resultados del algoritmo clonal.

Tabla 3. Resultados de Algoritmo Selección Clonal 1 (pob=100). Data A

SC1-100					
PROBLEMA	**MAX**	**MIN**	**PROM**	**P.ITER**	**P.MILISECS**
A01	0.8235	0.8235	0.8235	18.30	5.00
A02	0.6957	0.6957	0.6957	44.00	18.40
A03	0.7959	0.7959	0.7959	456.50	348.80
A04	0.7692	0.7692	0.7692	32.20	17.80
A05	0.6087	0.5600	0.5895	392.70	268.80
A06	0.7083	0.6800	0.6856	230.70	153.70
A07	0.6944	0.6136	0.6723	284.80	232.00
A08	0.8525	0.7049	0.8295	678.40	738.30
A09	0.5833	0.5688	0.5721	388.50	432.70
A10	0.7500	0.6786	0.7307	238.20	191.00
A11	0.9200	0.8000	0.8928	608.60	614.80
A12	0.7206	0.6316	0.6868	1,640.70	2,711.00
A13	0.7183	0.6615	0.6946	1,330.80	2,194.50
A14	0.5275	0.5100	0.5193	2,698.40	5,021.40
A15	0.6738	0.5802	0.6465	1,653.90	3,657.10
A16	0.5714	0.5127	0.5499	2,943.50	8,519.50
A17	0.5729	0.5510	0.5624	1,960.40	3,998.60
A18	0.4218	0.3815	0.4063	2,170.10	4,409.80
A19	0.4853	0.4345	0.4655	2,102.90	4,675.60
A20	0.7736	0.6531	0.7153	2,136.60	6,235.30
A21	0.5714	0.4862	0.5468	2,416.90	7,203.20
A22	0.9470	0.7101	0.7982	2,111.80	7,993.00
A23	0.8116	0.6350	0.7331	2,197.10	8,111.30
A24	0.7067	0.5870	0.6550	2,566.90	9,635.10
A25	0.5274	0.4460	0.4928	3,310.30	12,329.80
A26	0.4615	0.4231	0.4491	3,460.60	12,777.70
A27	0.4305	0.3947	0.4156	2,728.20	10,183.80
A28	0.5406	0.4466	0.5131	1,891.50	6,641.00
A29	0.4522	0.3750	0.4163	1,951.00	9,971.50
A30	0.6111	0.5590	0.5809	2,198.60	10,465.70
A31	0.5439	0.5024	0.5293	4,184.70	21,903.90
A32	0.4844	0.4235	0.4514	4,797.90	45,680.70
A33	0.4444	0.2031	0.3318	6,055.40	111,747.60
A34	0.5920	0.5505	0.5680	2,483.00	38,608.20
A35	0.7741	0.2944	0.4732	6,018.90	160,777.40

Tabla 4. Resultados de Algoritmo Selección Clonal 1 (pob=100). Data B

SC1-100					
PROBLEMA	**MAX**	**MIN**	**PROM**	**P.ITER**	**P.MILISECS**
B1	0.8095	0.8095	0.8095	59.30	31.70
B2	0.7222	0.6957	0.7196	380.10	212.30
B3	0.6071	0.5862	0.5981	378.70	266.40
B4	0.8889	0.8889	0.8889	19.00	7.90
B5	0.7500	0.7333	0.7450	202.70	135.30
B6	0.7391	0.6818	0.7277	68.80	38.90
B7	0.8148	0.7692	0.8103	401.80	322.20
B8	0.7222	0.7000	0.7160	371.30	249.90
B9	0.7576	0.6875	0.7506	491.70	390.30
B10	0.9000	0.7586	0.8610	202.90	148.20
B11	0.7273	0.7105	0.7193	525.50	554.90
B12	0.8276	0.7143	0.8072	650.30	532.80
B13	0.5962	0.5306	0.5813	506.60	877.10
B14	0.6333	0.4957	0.5835	1,583.70	3,575.30
B15	0.8333	0.7586	0.8132	599.40	528.50
B16	0.7356	0.6951	0.7220	854.30	2,486.00
B17	0.6552	0.6102	0.6398	940.30	1,368.40
B18	0.6027	0.5593	0.5882	669.00	891.40
B19	0.8000	0.7358	0.7846	620.90	752.00
B20	0.8710	0.7419	0.8201	920.20	2,679.90
B21	0.8333	0.6939	0.7452	809.70	1,995.70
B22	0.7258	0.5893	0.6704	756.80	1,112.30
B23	0.7816	0.6420	0.7312	819.50	1,579.80
B24	0.5660	0.5229	0.5444	2,336.40	10,402.30
B25	0.7600	0.7174	0.7367	778.60	2,081.70
B26	0.6017	0.5000	0.5632	2,933.90	24,801.90
B27	0.6712	0.5649	0.6196	2,099.50	7,581.50
B28	0.6459	0.5234	0.5858	3,489.70	13,874.10
B29	0.5657	0.5050	0.5423	2,353.70	5,631.60
B30	0.6805	0.4824	0.6036	4,267.10	31,298.10
B31	0.6477	0.2153	0.5048	7,365.40	110,595.40
B32	0.3040	0.2842	0.2947	11,720.10	257,207.60

Tabla 5. Resultados de Algoritmo Selección Clonal 2 (pob=100). Data A

SC2-100					
PROBLEMA	**MAX**	**MIN**	**PROM**	**P.ITER**	**P.MILISECS**
A01	0.8235	0.8235	0.8235	91.50	14.50
A02	0.6957	0.6957	0.6957	91.70	20.20
A03	0.7959	0.7959	0.7959	325.90	93.30
A04	0.7692	0.7692	0.7692	97.70	43.50
A05	0.6087	0.5833	0.5954	576.40	280.40
A06	0.7083	0.6800	0.7017	596.10	272.30
A07	0.6944	0.6585	0.6779	482.80	241.10
A08	0.8525	0.6056	0.7642	822.50	521.90
A09	0.5872	0.5669	0.5780	614.20	416.10
A10	0.7500	0.7097	0.7366	1,050.10	522.10
A11	0.9200	0.8367	0.8884	1,492.20	846.20
A12	0.6957	0.5769	0.6397	1,911.90	1,599.00
A13	0.6901	0.6053	0.6560	2,160.00	1,844.40
A14	0.4796	0.4015	0.4502	2,162.90	2,006.40
A15	0.6423	0.4805	0.5675	3,398.00	3,643.50
A16	0.5414	0.4205	0.5028	3,831.20	4,991.00
A17	0.5484	0.4531	0.5112	2,841.40	2,748.20
A18	0.4203	0.3672	0.3885	3,405.20	3,359.90
A19	0.4615	0.4041	0.4375	2,706.90	2,819.30
A20	0.7237	0.5789	0.6565	4,102.60	5,389.90
A21	0.5426	0.3840	0.4813	4,197.20	5,764.80
A22	0.7971	0.5526	0.6850	3,916.10	6,469.00
A23	0.6966	0.5267	0.6242	5,693.80	10,278.20
A24	0.6054	0.4105	0.5089	4,810.20	9,024.10
A25	0.4524	0.3418	0.4010	5,200.40	8,871.60
A26	0.4035	0.3246	0.3629	3,181.10	5,666.90
A27	0.3758	0.2950	0.3487	3,879.80	6,637.50
A28	0.5320	0.4676	0.5083	4,640.00	6,728.50
A29	0.4038	0.3293	0.3766	4,535.40	9,287.60
A30	0.5342	0.4551	0.4948	4,993.90	9,596.50
A31	0.4819	0.2682	0.4178	6,153.00	13,916.70
A32	0.3981	0.3402	0.3616	5,368.10	11,947.80
A33	0.3821	0.1536	0.1952	6,542.30	24,863.80
A34	0.5637	0.5329	0.5502	5,589.50	17,782.10
A35	0.3233	0.2334	0.2815	10,671.50	58,904.90

Tabla 6. Resultados de Algoritmo Selección Clonal 2 (pob=100). Data B

SC2-100					
PROBLEMA	**MAX**	**MIN**	**PROM**	**P.ITER**	**P.MILISECS**
B1	0.8095	0.8095	0.8095	102.30	41.70
B2	0.7222	0.7222	0.7222	107.30	42.60
B3	0.6071	0.5862	0.5994	549.50	254.90
B4	0.8889	0.8889	0.8889	51.70	25.50
B5	0.7500	0.6667	0.7400	444.30	188.60
B6	0.7391	0.7391	0.7391	196.00	84.70
B7	0.8148	0.7692	0.8057	339.90	161.80
B8	0.7222	0.7000	0.7168	689.70	313.00
B9	0.7576	0.6875	0.7126	404.50	210.00
B10	0.9000	0.8333	0.8867	416.90	199.20
B11	0.7273	0.6905	0.7117	999.70	539.50
B12	0.8276	0.8148	0.8250	445.20	203.10
B13	0.5962	0.5714	0.5886	709.80	426.90
B14	0.6000	0.4862	0.5560	2,436.30	2,332.40
B15	0.8333	0.7586	0.8111	903.20	459.10
B16	0.7391	0.6495	0.7071	2,096.40	1,539.90
B17	0.6441	0.5507	0.6040	1,716.80	1,164.80
B18	0.6111	0.5522	0.5793	2,112.40	1,430.20
B19	0.8000	0.6538	0.7240	662.50	391.90
B20	0.7778	0.6984	0.7212	1,703.90	1,180.80
B21	0.8333	0.6792	0.7495	1,646.40	1,112.50
B22	0.6885	0.5593	0.6498	2,253.60	1,505.70
B23	0.7386	0.5000	0.6795	2,240.50	1,970.20
B24	0.5446	0.4436	0.5060	3,286.70	3,181.80
B25	0.7556	0.5556	0.6860	1,464.90	898.30
B26	0.5605	0.4224	0.5010	5,741.00	9,793.80
B27	0.6107	0.4670	0.5527	4,400.30	5,938.10
B28	0.5268	0.3874	0.4654	5,044.80	8,879.30
B29	0.5259	0.4274	0.4683	3,031.50	3,259.50
B30	0.5843	0.0176	0.2786	6,724.20	18,656.00
B31	0.4561	0.2276	0.2813	10,705.70	72,141.60
B32	0.2577	0.2014	0.2401	14,199.80	144,491.30

Tabla 7. Resultados de Algoritmo Selección Clonal 1 (pob=1,000). Data A

SC1-1,000					
PROBLEMA	**MAX**	**MIN**	**PROM**	**P.ITER**	**P.MILISECS**
A01	0.8235	0.8235	0.8235	7.10	176.70
A02	0.6957	0.6957	0.6957	8.10	189.00
A03	0.7959	0.7959	0.7959	148.00	5,822.50
A04	0.7692	0.7692	0.7692	14.40	424.20
A05	0.6087	0.5862	0.6064	272.20	9,665.20
A06	0.7083	0.6800	0.7032	145.00	5,307.80
A07	0.6944	0.6591	0.6768	294.30	11,110.30
A08	0.8525	0.8525	0.8525	223.40	11,125.50
A09	0.5841	0.5688	0.5718	156.50	7,324.00
A10	0.7500	0.6897	0.7440	338.80	12,955.20
A11	0.9200	0.7551	0.8969	186.80	8,479.40
A12	0.7206	0.6769	0.7057	830.20	56,136.60
A13	0.7183	0.6765	0.7084	751.60	48,421.50
A14	0.5326	0.4854	0.5141	948.70	62,485.10
A15	0.6899	0.6462	0.6770	1,016.10	79,770.50
A16	0.5688	0.5161	0.5561	1,805.70	199,089.80
A17	0.5699	0.5490	0.5631	903.20	60,139.20
A18	0.4326	0.4027	0.4147	923.70	54,108.10
A19	0.5000	0.4472	0.4689	1,296.70	82,662.40
A20	0.7785	0.7020	0.7511	1,473.70	132,313.50
A21	0.5707	0.5222	0.5529	1,556.90	144,781.80
A22	0.9542	0.7388	0.8639	1,683.70	170,251.10
A23	0.8102	0.7132	0.7667	1,614.90	162,171.60
A24	0.7192	0.5985	0.6694	1,618.20	160,659.90
A25	0.5248	0.4453	0.5052	2,070.10	211,093.10
A26	0.4805	0.4444	0.4667	2,222.20	232,715.30
A27	0.4503	0.4204	0.4353	2,725.40	280,209.60
A28	0.5415	0.5265	0.5337	1,123.00	89,381.40
A29	0.4612	0.4218	0.4407	3,235.40	398,336.20
A30	0.6190	0.5435	0.5910	2,074.20	239,533.40
A31	0.5805	0.5235	0.5565	3,197.40	404,120.90
A32	0.5026	0.4402	0.4689	4,051.70	500,423.80
A33	0.4652	0.2188	0.3982	4,065.20	961,511.20
A34	0.5959	0.5225	0.5697	1,663.40	225,290.30
A35	0.8132	0.3925	0.7404	3,709.10	990,814.00

Tabla 8. Resultados de Algoritmo Selección Clonal 1 (pob=1,000). Data B

SC1-1,000					
PROBLEMA	**MAX**	**MIN**	**PROM**	**P.ITER**	**P.MILISECS**
B1	0.8095	0.8095	0.8095	12.50	316.70
B2	0.7222	0.7222	0.7222	25.20	714.80
B3	0.6071	0.6000	0.6064	124.60	4,006.60
B4	0.8889	0.8889	0.8889	8.30	199.20
B5	0.7500	0.7500	0.7500	55.70	1,643.40
B6	0.7391	0.7391	0.7391	35.70	1,048.00
B7	0.8148	0.8148	0.8148	93.30	3,069.40
B8	0.7222	0.7143	0.7206	198.00	6,356.60
B9	0.7576	0.7576	0.7576	84.80	2,946.50
B10	0.9000	0.9000	0.9000	63.40	2,204.50
B11	0.7273	0.7179	0.7263	515.70	23,650.80
B12	0.8276	0.8276	0.8276	254.40	9,007.90
B13	0.5962	0.5417	0.5854	258.90	10,563.20
B14	0.6404	0.5714	0.6114	1,336.10	109,475.70
B15	0.8333	0.8333	0.8333	323.90	12,962.30
B16	0.7391	0.6941	0.7288	450.30	26,979.70
B17	0.6552	0.6034	0.6489	479.00	27,111.10
B18	0.6111	0.5789	0.6012	462.60	21,738.90
B19	0.8000	0.7358	0.7936	314.10	14,368.60
B20	0.8710	0.6774	0.8129	444.10	20,755.60
B21	0.8333	0.7292	0.7867	713.20	31,873.60
B22	0.7258	0.6182	0.6800	516.50	24,123.60
B23	0.8111	0.6966	0.7612	842.70	46,232.20
B24	0.5673	0.5534	0.5607	1,048.20	65,995.20
B25	0.7600	0.7333	0.7539	407.40	22,315.50
B26	0.6029	0.5595	0.5809	1,971.90	247,161.30
B27	0.6980	0.6015	0.6488	1,246.00	92,088.10
B28	0.6588	0.5894	0.6302	2,379.10	212,671.90
B29	0.5566	0.5300	0.5459	1,287.70	111,912.40
B30	0.6975	0.6069	0.6546	3,067.80	413,548.00
B31	0.6645	0.2865	0.5837	4,708.30	4,911,359.30
B32	0.6056	0.2810	0.3284	6,022.00	2,497,259.70

Tabla 9. Resultados de Algoritmo Selección Clonal 2 (pob=1,000). Data A

SC2-1,000					
PROBLEMA	**MAX**	**MIN**	**PROM**	**P.ITER**	**P.MILISECS**
A01	0.8235	0.8235	0.8235	10.90	57.80
A02	0.6957	0.6957	0.6957	19.60	83.40
A03	0.7959	0.7959	0.7959	101.90	519.50
A04	0.7692	0.7692	0.7692	30.20	135.90
A05	0.6087	0.5926	0.6071	493.90	1,350.40
A06	0.7083	0.7083	0.7083	213.30	745.10
A07	0.6944	0.6829	0.6887	347.40	1,223.90
A08	0.8525	0.8525	0.8525	418.90	2,251.40
A09	0.5872	0.5833	0.5858	467.50	2,499.30
A10	0.7500	0.7500	0.7500	281.90	1,100.20
A11	0.9200	0.8542	0.9134	386.60	1,878.70
A12	0.7206	0.6883	0.7123	1,932.40	12,733.30
A13	0.7183	0.6944	0.7085	1,491.10	10,262.70
A14	0.5269	0.5049	0.5137	1,743.40	13,559.30
A15	0.6899	0.6549	0.6765	1,930.60	18,141.80
A16	0.5688	0.4740	0.5388	2,895.90	35,132.70
A17	0.5714	0.5481	0.5635	2,205.00	17,915.50
A18	0.4275	0.4024	0.4164	2,049.90	17,816.90
A19	0.5000	0.4626	0.4724	2,265.40	20,848.80
A20	0.7750	0.7436	0.7569	3,316.90	39,679.80
A21	0.5619	0.4977	0.5483	3,282.60	40,728.50
A22	0.9618	0.7551	0.8398	3,223.30	51,044.20
A23	0.8333	0.7305	0.7616	4,890.90	73,350.30
A24	0.7153	0.5854	0.6556	3,874.70	60,313.90
A25	0.5133	0.4324	0.4859	5,262.20	80,898.90
A26	0.4636	0.4238	0.4383	5,494.60	85,287.80
A27	0.4323	0.3869	0.4165	4,067.90	64,220.30
A28	0.5452	0.5252	0.5359	3,060.30	41,738.70
A29	0.4504	0.4156	0.4327	5,335.40	103,230.40
A30	0.6039	0.3854	0.5571	4,210.80	76,332.50
A31	0.5556	0.5000	0.5272	6,648.40	141,928.30
A32	0.4511	0.3990	0.4292	6,079.50	132,147.10
A33	0.4409	0.1966	0.3916	7,186.60	493,128.90
A34	0.5913	0.5546	0.5750	4,713.10	154,639.80
A35	0.7532	0.3186	0.5214	8,297.10	456,165.10

Tabla 10. Resultados de Algoritmo Selección Clonal 2 (pob=1,000). Data B

SC2-1,000					
PROBLEMA	**MAX**	**MIN**	**PROM**	**P.ITER**	**P.MILISECS**
B1	0.8095	0.8095	0.8095	32.30	115.20
B2	0.7222	0.7222	0.7222	55.40	209.40
B3	0.6071	0.6071	0.6071	161.20	560.30
B4	0.8889	0.8889	0.8889	16.00	75.90
B5	0.7500	0.7500	0.7500	142.10	443.90
B6	0.7391	0.7391	0.7391	75.60	286.90
B7	0.8148	0.8148	0.8148	91.70	390.30
B8	0.7222	0.7222	0.7222	263.50	774.50
B9	0.7576	0.7576	0.7576	152.90	631.80
B10	0.9000	0.9000	0.9000	126.60	548.50
B11	0.7273	0.7179	0.7226	500.90	2,099.80
B12	0.8276	0.8276	0.8276	161.40	606.60
B13	0.5962	0.5918	0.5957	450.70	1,814.60
B14	0.6264	0.5979	0.6138	1,755.00	13,946.00
B15	0.8333	0.8333	0.8333	314.10	1,262.60
B16	0.7391	0.7294	0.7375	789.00	4,911.20
B17	0.6552	0.6491	0.6525	1,304.70	6,827.80
B18	0.6129	0.6027	0.6092	1,077.20	5,311.00
B19	0.8000	0.8000	0.8000	343.60	1,805.50
B20	0.8710	0.8065	0.8473	1,067.80	5,810.40
B21	0.8333	0.7500	0.7981	950.00	5,061.30
B22	0.7258	0.7000	0.7066	1,068.40	5,745.40
B23	0.8111	0.7473	0.7787	1,833.10	12,599.50
B24	0.5673	0.5299	0.5555	2,725.40	21,721.50
B25	0.7600	0.7551	0.7577	822.00	3,969.20
B26	0.5936	0.5363	0.5720	4,047.90	65,086.10
B27	0.7192	0.6316	0.6798	3,658.50	44,366.40
B28	0.6202	0.5124	0.5783	4,732.90	74,733.30
B29	0.5631	0.5408	0.5510	2,379.50	22,359.30
B30	0.6591	0.0269	0.5432	6,347.30	173,021.60
B31	0.6034	0.2753	0.4529	9,710.10	675,551.20
B32	0.2907	0.1935	0.2683	13,385.30	1,431,061.20

Se procede con los resultados del algoritmo Iterated Greedy.

Tabla 11. Resultados de Algoritmo Iterated Greedy (d=0.7 x num. células). Data A

IG-d=0.7x					
PROBLEMA	**MAX**	**MIN**	**PROM**	**P.ITER**	**P.MILISECS**
A01	0.8235	0.8235	0.8235	101.00	3.20
A02	0.6957	0.6957	0.6957	47.00	1.50
A03	0.7959	0.7959	0.7959	78.80	5.30
A04	0.7692	0.7692	0.7692	34.70	3.20
A05	0.5862	0.5862	0.5862	289.00	20.10
A06	0.7083	0.7037	0.7056	42.40	6.20
A07	0.6829	0.6829	0.6829	55.00	8.30
A08	0.8525	0.8525	0.8525	305.60	76.60
A09	0.5688	0.5688	0.5688	1.00	0.00
A10	0.7500	0.7500	0.7500	46.30	9.50
A11	0.9200	0.9200	0.9200	20.50	3.10
A12	0.7206	0.7164	0.7198	199.20	189.80
A13	0.7183	0.7059	0.7171	276.70	263.30
A14	0.5326	0.5217	0.5293	371.40	510.20
A15	0.6899	0.6889	0.6891	179.80	347.80
A16	0.5743	0.5714	0.5727	508.20	1,592.30
A17	0.5743	0.5588	0.5677	414.10	720.00
A18	0.4306	0.4052	0.4213	435.00	801.90
A19	0.5040	0.4748	0.4897	398.50	843.30
A20	0.7791	0.7736	0.7782	431.60	1,674.10
A21	0.5798	0.5648	0.5717	289.20	1,106.90
A22	1.0000	1.0000	1.0000	140.90	1,076.60
A23	0.8511	0.8511	0.8511	96.40	736.60
A24	0.7351	0.7297	0.7343	270.20	2,041.00
A25	0.5329	0.5235	0.5292	424.40	3,130.60
A26	0.4859	0.4610	0.4778	442.20	3,349.60
A27	0.4658	0.4408	0.4559	468.20	3,442.30
A28	0.5482	0.5477	0.5481	270.90	1,617.40
A29	0.4656	0.4475	0.4598	659.50	8,641.40
A30	0.6267	0.6129	0.6173	765.40	9,780.00
A31	0.5965	0.5730	0.5878	748.50	13,056.10
A32	0.5083	0.4917	0.5003	565.90	9,409.70
A33	0.4792	0.4514	0.4682	729.40	31,207.00
A34	0.5953	0.5940	0.5945	203.40	7,451.40
A35	0.8403	0.8330	0.8395	327.50	64,897.00

Tabla 12. Resultados de Algoritmo Iterated Greedy (d=0.7 x num. células). Data B

IG d=0.7x					
PROBLEMA	**MAX**	**MIN**	**PROM**	**P.ITER**	**P.MILISECS**
B1	0.5556	0.5556	0.5556	1.00	1.60
B2	0.7222	0.7222	0.7222	21.00	3.20
B3	0.5862	0.5862	0.5862	194.00	12.10
B4	0.8889	0.8889	0.8889	37.00	0.00
B5	0.7500	0.7500	0.7500	44.60	1.60
B6	0.7391	0.7391	0.7391	25.00	3.00
B7	0.8148	0.8148	0.8148	107.00	5.00
B8	0.7222	0.7143	0.7183	417.00	30.20
B9	0.7576	0.7576	0.7576	145.30	15.00
B10	0.9000	0.9000	0.9000	40.80	4.70
B11	0.7273	0.7179	0.7254	298.20	56.50
B12	0.8276	0.8276	0.8276	41.80	3.60
B13	0.5962	0.5962	0.5962	97.90	23.10
B14	0.6404	0.5424	0.5718	202.30	206.40
B15	0.8333	0.8333	0.8333	22.00	7.80
B16	0.7391	0.7356	0.7384	79.10	42.00
B17	0.6552	0.6429	0.6527	100.90	52.90
B18	0.6111	0.5758	0.6024	114.30	58.30
B19	0.8000	0.8000	0.8000	60.60	15.30
B20	0.8710	0.8710	0.8710	46.90	31.20
B21	0.8333	0.8333	0.8333	35.90	26.90
B22	0.7258	0.7258	0.7258	54.80	37.90
B23	0.8111	0.8111	0.8111	46.60	58.60
B24	0.5673	0.5600	0.5629	227.50	408.90
B25	0.7600	0.7600	0.7600	113.40	38.90
B26	0.6034	0.6008	0.6030	290.90	2,008.90
B27	0.7248	0.6286	0.7152	137.50	767.10
B28	0.6729	0.6729	0.6729	133.20	1,242.90
B29	0.5700	0.5577	0.5637	175.70	339.80
B30	0.7290	0.6826	0.7070	692.80	26,811.80
B31	0.6798	0.6718	0.6781	415.70	74,679.70
B32	0.6184	0.5926	0.6120	741.60	293,049.10

Tabla 13. Resultados de Algoritmo Iterated Greedy (d=1 x num. células). Data A

IG-d=1x					
PROBLEMA	**MAX**	**MIN**	**PROM**	**P.ITER**	**P.MILISECS**
A01	0.8235	0.8235	0.8235	21.30	1.10
A02	0.6957	0.6957	0.6957	41.00	2.50
A03	0.7959	0.7959	0.7959	88.00	6.10
A04	0.7692	0.7692	0.7692	13.00	1.10
A05	0.6087	0.5862	0.5907	79.80	6.30
A06	0.7083	0.7083	0.7083	147.40	17.60
A07	0.6944	0.6829	0.6841	32.90	2.80
A08	0.8525	0.8525	0.8525	50.40	19.80
A09	0.5688	0.5688	0.5688	1.40	0.70
A10	0.7500	0.7500	0.7500	58.20	12.00
A11	0.9200	0.9200	0.9200	15.80	6.70
A12	0.7206	0.7164	0.7202	205.90	473.70
A13	0.7183	0.7059	0.7171	263.80	387.70
A14	0.5326	0.5196	0.5244	633.80	1,234.10
A15	0.6899	0.6889	0.6892	195.00	543.70
A16	0.5732	0.5578	0.5708	489.50	2,199.80
A17	0.5773	0.5567	0.5698	549.90	1,454.40
A18	0.4286	0.4091	0.4216	573.10	2,583.10
A19	0.5040	0.4771	0.4882	451.20	1,414.90
A20	0.7791	0.7758	0.7787	545.10	3,141.60
A21	0.5798	0.5707	0.5777	371.30	2,091.50
A22	1.0000	1.0000	1.0000	86.30	984.10
A23	0.8511	0.8511	0.8511	113.60	1,626.70
A24	0.7351	0.7273	0.7343	259.30	2,946.40
A25	0.5329	0.5205	0.5289	690.80	7,513.80
A26	0.4795	0.4621	0.4746	628.60	6,938.10
A27	0.4636	0.4467	0.4541	664.80	7,015.40
A28	0.5482	0.5415	0.5475	386.90	3,211.30
A29	0.4664	0.4411	0.4537	763.90	14,003.50
A30	0.6218	0.5959	0.6071	733.50	13,110.60
A31	0.5966	0.5469	0.5839	749.80	18,709.30
A32	0.5053	0.4822	0.4954	692.50	16,650.50
A33	0.4755	0.4497	0.4676	698.10	42,859.50
A34	0.5953	0.5940	0.5946	381.90	19,803.80
A35	0.8403	0.8403	0.8403	352.50	54,288.20

Tabla 14. Resultados de Algoritmo Iterated Greedy (d=1 x num. células). Data B

IG d=1x					
PROBLEMA	**MAX**	**MIN**	**PROM**	**P.ITER**	**P.MILISECS**
B1	0.8095	0.8095	0.8095	9.00	1.50
B2	0.7222	0.7222	0.7222	36.70	1.60
B3	0.6071	0.6071	0.6071	55.50	6.80
B4	0.8889	0.8889	0.8889	4.00	0.00
B5	0.7500	0.7500	0.7500	31.80	6.30
B6	0.7391	0.7391	0.7391	16.70	3.00
B7	0.8148	0.8148	0.8148	21.40	0.00
B8	0.7143	0.7143	0.7143	14.00	1.60
B9	0.7576	0.7576	0.7576	57.10	7.90
B10	0.9000	0.9000	0.9000	12.00	0.00
B11	0.7273	0.7273	0.7273	84.70	23.20
B12	0.8276	0.8276	0.8276	9.60	0.80
B13	0.5962	0.5962	0.5962	105.00	26.80
B14	0.6404	0.6404	0.6404	195.80	282.40
B15	0.8333	0.7333	0.7933	26.40	3.20
B16	0.7391	0.7391	0.7391	79.90	56.10
B17	0.6552	0.6500	0.6541	209.70	146.70
B18	0.6129	0.6111	0.6113	225.00	160.90
B19	0.8000	0.8000	0.8000	49.80	19.00
B20	0.8710	0.8710	0.8710	36.10	28.10
B21	0.8333	0.8333	0.8333	34.30	27.00
B22	0.7258	0.7258	0.7258	61.10	54.50
B23	0.8111	0.8111	0.8111	55.40	95.30
B24	0.5673	0.5619	0.5640	349.80	852.80
B25	0.7600	0.7556	0.7596	169.00	84.20
B26	0.6068	0.6026	0.6036	348.20	3,439.90
B27	0.7248	0.5894	0.7113	169.40	1,260.00
B28	0.6729	0.6561	0.6708	285.10	3,490.80
B29	0.5730	0.5524	0.5637	573.80	1,561.60
B30	0.6981	0.6647	0.6817	656.40	34,446.50
B31	0.6798	0.6783	0.6792	484.90	118,474.30
B32	0.6134	0.3118	0.3717	497.90	268,493.80

Tabla 15. Resultados de Algoritmo Iterated Greedy (d=2 x num. células). Data A

IG-d=2x					
PROBLEMA	**MAX**	**MIN**	**PROM**	**P.ITER**	**P.MILISECS**
A01	0.8235	0.8235	0.8235	15.00	1.60
A02	0.6957	0.6957	0.6957	54.80	5.40
A03	0.7959	0.7959	0.7959	18.70	7.80
A04	0.7692	0.7692	0.7692	12.30	0.00
A05	0.6087	0.5926	0.6013	454.10	144.10
A06	0.7083	0.7083	0.7083	235.50	77.30
A07	0.6944	0.6829	0.6887	225.80	118.60
A08	0.8525	0.8525	0.8525	45.00	43.20
A09	0.5872	0.5688	0.5706	53.90	50.50
A10	0.7500	0.7500	0.7500	384.60	225.70
A11	0.9200	0.9200	0.9200	24.50	23.80
A12	0.7206	0.6866	0.7127	539.70	1,805.30
A13	0.7183	0.6857	0.7059	479.10	1,648.80
A14	0.5217	0.5000	0.5092	519.80	2,476.70
A15	0.6899	0.6835	0.6882	638.90	4,291.40
A16	0.5723	0.5556	0.5643	611.50	6,777.90
A17	0.5728	0.5421	0.5527	699.80	4,263.40
A18	0.4194	0.4063	0.4131	571.70	3,382.80
A19	0.5000	0.4714	0.4825	436.40	2,949.00
A20	0.7791	0.7711	0.7772	430.90	5,040.90
A21	0.5759	0.5619	0.5674	615.80	7,409.80
A22	1.0000	0.9624	0.9858	280.50	6,200.90
A23	0.8511	0.8322	0.8430	336.80	7,406.60
A24	0.7351	0.7143	0.7285	388.30	8,480.90
A25	0.5287	0.5031	0.5113	481.80	10,856.40
A26	0.4641	0.4403	0.4537	474.20	10,529.50
A27	0.4403	0.4268	0.4333	548.30	11,977.90
A28	0.5469	0.5413	0.5426	507.70	8,679.80
A29	0.4582	0.4449	0.4511	533.00	19,345.20
A30	0.6118	0.5915	0.5953	578.30	20,347.90
A31	0.5829	0.5526	0.5704	648.90	32,695.40
A32	0.4974	0.4697	0.4793	536.80	25,386.80
A33	0.4589	0.4400	0.4512	741.20	92,207.80
A34	0.5953	0.5916	0.5943	452.20	46,433.70
A35	0.8403	0.8268	0.8330	532.40	161,005.60

Tabla 16. Resultados de Algoritmo Iterated Greedy (d=2 x num. células). Data B

IG d=2x					
PROBLEMA	**MAX**	**MIN**	**PROM**	**P.ITER**	**P.MILISECS**
B1	0.8095	0.8095	0.8095	17.40	3.20
B2	0.7222	0.7222	0.7222	36.50	4.30
B3	0.6071	0.6071	0.6071	166.10	42.00
B4	0.8889	0.8889	0.8889	6.40	0.00
B5	0.7500	0.7500	0.7500	100.70	23.70
B6	0.7391	0.7391	0.7391	25.60	4.60
B7	0.8148	0.8148	0.8148	19.20	6.40
B8	0.7222	0.7143	0.7198	394.00	130.60
B9	0.7576	0.7576	0.7576	81.30	27.10
B10	0.9000	0.9000	0.9000	38.00	12.60
B11	0.7273	0.7143	0.7213	220.00	163.40
B12	0.8276	0.8276	0.8276	25.30	9.30
B13	0.5962	0.5962	0.5962	246.40	175.80
B14	0.6404	0.6404	0.6404	399.30	1,332.80
B15	0.8333	0.8333	0.8333	98.90	66.90
B16	0.7391	0.7391	0.7391	79.50	144.90
B17	0.6552	0.6429	0.6498	386.30	661.20
B18	0.6129	0.6111	0.6113	478.70	816.20
B19	0.8000	0.8000	0.8000	41.80	36.90
B20	0.8710	0.8710	0.8710	130.20	277.10
B21	0.8333	0.8333	0.8333	126.70	279.40
B22	0.7258	0.7258	0.7258	369.40	829.50
B23	0.8111	0.8111	0.8111	156.80	607.00
B24	0.5600	0.5340	0.5516	503.20	2,736.30
B25	0.7600	0.7556	0.7596	390.00	496.60
B26	0.6034	0.5983	0.6015	612.80	12,947.30
B27	0.7248	0.7014	0.7207	385.90	6,083.50
B28	0.6729	0.6462	0.6631	539.80	13,653.20
B29	0.5600	0.5391	0.5519	783.90	4,920.60
B30	0.6994	0.6364	0.6692	483.80	47,901.30
B31	0.6786	0.6708	0.6745	482.00	229,854.50
B32	0.3129	0.3124	0.3127	556.60	572,147.30

4.4 Comparación

Para comenzar la comparación entre los algoritmos, se van a disponer en una tabla los mejores resultados de cada uno de los algoritmos, señalando en color el mejor resultado entre todos los métodos utilizados.

Tabla 17. Comparativo de resultados entre algoritmos. Data A

	ALGORITMOS						
PROBLEMA	**SC1**	**SC2**	**SC1-1000**	**SC2-1000**	**IG-d=0,7x**	**IG-d=1x**	**IG-d=2x**
A01	0.8235	0.8235	0.8235	0.8235	0.8235	0.8235	0.8235
A02	0.6957	0.6957	0.6957	0.6957	0.6957	0.6957	0.6957
A03	0.7959	0.7959	0.7959	0.7959	0.7959	0.7959	0.7959
A04	0.7692	0.7692	0.7692	0.7692	0.7692	0.7692	0.7692
A05	0.6087	0.6087	0.6087	0.6087	0.5862	0.6087	0.6087
A06	0.7083	0.7083	0.7083	0.7083	0.7083	0.7083	0.7083
A07	0.6944	0.6944	0.6944	0.6944	0.6829	0.6944	0.6944
A08	0.8525	0.8525	0.8525	0.8525	0.8525	0.8525	0.8525
A09	0.5833	0.5872	0.5841	0.5872	0.5688	0.5688	0.5872
A10	0.7500	0.7500	0.7500	0.7500	0.7500	0.7500	0.7500
A11	0.9200	0.9200	0.9200	0.9200	0.9200	0.9200	0.9200
A12	0.7206	0.6957	0.7206	0.7206	0.7206	0.7206	0.7206
A13	0.7183	0.6901	0.7183	0.7183	0.7183	0.7183	0.7183
A14	0.5275	0.4796	0.5326	0.5269	0.5326	0.5326	0.5217
A15	0.6738	0.6423	0.6899	0.6899	0.6899	0.6899	0.6899
A16	0.5714	0.5414	0.5688	0.5688	0.5743	0.5732	0.5723
A17	0.5729	0.5484	0.5699	0.5714	0.5743	0.5773	0.5728
A18	0.4218	0.4203	0.4326	0.4275	0.4306	0.4286	0.4194
A19	0.4853	0.4615	0.5000	0.5000	0.5040	0.5040	0.5000
A20	0.7736	0.7237	0.7785	0.7750	0.7791	0.7791	0.7791
A21	0.5714	0.5426	0.5707	0.5619	0.5798	0.5798	0.5759
A22	0.9470	0.7971	0.9542	0.9618	1.0000	1.0000	1.0000
A23	0.8116	0.6966	0.8102	0.8333	0.8511	0.8511	0.8511
A24	0.7067	0.6054	0.7192	0.7153	0.7351	0.7351	0.7351
A25	0.5274	0.4524	0.5248	0.5133	0.5329	0.5329	0.5287
A26	0.4615	0.4035	0.4805	0.4636	0.4859	0.4795	0.4641
A27	0.4305	0.3758	0.4503	0.4323	0.4658	0.4636	0.4403
A28	0.5406	0.5320	0.5415	0.5452	0.5482	0.5482	0.5469
A29	0.4522	0.4038	0.4612	0.4504	0.4656	0.4664	0.4582
A30	0.6111	0.5342	0.6190	0.6039	0.6267	0.6218	0.6118
A31	0.5439	0.4819	0.5805	0.5556	0.5965	0.5966	0.5829
A32	0.4844	0.3981	0.5026	0.4511	0.5083	0.5053	0.4974
A33	0.4444	0.3821	0.4652	0.4409	0.4792	0.4755	0.4589
A34	0.5920	0.5637	0.5959	0.5913	0.5953	0.5953	0.5953
A35	0.7741	0.3233	0.8132	0.7532	0.8403	0.8403	0.8403

Tabla 18. Comparativo de resultados entre algoritmos. Data B

	ALGORITMOS						
PROBLEMA	**SC1**	**SC2**	**SC1-1000**	**SC2-1000**	**IG-d=0,7x**	**IG-d=1x**	**IG-d=2x**
B1	0.8095	0.8095	0.8095	0.8095	0.5556	0.8095	0.8095
B2	0.7222	0.7222	0.7222	0.7222	0.7222	0.7222	0.7222
B3	0.6071	0.6071	0.6071	0.6071	0.5862	0.6071	0.6071
B4	0.8889	0.8889	0.8889	0.8889	0.8889	0.8889	0.8889
B5	0.7500	0.7500	0.7500	0.7500	0.7500	0.7500	0.7500
B6	0.7391	0.7391	0.7391	0.7391	0.7391	0.7391	0.7391
B7	0.8148	0.8148	0.8148	0.8148	0.8148	0.8148	0.8148
B8	0.7222	0.7222	0.7222	0.7222	0.7222	0.7143	0.7222
B9	0.7576	0.7576	0.7576	0.7576	0.7576	0.7576	0.7576
B10	0.9000	0.9000	0.9000	0.9000	0.9000	0.9000	0.9000
B11	0.7273	0.7273	0.7273	0.7273	0.7273	0.7273	0.7273
B12	0.8276	0.8276	0.8276	0.8276	0.8276	0.8276	0.8276
B13	0.5962	0.5962	0.5962	0.5962	0.5962	0.5962	0.5962
B14	0.6333	0.6000	0.6404	0.6264	0.6404	0.6404	0.6404
B15	0.8333	0.8333	0.8333	0.8333	0.8333	0.8333	0.8333
B16	0.7356	0.7391	0.7391	0.7391	0.7391	0.7391	0.7391
B17	0.6552	0.6441	0.6552	0.6552	0.6552	0.6552	0.6552
B18	0.6027	0.6111	0.6111	0.6129	0.6111	0.6129	0.6129
B19	0.8000	0.8000	0.8000	0.8000	0.8000	0.8000	0.8000
B20	0.8710	0.7778	0.8710	0.8710	0.8710	0.8710	0.8710
B21	0.8333	0.8333	0.8333	0.8333	0.8333	0.8333	0.8333
B22	0.7258	0.6885	0.7258	0.7258	0.7258	0.7258	0.7258
B23	0.7816	0.7386	0.8111	0.8111	0.8111	0.8111	0.8111
B24	0.5660	0.5446	0.5673	0.5673	0.5673	0.5673	0.5600
B25	0.7600	0.7556	0.7600	0.7600	0.7600	0.7600	0.7600
B26	0.6017	0.5605	0.6029	0.5936	0.6034	0.6068	0.6034
B27	0.6712	0.6107	0.6980	0.7192	0.7248	0.7248	0.7248
B28	0.6459	0.5268	0.6588	0.6202	0.6729	0.6729	0.6729
B29	0.5657	0.5259	0.5566	0.5631	0.5700	0.5730	0.5600
B30	0.6805	0.5843	0.6975	0.6591	0.7290	0.6981	0.6994
B31	0.6477	0.4561	0.6645	0.6034	0.6798	0.6798	0.6786
B32	0.3040	0.2577	0.6056	0.2907	0.6184	0.6134	0.3129

Se calcula, por cada problema, el mejor resultado de todos los métodos, para posteriormente calcular un ratio consistente en la división, en cada método, del total de coincidencias con el mejor obtenido dividido por el total de problemas de cada batería. De esta forma podemos ver cuáles son los métodos con los que se obtienen mejores respuestas.

Tabla 19. Ratio mejor resultado para cada algoritmo

	ALGORITMOS						
PROBLEMA	**SC1**	**SC2**	**SC1-1000**	**SC2-1000**	**IG-d=0.7x**	**IG-d=1x**	**IG-d=2x**
DATA A	34.29%	31.43%	45.71%	40.00%	77.14%	74.29%	54.29%
DATA B	62.50%	53.13%	75.00%	75.00%	84.38%	90.63%	81.25%

Por otra parte, se puede ver cómo se comporta en función del tiempo que tarda en resolver el problema cada uno de los procedimientos.

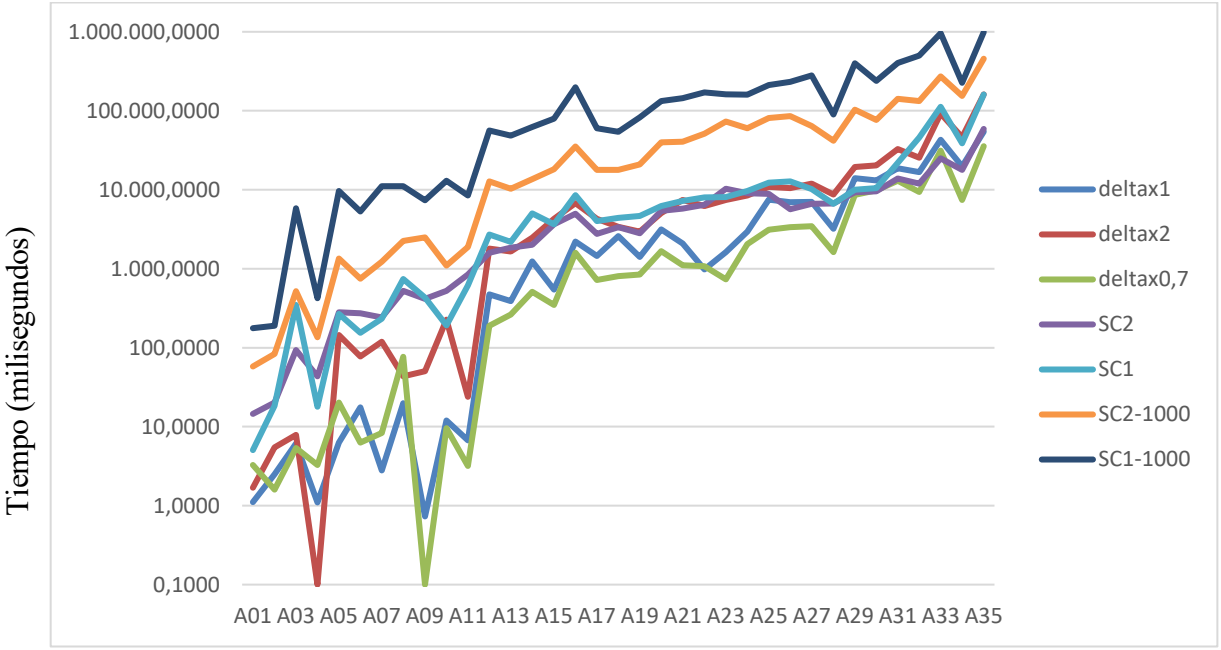


Figura 10. Gráfica de tiempos de ejecución por algoritmo y problema. Data A.

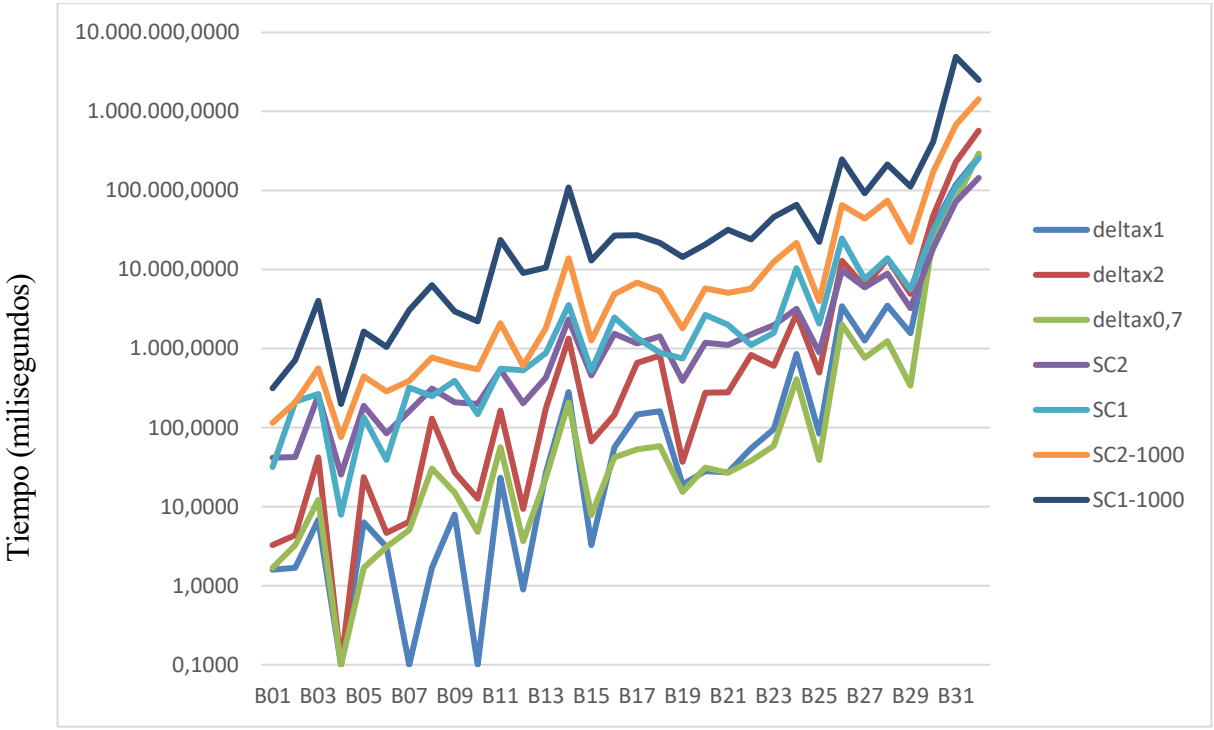


Figura 11. Gráfica de tiempos de ejecución por algoritmo y problema. Data B.

Además, otro aspecto importante consiste en la dispersión de los resultados obtenidos en las 10 pruebas realizadas con cada problema. Para ello se adjuntan dos tablas con la media de la desviación típica de cada una de las soluciones, a lo que se le ha llamado variabilidad en los resultados.

Tabla 20. Variabilidad entre las pruebas realizadas por problema y algoritmo. Data A

	ALGORITMOS						
PROBLEMA	**SC1**	**SC2**	**SC1-1000**	**SC2-1000**	**IG-d=0.7x**	**IG-d=1x**	**IG-d=2x**
A01	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
A02	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
A03	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
A04	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
A05	0.0154	0.0089	0.0071	0.0051	0.0000	0.0095	0.0071
A06	0.0109	0.0116	0.0085	0.0000	0.0024	0.0000	0.0000
A07	0.0253	0.0169	0.0160	0.0061	0.0000	0.0036	0.0061
A08	0.0508	0.0792	0.0000	0.0000	0.0000	0.0000	0.0000
A09	0.0044	0.0077	0.0057	0.0017	0.0000	0.0000	0.0058
A10	0.0237	0.0156	0.0191	0.0000	0.0000	0.0000	0.0000
A11	0.0456	0.0411	0.0540	0.0208	0.0000	0.0000	0.0000
A12	0.0295	0.0434	0.0174	0.0101	0.0018	0.0013	0.0115
A13	0.0202	0.0285	0.0157	0.0096	0.0039	0.0039	0.0092
A14	0.0060	0.0315	0.0146	0.0064	0.0046	0.0043	0.0065
A15	0.0298	0.0595	0.0184	0.0124	0.0004	0.0005	0.0019
A16	0.0213	0.0366	0.0151	0.0294	0.0009	0.0047	0.0066
A17	0.0081	0.0340	0.0065	0.0090	0.0053	0.0064	0.0097
A18	0.0117	0.0158	0.0089	0.0071	0.0085	0.0055	0.0046
A19	0.0144	0.0170	0.0147	0.0115	0.0096	0.0080	0.0079
A20	0.0407	0.0453	0.0262	0.0098	0.0019	0.0011	0.0031
A21	0.0258	0.0563	0.0182	0.0190	0.0048	0.0037	0.0044
A22	0.0689	0.0920	0.0661	0.0690	0.0000	0.0000	0.0185
A23	0.0551	0.0528	0.0348	0.0315	0.0000	0.0000	0.0087
A24	0.0341	0.0545	0.0448	0.0351	0.0018	0.0025	0.0075
A25	0.0271	0.0331	0.0255	0.0259	0.0031	0.0041	0.0078
A26	0.0141	0.0232	0.0123	0.0119	0.0071	0.0053	0.0069
A27	0.0129	0.0270	0.0105	0.0132	0.0076	0.0054	0.0042
A28	0.0281	0.0210	0.0052	0.0071	0.0001	0.0021	0.0018
A29	0.0204	0.0198	0.0104	0.0115	0.0068	0.0088	0.0035
A30	0.0162	0.0291	0.0228	0.0625	0.0041	0.0087	0.0061
A31	0.0152	0.0566	0.0167	0.0209	0.0077	0.0145	0.0094
A32	0.0198	0.0238	0.0178	0.0167	0.0055	0.0059	0.0092
A33	0.1087	0.0678	0.0717	0.0735	0.0076	0.0081	0.0058
A34	0.0130	0.0093	0.0221	0.0122	0.0006	0.0006	0.0011
A35	0.1681	0.0329	0.1251	0.1625	0.0023	0.0000	0.0052
PROMEDIO	0.0281	0.0312	0.0215	0.0203	0.0028	0.0034	0.0051

Tabla 21. Variabilidad entre las pruebas realizadas por problema y algoritmo. Data B

	ALGORITMOS						
PROBLEMA	**SC1**	**SC2**	**SC1-1000**	**SC2-1000**	**IG-d=0,7x**	**IG-d=1x**	**IG-d=2x**
B1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
B2	0.0084	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
B3	0.0104	0.0096	0.0023	0.0000	0.0000	0.0000	0.0000
B4	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
B5	0.0081	0.0263	0.0000	0.0000	0.0000	0.0000	0.0000
B6	0.0242	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
B7	0.0144	0.0192	0.0000	0.0000	0.0000	0.0000	0.0000
B8	0.0069	0.0071	0.0033	0.0000	0.0042	0.0000	0.0038
B9	0.0222	0.0282	0.0000	0.0000	0.0000	0.0000	0.0000
B10	0.0634	0.0281	0.0000	0.0000	0.0000	0.0000	0.0000
B11	0.0060	0.0133	0.0029	0.0049	0.0039	0.0000	0.0052
B12	0.0407	0.0054	0.0000	0.0000	0.0000	0.0000	0.0000
B13	0.0199	0.0072	0.0166	0.0014	0.0000	0.0000	0.0000
B14	0.0413	0.0373	0.0207	0.0103	0.0474	0.0000	0.0000
B15	0.0314	0.0284	0.0000	0.0000	0.0000	0.0516	0.0000
B16	0.0162	0.0276	0.0149	0.0032	0.0015	0.0000	0.0000
B17	0.0143	0.0353	0.0163	0.0028	0.0052	0.0022	0.0038
B18	0.0157	0.0219	0.0100	0.0044	0.0144	0.0006	0.0006
B19	0.0262	0.0568	0.0203	0.0000	0.0000	0.0000	0.0000
B20	0.0568	0.0243	0.0565	0.0312	0.0000	0.0000	0.0000
B21	0.0405	0.0539	0.0299	0.0212	0.0000	0.0000	0.0000
B22	0.0490	0.0453	0.0291	0.0111	0.0000	0.0000	0.0000
B23	0.0368	0.0707	0.0347	0.0225	0.0000	0.0000	0.0000
B24	0.0156	0.0330	0.0040	0.0099	0.0023	0.0022	0.0076
B25	0.0164	0.0695	0.0106	0.0024	0.0000	0.0014	0.0014
B26	0.0297	0.0400	0.0128	0.0202	0.0008	0.0012	0.0014
B27	0.0377	0.0454	0.0286	0.0264	0.0304	0.0428	0.0089
B28	0.0357	0.0425	0.0269	0.0345	0.0000	0.0053	0.0099
B29	0.0174	0.0365	0.0098	0.0081	0.0053	0.0056	0.0057
B30	0.0581	0.1752	0.0285	0.1861	0.0135	0.0130	0.0187
B31	0.1742	0.0660	0.1142	0.1443	0.0030	0.0007	0.0027
B32	0.0064	0.0185	0.0979	0.0281	0.0082	0.1243	0.0002
PROMEDIO	0.0295	0.0335	0.0185	0.0179	0.0044	0.0078	0.0022

Se ha finalizado con la exposición de todos los resultados correspondientes a los cálculos obtenidos a través de los procedimientos utilizados, de los problemas propuestos. Para ayudar al análisis y conclusiones del capítulo posterior se han realizado algunas gráficas y tablas de comparación entre métodos.

5 CONCLUSIÓN

La vida es el arte de sacar conclusiones suficientes
a partir de datos insuficientes.

- Samuel Butler -

Para finalizar este documento tras haber expuesto los cálculos y las diversas comparaciones se pueden llegar a varias conclusiones observando los resultados:

- Teniendo en cuenta los mejores resultados de cada uno de los algoritmos en las tablas 17 y 18 se puede afirmar que la función objetivo suele tomar mejores valores finales en la metaheurística Iterated Greedy propuesta que con las variaciones de la metaheurística de selección clonal. Concretamente para los problemas del set Data A cuando el valor del parámetro delta de esta, toma el valor 0,7 por el número de células posibles con los datos iniciales del problema. Y para los problemas del set Data B cuando se corresponde directamente con el número de células posibles.

  Otra característica por recalcar es que las soluciones en los problemas Data B son más homogéneas en todos los algoritmos, solamente se encuentran más diferencias en los problemas de gran tamaño.

- Se puede notificar que en el set de problemas Data B, los resultados son similares en todos los algoritmos y se va notando más diferencia en los más grandes, sin embargo, en el set Data A los cambios comienzan en problemas de tamaño menor.

- Por otro lado, si se observa el tiempo que tardan en realizar los cálculos, se deduce que los métodos que menos tiempo invierten en los cálculos son el algoritmo Iterated Greedy con los multiplicadores 0,7 y 1. Por el contrario, las variaciones del algoritmo de selección clonal son más lentos y aumenta el tiempo conforme el tamaño de la población que se usa.

- El presentar la variabilidad de los resultados calculados ayuda a saber si para un problema se pueden obtener efectos muy diversos, lo que puede significar que el factor aleatorio para ese problema y ese algoritmo sea más alto. Como consecuencia tendríamos que realizar más pruebas para conseguir una mejor solución. Si nos fijamos en la tabla 20 para los problemas Data A el método por el que se consigue menor variabilidad es el Iterated Greedy para el valor de delta con el multiplicador 0,7 y en la tabla 21, Data B, para el valor de delta con el multiplicador 2.

Con todas estas conclusiones, podemos observar que la línea de investigación de metaheurísticas no poblacionales como el caso del Iterated Greedy puede dar buenos resultados para el problema de formación de células de producción.

Este tipo de estudios reflejan la cantidad de datos a la que se enfrentan los entornos de fabricación para proporcionar soluciones buenas y rápidas. Existen numerosas metodologías y modelos para resolver el problema de la formación celular, quizás para futuros estudios sería interesante un enfoque para resolver problemas con datos reales, porque muchos de los softwares existentes tienen potencial para ser usados en aplicaciones industriales.

REFERENCIAS

Adil, G. K., Rajamani, D., & Strong, D. (1996) 'Cell formation considering alternate routings', *International Journal of Production Research*, 34(5), pp. 1361–1380.

Askin, R. G., & Subramanian, S. (1987) 'A cost-based heuristic for group technology configuration', *International Journal of Production Research*, 25, pp. 101–113.

Boctor, F. A. (1991) 'Linear formulation of the machine-part cell formation problem.', *International Journal of Production Research*, 29(2), pp. 343–356.

Boe, W., & Cheng, C. H. (1991) 'A close neighbor algorithm for designing cellular manufacturing systems', *International Journal of Production Research,* 29(10), pp. 2097–2116.

Brown, E., & Sumichrast, R. (2001) 'CF-GGA: a grouping genetic algorithm for the cell formation problem', *International Journal of Production Research*, 36, pp. 3651–3669.

Burnet, F. M. (1959) *The Clonal Selection Theory of Acquired Immunity*. University Press.

Bychkov, I. and Batsyn, M. (2017) 'An efficient exact model for the cell formation problem with a variable number of production cells', *Computers and Operations Research*, 91, pp. 112–120. doi: 10.1016/j.cor.2017.11.009.

Carrie, S. (1973) 'Numerical taxonomy applied to group technology & plant layout', *International Journal of Production Research*, 11, pp. 399–416.

De Castro, L. N. and Von Zuben, F. J. (2002) 'Learning and optimization using the clonal selection principle', *IEEE Transactions on Evolutionary Computation*, 6(3), pp. 239–251. doi: 10.1109/TEVC.2002.1011539.

Chan, H. M., & Milner, D. A. (1982) 'Direct clustering algorithm for group formation in cellular manufacture', *Journal of Manufacturing System*, 1, pp. 65–75.

Chandrasekaran, M. P., & Rajagopalan, R. (1986a) 'An ideal seed nonhierarchical clustering algorithm for cellular manufacturing', *International Journal of Production Research*, 24, pp. 451–463.

Chandrasekaran, M. P., & Rajagopalan, R. (1986b) 'MODROC: An extension of rank order clustering of group technology', *International Journal of Production Research*, 24(5), pp. 1221–1233.

Chandrasekharan, M. P.,&Rajagopalan, R. (1987) 'ZODIAC: An algorithm for concurrent formation of part-families and machine-cells', *International Journal of Production Research*, 24(2), pp. 835–850.

Chandrasekharan, M. P.,&Rajagopalan, R. (1989) 'Groupability: Analysis of the properties of binary datamatrices for group technology', *International Journal of Production Research*, 27(6), pp. 1035–1052.

Colin, R. (1993) 'Improving the efficiency of tabu search for machine sequencing problems', *Journal of the Operational Research Society*. Taylor & Francis, 44(4), pp. 375–382. doi: 10.1057/jors.1993.67.

Cuatrecasas Arbós, L. (2012) *Organización de la producción y dirección de operaciones: Sistemas actuales de gestión eficiente y competitiva*. Ediciones Díaz de Santos.

D´ıaz, A., Glover, F., Ghaziri, H.M., Gonzalez, J.L., Laguna, M, Moscato, P. y T. and F.T. (1996)

Optimización heurística y redes neuronales. Madrid: Paraninfo.

Dasgupta, D., Ji, Z. and Gonzalez, F. (2003) 'Artificial immune system (AIS) research in the last five years', in *2003 Congress on Evolutionary Computation, CEC 2003 - Proceedings*. IEEE Computer Society, pp. 123–130. doi: 10.1109/CEC.2003.1299565.

Dimopoulos, C. and Zalzala, A. M. S. (2000) 'Recent developments in evolutionary computation for manufacturing optimization: Problems, solutions, and comparisons', *IEEE Transactions on Evolutionary Computation*, 4(2), pp. 93–113. doi: 10.1109/4235.850651.

García Díaz, R. (2002) *Formación de células de fabricación con múltiples planes de proceso para cada pieza*. Sevilla.

Goldberg, D. E. (1989) *Goldberg Genetic Algorithms in Search, Optimization & Machine Learning*. Edited by Addison-Wesley.

Gonçalves, J. F. and Resende, M. G. C. (2004) 'An evolutionary algorithm for manufacturing cell formation *'. doi: 10.1016/j.cie.2004.07.003.

HamI., Hitomi K., Y. (1985) *Layout planning for group technology. In Group technology*, *International series in management science/operations research*. Springer US.

Harhalakis, G., Ioannou, G., Minis, I., & Nagi, R. (no date) 'Manufacturing cell formation under random product demand', *International Journal of Production Research*, 32(1), pp. 47–64.

Islam, K. M. S.,&Sarker, B. R. (2000) 'A similarity coefficient measure and machine parts grouping in cellular manufacturing systems', *International Journal of Production Research*, 38(3), pp. 699–720.

Johnson, S. M. (1954) 'Optimal two- and three-stage production schedules with setup times included', *Naval Research Logistics Quarterly*. Wiley, 1(1), pp. 61–68. doi: 10.1002/nav.3800010110.

Kareem Sakran, H., Majeed Mahbuba, H. and Saleh Jafer, A. (2016) 'a Review of a Basic Concept of Cellular Manufacturing', *International Journal of Design and Manufacturing Technology*, 87682(81), pp. 30–37.

King, J. R., & Nakornchai, V. (1982) 'Machine-component group formation in group technology: Review and extension', *International Journal of Production Research*, 20(2), pp. 117–133.

King, J. R. (1980) 'Machine-component grouping in production flow analysis: An approach using a rank order clustering algorithm', *International Journal of Production Research*, 18(2), pp. 213–232.

Kumar, K. R., Kusiak, A., & Vannelli, A. (1986) 'Grouping of parts and components in flexible manufacturing systems', *European Journal of Operations Research*, 24, pp. 387–397.

Kumar, K. R., & V. (1987) 'Strategic subcontracting for efficient disaggregated manufacturing', *International Journal of Production Research*, 25(12), pp. 1715–1728.

Kusiak, A., & Cho, M. (1992) 'Similarity coefficient algorithm for solving the group technology problem', *International Journal of Production Research*, 30, pp. 2633–2646.

Kusiak, A., & Chow, W. (1987a) 'Efficient solving of the group technology problem. Journal of Manufacturing Systems', 6(2), pp. 117–124.

Kusiak, A., & Chow, W. (1987b) 'Efficient solving of the group technology problem', *Journal of Manufacturing Systems*, 6(2), pp. 117–124.

Li, M. L. (2003) 'The algorithm for integrating all incidence matrices in multidimensional group technology', *International Journal Production Economics*, 86, pp. 121–131.

Martí, R. (2001) 'Procedimientos Metaheurísticos en Optimización Combinatoria', *Departament d'Estadística i Investigació Operativa*, pp. 1–60.

McCormick, W. T., Schweitzer, P. J., & White, T. W. (1972) 'Problem decomposition and data

reorganization by a clustering technique', *Operations Research*, 20, pp. 993–1009.

Moon, Y. B., & Chi, S. C. (1992) 'Generalized part family formation using neural network techniques', *Journal of Manufacturing Systems*, 11(3), pp. 149–159.

Mosier, C. T.,&Taube, L. (1985) 'Weighted similarity measure heuristics for the group technology machine clustering problem', *Omega*, 13(6), pp. 577–583.

Mosier, C. T., & Taube, L. (1985) 'The facets of group technology & their impact on implementation', *Omega*, 13(6), pp. 381–391.

Nagi, R., Harhalakis, G., & Proth, J. M. (1990) 'Multiple routeings and capacity considerations in group technology applications', *International Journal of Production Research*, 28(12), pp. 1243–1257.

Nair, G. J. (1999) 'Accord: A bicriterion algorithm for cell formation using ordinal and ratio level data', *International Journal of Production Research*, 37(3), pp. 539–556.

Nawaz, M., Enscore, E. E. and Ham, I. (1983) 'A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem', *Omega*, 11(1), pp. 91–95. doi: 10.1016/0305-0483(83)90088-9.

Osman, I.H. and Kelly, J. P. (1996) *Meta-Heuristics: Theory and Applications*. Boston: Ed. Kluwer Academic.

Osman, I. and Potts, C. (1989) 'Simulated annealing for permutation flow-shop scheduling', *Omega*, 17(6), pp. 551–557. doi: 10.1016/0305-0483(89)90059-5.

Pa Rkin, R. E., & Li, M. L. (1997) 'The multi-dimensional aspects of a group technology algorithm', *International Journal of Production Research*, 35(8), pp. 2345–2358.

Pérez González, P. (2014) *Programación y Control de la Producción*. Universidad de Sevilla.

Popovic, D. (2018) *Off-site manufacturing systems development in timber house building Towards mass customization-oriented manufacturing.*

Quesada Campos, J. D. (2019) *Concepto de Manufactura Celular*, *LinkedIn*. Available at: https://www.linkedin.com/pulse/concepto-de-manufactura-celular-josé-david-quesada-campos/?originalSubdomain=es (Accessed: 15 July 2020).

Riba, C. *et al.* (2006) 'Familia, portafolio y gama de productos ', *IIngeniería Concurrente: Una metodología integradora*, (July), pp. 37–47.

Ruiz, R. and Stützle, T. (2007) 'A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem', *European Journal of Operational Research*, 177(3), pp. 2033–2049. doi: 10.1016/j.ejor.2005.12.009.

Sandbothe, R. A. (1998) 'Two observations on the grouping efficacy measure for goodness of block diagonal forms', *International Journal of Production Research*, 36(11), pp. 3217–3222.

Sarker, B. R., & Khan, M. (2001) 'A comparison of existing grouping eciency measures and a new weighted grouping eciency measure', *IIE Transactions*, (33), pp. 11–27.

Seifoddini, H.,&Djassemi, M. (1991) 'The production data based similarity coefficient versus Jaccard's similarity coefficient', *Computers Industrial Engineering*, 21, pp. 263–266.

Seifoddini, H., & Djassemi, M. (1996) 'The threshold value of a quality index for formation of cellular manufacturing systems', *International Journal of Production Research*, 34(12), pp. 3401–3416.

Seifoddini, H., & Wolfe, P. M. (1986) 'Application of the similarity coefficient method in group technology', *IIE Transactions*, 18(3), pp. 266–270.

Seifoddini, H. (1989) 'Single linkage versus average linkage clustering in machine cells formation applications', *Computers and Industrial Engineering*, 16(3), pp. 419–426.

Shargal, M., Shekhar, S., & Irani, S. A. (1995) 'Evaluation of search algorithms and clustering efficiency measures for machine-part matrix clustering', *IIE Transactions*, 27(1), pp. 43–59.

Srinivasan, G., Narendran, T., &Mahadevan, B. (1990) 'An assignment model for the part-families problem in group technology', *International Journal of Production Research*, 28(1), pp. 145–152.

Stanfel, L. (1985) 'Machine clustering for economic production', *Engineering Economics, Costs and Production*, 9, pp. 73–78.

Suresh Kumar, C. and Chandrasekharan, M. P. (1990) 'Grouping efficacy: A quantitative criterion for goodness of block diagonal forms of binary matrices in group technology', *International Journal of Production Research*. Taylor & Francis Group , 28(2), pp. 233–243. doi: 10.1080/00207549008942706.

Taillard, E. (1990) 'Some efficient heuristic methods for the flow shop sequencing problem', *European Journal of Operational Research*. North-Holland, 47(1), pp. 65–74. doi: 10.1016/0377-2217(90)90090-X.

Talbi, E.-G. (2009) *Metaheuristics, from design to implementation*, *Foreign Affairs*. doi: 10.1017/CBO9781107415324.004.

Ulutas, B. H. (2019) 'An immune system based algorithm for cell formation problem', *Journal of Intelligent Manufacturing*. Springer US, 30(8), pp. 2835–2852. doi: 10.1007/s10845-018-1407-x.

Viswanathan, S. (1996) 'A new approach for solving the P-median problem in group technology', *International Journal of Production Research*, 34(10), pp. 2691–2700.

Waghodekar, P. H., & Sahu, S. (1984) 'Machine-component cell formation in group technology, MACE', *International Journal of Production Research*, 22, pp. 937–948.

Widmer, M. and Hertz, A. (1989) 'A new heuristic method for the flow shop sequencing problem', *European Journal of Operational Research*. North-Holland, 41(2), pp. 186–193. doi: 10.1016/0377-2217(89)90383-4.

Won, Y., & Kim, S. (1997) 'Multiple criteria clustering algorithm for solving the group technology problem with multiple process routings', *Computers and Industrial Engineering*, 32(1), pp. 207–220.

Yang, M. S.,&Yang, J. H. (2008) 'Machine-part cell formation in group technology using a modified ART1 method', *European Journal of Operations Research*, 188(1), pp. 140–152.

Zolfaghari, S., & Liang, M. (1997) 'An objective-guided orthosynapse Hopfield network approach to machine grouping problems', *International Journal of Production Research*, 35(10), pp. 2773–2792.

Zolfaghari, S., & Liang, M. (2002) 'Comparative study of simulated annealing, genetic algorithms and tabu search for binary and comprehensive machine-grouping problems', *International Journal of Production Research*, 40, pp. 2141–215.

Zuben, L. N. de C. and F. J. V. (2001) 'The clonal selection algorithm with engineering applications. In Workshop on Artificial Immune Systems and Their Applications"'.

ANEXO A. MATRICES RESULTADO DATA A

Se representa la matriz del mejor resultado obtenido para cada problema.

A1 (SC1)

	7	9	10	11	6	8	12
1	1	1	1	1	0	0	0
4	1	1	0	1	0	0	0
2	0	0	0	0	1	1	0
3	0	0	0	0	1	1	1
5	0	0	0	0	1	0	1

A2 (SC1)

	6	11	12	7	8	9	10
1	1	1	1	0	0	0	1
2	0	0	0	1	1	1	1
3	0	1	0	0	1	1	1
4	1	0	0	1	1	1	0
5	0	1	0	1	0	1	1

A3 (SC1)

	9	12	14	15	20	23	6	7	8	10	11	13	16	17	18	19	21	22
2	1	1	0	1	1	1	1	0	1	0	1	1	1	1	1	0	0	0
3	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
5	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
4	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

A4 (SC1)

	10	13	7	8	9	11	12	14
1	1	1	0	1	0	0	0	0
4	1	1	0	0	0	0	0	0
6	1	1	0	0	0	0	0	0
2	0	1	1	1	1	1	1	1
3	0	0	0	0	1	0	1	1
5	0	0	1	0	1	1	1	1

A5 (SC1)

	12	18	9	14	10	13	8	11	15	16	17
2	1	1	0	0	0	0	1	0	0	0	0
5	1	0	0	0	0	0	0	0	1	0	0
3	0	1	0	0	0	0	0	0	0	0	1
1	0	0	1	1	1	0	0	0	0	0	0
4	0	0	0	0	1	1	1	0	0	0	0
7	0	0	0	1	1	1	0	1	0	1	0
6	0	0	0	0	0	0	1	1	1	1	1

A6 (SC1)

	11	18	12	15	17	8	10	14	9	13	16
6	1	1	0	0	0	0	1	0	0	0	0
4	1	0	1	0	1	0	0	0	0	0	0
7	0	0	1	1	1	0	0	0	0	0	0
1	0	1	0	0	0	1	1	1	0	0	0
5	0	0	0	0	0	0	1	1	0	0	0
2	0	0	0	0	0	1	0	0	1	1	0
3	0	0	0	0	0	0	0	0	1	1	1

A7 (SC1)

	15	16	17	18	11	12	13	14	9	10	19	20
4	1	1	1	1	0	0	0	1	0	0	0	0
5	1	1	1	1	0	0	0	0	0	0	0	0
6	1	1	0	1	0	0	0	0	0	0	1	0
2	1	0	0	1	1	1	1	1	1	0	0	0
3	1	1	1	0	1	1	1	1	0	0	0	0
1	0	0	0	0	1	1	0	0	1	1	0	0
7	0	0	0	0	0	0	0	0	0	0	1	1
8	0	0	0	0	0	0	0	0	0	0	1	1

A8 (SC1)

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

A9 (SC2)

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

A10 (SC1)

	13	18	11	14	12	17	19	20	15	16
5	1	0	0	0	0	0	0	0	0	0
6	1	1	0	0	0	0	0	0	0	0
9	0	1	0	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0	0	0	1
2	0	0	0	0	0	0	1	1	0	0
3	0	0	0	0	1	1	1	1	0	0
4	0	0	0	0	1	0	1	1	0	0
8	0	0	0	0	1	1	1	0	0	0
7	0	0	0	0	0	0	0	0	1	1
10	0	0	1	0	0	0	0	0	1	1

A11 (SC1)

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

A12 (SC1)

	15	20	24	26	28	29	30	16	17	31	33	34	37	36	38	25	27	21	22	23	32	18	19	35
6	1	0	1	1	1	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
8	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	1	1	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0
14	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	1	1	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

A13 (SC1)

	19	23	24	26	28	29	30	36	25	27	20	17	18	35	38	15	16	31	33	34	37	21	22	32
6	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	1	1	0	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	1	0	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0
2	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	1	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	0	1	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
13	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	1	1

A14 (SC1-1000)

	18	24	27	37	25	30	36	40	17	19	23	26	38	39	32	35	20	22	28	29	31	33	34	21
13	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	1	0	0	0	1	1	1	1	0	0	0	1	1	1	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
12	1	0	1	0	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
16	0	0	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	0	1	1	1	1	1	1	0	1	1	1	0	0	1	0	1	0
2	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	1	0	0
8	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	1	0	0	0	0
3	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	1	1	0	0	0	1	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
14	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0
6	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	1	1	0
9	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

A15 (SC1-1000)

	21	33	35	44	39	41	43	45	17	26	32	36	19	29	22	24	27	30	31	37	40	42	18	20	23	25	28	34	38	46
9	1	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	1	1	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
6	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	1	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0
13	0	0	0	0	0	0	0	1	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
5	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	1	0	0
14	0	1	0	0	0	0	1	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0	1	1	0	1	1	0	0	1	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	1	1	1
7	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	1	1	1
8	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	1	1	1	1	0	1
12	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	1	1	0

A16 (IG-d=0.7x)

	33	51	52	27	36	38	40	43	46	22	23	24	28	49	50	55	21	25	30	31	35	37	39	45	57	59	32	19	17	29	41	42	47	18	20	26	34	44	48	53	54	56	58
3	1	1	1	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	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
11	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	1	0	1	0	1	0	0	1	0	0	1	1	0	0	0	1	0	0	0	0	1	1	0	1	1
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	1	1	0	1	1	0	0	0	1	1	0	0	0	0	1	0	1	1	1	1	0	1	1	0	1	1	0	0	0	1	1	0	0	0	1	0	1	1	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	
13	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	1	1	1	1	1
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
16	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	1	1	1	0	1

A17 (IG-d=1x)

	24	19	21	42	22	38	41	29	34	37	40	25	31	32	36	39	20	23	26	27	28	30	33	35
7	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0
8	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
9	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
2	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0	0
13	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0	1	1	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0
12	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
15	0	1	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	1	1
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	1	1	1
4	1	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	1	1	1	0	1	1	1
5	0	0	0	1	1	0	1	1	1	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
6	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	1	1

A18 (SC1-1000)

	30	22	31	32	37	38	21	26	29	34	40	33	35	39	23	24	25	27	28	36
4	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
19	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
12	0	0	1	1	1	1	0	0	0	1	0	0	0	0	0	1	0	0	0	0
13	1	1	1	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	1	1	1	0	0	1	0	0	0	0	0	0	0	1
6	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	0	0
7	0	0	0	0	0	1	1	1	1	1	1	0	0	1	0	0	0	1	0	1
8	1	0	0	0	0	1	0	1	1	1	1	0	1	0	0	1	0	0	0	0
9	0	0	0	0	0	0	0	1	1	1	0	0	0	1	0	0	0	0	0	1
5	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0
10	1	0	1	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0
11	0	0	0	0	0	1	0	0	0	1	0	1	1	1	0	1	1	0	0	0
14	0	0	0	0	1	1	0	0	0	1	0	1	0	1	0	0	0	0	0	0
2	1	1	0	0	1	0	0	0	0	0	1	1	1	0	1	0	1	1	1	0
3	0	0	1	0	0	1	0	0	0	1	0	1	0	0	1	1	0	0	1	1
15	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	1	0	1	1
16	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	1
17	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	1	1	0	1
18	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	1
20	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	1	0	0	1	0

A19 (IG-d=1x)

	27	31	37	43	32	38	40	30	33	25	34	39	42	36	24	26	28	29	35	41
3	1	1	0	1	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1
12	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	1	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
22	1	1	1	1	1	1	0	0	1	0	0	0	0	0	1	1	0	1	0	0
18	0	0	0	0	1	1	1	0	0	0	0	1	0	1	0	1	0	0	1	0
19	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
21	1	0	0	0	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0
23	1	0	0	0	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0
13	0	1	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	1	0	1	1	1	0	1	1	0	0	0	1	0	0	0	0
8	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
17	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	1	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	0
1	1	1	1	0	0	1	0	0	0	0	0	0	1	1	1	1	1	1	1	1
2	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	1	1	0
10	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	0
15	0	1	1	1	0	0	1	0	0	1	0	0	0	1	1	1	1	1	1	1

A20 (IG-d=1x)

	28	34	39	42	46	36	53	54	55	21	23	25	35	37	40	43	45	49	22	27	30	32	33	38	44	47	51	24	26	29	31	41	48	50	52
6	1	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	1	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
12	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	1	1	1

A21 (IG-d=1x)

	28	34	39	42	43	46	24	26	29	31	41	48	50	52	21	23	25	35	37	40	49	22	27	30	32	33	44	47	51	36	38	45	53	54	55
5	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	1	1	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
9	1	1	1	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
10	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
20	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	1	1	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1
12	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
15	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	1	1	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0
19	0	0	0	0	0	0	1	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	1	1	1	1	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	1	0	1	0	1	0	1	0	0	0	0	0	0
8	0	0	0	0	1	0	0	0	1	0	0	0	0	0	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1
2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	1	1	1	0	1	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	1	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	1	1	1	0	1	0	0	0	0	0	0
1	0	1	0	0	1	0	0	1	0	1	0	0	1	1	1	0	0	1	0	1	0	0	0	0	0	0	1	0	0	0	1	1	0	1	1

A22 (IG-d=1x)

	32	43	45	52	61	62	63	26	35	36	39	47	48	55	58	28	29	42	50	51	54	25	33	40	41	57	34	37	38	46	59	60	27	49	56	30	31	44	53	64
4	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

A23 (IG-d=1x)

	32	43	45	52	61	62	63	34	37	38	46	59	60	30	31	44	53	64	25	33	40	41	57	28	29	42	50	51	54	27	49	56	26	35	36	39	47	48	55	58
4	1	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	1	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	1	0	1	1	0	0	0	0	0	0	0	0	1	0	0
15	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

A24 (IG-d=1x)

	34	37	38	46	59	60	32	43	45	52	61	62	63	26	35	36	39	47	48	55	58	30	31	44	53	64	25	33	40	41	57	28	29	42	50	51	54	27	49	56
2	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
5	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
11	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	1	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
3	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0
21	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	1	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	1	1	1	0	1	1	0	0	0
15	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	1	1	1	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1
14	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
24	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

A25 (IG-d=1x)

	28	29	42	50	51	54	26	36	39	47	58	34	37	38	46	59	60	49	27	56	30	53	63	64	33	57	32	35	43	45	52	61	62	44	48	25	40	41	31	55
6	0	0	1	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
8	1	1	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	1	1	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0
15	1	1	0	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0
18	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
20	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	1	0	0	0	0	0	1	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	1	1	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
23	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0
21	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0	0	0	0	0
16	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	1	0	1	1	0	1	0	0	0	0	0	0	0	0
10	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
13	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0
22	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1

A26 (IG-d=0,7x)

	35	49	52	37	38	46	59	60	26	36	39	47	58	29	50	54	62	25	40	34	30	33	44	48	31	41	55	32	43	45	61	63	53	57	64	27	56	28	42	51
4	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0
7	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0
14	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
5	0	0	0	0	0	1	1	1	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
3	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	1	1	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
8	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0
15	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1
13	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	0	1	1	0	0	0	0	1	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	1	0	0	0
9	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
23	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
24	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	0	0	0	0	0	0	0	0	0	1	1	0	0	0
12	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
18	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

A27 (IG-d=0,7x)

	37	38	51	64	33	34	57	39	46	47	58	26	49	56	32	35	52	28	29	42	50	54	62	25	31	40	43	60	61	63	27	45	30	36	59	44	48	41	53	55
2	1	1	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
19	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	1	
21	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0
11	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
7	1	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1
24	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
14	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0	1	1	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0
18	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
23	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	1	0	0	0	0	0	0	0
5	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0
9	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	1	1	0	0	1	0	0
10	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0	0
12	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1

A28 (IG-d=1x)

	36	28	29	30	31	32	34	35	37	40	41	43	46	49	42	45	52	38	39	33	44	47	48	50	51	53	54
9	1	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1	0	1	1	0	1	1	1	1	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0
2	0	1	1	0	1	0	1	1	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	1	1	1	1	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	1	1	1	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
5	1	1	0	1	0	1	1	1	1	0	0	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0
7	1	1	1	1	0	1	1	1	1	1	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0
8	1	1	1	1	0	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	1	0	0	0	0	1	1	1	1	1	1	0	1	0	0	0	0	1	1	1	0	0	0	0	0	0	0
13	0	1	1	0	1	0	1	0	1	1	0	1	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0
14	0	0	0	1	1	0	1	0	1	1	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0
16	0	0	1	1	0	1	1	1	0	1	1	1	1	1	1	1	0	0	0	0	1	0	1	0	0	0	0
19	0	0	1	1	0	1	0	0	1	1	1	1	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0
22	0	1	1	0	0	1	1	0	0	1	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	1	0	1	0	0	0	1
18	0	1	0	0	0	0	0	0	0	0	0	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	1
11	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0	0	0	0
12	0	0	0	0	0	0	0	0	1	1	1	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0
6	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	0	1	1
17	0	0	0	0	1	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	1	1	1	1	0	0	1
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	1	0	1	1	0
21	0	0	0	0	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	1	1	1	0	1	1	1
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	1
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	1	1	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	1	0	1	1	1	0	1

A29 (IG-d=1x)

	62	63	74	45	67	68	69	70	71	29	30	31	32	33	35	40	43	44	52	66	72	73	41	54	55	57	65	34	36	37	38	39	60	64	42	46	47	48	49	50	51	53	56	58	59	61
25	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
11	1	0	0	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
12	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	1	1	1	1	1	0	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	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
24	0	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	1	1	0	0	1	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	1	1	0	0	0	1	1	1	0	0	0	0	1	1	0	1	0	0	1	0	0	0	0	0
9	0	0	0	0	1	1	0	0	0	1	1	0	1	1	1	0	1	1	1	0	0	1	1	0	0	0	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0
10	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0
21	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1
3	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0
5	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	0	0	1	1	1	1	1	1	1	1	0	0	1	

A30-(IG d=0,7x)

	43	31	33	51	52	60	49	68	32	42	50	53	61	69	70	36	37	45	54	55	58	65	35	48	56	67	34	38	40	41	62	63	71	39	44	59	46	57	64	66	47
9	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
30	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	1	0	0	1	1	1	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0
8	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	1	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	0	
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

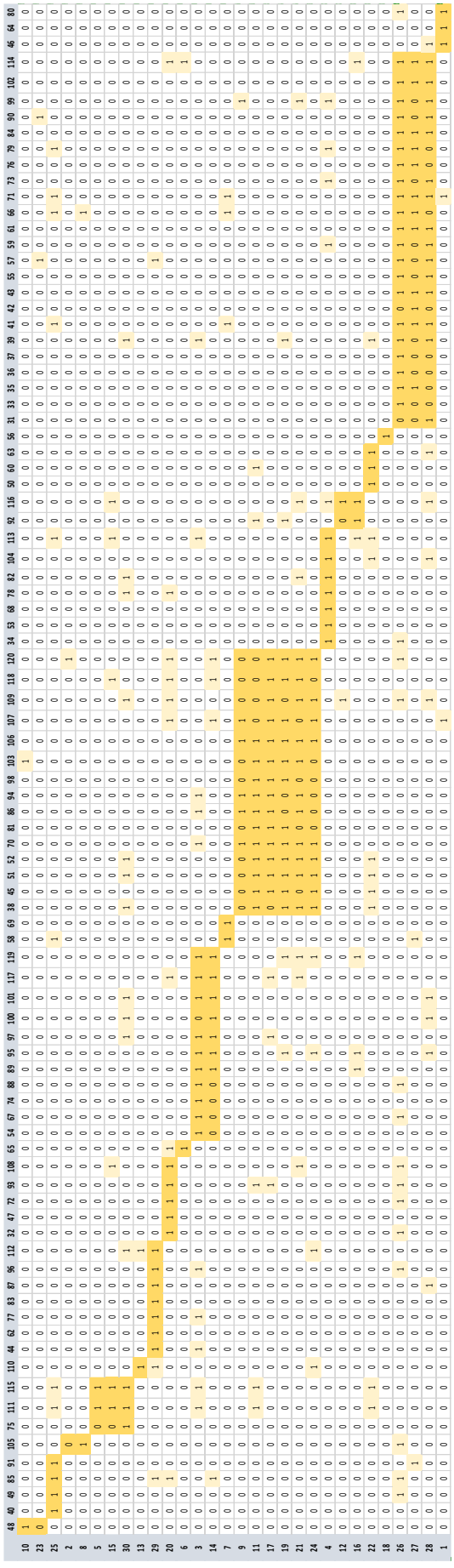
A31-(IG d=1x)

	70	71	75	79	80	50	52	54	55	57	31	37	48	58	60	62	64	65	66	67	59	61	63	68	35	43	44	45	46	32	33	39	47	49	51	53	56	76	69	72	74	78	73	77	34	36	38	40	41	42
25	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
27	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0
29	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
11	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
12	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
14	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
26	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0
30	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0
28	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	0
10	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1

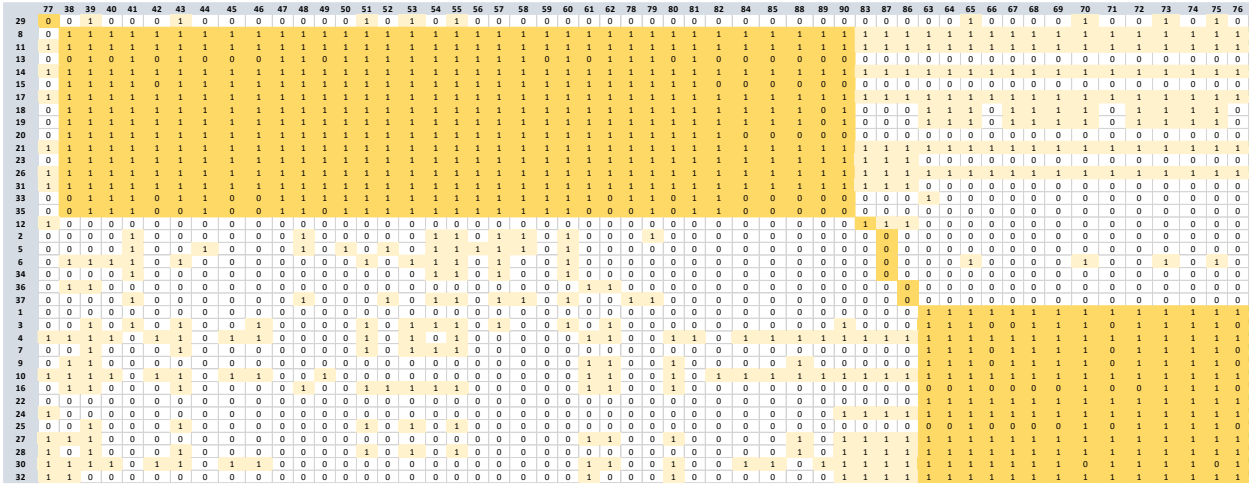
A32-(IG-d=0,7x)

	47	52	73	38	39	51	66	71	40	42	44	48	70	35	36	37	56	57	59	31	34	63	72	75	46	79	43	45	64	65	69	67	77	32	33	41	49	61	62	74	78	53	54	55	58	60	80	50	68	76
4	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
6	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	1	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
28	0	0	0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0
27	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	1	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	1	1	1	0	0	0	1	0	0
29	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0
7	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0
17	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	1	0	0	0
19	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	1
2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	1

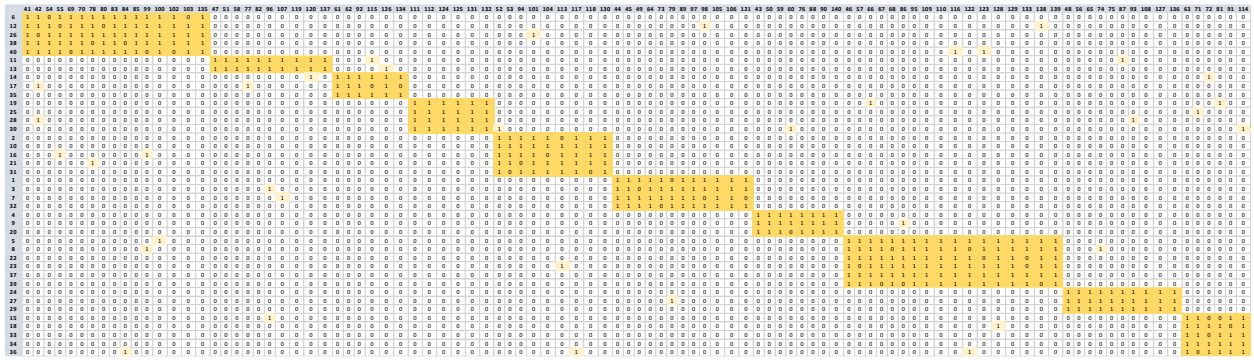
A33 (IG-d=0,7x)



A34 (SC1-1000)



A35(IG-d=1x)



ANEXO B. MATRICES RESULTADO DATA B

Se representa la matriz del mejor resultado obtenido para cada problema.

B1 (SC1)

	7	9	11	8	10	12
1	1	1	1	1	0	0
4	1	1	1	0	0	0
5	0	1	1	1	0	0
2	0	0	1	1	1	1
3	0	0	0	1	1	1
6	0	0	0	1	1	1

B2 (SC1)

	7	9	12	10	11	13	8
4	1	1	1	0	0	0	0
6	1	0	1	0	0	0	0
2	0	0	0	1	1	1	0
3	0	0	0	1	1	1	1
1	1	0	0	1	0	0	1
5	0	0	0	0	1	0	1

B3 (SC1)

	8	10	12	16	7	11	13	9	14	15	17
1	1	1	1	1	0	0	0	0	0	0	0
2	1	0	0	0	0	1	1	1	0	0	0
5	0	0	0	1	1	1	0	0	0	0	0
6	0	1	0	0	1	0	1	0	0	0	0
3	0	0	0	1	0	0	0	1	1	1	1
4	0	1	1	0	0	0	0	1	0	1	1

B4 (SC1)

	9	10	12	8	11
1	1	0	1	0	0
2	1	1	0	0	0
4	1	1	1	0	0
6	1	1	1	0	0
3	0	0	0	1	1
5	0	0	0	1	1
7	0	0	0	1	1

B5 (SC1)

	8	13	9	10	12	11	14	15
2	1	1	0	0	0	0	0	0
4	0	1	0	0	0	0	0	0
1	0	0	1	1	1	0	0	0
7	0	0	0	1	1	0	0	1
3	0	0	0	0	0	1	1	0
6	0	0	0	0	0	1	0	0
5	0	0	0	0	0	0	0	1

B6 (SC1)

	10	13	9	11	15	8	12	14
2	1	1	0	0	0	0	0	0
6	1	1	0	0	0	0	0	0
4	0	0	1	1	1	0	0	0
1	0	0	0	0	0	1	1	1
3	0	0	1	1	0	1	0	1
5	0	0	0	0	0	1	1	1
7	0	0	1	0	1	1	0	1

B7 (SC1)

	11	12	15	16	8	13	17	9	10	14
3	1	1	1	1	0	0	0	0	0	1
6	1	1	1	1	0	0	0	0	0	0
7	1	1	1	0	0	0	0	0	0	0
2	0	0	0	0	1	1	1	0	0	0
4	0	0	1	0	1	1	1	0	0	0
1	0	0	0	0	0	0	0	1	0	1
5	0	0	0	0	0	0	1	1	1	1

B8 (SC1)

	10	11	14	15	9	12	13	16
3	1	0	0	0	0	1	0	0
7	1	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0
8	0	1	1	0	0	0	0	0
6	0	1	0	1	0	0	1	0
2	0	0	0	0	1	1	1	0
4	0	0	0	0	1	0	1	1
5	0	0	0	0	0	0	1	1

B9 (SC1)

	9	13	15	17	12	14	10	11	16	18
2	1	1	1	1	0	0	0	0	0	1
3	1	1	1	1	0	0	0	0	0	0
4	0	0	0	0	1	1	1	0	0	1
5	0	0	0	0	1	1	1	0	0	0
6	1	1	0	0	1	1	0	0	0	0
1	0	0	0	0	0	0	1	1	1	1
7	0	0	0	0	0	1	1	1	1	1
8	0	0	0	0	0	0	0	1	1	1

B10 (SC1)

	10	13	17	9	12	14	16	18	11	15
3	1	1	1	0	0	0	0	1	0	0
4	1	1	1	0	0	0	0	0	0	0
8	1	1	1	0	0	0	0	1	0	0
2	0	0	0	1	1	1	1	1	0	0
5	0	0	0	1	1	1	1	0	0	0
7	0	0	0	1	1	1	1	1	0	0
1	0	0	0	0	0	0	0	0	1	1
6	0	0	0	0	0	0	0	0	1	1

B11 (SC1)

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

B12 (SC1)

	12	15	18	11	13	16	14	17
3	1	1	1	0	0	0	0	0
6	1	1	1	0	0	0	0	0
7	0	1	1	0	0	0	0	0
1	0	0	1	1	1	1	0	0
2	0	0	0	1	0	1	0	0
4	0	0	0	1	1	1	0	0
8	0	0	0	1	1	1	0	0
5	0	0	0	0	0	0	1	1
9	0	0	0	0	0	0	1	1
10	1	0	0	0	0	0	0	1

B13 (IG-d=1x)

	11	12	13	16	17	19	14	15	18	20	21	22
4	1	1	1	0	0	0	1	0	0	0	0	0
5	1	1	0	0	0	0	0	0	0	0	0	0
9	0	1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	1	1	1	0	0	0	0	0	0
3	0	0	0	1	0	0	1	0	0	0	0	0
7	0	0	0	1	1	0	0	1	1	0	0	0
1	0	0	0	1	1	0	1	1	1	0	1	1
6	0	0	0	1	0	0	1	0	1	1	1	0
8	0	0	0	0	1	1	1	0	1	1	1	1
10	0	1	0	0	0	0	1	1	1	1	0	1

B14 (SC1-1000)

	12	14	19	24	25	28	30	31	46	47	11	26	44	42	43	45	16	20	23	32	35	36	38	40	41	13	15	21	22	27	29	34	39	48	17	18	33	37
3	1	1	1	1	1	1	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	1	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
6	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
10	0	0	1	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0
8	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
4	1	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	1	1	0	0	0	1	1	1	1	1	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

B15 (SC1)

	13	14	16	12	17	18	19	21	15	20
1	1	1	1	0	0	0	0	0	0	0
3	1	1	0	0	0	0	0	0	0	0
5	1	1	1	0	0	0	0	1	0	0
2	0	0	0	1	1	1	0	0	0	0
8	0	0	0	1	1	1	0	0	0	0
10	0	0	0	1	0	1	0	0	0	0
9	0	0	0	0	0	0	1	1	0	0
11	0	0	0	0	0	0	1	1	0	0
4	0	0	0	0	0	0	0	0	1	1
6	0	0	0	0	0	0	0	0	1	1
7	0	0	0	0	1	0	0	0	1	0

B16 (SC2)

	15	17	20	21	25	28	29	12	13	14	18	22	26	27	31	32	33	16	19	23	24	30
7	1	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1
9	1	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	1	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1	1	1	0	1	1	1	1	1	1	0	0	0	0	0
4	0	0	0	0	0	0	0	1	1	1	0	1	1	1	1	1	1	0	0	0	0	0
5	0	0	0	0	0	0	0	1	1	1	1	0	1	1	1	1	1	0	0	0	0	0
10	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	1	1	1	0	0
2	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	1	0	1
3	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1

B17 (IG-d=1x)

	24	18	20	27	30	13	16	25	26	28	14	17	21	22	29	15	19	23	31
2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
3	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0
1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0
5	1	0	1	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0
6	0	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0
10	0	0	0	1	0	0	0	0	1	0	1	0	1	1	1	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0
9	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	1	1
12	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	1

B18 (IG-d=1x)

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

B19 (SC1)

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

B20 (SC1)

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

B21 (SC1)

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

B22 (SC1)

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

B23 (SC1-1000)

	19	25	34	18	24	27	32	36	20	23	26	29	31	35	37	21	22	28	30	33
6	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	1	1	0	0	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0
14	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	1	1	0	1	1	1	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	1	1	1	0	1	1	0	0	0	0	0
12	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0
16	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	1

B24 (SC1-1000)

	37	29	32	34	41	38	25	31	36	39	40	19	21	28	42	20	23	24	26	27	30	33	35	22
16	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0
7	1	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0
8	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	1	0	1	0	0	0	1	1	1	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0
10	0	0	0	0	0	0	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	1	1	0
11	0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0
14	0	0	0	0	1	1	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0	0
3	0	0	0	0	1	0	0	0	0	0	0	0	1	1	1	0	1	1	1	1	0	1	1	0
4	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	1	1	1	1	1	1	1	0
15	0	1	0	1	1	0	0	0	0	0	0	0	0	1	1	1	1	0	1	1	1	1	1	1
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	1	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

B25 (SC1)

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

B26 (IG-d=1x)

	59	60	61	62	63	64	65	66	67	68	21	22	23	24	25	26	27	28	29	30	31	69	70	71	33	34	35	36	40	41	42	43	44	45	46	47	49	54	32	37	38	39	48	50	51	52	53	56	57	58	55
10	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	1	1	1	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	1	1	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	1	1	1	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	1	0	0	0	0	0	0	1	1	1	1	0	1	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0
19	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	1	0	0	0	0	0	1	0	1	0	0	0	0	1	0	0	0	1	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	1	0	1	1	1	1	1	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	1	1	1	1	1	1
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	1	1	1	1	1	1
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	1	1	0
18	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	1	0	0	0	0
20	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	1	1

B27 (IG-d=1x)

	27	33	35	40	45	50	53	28	37	44	31	38	39	42	47	29	34	52	54	32	43	49	30	36	41	46	48	51
2	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
6	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
15	0	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
21	1	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	1	1	1	0	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
3	1	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0
8	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0
20	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	1	0	0	0	0	0
4	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	1	1	1	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	1
14	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1
25	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	1

B28 (IG-d=1x)

	44	54	56	33	40	48	51	57	60	63	29	30	35	41	49	50	53	59	31	37	38	45	46	52	34	36	42	43	58	62	32	39	47	55	61
2	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0
6	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0
8	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
20	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0
4	0	1	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
10	0	0	0	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	1	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	1	1	0	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	1	1	1	1	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
28	0	0	0	1	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
16	0	0	0	0	1	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0
24	0	0	0	0	0	0	1	1	0	0	1	1	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
5	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	1	1	1	1	0	0	1	0	0	0	0	0	0	1	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	0	0	0	0	0
14	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	1	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	1	0	1	1	1	0	1	1	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	1	1	1
23	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
27	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	1	1	1	1	1

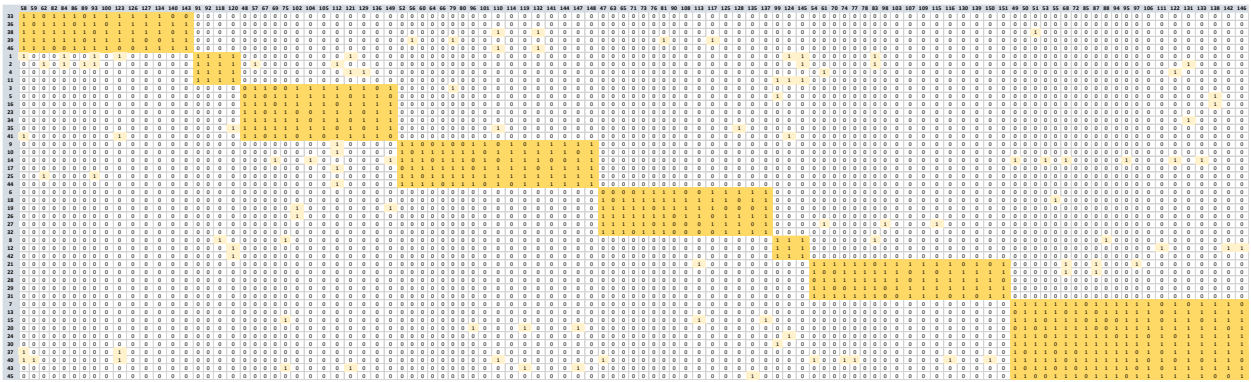
B29 (IG-d=1x)

	37	49	40	43	41	46	38	42	47	48	45	36	39	44	50
9	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0
13	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1
18	1	0	0	0	0	0	0	1	0	0	1	0	0	0	0
25	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0
14	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0
17	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0
16	0	0	1	1	0	0	0	0	0	1	0	0	0	0	1
22	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
26	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1
28	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0
19	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
35	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
30	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0
32	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0
8	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0
23	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
33	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
11	0	0	0	0	0	0	0	1	0	0	1	1	0	0	0
2	0	1	0	0	0	0	0	0	0	0	0	1	1	0	0
20	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
24	0	1	0	0	0	0	0	0	1	0	0	1	1	0	0
31	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
5	0	0	0	0	1	0	1	0	1	0	0	0	0	1	1
7	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1
12	0	0	1	0	0	0	0	0	0	1	0	0	0	1	1
15	0	0	1	0	1	0	0	0	0	0	0	0	0	1	1
29	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
34	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

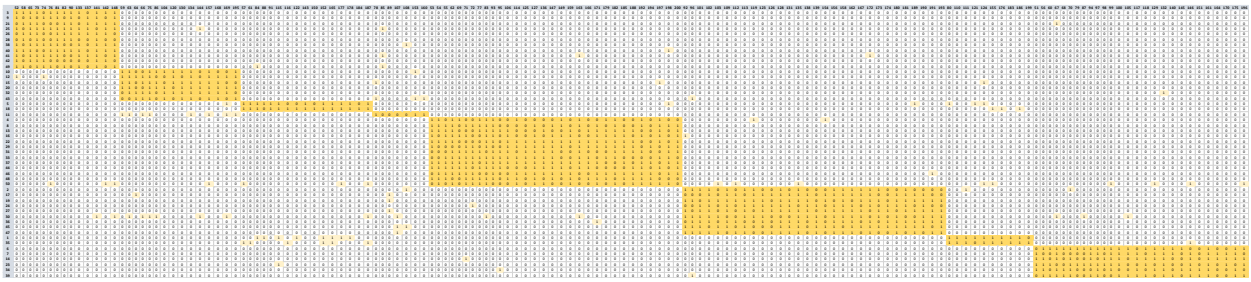
B30 (IG-d=0,7x)

	76	80	67	70	73	74	75	52	55	59	57	58	77	81	85	91	83	69	71	61	64	88	87	60	62	63	51	53	86	89	90	54	56	65	66	68	72	78	79	82	84
33	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
34	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
35	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
38	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0
32	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0
14	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0
37	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
50	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
39	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
40	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
41	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
42	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
27	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
49	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
47	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
48	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
45	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0
46	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
26	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
29	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
30	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0
43	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
44	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1

B31 (IG-d=1x)



B32 (IG-d=0,7x)



ANEXO C. DESARROLLO DEL CÓDIGO

Seguidamente se desarrolla el código desarrollado para cada uno de los algoritmos.

C.1 Selección clonal 1

```
#include <schedule.h>
#include <time.h>



float evalua(VECTOR_INT anticuerpo_copia, int dimension, int trabajos, int maquinas,
MAT_INT inicial, int operaciones);
int extraer (MAT_FLOAT poblacion,MAT_FLOAT clones, int pob, int dimension);
void mutar (MAT_FLOAT clones, int dimension, int indice);
VECTOR_INT baraja_vector(VECTOR_INT anticuerpo_copia, int dimension);
float funcion_final();


int main()
{
    //Se declara el problema a analizar
    char problema[12]="dataA1.txt";
    int k;
    VECTOR_FLOAT solucion =DIM_VECTOR_FLOAT(10);
    for(k=0;k<10;k++)
    {
        solucion[k]=funcion_final(problema);
    }
    printf("\nEl vector de 10 iters \n");
    print_float_vector(solucion,10);


}


float funcion_final(char problema[])
{
    //Se contabiliza el tiempo que va a tardar
    srand((int)time(0));
    clock_t t_ini,t_fin;
    double secs;
    t_ini=clock();
    //leemos el archivo de texto donde tenemos los datos
    FILE *fichero;


    fichero = fopen(problema,"r");//rnl
    if (fichero==NULL)
    {
        printf( "No se puede abrir el fichero.\n" );
        system("pause");
        exit (EXIT_FAILURE);
    }
    int trabajos,maquinas;
    //primero vemos cuantas filas y columnas hay para poder declarar la variable
    para la matriz
    fscanf (fichero, "%d %d\n", &trabajos, &maquinas);

    MAT_INT inicial =DIM_MAT_INT(trabajos, maquinas);
    int i,j;
    int vueltas=0,operaciones=0;
    float mejor=0;

    //Vamos leyendo de uno en uno y vamos metiendo en la matriz
```

```
58      for (i = 0; i < trabajos; i++)
59      {
60          for (j = 0; j < maquinas; j++)
61          {
62              fscanf (fichero, "%d\n", &inicial[i][j]);
63              if(inicial[i][j]==1)
64              {
65                  operaciones++;
66              }
67          }
68      }
69
70      fclose(fichero);
71
72
73      // ver numero de celulas
74
75      int num_celulas;
76      num_celulas=Min(trabajos,maquinas);
77
78       // creacion del vector anticuerpo
79      int dimension= trabajos+maquinas+num_celulas-1;
80      VECTOR_INT anticuerpo = DIM_VECTOR_INT(dimension);
81
82      for (i=0;i<dimension;i++)
83      {
84          if (i<trabajos+maquinas)
85          {
86              anticuerpo[i]=i+1;
87          }
88          else
89          {
90              anticuerpo[i]=0;
91          }
92
93      }
94
95      //generar poblacion aleatoria inicial -> meterla en una matriz y evaluar
96      VECTOR_INT anticuerpo_copia = DIM_VECTOR_INT(dimension);
97
98      int pob=100;
99      int queda=pob*0.9;
100     MAT_FLOAT poblacion = DIM_MAT_FLOAT(pob,dimension+3);
101     MAT_FLOAT clones = DIM_MAT_FLOAT(pob*3,dimension+3);
102
103     for(i=0;i<pob;i++)
104     {
105         anticuerpo_copia=baraja_vector(anticuerpo,dimension);
106         for(j=0;j<dimension;j++)
107         {
108             poblacion[i][j]=anticuerpo_copia[j];
109         }
110     }
111
112     //declarar variables para tener en cuenta el número de vueltas
113     int vueltas2=0;
114     int vueltas3=0;
115
116     VECTOR_INT mejorvector=DIM_VECTOR_INT(dimension);
117
118     while (vueltas<1000)//empezaría el bucle
119     {
120         vueltas2++;
121         VECTOR_INT copia=DIM_VECTOR_INT(dimension);
122         float f_objetivo_total=0;
123         float efectividad=0;
124
125         //Se calcula la efectividad de cada individuo
126         for(i=0;i<pob;i++)
127         {
128             for(j=0;j<dimension;j++)
129             {
130                 copia[j]=poblacion[i][j];
```

```
131             }
132             efectividad=evalua(copia,dimension,trabajos,maquinas,inicial,operaciones);
133             f_objetivo_total=f_objetivo_total+efectividad;
134             poblacion[i][j]=efectividad;
135         }
136
137          //Ordenamos la matriz de menor a mayor eficacia
138
139         int flag=0;
140         float auxi=0;
141         while(flag==0)
142         {
143             flag=1;
144             for(i=0;i<pob-1;i++)
145             {
146                 if(poblacion[i][dimension]>poblacion[i+1][dimension])
147                 {
148                     for(j=0;j<dimension+1;j++)
149                     {
150                         auxi=poblacion[i+1][j];
151                         poblacion[i+1][j]=poblacion[i][j];
152                         poblacion[i][j]=auxi;
153                     }
154
155                     flag=0;
156                 }
157             }
158         }
159
160         //Cargar las probabilidades de seleccion en la matriz
161         //primero inicializamos en i=0
162
163         poblacion[0][dimension+1]=poblacion[0][dimension]/f_objetivo_total;
164         poblacion[0][dimension+2]=poblacion[0][dimension+1];
165         for(i=1;i<pob;i++)
166         {
167             poblacion[i][dimension+1]=poblacion[i][dimension]/f_objetivo_total;
168
                poblacion[i][dimension+2]=poblacion[i-1][dimension+2]+poblacion[i][dimensi
                on+1];
169
170         }
171
172
173         //extraemos los vectores elegidos
174         int indice=0;
175         indice=extraer(poblacion,clones,pob,dimension);
176
177         // copiar los seleccionados en una matriz (clonar)
178
179         for(i=0;i<indice*2;i++)
180         {
181             for(j=0;j<dimension+3;j++)
182             {
183                 clones[i+indice][j]= clones[i][j];
184             }
185         }
186
187         //madurar los clones
188         mutar(clones,dimension,indice);
189
190         for(i=0;i<(indice*3);i++)
191         {
192             for(j=0;j<dimension;j++)
193             {
194                 copia[j]=clones[i][j];
195             }
196
                clones[i][dimension]=evalua(copia,dimension,trabajos,maquinas,inicial,oper
                aciones);
197         }
198
199         //Buscar el maximo de eficacia
```

```
200             float max=0;
201             for(i=0;i<indice*3;i++)
202             {
203                 if(clones[i][dimension]>max)
204                 {
205                     max=clones[i][dimension];
206                 }
207                 if(max>mejor)
208                 {
209                     mejor=max;
210                     for (j=0;j<dimension;j++)
211                     {
212                         mejorvector[j]=clones[i][j];
213                     }
214                     vueltas3=vueltas2;
215                     t_fin=clock();
216 
217                     vueltas=0;
218                 }
219             }
220 
221             //Nos quedamos con el 90% de los mejores, es decir, eliminamos el 10% de los
                peores
222             for(i=0;i<queda;i++)
223             {
224                 for(j=0;j<dimension+2;j++)
225                 {
226                     poblacion[i][j]=clones[i][j];
227                 }
228 
229             }
230 
231             VECTOR_INT nuevo=DIM_VECTOR_INT(dimension);
232             if(indice*3<queda)
233             {
234                 for(i=0;i<queda;i++)
235                 {
236                     for(j=0;j<dimension+2;j++)
237                     {
238                         poblacion[i][j]=clones[i][j];
239                     }
240 
241                 }
242                 for(i=indice*3;i<pob;i++)
243                 {
244                     nuevo=baraja_vector(anticuerpo,dimension);
245                     for(j=0;j<dimension;j++)
246                     {
247                         poblacion[i][j]=nuevo[j];
248                     }
249                 }
250             }
251             else
252             {
253                 float best=0;
254                 int position=0;
255 
256                 for(i=0;i<queda;i++)
257                 {
258                     for(j=0;j<(indice*3);j++)
259                     {
260                         if(clones[i][dimension]>best)
261                         {
262                             best=clones[i][dimension];
263                             position=j;
264                         }
265                     }
266 
267                     for(j=0;j<dimension;j++)
268                     {
269                         poblacion[position][j]=clones[position][j];
270                     }
271                     clones[position][dimension]=0;
```

```
272                 }
273
274                 for(i=queda;i<pob;i++)
275                 {
276                     nuevo=baraja_vector(anticuerpo,dimension);
277                     for(j=0;j<dimension;j++)
278                     {
279                         poblacion[i][j]=nuevo[j];
280                     }
281                 }
282             }
283
284             vueltas++;
285
286         }
287
288         secs = (double)(t_fin - t_ini) / CLOCKS_PER_SEC;
289         printf("%s %d  %f  %.16g ",problema,vueltas3,mejor,secs*1000);
290         print_vector(mejorvector,dimension);
291         return mejor;
292     }
293
294
295     float evalua(VECTOR_INT anticuerpo_copia, int dimension, int trabajos, int maquinas,
    MAT_INT inicial, int operaciones)
296     {
297         int i,j,k,h;
298         int posfil=0,poscol=0;
299         int vacios=0,excep=0;
300         int falta_fila=0,falta_columna=0;
301         float penalizacion=0;
302         float resultado=0;
303         int filas[trabajos];
304         int columnas[maquinas];
305         int celulas[trabajos][maquinas];
306         for(i=0;i<trabajos;i++)
307         {
308             for(j=0;j<maquinas;j++)
309             {
310                 celulas[i][j]=0;
311             }
312         }
313
314
315         for (i=0;i<dimension;i++)
316         {
317             for(h=0;h<dimension-i && anticuerpo_copia[i+h]!=0;h++)
318             {
319                 if(anticuerpo_copia[i+h]<=trabajos)
320                 {
321                     filas[h-posfil]=anticuerpo_copia[i+h];
322                     poscol++;
323                 }
324                 else
325                 {
326                     columnas[h-poscol]=anticuerpo_copia[i+h];
327                     posfil++;
328                 }
329             }
330             i=i+h;
331             // penalizar si no tiene trabajos o maquinas
332             if(poscol==0 && posfil!=0)
333             {
334                 falta_fila=1;
335             }
336             if(posfil==0 && poscol!=0)
337             {
338                 falta_columna=1;
339             }
340
341
342             //rellenar una matriz donde los espacios ocupados por celulas sean igual a 1
343
```

```
344             for(j=0;j<poscol && posfil!=0 && poscol!=0;j++)
345             {
346                 for(k=0;k<posfil;k++)
347                 {
348                     celulas[filas[j]-1][columnas[k]-trabajos-1]=1;
349                 }
350             }
351             poscol=0;
352             posfil=0;
353             for(j=0;j<poscol;j++)
354             {
355                 filas[j]=0;
356             }
357             for(j=0;j<posfil;j++)
358             {
359                 columnas[j]=0;
360             }
361 
362         }
363 
364         //Se calculas los vacios y excepciones
365         for(i=0;i<trabajos;i++)
366         {
367             for(j=0;j<maquinas;j++)
368             {
369                 if(inicial[i][j]==0 && celulas[i][j]==1)
370                 {
371                     vacios++;
372                 }
373                 if(inicial[i][j]==1 && celulas[i][j]==0)
374                 {
375                     excep++;
376                 }
377             }
378         }
379 
380         resultado=(1-(float)excep/(operaciones))/(1+(float)vacios/(operaciones));
381         penalizacion=0.5*resultado*falta_fila+0.5*resultado*falta_columna;
382 
383 
384         free_mat_int(celulas,trabajos);
385 
386         return (resultado-penalizacion);
387 
388     }
389 
390 
391 
392     int extraer (MAT_FLOAT poblacion, MAT_FLOAT clones, int pob, int dimension)
393 
394     {
395         int i,j;
396         int indice=0;
397         float aleatorio=0;
398         //Se crea un numero aleatorio y se selecciona el individuo que corresponde a ese
            número
399         aleatorio=(float)(rand()%10000)/10000;
400         for(i=0;i<pob;i++)
401         {
402             if(poblacion[i][dimension+2]>=aleatorio)
403             {
404                     for(j=0;j<dimension+3;j++)
405                     {
406                         clones[indice][j]=poblacion[i][j];
407                     }
408                     indice++;
409             }
410         }
411         return indice;
412 
413     }
414 
415     void mutar (MAT_FLOAT clones, int dimension, int indice)
```

```
416  {
417      int i,j,posicion1,posicion2;
418      int aux=0;
419
420      //Cambiar dos posiciones de cada vector anticuerpo
421      for (i=indice;i<indice*3;i++)
422      {
423          posicion1=rand()%dimension;
424          posicion2=rand()%dimension;
425          aux=clones[i][posicion1];
426          clones[i][posicion1]=clones[i][posicion2];
427          clones[i][posicion2]=aux;
428
429      }
430
431  }
432
433
434
435  VECTOR_INT baraja_vector(VECTOR_INT anticuerpo_copia, int dimension)
436  {
437      //Para crear nuevos vectores aleatorios
438      int i,temp1=0,temp2=0;
439      int flag=0;
440      int indices[dimension];
441      for(i=0;i<dimension;i++)
442      {
443          indices[i]=rand()%100;
444      }
445      while(flag==0)
446      {
447          flag=1;
448          for(i=0;i<dimension-1;i++)
449          {
450              if(indices[i]>indices[i+1])
451              {
452                  temp1=indices[i];
453                  indices[i]=indices[i+1];
454                  indices[i+1]=temp1;
455                  temp2=anticuerpo_copia[i];
456                  anticuerpo_copia[i]=anticuerpo_copia[i+1];
457                  anticuerpo_copia[i+1]=temp2;
458                  flag=0;
459              }
460          }
461      }
462
463      free_vector(indices);
464
465      return anticuerpo_copia;
466  }
467
```

C.2 Selección clonal 2

```
#include <schedule.h>
#include <time.h> //rnl

float evalua(VECTOR_INT anticuerpo_copia, int dimension, int trabajos, int maquinas, MAT_INT inicial, int operaciones);
int extraer (MAT_FLOAT poblacion,MAT_FLOAT clones, int pob, int dimension);
void mutar (MAT_FLOAT clones, int dimension, int indice, MAT_FLOAT poblacion, int trabajos, int maquinas, MAT_INT inicial, int operaciones);
VECTOR_INT baraja_vector(VECTOR_INT anticuerpo_copia, int dimension);
float funcion_final();

int main()
{
    //Se declara el problema a analizar
    int k;
    VECTOR_FLOAT solucion =DIM_VECTOR_FLOAT(10);

    char problema1[12]="dataA1.txt";

    for(k=0;k<10;k++)
    {
        solucion[k]=funcion_final(problema1);
    }
    printf("\nEl vector de 10 iters \n");
    print_float_vector(solucion,10);


}


float funcion_final(char problema[])
{
    //Se declara el tiempo
    srand((int)time(0));
    clock_t t_ini,t_fin;
    double secs;
    t_ini=clock();
    //leemos el archivo de texto donde tenemos los datos
    FILE *fichero;
    fichero = fopen(problema,"r");
    if (fichero==NULL)
    {
        printf( "No se puede abrir el fichero.\n" );
        system("pause");
        exit (EXIT_FAILURE);
    }
    int trabajos,maquinas;
    //primero vemos cuantas filas y columnas hay para poder declarar la variable para la matriz
    fscanf (fichero, "%d %d\n", &trabajos, &maquinas);

    MAT_INT inicial =DIM_MAT_INT(trabajos, maquinas);
    int i,j;
    int vueltas=0,operaciones=0;
    float mejor=0;

    //Vamos leyendo de uno en uno y vamos metiendo en la matriz
    for (i = 0; i < trabajos; i++)
    {
        for (j = 0; j < maquinas; j++)
        {
            fscanf (fichero, "%d\n", &inicial[i][j]);
            if(inicial[i][j]==1)
            {
                operaciones++;
            }
        }
    }

    fclose(fichero);

    // ver numero de celulas

```

```
71        int num_celulas;
72        num_celulas=Min(trabajos,maquinas);
73
74         // creacion del vector anticuerpo
75        int dimension= trabajos+maquinas+num_celulas-1;
76        VECTOR_INT anticuerpo = DIM_VECTOR_INT(dimension);
77
78        for (i=0;i<dimension;i++)
79        {
80            if (i<trabajos+maquinas)
81            {
82                anticuerpo[i]=i+1;
83            }
84            else
85            {
86                anticuerpo[i]=0;
87            }
88
89        }
90        //generar poblacion aleatoria inicial -> meterla en una matriz y evaluar
91        VECTOR_INT anticuerpo_copia = DIM_VECTOR_INT(dimension);
92        int pob=100;
93        int queda=pob*0.9;
94        MAT_FLOAT poblacion = DIM_MAT_FLOAT(pob,dimension+3);
95        MAT_FLOAT clones = DIM_MAT_FLOAT(pob,dimension+3);
96
97        for(i=0;i<pob;i++)
98        {
99            anticuerpo_copia=baraja_vector(anticuerpo,dimension);
100           for(j=0;j<dimension;j++)
101           {
102               poblacion[i][j]=anticuerpo_copia[j];
103           }
104       }
105
106       int vueltas2=0;
107       int vueltas3=0;
108       VECTOR_INT mejorvector=DIM_VECTOR_INT(dimension);
109       while (vueltas<1000)//empezaría el bucle
110       {
111
112           vueltas2++;
113           VECTOR_INT copia=DIM_VECTOR_INT(dimension);
114           float f_objetivo_total=0;
115           float efectividad=0;
116
117           for(i=0;i<pob;i++)
118           {
119               for(j=0;j<dimension;j++)
120               {
121                   copia[j]=poblacion[i][j];
122               }
123               efectividad=evalua(copia,dimension,trabajos,maquinas,inicial,operaciones);
124               f_objetivo_total=f_objetivo_total+efectividad;
125               poblacion[i][j]=efectividad;
126           }
127
128            //Ordenamos la matriz de menor a mayor eficacia
129
130           int flag=0;
131           float auxi=0;
132           while(flag==0)
133           {
134               flag=1;
135               for(i=0;i<pob-1;i++)
136               {
137                   if(poblacion[i][dimension]>poblacion[i+1][dimension])
138                   {
139                       for(j=0;j<dimension+1;j++)
140                       {
141                           auxi=poblacion[i+1][j];
142                           poblacion[i+1][j]=poblacion[i][j];
143                           poblacion[i][j]=auxi;
```

```
144                     }
145
146                     flag=0;
147                 }
148             }
149         }
150
151         //Cargar las probabilidades de seleccion en la matriz
152         //primero inicializamos en i=0
153
154         poblacion[0][dimension+1]=poblacion[0][dimension]/f_objetivo_total;
155         poblacion[0][dimension+2]=poblacion[0][dimension+1];
156         for(i=1;i<pob;i++)
157         {
158             poblacion[i][dimension+1]=poblacion[i][dimension]/f_objetivo_total;
159
                poblacion[i][dimension+2]=poblacion[i-1][dimension+2]+poblacion[i][dimensi
                on+1];
160
161         }
162
163
164         //extraemos los vectores elegidos
165         int indice=0;
166         indice=extraer(poblacion,clones,pob,dimension);
167
168         //madurar los clones
169         mutar (clones, dimension, indice, poblacion, trabajos, maquinas, inicial,
                operaciones);
170
171         //Buscar el maximo de eficacia
172         float max=0;
173         for(i=0;i<pob;i++)
174         {
175             if(poblacion[i][dimension]>max)
176             {
177                 max=poblacion[i][dimension];
178             }
179             if(max>mejor)
180             {
181                 mejor=max;
182                 for (j=0;j<dimension;j++)
183                 {
184                     mejorvector[j]=clones[i][j];
185                 }
186                 vueltas3=vueltas2;
187                 t_fin=clock();
188
189                 vueltas=0;
190             }
191         }
192
193          //Quitamos los malos
194         float peor=1;
195         int position=0;
196         VECTOR_INT nuevo=DIM_VECTOR_INT(dimension);
197
198         for(i=0;i<(pob-queda);i++)
199         {
200             for(j=0;j<pob;j++)
201             {
202                 if(poblacion[i][dimension]<peor)
203                 {
204                     peor=poblacion[i][dimension];
205                     position=j;
206                 }
207             }
208             nuevo=baraja_vector(anticuerpo,dimension);
209             for(j=0;j<dimension;j++)
210             {
211                 poblacion[position][j]=nuevo[j];
212             }
213             poblacion[position][dimension]=1;
```

```
214             }
215
216
217             vueltas++;
218         }
219
220         secs = (double)(t_fin - t_ini) / CLOCKS_PER_SEC;
221         printf("%s %d  %f  %.16g ",problema,vueltas3,mejor,secs*1000);
222         print_vector(mejorvector,dimension);
223
224         return mejor;
225     }
226
227
228     float evalua(VECTOR_INT anticuerpo_copia, int dimension, int trabajos, int maquinas,
        MAT_INT inicial, int operaciones)
229     {
230         int i,j,k,h;
231         int posfil=0,poscol=0;
232         int vacios=0,excep=0;
233         int falta_fila=0,falta_columna=0;
234         float penalizacion=0;
235         float resultado=0;
236         int filas[trabajos];
237         int columnas[maquinas];
238         int celulas[trabajos][maquinas];
239         for(i=0;i<trabajos;i++)
240         {
241             for(j=0;j<maquinas;j++)
242             {
243                 celulas[i][j]=0;
244             }
245         }
246
247         for (i=0;i<dimension;i++)
248         {
249             for(h=0;h<dimension-i && anticuerpo_copia[i+h]!=0;h++)
250             {
251                 if(anticuerpo_copia[i+h]<=trabajos)
252                 {
253                     filas[h-posfil]=anticuerpo_copia[i+h];
254                     poscol++;
255                 }
256                 else
257                 {
258                     columnas[h-poscol]=anticuerpo_copia[i+h];
259                     posfil++;
260                 }
261             }
262             i=i+h;
263             // penalizar si no tiene trabajos o maquinas
264             if(poscol==0 && posfil!=0)
265             {
266                 falta_fila=1;
267             }
268             if(posfil==0 && poscol!=0)
269             {
270                 falta_columna=1;
271             }
272             //rellenar una matriz donde los espacios ocupados por celulas sean igual a 1
273
274             for(j=0;j<poscol && posfil!=0 && poscol!=0;j++)
275             {
276                 for(k=0;k<posfil;k++)
277                 {
278                     celulas[filas[j]-1][columnas[k]-trabajos-1]=1;
279                 }
280             }
281             poscol=0;
282             posfil=0;
283             for(j=0;j<poscol;j++)
284             {
285                 filas[j]=0;
```

```
286             }
287             for(j=0;j<posfil;j++)
288             {
289                 columnas[j]=0;
290             }
291         }
292 
293         for(i=0;i<trabajos;i++)
294         {
295             for(j=0;j<maquinas;j++)
296             {
297                 if(inicial[i][j]==0 && celulas[i][j]==1)
298                 {
299                     vacios++;
300                 }
301                 if(inicial[i][j]==1 && celulas[i][j]==0)
302                 {
303                     excep++;
304                 }
305             }
306         }
307 
308         resultado=(1-(float)excep/(operaciones))/(1+(float)vacios/(operaciones));
309         penalizacion=0.5*resultado*falta_fila+0.5*resultado*falta_columna;
310 
311 
312         return (resultado-penalizacion);
313 
314     }
315 
316 
317 
318     int extraer (MAT_FLOAT poblacion, MAT_FLOAT clones, int pob, int dimension)
319 
320     {
321         int i,j;
322         int indice=0;
323         float aleatorio=0;
324 
325         aleatorio=(float)(rand()%10000)/10000;
326         for(i=0;i<pob;i++)
327         {
328             if(poblacion[i][dimension+2]>=aleatorio)
329             {
330                     for(j=0;j<dimension+3;j++)
331                     {
332                         clones[indice][j]=poblacion[i][j];
333                     }
334                     indice++;
335             }
336         }
337         return indice;
338 
339     }
340 
341 
342     void mutar (MAT_FLOAT clones, int dimension, int indice, MAT_FLOAT poblacion, int
    trabajos, int maquinas, MAT_INT inicial, int operaciones)
343     {
344         int i,j,posicion1,posicion2;
345         int aux=0, mejorados=0;
346         int copia[dimension];
347         float ef=0;
348 
349         //Cambiar dos posiciones de cada vector anticuerpo
350         for (i=0;i<indice;i++)
351         {
352             posicion1=rand()%dimension;
353             posicion2=rand()%dimension;
354             aux=clones[i][posicion1];
355             clones[i][posicion1]=clones[i][posicion2];
356             clones[i][posicion2]=aux;
357             for(j=0;j<dimension;j++)
```

```
358             {
359                 copia[j]=clones[i][j];
360             }
361             ef=evalua(copia,dimension,trabajos,maquinas,inicial,operaciones);
362             if(ef>clones[i][dimension])
363             {
364                 for(j=0;j<dimension;j++)
365                 {
366                     poblacion[mejorados][j]=clones[i][j];
367                 }
368                 poblacion[mejorados][dimension]=ef;
369                 mejorados++;
370             }
371 
372         }
373 
374         free_vector(copia);
375 
376     }
377 
378 
379 
380     VECTOR_INT baraja_vector(VECTOR_INT anticuerpo_copia, int dimension)
381     {
382         int i,temp1=0,temp2=0;
383         int flag=0;
384         int indices[dimension];
385         for(i=0;i<dimension;i++)
386         {
387             indices[i]=rand()%100;
388         }
389         while(flag==0)
390         {
391             flag=1;
392             for(i=0;i<dimension-1;i++)
393             {
394                 if(indices[i]>indices[i+1])
395                 {
396                     temp1=indices[i];
397                     indices[i]=indices[i+1];
398                     indices[i+1]=temp1;
399                     temp2=anticuerpo_copia[i];
400                     anticuerpo_copia[i]=anticuerpo_copia[i+1];
401                     anticuerpo_copia[i+1]=temp2;
402                     flag=0;
403                 }
404             }
405         }
406 
407         free_vector(indices);
408 
409         return anticuerpo_copia;
410     }
411 
```

C.3 Iterated Greedy *d=N*

```
#include <schedule.h>
#include <time.h> //rnl

float evalua(VECTOR_INT anticuerpo_copia, int dimension, int trabajos, int maquinas, MAT_INT inicial, int operaciones);
float funcion_final();


int main()
{
    int k;
    VECTOR_FLOAT solucion =DIM_VECTOR_FLOAT(10);

    char problema1[12]="dataA1.txt";

    for(k=0;k<10;k++)
    {
        solucion[k]=funcion_final(problema1);
    }
    printf("\nEl vector de 10 iters \n");
    print_float_vector(solucion,10);


}


float funcion_final(char problema[])
{
    srand((int)time(0));
    clock_t t_ini,t_fin;
    double secs;
    t_ini=clock();
    //leemos el archivo de texto donde tenemos los datos
    FILE *fichero;
    fichero = fopen(problema,"r");
    if (fichero==NULL)
    {
        printf( "No se puede abrir el fichero.\n" );
        system("pause");
        exit (EXIT_FAILURE);
    }
    int trabajos,maquinas;
    //primero vemos cuantas filas y columnas hay para poder declarar la variable para la matriz
    fscanf (fichero, "%d %d\n", &trabajos, &maquinas);

    MAT_INT inicial =DIM_MAT_INT(trabajos, maquinas);
    int i,j;
    int vueltas=0,operaciones=0;
    float mejor=0;
    int iteracion=0;

    //Vamos leyendo de uno en uno y vamos metiendo en la matriz
    for (i = 0; i < trabajos; i++)
    {
        for (j = 0; j < maquinas; j++)
        {
            fscanf (fichero, "%d\n", &inicial[i][j]);
            if(inicial[i][j]==1)
            {
                operaciones++;
            }
        }
    }

    fclose(fichero);

    // ver numero de celulas

    int num_celulas;
    num_celulas=Min(trabajos,maquinas);

     // creacion del vector anticuerpo
```

```
72      int dimension= trabajos+maquinas+num_celulas-1;
73      VECTOR_INT vect_mejor=DIM_VECTOR_INT(dimension);
74      VECTOR_INT anticuerpo = DIM_VECTOR_INT(dimension);
75      VECTOR_INT anticuerpo_b = DIM_VECTOR_INT(dimension);
76      VECTOR_INT copia_r = DIM_VECTOR_INT(dimension);
77      VECTOR_INT copia_d = DIM_VECTOR_INT(dimension);
78
79      for (i=0;i<dimension;i++)
80      {
81          if (i<trabajos+maquinas)
82          {
83              anticuerpo[i]=i+1;
84          }
85          else
86          {
87              anticuerpo[i]=0;
88          }
89
90      }
91
92      float obj=evalua(anticuerpo,dimension,trabajos,maquinas,inicial,operaciones);
93      float obj_b;
94      obj_b=obj;
95      int delta=num_celulas*1;
96      VECTOR_INT anticuerpo_prima = DIM_VECTOR_INT(dimension);
97      VECTOR_INT vect_best_pos=DIM_VECTOR_INT(dimension);
98      int vueltas2=0;
99      int vueltas3=0;
100
101     while (vueltas<1000)
102     {
103         vueltas2++;
104         copy_vector(anticuerpo,anticuerpo_prima,dimension);
105         int i,j,k;
106         int aleatorio=0;
107         int nueva_long=dimension;
108         int aux=0,temp=0;
109         float best_pos=0;
110         float eficacia=0;
111
112         for(i=0;i<delta;i++)
113         {
114             aleatorio=rand()%(dimension-i);
115             copia_r[i]=anticuerpo_prima[aleatorio];
116             nueva_long=extract_vector(anticuerpo_prima,dimension-i,aleatorio);
117         }
118
119         for(i=0;i<nueva_long;i++)
120         {
121             copia_d[i]=anticuerpo_prima[i];
122         }
123         for(i=0;i<delta;i++)
124         {
125             copia_d[i+nueva_long]=copia_r[i];
126         }
127
128
129         for(i=0;i<delta;i++)
130         {
131             aux=copia_d[nueva_long+i];
132             temp=extract_vector(copia_d,dimension,nueva_long+i);
133             for(j=0;j<(nueva_long+i);j++)
134             {
135                 insert_vector(copia_d,dimension,aux,j);
136
                    eficacia=evalua(copia_d,dimension,trabajos,maquinas,inicial,operacione
                    s);
137                 if(eficacia>best_pos)
138                 {
139                     best_pos=eficacia;
140                     for(k=0;k<dimension;k++)
141                     {
142                         vect_best_pos[k]=copia_d[k];
```

```
                    }
                }
                temp=extract_vector(copia_d,dimension,j);
            }
            for(k=0;k<dimension;k++)
            {
                copia_d[k]=vect_best_pos[k];
            }
            best_pos=0;

        }

        eficacia=evalua(copia_d,dimension,trabajos,maquinas,inicial,operaciones);
        if(eficacia>obj)
        {
            copy_vector(copia_d,anticuerpo,dimension);
            obj=eficacia;
        }
        if(eficacia>mejor)
        {
            mejor=eficacia;
            for(k=0;k<dimension;k++)
            {
                vect_mejor[k]=copia_d[k];
            }
            iteracion=vueltas;


                vueltas3=vueltas2;
                t_fin=clock();

        }

        vueltas++;
    }

    secs = (double)(t_fin - t_ini) / CLOCKS_PER_SEC;
    printf("%s %d  %f  %.16g ",problema,vueltas3,mejor,secs*1000);
    print_vector(vect_mejor,dimension);


    return mejor;
}



float evalua(VECTOR_INT anticuerpo_copia, int dimension, int trabajos, int maquinas,
MAT_INT inicial, int operaciones)
{
    int i,j,k,h;
    int posfil=0,poscol=0;
    int vacios=0,excep=0;
    int falta_fila=0,falta_columna=0;
    float penalizacion=0;
    float resultado=0;
    int filas[trabajos];
    int columnas[maquinas];
    int celulas[trabajos][maquinas];

    for(i=0;i<trabajos;i++)
    {
        for(j=0;j<maquinas;j++)
        {
            celulas[i][j]=0;
        }
    }

    for (i=0;i<dimension;i++)
    {
        for(h=0;h<dimension-i && anticuerpo_copia[i+h]!=0;h++)
        {
            if(anticuerpo_copia[i+h]<=trabajos)
```

```
215                 {
216                     filas[h-posfil]=anticuerpo_copia[i+h];
217                     poscol++;
218                 }
219                 else
220                 {
221                     columnas[h-poscol]=anticuerpo_copia[i+h];
222                     posfil++;
223                 }
224             }
225             i=i+h;
226             // penalizar si no tiene trabajos o maquinas
227             if(poscol==0 && posfil!=0)
228             {
229                 falta_fila=1;
230             }
231             if(posfil==0 && poscol!=0)
232             {
233                 falta_columna=1;
234             }
235             //rellenar una matriz donde los espacios ocupados por celulas sean igual a 1
236 
237             for(j=0;j<poscol && posfil!=0 && poscol!=0;j++)
238             {
239                 for(k=0;k<posfil;k++)
240                 {
241                     celulas[filas[j]-1][columnas[k]-trabajos-1]=1;
242                 }
243             }
244             poscol=0;
245             posfil=0;
246             for(j=0;j<poscol;j++)
247             {
248                 filas[j]=0;
249             }
250             for(j=0;j<posfil;j++)
251             {
252                 columnas[j]=0;
253             }
254 
255         }
256 
257         for(i=0;i<trabajos;i++)
258         {
259             for(j=0;j<maquinas;j++)
260             {
261                 if(inicial[i][j]==0 && celulas[i][j]==1)
262                 {
263                     vacios++;
264                 }
265                 if(inicial[i][j]==1 && celulas[i][j]==0)
266                 {
267                     excep++;
268                 }
269             }
270         }
271 
272         resultado=(1-(float)excep/(operaciones))/(1+(float)vacios/(operaciones));
273         penalizacion=0.5*resultado*falta_fila+0.5*resultado*falta_columna;
274 
275 
276         return (resultado-penalizacion);
277 
278     }
279 
```