Trabajo Fin de Máster

Optimización del proceso de llenado de placas PCR en secuenciación Sanger

Diego Noceda Davila

Máster en Técnicas Estadísticas

Curso 2018-2019

Propuesta de Trabajo Fin de Máster

Título en galego: Optimización do proceso de enchido de placas PCR en secuenciación Sanger
Título en español: Optimización del proceso de llenado de placas PCR en secuenciación Sanger
English title: Optimization of the PCR plate filling process in Sanger sequencing
Modalidad: A
Autor/a: Diego Noceda Davila, Universidade da Coruña
Director/a: Luisa Carpente Rodríguez, Universidade da Coruña; Silvia Lorenzo Freire, Universidade da Coruña
Breve resumen del trabajo: Se trata de resolver un problema de optimización asociado al proceso de secuenciación de ADN. En concreto, estudiar la colocación de las muestras de ADN en placas PCR teniendo en cuenta las restricciones y los costes asociados. Además, se propondrán heurísticas que permitan obtener buenas soluciones en tiempos razonables.
Recomendaciones:
Otras observaciones: Este TFM ha sido propuesto por Diego Noceda Davila con el consentimiento de Luisa Carpente Rodríguez y Silvia Lorenzo Freire.

Índice general

Resumen

Resumen

En este TFM se resolverá un problema de optimización asociado al proceso de secuenciación de ADN por medio del método Sanger. Aunque el método Sanger consta de varias fases, el problema de optimización que nos interesa está relacionado con la fase de preparación de las placas PCR para ser procesadas en los termocicladores, de forma que se puedan aislar y amplificar determinados fragmentos de ADN. En concreto, nuestro objetivo consistirá en estudiar el problema de optimización relacionado con la colocación de las muestras de ADN (junto con los reactivos correspondientes), en las placas PCR, teniendo en cuenta las restricciones asociadas a las muestras de ADN y a las características de las placas PCR. Adicionalmente, las restricciones de capacidad de procesado del laboratorio exigen generalizar el problema, haciendo necesario adaptar la exploración del espacio de soluciones factibles.

Para lograr este objetivo, se propondrán heurísticas que permitan obtener buenas soluciones en tiempos computacionales razonables. Habrá que hacer, para ello, un estudio de simulación.

Abstract

In this masters degree project an optimization problem associated with the DNA sequencing process by means of the Sanger method will be solved. Although the Sanger method consists of several phases, the optimization problem we are interested in is related to the preparation phase of the PCR plates to be processed in the thermocyclers, so that certain DNA fragments can be isolated and amplified. Specifically, our objective would be to study the optimization problem related to the placement of DNA samples (together with the corresponding reagents) on the PCR plates. Taking into account the restrictions associated with DNA samples and the characteristics of the PCR plates. In addition, the laboratory's processing capacity restrictions demand a generalization of the problem, making it necessary to adapt the space exploration of feasible solutions.

In order to achieve this objective, heuristics that allow to obtain good solutions in reasonable computational times will be proposed. For this, a simulation study will have to be carried out.

Capítulo 1

Introducción

El análisis del ADN permite obtener información acerca de la genética de una persona; esta información puede ser utilizada para diagnosticar enfermedades hereditarias, mutaciones en genes asociados con el desarrollo de desórdenes genéticos o ,incluso, para determinar la ascendencia de una persona. En un sentido más amplio, el conocido como examen genético, formado por el análisis de ADN, ARN, proteínas y procesos metabólicos, identifica cambios en las proteínas, genes y cromosomas. Los resultados de este examen pueden confirmar o descartar la presencia de una condición genética y determinar la probabilidad de que un ser humano desarrolle una enfermedad genética. Existen cientos de pruebas entre las cuales se incluyen las siguientes:

- Exploración prenatal. Utilizada para detectar cambios en genes y cromosomas de los fetos, permite a los progenitores conocer los riesgos de enfermedades genéticas antes del nacimiento del bebé.

- Monitorización en recién nacidos. Que permite identificar y tratar enfermedades genéticas lo antes posible.

- Examen predictivo. Permite detectar mutaciones genéticas asociadas con enfermedades que aparecen a lo largo de la vida del ser humano, muy utilizada cuando un miembro de la familia presenta una enfermedad concreta.

- Huella genética. Utilizada comúnmente para propósitos legales y forenses, permite identificar a un individuo concreto o establecer relaciones biológicas entre dos o más individuos (relaciones de paternidad, maternidad, etc.).

Para llevar a cabo el proceso de análisis del ADN, es necesario disponer de muestras, que pueden ser obtenidas a partir de cualquier célula del cuerpo que tenga núcleo, como, por ejemplo, una gota de sangre, saliva o un pelo. Para poder estudiar la muestra es necesario procesarla. En concreto, debe realizarse la secuenciación del ADN que, básicamente, trata de determinar la secuencia de nucleótidos (Adenina, Timina, Citosina y Guanina) de un fragmento de ADN. Por ejemplo, la Figura 1.1 muestra un ejemplo de una secuencia de ADN.

Existen muchos métodos para realizar la secuenciación de ADN, pero probablemente el más conocido y uno de los más utilizados es el método de secuenciación de Sanger o método de terminación de cadena, desarrollado por Fred Sanger et. al. (1977) [3]. En la actualidad, es posible secuenciar todo el genoma con métodos muy avanzados, conocidos como de nueva generación. Estos métodos son rápidos, de bajo coste y es posible paralelizar las reacciones de secuenciación. Sin embargo, cuando se trata de secuenciar un fragmento individual de ADN, el método de Sanger sigue siendo ampliamente utilizado.

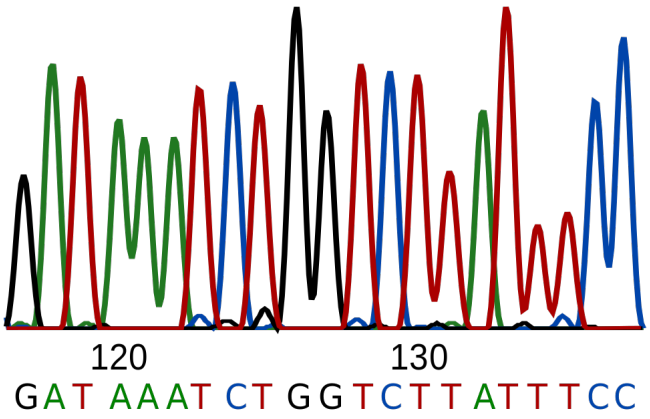


Figura 1.1: Ejemplo de una secuencia de ADN. Fuente: Wikipedia.

1.1. Reacción en cadena de la polimerasa (PCR)

El método de secuenciación de Sanger requiere realizar muchas copias de una región de ADN concreta. La técnica PCR permite justamente crear muchas copias (hasta miles de millones) de una región concreta de ADN *in vitro*, lo que es muy útil en ambientes de laboratorio. La técnica de PCR coloca el ADN y una serie de componentes químicos y se someten a ciclos de calentamiento y enfriamiento que permiten sintetizar el ADN. Los pasos básicos son los siguientes:

- Desnaturalización. Se calienta la reacción para separar las cadenas de ADN. Este paso genera los moldes de ADN utilizados en las fases siguientes.
- Templado. Se enfría la reacción de forma que los cebadores (pedazos cortos de ADN utilizados cono conectores) puedan unirse a las secuencias de los moldes de ADN.
- Extensión. La temperatura se eleva de nuevo para sintetizar las nuevas cadenas de ADN.

Este ciclo se repite unas 30 veces y puede tardar entre 2 y 4 horas según la longitud de la cadena a copiar. Una vez producido el resultado, éste puede ser utilizado de diversas maneras, por ejemplo, en la ciencia forense podría ser visualizado con técnicas de electroforesis y en otro tipo de análisis podría utilizarse para ser secuenciado.

1.2. El problema

En la actualidad, el número de servicios prestados relativos al análisis de ADN se han multiplicado, abarcando desde pruebas para el diagnóstico de enfermedades hasta análisis forense y legal. Además, con el paso del tiempo los procesos son menos costosos, por ejemplo, la secuenciación completa del genoma humano completada en 2003 costó en torno a 100 millones de dólares, mientras que en el año 2015 se consiguió realizar por poco más de 1000 dólares. Este crecimiento ha creado un nicho de mercado en el que cada vez más empresas y laboratorios compiten ferozmente por ofrecer el mejor servicio. En esta situación se encuentra la empresa Health in Code, especializada en, entre otros servicios, el diagnóstico de enfermedades cardiovasculares. Las necesidades de esta empresa para ser competitiva pasan por reducir sus costes de operación y los tiempos de entrega de resultados, lo que repercutirá directamente en la calidad del servicio que proporcionan a sus clientes. Concretamente, la empresa utiliza la técnica PCR para la copia de los fragmentos de ADN y el método de Sanger para su secuenciación.

Figura 1.2: Ejemplo de placa PCR utilizada en laboratorio.

La técnica PCR replica las cadenas de ADN *in vitro*, es decir, no necesita de un organismo vivo para llevar a cabo las copias. Por esta razón, será necesario colocar las muestras de ADN junto con todos los reactivos correspondientes en equipamiento de laboratorio. Concretamente, se utilizan unas placas denominadas placas PCR, las cuales disponen de una serie de huecos o pocillos en los que colocar todo el material. La Figura 1.2 muestra un ejemplo de una de estas placas. La estructura física de estas placas se compone de 8 filas y 12 columnas, obteniendo un total de 96 pocillos que pueden ser utilizados. Además, puede verse cómo las filas y las columnas de la placa están marcadas formando un pequeño sistema de coordenadas.

Cada muestra de ADN, junto con los componentes químicos correspondientes, se coloca en uno de los pocillos disponibles. Si se busca replicar la misma región de ADN en múltiples muestras se utilizarán los mismos componentes químicos en todas ellas, en cuyo caso se dice que esas muestras pertenecen al mismo grupo. Los componentes químicos utilizados producen sus reacciones en ciertos valores de temperatura, por lo que las muestras deberán procesarse a la temperatura concreta que indiquen los químicos. Cabe destacar que, si bien todas las muestras del mismo grupo se procesan a la misma temperatura pues llevan los mismos componentes químicos, dos grupos distintos pueden ser procesados a la misma temperatura.

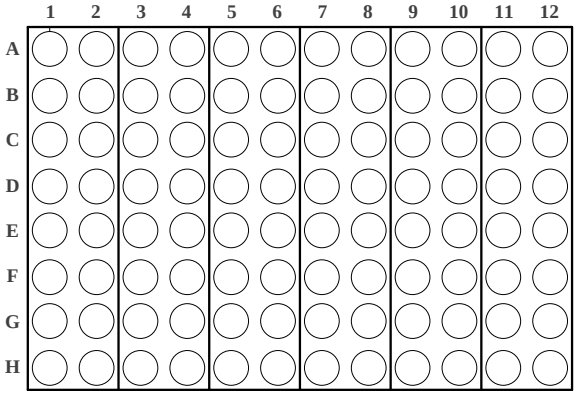


Figura 1.3: Ejemplo de placa PCR vacía dividida en 6 franjas.

La herramienta que permite asignar una temperatura de procesado a los distintos pocillos de

las placas PCR así como realizar el ciclado de la técnica PCR se conoce como termociclador. En este aparato se colocan las placas donde se han organizado las distintas muestras y sus respectivos componentes químicos. Los termocicladores presentan una característica importante, y es que no son capaces de proporcionar temperaturas de procesado individuales a cada pocillo, por el contrario, definen el concepto de franja como unidad organizativa. Estas franjas son divisiones lógicas de las placas PCR, normalmente cada una de las franjas ocupa dos columnas enteras de la placa, es decir, 16 pocillos. La Figura 1.3 muestra un diagrama de una placa PCR vacía, indicando la división lógica en 6 franjas de 16 muestras cada una (2 columnas x 8 filas). Los termocicladores son capaces de establecer una temperatura de procesado distinta para cada una de las franjas, es decir, una placa podría podría llegar a procesar, como máximo, 6 temperaturas distintas. Adicionalmente, debido a su funcionamiento, el termociclador no puede establecer dos temperaturas de procesado que difieran en más de 5º C en dos franjas adyacentes. Por ejemplo, si en la primera franja se establece una temperatura de 50 ºC, la temperatura de la segunda franja no puede superar los 55 ºC en ningún caso.

1.2.1. Estructura del problema

El coste de procesar cada placa es alto tanto en dinero como en tiempo, llegando a costar varios miles de euros cada una. Por esta razón, con el objetivo de reducir costes operacionales, la principal necesidad del laboratorio es utilizar el menor número posible de placas para llevar a cabo su trabajo. Para conseguirlo será necesario encontrar una organización para las muestras que permita aprovechar los pocillos de las placas al máximo. Para ello es necesario tener ciertos factores en cuenta. Por un lado, las restricciones impuestas por el termociclador exigen que las temperaturas se establezcan a nivel de franja, donde dos franjas adyacentes no pueden tener temperaturas de procesado cuya diferencia supere los 5º C. Por otro lado, para asegurar el buen funcionamiento de los experimentos, para cada grupo de muestras presente en una placa es necesario reservar, en la misma placa, un pocillo que contenga una muestra de control de experimento. Por último, como es lógico, es necesario tener en cuenta las dimensiones físicas de las placas a utilizar.

Atendiendo a estas restricciones, se puede plantear el problema de optimización como aquel que busca reorganizar las muestras de ADN en las placas PCR de forma que el número de placas totales utilizadas sea mínimo, siempre teniendo en cuenta las restricciones impuestas por el termociclador y la presencia del indicador de control.

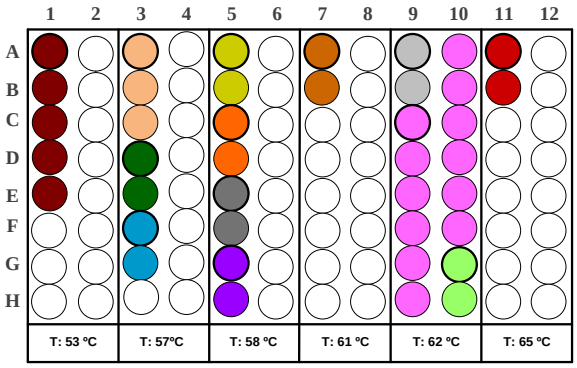


Figura 1.4: Ejemplo de placa PCR con muestras y temperaturas.

Las características del problema recuerdan al conocido como problema de *bin packing*, donde se buscaría colocar de forma óptima las muestras en las placas. El *bin packing* es un conocido problema de optimización NP-duro. Por esta razón, no siempre es posible obtener la solución óptima del problema en un tiempo razonable. Si bien es cierto que existen algoritmos heurísticos para la resolución del bin packing, el problema estudiado considera características y restricciones adicionales, tales como los distintos rangos de temperaturas o las restricciones asociadas a los indicadores de control necesarios.

Estas características particulares plantean un problema de optimización distinto a los estudiados en la literatura, por lo que será necesario plantear una nueva forma de resolverlo.

La Figura 1.4 muestra un diagrama de ejemplo de la configuración de una placa, representando las franjas que componen la placa, así como los distintos grupos de muestras, cada uno con un color de forma que sean fácilmente diferenciables. Además, los huecos reservados para colocar el indicador de control se han marcado con una línea gruesa. Por último, las franjas tienen información asociada sobre su temperatura de procesado, que viene definida por la temperatura de reacción de los componentes químicos añadidos a las muestras, esto es en esencia, que la temperatura de procesado de una muestra viene definida por el grupo al que pertenece. Para realizar la organización ha sido necesario conocer qué muestras de ADN se van a analizar y qué componentes químicos adicionales necesita cada una, pues eso va a determinar su temperatura de procesado así como el grupo al que pertenece cada muestra. Con esa información, es posible ubicar las muestras en las franjas de forma que se cumplan todas las restricciones.

1.2.2. Objetivos del problema

La necesidad principal de la empresa es la de reducir sus costes operacionales. Teniendo en cuenta que el mayor coste en el que incurre es el procesado de las propias placas PCR, valorado en varios miles de euros, parece claro que el primer objetivo del problema de optimización será minimizar el número total de placas utilizadas. Además, se ha visto que por cada grupo de muestras presente en una placa es necesario colocar un indicador de control asociado; pues bien, como la cantidad total de indicadores ubicados a lo largo de las placas depende directamente de la organización de los grupos, se plantea también la necesidad de minimizar el número de pocillos ocupados en total. De esta forma, lo que se hace es minimizar el número de indicadores de control utilizados, algo que asegura que la organización de las muestras es la correcta. Por último, como la empresa recibe nuevas muestras de forma periódica para procesar, ha creído conveniente procurar que el porcentaje de ocupación de las placas sea el mayor posible. La razón es que de esta forma las placas que se consigan llenar al 100 % puedan procesarse directamente y aquellas que tengan huecos libres sean reservadas para ubicar las nuevas muestras.

Capítulo 2

Estado del arte

El trabajo principal que compone el estado del arte del problema a resolver ha sido planteado por Carpente et al. [1]. En él, las autoras proponen un estudio completo del problema, diseñando un modelo de programación lineal, un algoritmo aproximado que consiga buenas soluciones en tiempos razonables y un estudio comparativo de los resultados obtenidos.

2.1. Software comercial

Es habitual que los laboratorios y empresas utilicen sistemas comerciales de administración de información de laboratorio, conocidos como LIMS por sus siglas en inglés. Estos sistemas permiten administrar toda la información relacionada con la actividad del laboratorio, soportando las operaciones habituales, automatizar flujos de trabajo, integrar instrumentación y gestionar bases de datos. Entre otras funciones, los LIMS como el utilizado por la empresa Health in Code permiten organizar las muestras disponibles en placas para realizar la técnica PCR. Sin embargo, esta disposición es muy mejorable y el cálculo de la solución consume una gran cantidad de tiempo. La empresa, tras realizar un estudio de todo el software comercial disponible, ha concluido que no existe una opción que realmente optimice este proceso. Por esta razón, se ha embarcado en el diseño e implementación de una solución a medida que se integre con el resto del sistema. Este desarrollo consta de dos partes, por una parte está la resolución del propio problema de optimización y por otra parte la implementación del software que realizará la integración con el sistema utilizado por la empresa. El estado del arte estudiado se centra en la primera parte, que corresponde con el planteamiento y resolución del problema de optimización relacionado con la colocación de las muestras en placas PCR de forma óptima teniendo en cuenta las restricciones planteadas.

2.2. Problema de programación Lineal

Se ha desarrollado un problema de programación lineal entera que tiene en cuenta todas las restricciones comentadas y cuyo objetivo principal es minimizar el número de placas utilizadas.

Parámetros

- Conjunto de placas disponibles: $\{1, \ldots, p\}$, con índice q.
- Franjas disponibles. Cada placa está compuesta por 6 franjas, por lo que el conjunto resultante será $\{1, \ldots, 6p\}$, con índice l.
- Peso asignado a cada placa en la función objetivo: w_q, para cada $q \in \{1, \ldots, p\}$.

- Conjunto de grupos: $\{1, \ldots, n\}$, con índice i.

- Cantidad de muestras por grupo i: N_i, donde $i \in \{1, \ldots, n\}$.

- Temperatura de procesado de cada grupo T_i. Distintos grupos pueden ser procesados a la misma temperatura; por tanto, se considera el conjunto de temperaturas distintas utilizadas, es decir $\{T^1, \ldots, T^m\} \subseteq \{T_1, \ldots, T_n\}$.

Variables de decisión

Para cada franja $l \in \{1, \ldots, 6p\}$, se definen las siguientes variables:

- Temperatura de procesado de l: t_l.

- Para cada grupo $i \in \{1, \ldots, n\}$ se tiene:

  - Número de muestras del grupo i en la franja l: n_{il}.

  - La variable binaria x_{il}, tal que:
    $x_{il} = \begin{cases} 1 & \text{si el indicador del grupo } i \text{ está presente en la franja } l; \\ 0 & \text{en otro caso.} \end{cases}$

- Para las temperaturas se tiene una variable y_{jl} tal que:
  $y_{jl} = \begin{cases} 1 & \text{si la franja } \ l \text{ tiene asignada la temperatura } T^j; \\ 0 & \text{en otro caso.} \end{cases}$

Objetivo y restricciones

$$\min z = \sum_{i=1}^{n} \sum_{q=1}^{p} w_q \sum_{l=6(q-1)+1}^{6q} n_{il} \tag{2.1}$$

sujeto a:

$$\sum_{i=1}^{n}(n_{il}+x_{il}) \leq 16 \qquad \forall l \in \{1,\ldots,6p\} \tag{2.2}$$

$$t_l - t_{l-1} \leq 5 \qquad \forall l \in \{6(q-1)+1,\ldots,6q-1\}, \forall q \in \{1,\ldots,p\} \tag{2.3}$$

$$t_{l+1} - t_l \leq 5 \qquad \forall l \in \{6(q-1)+1,\ldots,6q-1\}, \forall q \in \{1,\ldots,p\} \tag{2.4}$$

$$\sum_{j=1}^{m} y_{jl} \cdot T^j = t_l \qquad \forall l \in \{1,\ldots,6p\} \tag{2.5}$$

$$\sum_{j=1}^{m} y_{jl} \leq 1 \qquad \forall l \in \{1,\ldots,6p\} \tag{2.6}$$

$$y_{jl} \geq \frac{\sum_{i:T_i=T^j} n_{il}}{\sum_{i=1}^{n} N_i} \qquad \forall l \in \{1,\ldots,6p\}, \forall j \in \{1,\ldots,m\} \tag{2.7}$$

$$\sum_{l=1}^{6p} n_{il} = N_i \qquad \forall i \in \{1,\ldots,n\} \tag{2.8}$$

$$n_{il} - x_{il} \geq 0 \qquad \forall l \in \{1,\ldots,6p\}, \forall i \in \{1,\ldots,n\} \tag{2.9}$$

$$\sum_{l'=6(q-1)+1}^{6q} N_i x_{il'} - n_{il} \geq 0 \qquad \forall i \in \{1,\ldots,n\}, \forall l \in \{6(q-1)+1,\ldots,6q\}, \forall q \in \{1,\ldots,p\} \tag{2.10}$$

$$\sum_{l=6(q-1)+1}^{6q} x_{il} \leq 1 \qquad \forall i \in \{1,\ldots,n\}, \forall q \in \{1,\ldots,p\} \tag{2.11}$$

La función objetivo (2.1) minimiza el número de placas utilizadas, así como la cantidad de huecos ocupados. Para ello, se multiplica la cantidad de muestras de cada placa por un peso, de forma que se penalice más utilizar las placas de las últimas posiciones. Los pesos se toman en orden ascendente, en la práctica se ha decidido utilizar $w_q = q$, $\forall q \in \{1,\ldots,p\}$. Por otro lado, la restricción (2.2) establece el tamaño de cada franja en 16 celdas. Las restricciones (2.3) y (2.4) garantizan el cumplimiento de la restricción de diferencia de temperatura entre franjas adyacentes. Las restricciones (2.5), (2.6) y (2.7) aseguran que las temperaturas de procesado de cada franja coinciden con las temperaturas de los grupos que contienen. La restricción (2.8) comprueba que todas las muestras están colocadas en algún hueco de alguna de las placas de la solución. Finalmente, las restricciones (2.9), (2.10) y (2.11) tratan la presencia de los indicadores de control.

2.3. Análisis de la metaheurística propuesta en la literatura

Las heurísticas son técnicas diseñadas para resolver problemas de forma aproximada cuando no es posible hacerlo de forma exacta, o en los casos en los que el tiempo utilizado para calcular una solución exacta no es razonable. La forma de conseguirlo es relajar en términos de optimalidad o precisión a cambio de velocidad de ejecución. Siguiendo con esta definición, se conocen como metaheurísticas aquellas técnicas que no son dependientes del problema a resolver, sino que establecen el marco que permite definir la heurística concreta en cada caso.

2.3.1. Simulated Annealing

Originalmente desarrollado por Kirkpatrick et al. (1983) [2], el algoritmo se sustenta en una analogía entre el proceso de templado del metal y la optimización combinatoria. Para implementarlo, se define el problema en términos de un espacio de soluciones con un vecindario y una función de coste asociada.

El algoritmo parte de una solución inicial y se mueve de forma aleatoria por el espacio de soluciones, seleccionando un vecino a la solución actual y comparando la función de coste de ambas soluciones. Si la nueva solución aporta un mejor resultado se escoge como nueva solución. En cambio, aunque la nueva solución sea peor, existe la posibilidad de aceptarla igual.

La probabilidad de aceptar un movimiento depende de un parámetro de control T, que se corresponde con la temperatura del metal en la analogía del templado. Cuanto menor es el valor de T, menor es la probabilidad de aceptar los movimientos. Para evitar óptimos locales, el valor inicial de T es alto y desciende tras cada movimiento realizado.

2.3.2. Solución Inicial

Para obtener la solución inicial se ordenan los grupos en orden ascendente atendiendo a la temperatura de procesado de sus muestras. Una vez ordenados los grupos, se colocan las muestras de cada grupo de forma secuencial en las placas, teniendo en cuenta que es necesario reservar un hueco para colocar el indicador del grupo, que será depositado justo después de la última muestra del grupo.

Cuando se colocan todas las muestras de un grupo, puede procederse a colocar el siguiente lo más cerca posible del anterior, incluso en la misma franja si tienen la misma temperatura de procesado. De no ser el caso, nunca se pueden colocar dos grupos en la misma franja.

Si las muestras de dos grupos consecutivos no tienen la misma temperatura de procesado, se comienza a rellenar la siguiente franja a la última utilizada siempre que se cumpla la restricción de diferencia de 5ºC entre franjas consecutivas. De no cumplirse, es necesario dejar franjas vacías en medio hasta satisfacer la restricción de temperatura entre franjas consecutivas.

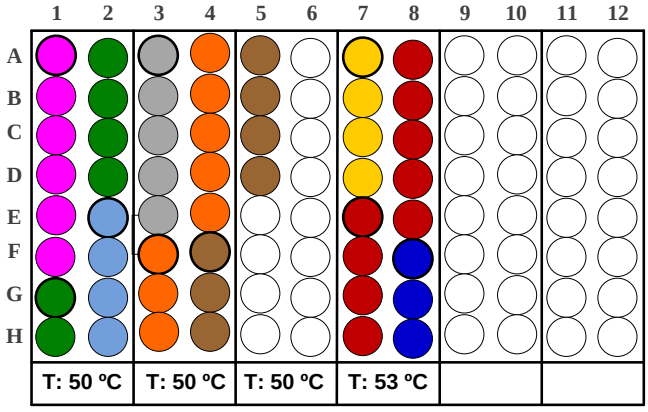


Figura 2.1: Diagrama de ejemplo de solución inicial..

Por ejemplo, la Figura 2.1 muestra un diagrama de la solución inicial de un fichero ficticio, que consta de 52 muestras repartidas en 10 grupos y 2 valores de temperatura. El diagrama muestra cómo, debido a que los grupos han sido ordenados en función de su temperatura de procesado de forma ascendente, las primeras franjas se rellenan con las muestras del grupo de menor temperatura, 50 °C. En el momento en el que se terminan de colocar estas muestras y se selecciona la siguiente temperatura, también se avanza a la siguiente franja disponible, ya que no es posible procesar una misma franja a dos temperaturas distintas.

2.3.3. Movimientos

La definición de los movimientos es un apartado crítico en la implementación del simulated annealing, ya que es la forma de mejorar la solución y recorrer el espacio de soluciones. Se han diseñado los siguientes movimientos:

Intercambio de franjas

Este movimiento selecciona dos franjas de dos placas y las intercambia siempre que sea posible. Nótese que aunque deba cumplirse la restricción de la diferencia de temperatura entre franjas adyacentes, la temperatura de ambas franjas puede diferir.

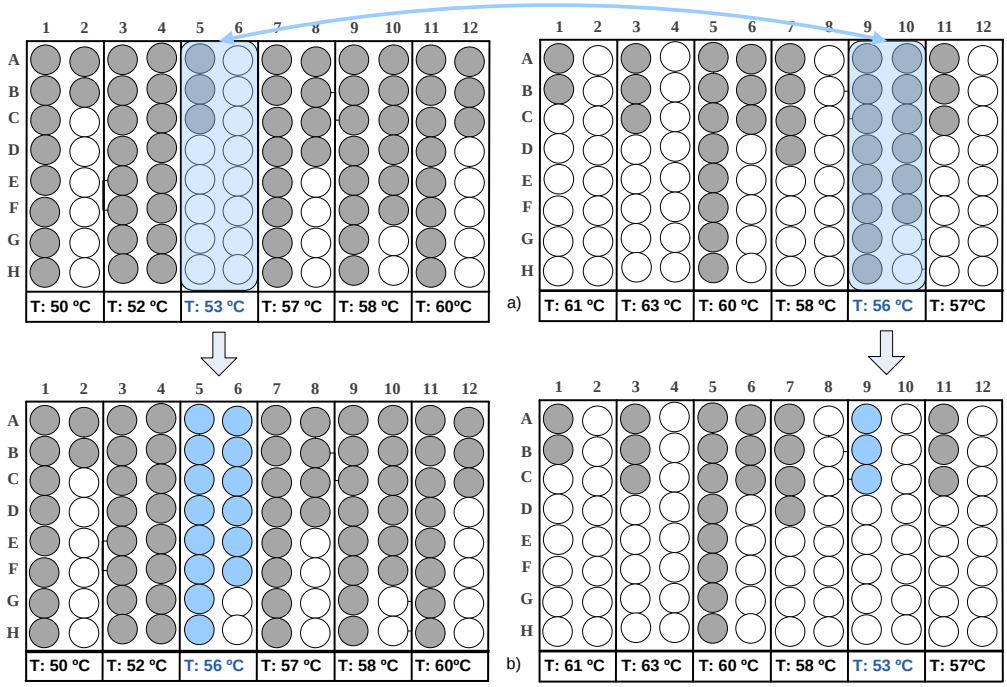


Figura 2.2: Diagrama de movimiento de intercambio de franjas.

La Figura 2.2 muestra un diagrama del funcionamiento del movimiento de intercambio de franjas, donde se pretende intercambiar las dos franjas resaltadas en color azul. Atendiendo tanto a sus valores de temperatura de procesado como a los de sus respectivas franjas adyacentes, se determina que el movimiento es factible. Concretamente, la temperatura de procesado de la franja de la primera placa es de 53 °C y, al ubicarla en la segunda placa se observa que las temperaturas de procesado de las franjas adyacentes anterior y posterior son, respectivamente, 58°C y 57°C, satisfaciendo en ambos casos la condición de que la diferencia entre las temperaturas sea menor o igual a 5°C. En el caso de la franja de la segunda placa, ésta presenta una temperatura de procesado de 56°C y, en su nueva ubicación, las franjas adyacentes anterior y posterior presentan temperaturas de 52 °C y 57 °C, por lo que la diferencia de temperaturas no excede los 5°C en ningún caso.

La Figura 2.2 a) muestra la configuración de las placas antes del intercambio. Atendiendo a la ocupación de las placas, donde los huecos en color gris son aquellos que están ocupados y los huecos de color blanco aquellos vacíos, se concluye que la primera placa tiene $10+16+3+12+14+11=66$ huecos ocupados, lo que corresponde a un 68.75 % de ocupación y, por su parte, la segunda placa presenta una ocupación de $2+3+11+4+14+3=37$, que corresponde a un 38.54 % de ocupación.

La Figura 2.2 b) muestra la configuración después del intercambio. Siguiendo la lógica anterior, se concluye que la primera placa ahora presenta una ocupación de $10+16+14+12+14+11=77$ huecos ocupados, lo que corresponde a un 80.20 % de ocupación y la segunda placa presenta una ocupación de $2+3+11+4+3+3=26$ huecos ocupados, lo que corresponde a un 27.08 % de ocupación.

De cara a decidir si el movimiento ha mejorado o empeorado la solución es necesario valorar la función objetivo. El primer objetivo, relativo al número de placas total utilizadas no cambia, así como

tampoco lo hace la cantidad de huecos ocupados en total ($66 + 37 = 103 = 77 + 26$). Lo que sí ha variado es el porcentaje de ocupación de las franjas, pasando de 68.75 % y 38.54 % antes del intercambio a 80.20 % y 27.08 % después del intercambio. Para valorar este cambio se utiliza el orden lexicográfico definido, el cual permite decidir que el movimiento ha mejorado la solución.

Agrupamiento de muestras

Este movimiento selecciona un grupo que esté disperso en varias placas con el objetivo de reagrupar sus muestras, lo que permite reducir el número de huecos ocupados en total, eliminando indicadores de control de grupo sobrantes. Hay varias formas de seleccionar la forma en la que se lleva a cabo el movimiento, primero se intenta agrupar las muestras en una de las placas donde éstas estén ubicadas, intercambiando muestras de otros grupos de ser necesario, siempre realizando los movimientos mínimos para no alterar la factibilidad de la solución. En caso de no ser posible, puede buscarse otra placa donde haya una franja con temperatura similar y huecos libres para todas las muestras o, incluso, crear una placa nueva donde alojar el grupo.

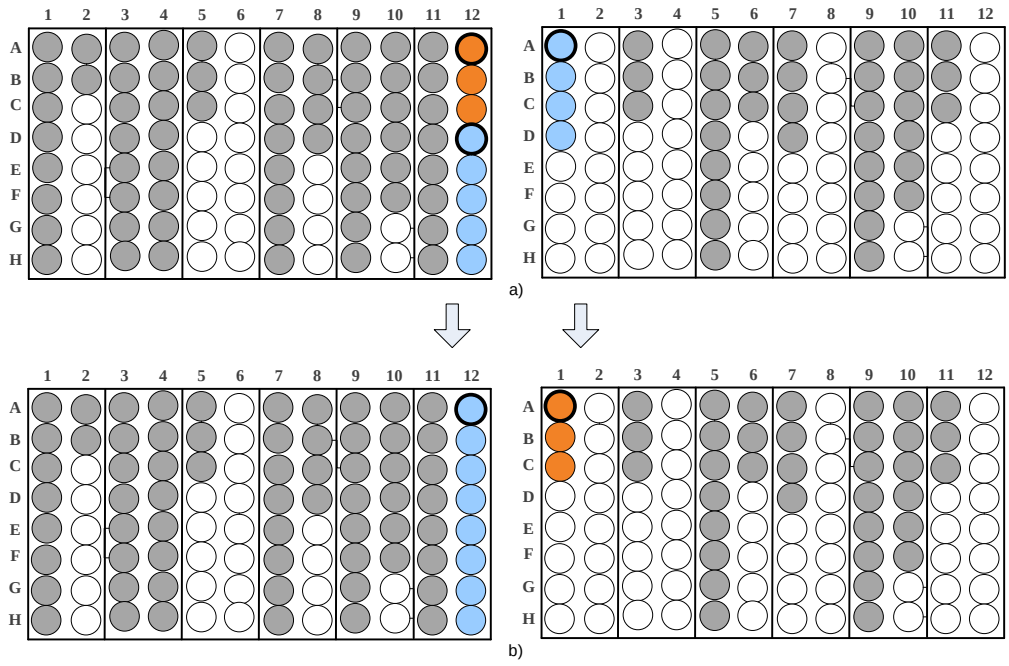


Figura 2.3: Diagrama de movimiento de agrupamiento de muestras.

La Figura 2.3 muestra un diagrama del funcionamiento del movimiento de agrupamiento de muestras. En el ejemplo mostrado se ha encontrado un grupo que está repartido en dos placas distintas (marcado en azul), por lo que reagrupar todas las muestras en la misma placa permitirá reducir el número de indicadores de grupo necesarios (marcados con la línea sombreada), bajando de los 2 actuales a solamente 1. Para llevar a cabo el movimiento se opta por seleccionar otro grupo con la misma temperatura de procesado (marcado en naranja), cuyo tamaño sea similar a la cantidad de muestras que es necesario agrupar. La Figura 2.3 b) muestra el resultado tras realizar el agrupamiento en base a intercambiar las muestras de ambos grupos. La ocupación total de las placas ha pasado de 103 huecos ocupados a 102, por tanto se considera que el movimiento ha mejorado la solución.

Movimiento de escape

En ocasiones, la solución puede alcanzar un punto de estancamiento, en el cual el algoritmo no consigue mejorar. Para intentar solventarlo, se ha planteado un movimiento de escape, que es capaz de modificar la solución lo suficiente para que el algoritmo pueda encontrar nuevas vías de mejora.

2.4. Resultados

Con el objetivo de poder realizar comparativas a lo largo del trabajo, se exponen a continuación los principales resultados alcanzados en el estado del arte. Concretamente, se muestran aquellos obtenidos por la metaheurística propuesta, la cual ha demostrado mejorar las alternativas comerciales y ser competitiva tanto en tiempos como calidad de los resultados.

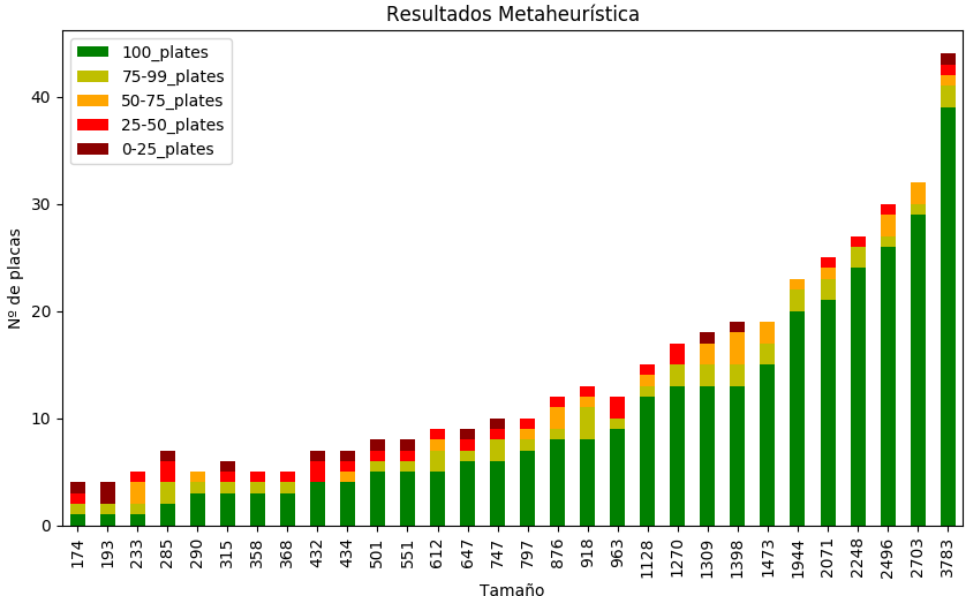


Figura 2.4: Resumen de resultados de la metaheurística.

La Figura 2.4 muestra un resumen de la solución obtenida para cada uno de los ficheros de muestras disponible, la altura de las barras representa la cantidad de placas utilizadas en la solución, que se corresponde con el primer objetivo a minimizar. Cada una de estas barras está dividida en segmentos de distintos colores, los cuales determinan el número de placas ocupadas con un cierto porcentaje de ocupación. Por ejemplo, para el fichero con 285 muestras, la solución ha llenado 2 placas al 100 % (verde), otras 2 placas a un porcentaje entre 75 y 99 % (amarillo), 2 placas a un porcentaje entre 25 y 50 % (rojo claro) y una última placa entre un 0 y un 25 % (rojo oscuro). Esta representación permite visualizar de forma resumida y con facilidad las principales características de las soluciones alcanzadas.

De los tres objetivos definidos para el problema de optimización, el segundo de ellos trata de minimizar la ocupación total de las placas. La Figura 2.5 muestra la ocupación total obtenida por la metaheurística del estado del arte en todos los ficheros disponibles. La importancia de este valor reside en el hecho de que la diferencia entre el número de muestras del fichero y la ocupación total de las placas es justamente la cantidad de indicadores que han sido colocados a lo largo de la solución. Como mínimo, esta diferencia debe ser igual al número de grupos en los que se organizan las muestras, ya que para cada uno es necesario colocar un indicador. Es más, es necesario un indicador por cada placa en la que esté presente un grupo. En resumen, un valor demasiado alto en la ocupación total de las

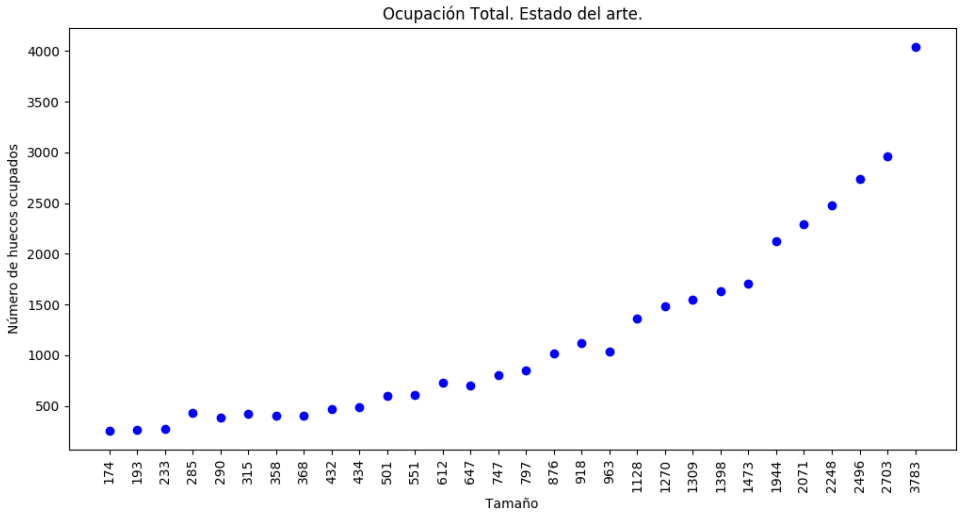


Figura 2.5: Resumen de la ocupación total de la metaheurística.

placas puede sugerir que la organización de las muestras no es buena, ya que se estaría ocupando más espacio del estrictamente necesario.

Figura 2.6: Tiempos de ejecución de la metaheurística.

Por otra parte, la Figura 2.6 muestra un resumen de los tiempos de ejecución que el algoritmo ha necesitado para calcular la mejor solución en cada caso. Cabe destacar que estos tiempos crecen muy rápidamente para los problemas más grandes. Este hecho posiblemente esté relacionado con la necesidad de utilizara el movimiento de escape con mayor frecuencia a medida que el fichero crece. En el Capítulo 4 se analizará el rendimiento del algoritmo en detalle.

Capítulo 3

Análisis del problema

El presente capítulo expone un estudio llevado a cabo principalmente a través de un análisis de los ficheros de entrada disponibles. El objetivo es determinar las principales características y propiedades de dichos ficheros de cara a conseguir mayor conocimiento sobre el problema.

3.1. Análisis de los ficheros de entrada

Todos los ficheros utilizados en presente trabajo parten de sesiones reales de trabajo del laboratorio. En concreto, se trata de los mismos datos utilizados en el estado del arte, lo que permite realizar un análisis y comparativas justas y realistas. Se dispone de 36 ficheros de datos en total, divididos en dos conjuntos, uno de ellos cuenta con 30 ficheros, cada uno con información de una sesión de trabajo del laboratorio, mientras que el otro conjunto se compone de 6 ficheros de 2 o 3 placas escogidas al azar de una sesión del laboratorio.

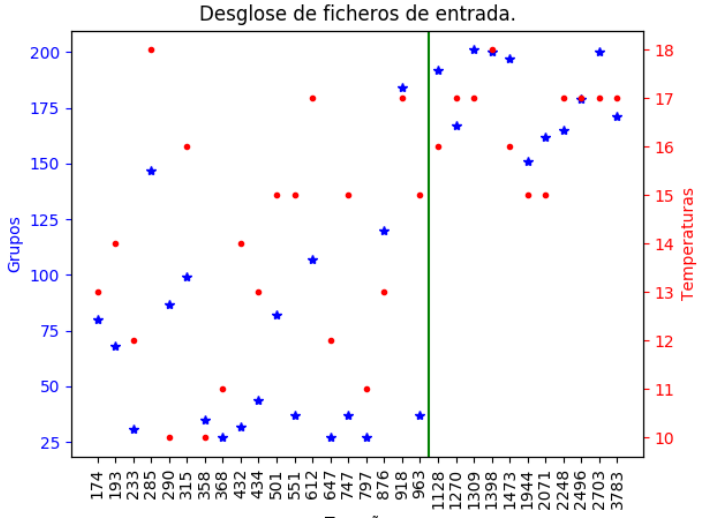


Figura 3.1: Desglose de número de grupos y temperaturas para los ficheros de entrada.

La información utilizada por el laboratorio para el tratamiento de las muestras y aquella necesaria para el proceso PCR es extensa; por tanto, es necesario procesar los ficheros de datos para mantener la información útil desde el punto de vista de la optimización y desechar aquella que resulte irrelevante.

Esta información consta básicamente de tres parámetros: el identificador de una muestra de ADN concreta, el grupo al que esta muestra pertenece y la temperatura de procesado del grupo. El identificador de la muestra es imprescindible, en el sentido de que son las propias muestras los objetos a asignar en las placas que componen la solución. Por otra parte, la información de los grupos es importante para conocer cómo ubicar los indicadores de control. Por último, la temperatura de procesado es importante debido a las restricciones impuestas por los termocicladores. En estas tres características y sus relaciones se basa el análisis llevado a cabo en este capítulo.

La Figura 3.1 muestra un desglose de los parámetros de todos los ficheros disponibles, ordenados según su tamaño; esto es, la cantidad de muestras en cada uno. En azul se indica la cantidad de grupos que existe en el fichero, en rojo la cantidad de temperaturas distintas de procesado. A la vista del gráfico, los ficheros más grandes (a la derecha de la línea verde de separación) se ven más compactos, es decir, en todos los casos el número de grupos oscila entre 150 y 200 y la cantidad de temperaturas entre 15 y 18. Es importante destacar que dentro de este conjunto la cantidad de muestras de cada fichero crece muy rápidamente, aunque no parece afectar a las demás propiedades. Por otra parte, el resto de ficheros tienen un comportamiento más errático, donde no parece haber una relación directa entre el tamaño de la entrada y sus propiedades.

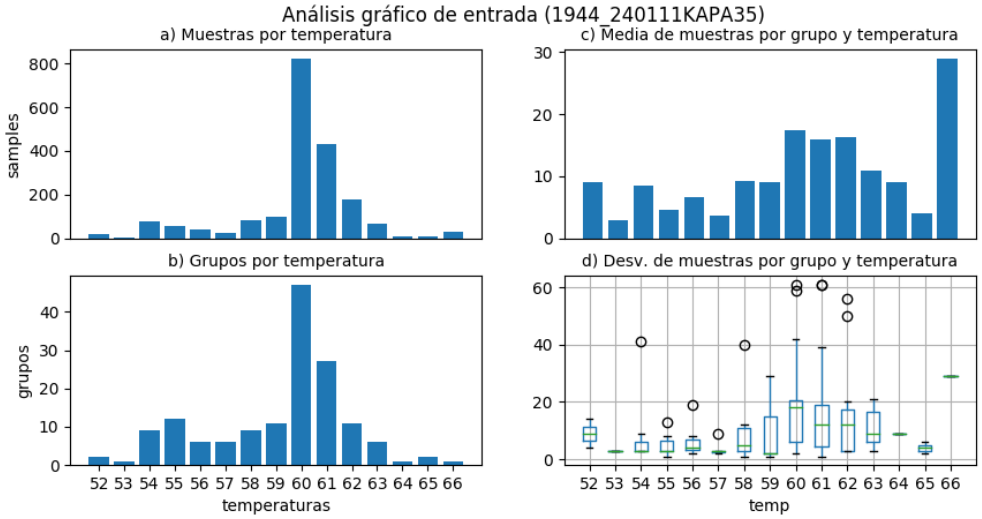


Figura 3.2: Desglose de número de grupos y temperaturas para los ficheros de entrada.

A nivel individual, cada fichero de entrada puede ser analizado por separado. Esto permite conocer la estructura interna del problema en función de la organización de las muestras que lo componen en grupos y temperaturas. Un ejemplo de esta organización puede verse en la Figura 3.2, que corresponde a un fichero de entrada con 1994 muestras, 151 grupos y 15 temperaturas distintas. La Figura 3.2 a) muestra el reparto total de muestras para cada valor de temperatura presente en el fichero. El comportamiento general en todos los ficheros estudiados es que las temperaturas centrales, aproximadamente aquellas comprendidas entre 58 y 62 ºC, son las más comunes y las que concentran el mayor número de muestras. La Figura 3.2 b) muestra el reparto de grupos para cada valor de temperatura. Puede verse que esta gráfica y la anterior son muy similares, comportamiento que se encuentra en varios de los ficheros de entrada, aunque puede diferir.

Por ejemplo, la Figura 3.3 muestra un análisis similar al anterior, en este caso tratando un fichero de 797 muestras. Por el reparto de grupos y temperaturas, la estructura de las gráficas a) y b) en este caso son distintas. La cantidad de grupos por temperatura es estable y pequeña (entre 1 y 3 grupos para todas las temperaturas). Sin embargo, hay grandes diferencias en la cantidad total de muestras por temperatura. Las gráficas c) y d) completan la información del fichero y explican los motivos por

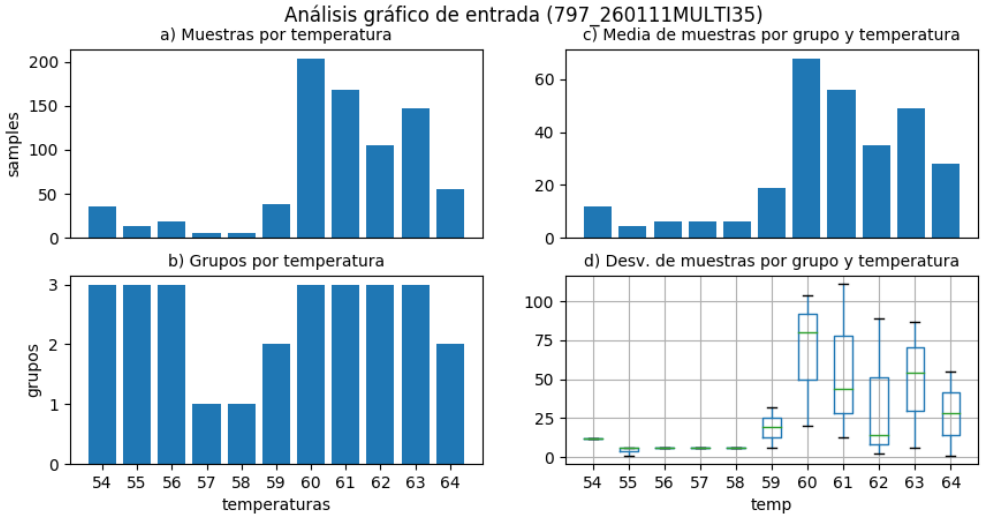


Figura 3.3: Desglose de número de grupos y temperaturas para los ficheros de entrada.

los cuales no se mantiene la estructura. Puede observarse que, en el caso de las temperaturas más bajas (entre 55 y 59 °C), los valores para la media de muestras son bajos y la varianza reducida. En cambio, en los valores de temperatura más altos (a partir de 60 °C), el tamaño de los grupos es mayor en media y hay más variabilidad.

En parte, este estudio caracteriza el comportamiento del fichero de entrada, algo que podría utilizarse para, por ejemplo, realizar una clasificación de los distintos ficheros. Sin embargo, debido a la gran disparidad en las características entre los ficheros disponibles, no existe una estructura clara que permita dicha clasificación o algún tipo de distinción.

Se han realizado análisis gráficos similares a los presentados para todos los ficheros disponibles. Todos ellos se encuentran en el Anexo A.

3.2. Marco de comportamiento

Si bien las características de los ficheros de entrada estudiados hasta ahora no permiten una clasificación detallada de los mismos, es posible trazar marcos en el comportamiento general esperado del problema. Esto se traduce en que es posible acotar ciertos valores de la función objetivo dependiendo de la estructura interna del fichero a resolver. Por ejemplo, suponiendo un fichero muy pequeño de una docena de muestras, ubicarlas a todas ellas en una placa es trivial, es más, una franja es suficiente. Sin embargo, si éstas se reparten en dos temperaturas distintas, será imposible que todas las muestras residan en una misma franja debido a las restricciones del termociclador que impide que una franja se procese a más de una temperatura. Es más, si todas las muestras deben procesarse a una temperatura distinta, serían necesarias todas las franjas de dos placas para organizarlas. Generalizando este aspecto, se concluye que el tamaño de una solución va a ocupar, como mínimo, tantas franjas como valores de temperatura distintos haya presentes. Por otra parte, por cada grupo que se haya definido es necesario tener en cuenta que se debe ubicar un indicador de control en cada placa en la que el grupo esté presente. El estudio de estos parámetros para cada fichero que se pretenda resolver permitirá acotar el comportamiento de la solución óptima. A continuación, se desarrollan dos cotas que permiten modelar distintos valores de la función objetivo.

3.2.1. Cota inferior para el número de placas necesarias en cualquier solución

El valor más importante a la hora de evaluar una solución calculada es el número de placas que ésta utiliza, ya que para el laboratorio cada placa se traduce en un gasto de varios miles de euros. Minimizar el número de placas utilizadas es, por tanto, crítico. A continuación, se presenta una ecuación que modela una cota del número de placas imprescindibles en una solución; esto es, debido a las características del fichero correspondiente, será inviable utilizar menos placas de las que marca la cota. Esta cota servirá como herramienta para comparar soluciones distintas y también como medida de calidad de un determinado algoritmo. La ecuación es la siguiente:

$$LB_p = \lceil \sum_{j=1}^{m} \lceil \sum_{i=1}^{n} (N_i + 1) h(i,j)/16 \rceil / 6 \rceil \tag{3.1}$$

Donde m es el número de temperaturas diferentes presentes en el fichero, n el número de grupos, N_i la cantidad de muestras del grupo i y h una función que permite discernir los sumandos apropiados para cada grupo y temperatura, con la siguiente expresión:

$$h(i,j) = \begin{cases} 1 & \text{si el grupo i tiene la temperatura } T_j \text{ ;} \\ 0 & \text{en otro caso.} \end{cases}$$

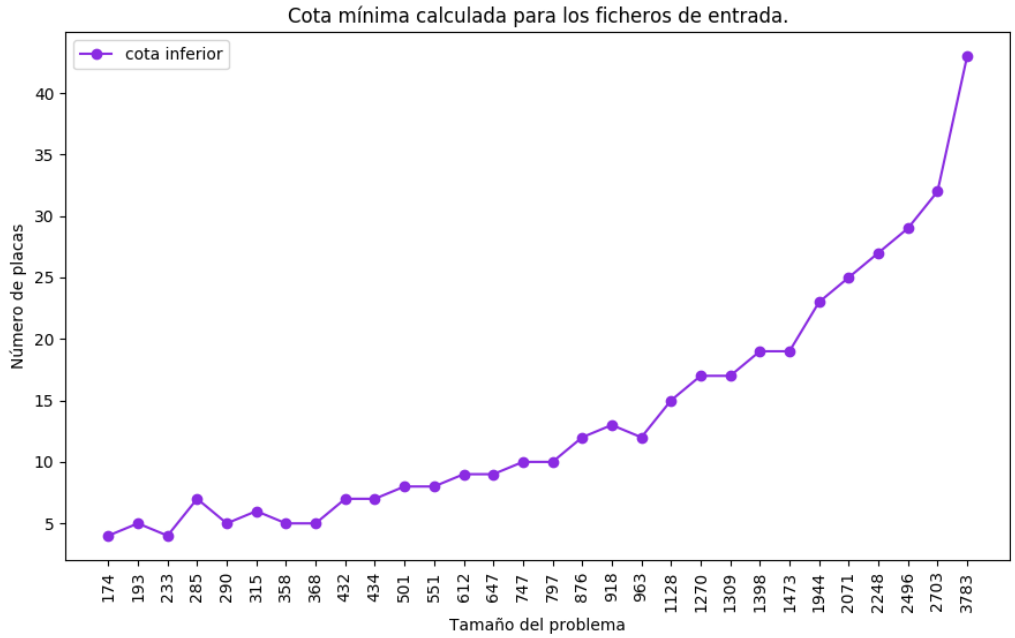


Figura 3.4: Valores para la cota inferior de placas necesarias en los ficheros de entrada.

Para el cálculo de la cota se tienen en cuenta los parámetros estudiados, es decir, el tamaño del problema, la distribución de los grupos y temperaturas y, adicionalmente, las restricciones de tamaño de las placas, tanto en número de franjas como la capacidad de las mismas. Para realizar el cálculo, se organizan las muestras en función de su temperatura de procesado, utilizando para ello la función auxiliar h, luego se suman y se calcula el número de franjas necesario para ubicarlas a todas. Haciendo esto con todas las temperaturas presentes en el fichero es posible conocer el número mínimo de franjas

a utilizar. Por último, teniendo en cuenta la capacidad de cada placa para albergar 6 franjas se obtiene el número mínimo de placas. Como nota adicional, se asume que cada grupo necesita únicamente un indicador de control, cosa que será cierta en la gran mayoría de casos, aunque algunos ficheros estudiados contienen grupos realmente grandes que por sí solos ocupan más de una placa.

En cualquier caso, es necesario tener en cuenta que en ocasiones no será posible que la solución real alcance dicho valor. Por ejemplo, en un problema con 6 temperaturas distintas, donde la cantidad de muestras de cada temperatura (contando los indicadores de grupo) no supere 16 en ningún caso, esto es, todas las muestras de cada temperatura puedan ser ubicadas en una sola franja, la cota calculada tendría el valor 1, ya que una placa es suficiente para almacenar 6 franjas. Sin embargo, es posible que estas franjas no puedan ser colocadas de forma continua, por ejemplo por no cumplir la restricción de diferencia de temperatura entre franjas adyacentes. En este caso, será necesario utilizar más de una placa para acomodar todas las muestras de la solución. Si bien en la práctica no se ha detectado un caso como el planteado, es posible que una organización concreta de las franjas y sus temperaturas haga necesario incluir huecos vacíos para acomodar todas las temperaturas posibles.

La Figura 3.4 muestra la cota calculada para los ficheros de entrada. Como es lógico, el valor de la cota crece a medida que es necesario acomodar una cantidad mayor de muestras. Sin embargo, se observa que no es una función estrictamente creciente, ya que ficheros de tamaño similar pueden tener características muy distintas que modifiquen el comportamiento.

3.2.2. Cota inferior para la ocupación de las placas

El segundo objetivo del problema es minimizar la ocupación total de las placas utilizadas, entendiendo por ocupación total el número de huecos ocupados en cada una de las placas que componen la solución. Si bien es cierto que el número de muestras para un problema dado es estático, la cantidad necesaria de indicadores de control de grupo no lo es, ya que depende directamente de la cantidad de grupos presentes, así como de su colocación en las placas. Es posible estudiar la ocupación mínima que va a requerir una solución ante un problema dado, ya que el fichero de entrada permite conocer tanto la cantidad de muestras presentes como el número de grupos que lo componen. La expresión que modela esta cota es sencilla en su forma básica y se muestra a continuación:

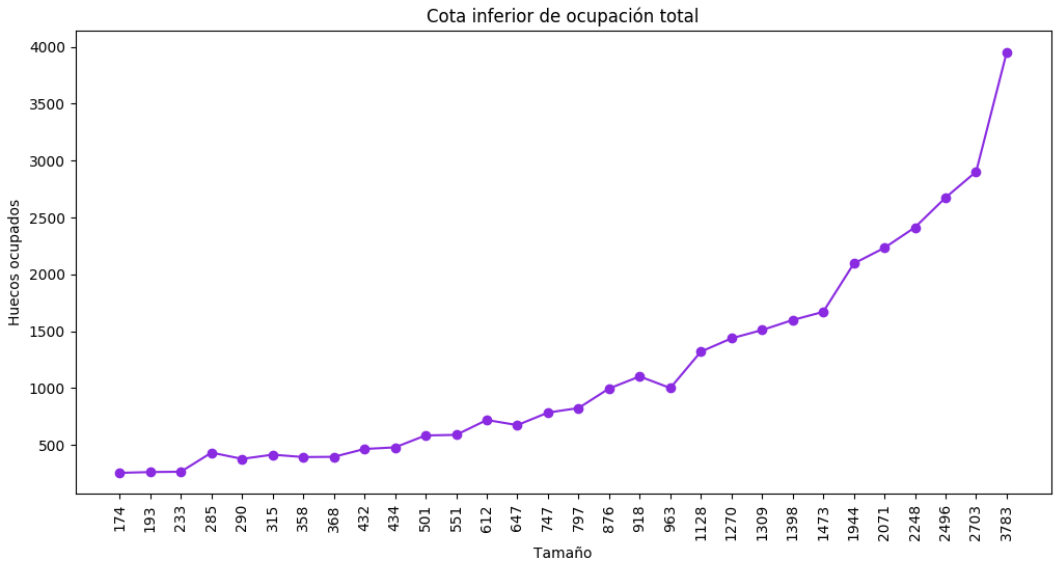


Figura 3.5: Valores para la cota inferior de placas necesarias en los ficheros de entrada.

$$LB_{wb} = \sum_{i=1}^{n} N_i + n \tag{3.2}$$

La cota asume que es posible ubicar cada grupo utilizando únicamente 1 indicador de control. Sin embargo, ya se ha comentado que en la práctica puede no cumplirse para todos los grupos, lo que obligaría a utilizar más de 1 indicador de control. En cualquier caso, esta cota permite comparar la solución obtenida por distintos algoritmos y medir la calidad de los mismos.

La Figura 3.5 muestra un gráfico con la cota de ocupación total calculada para todos los ficheros de entrada disponibles. Lógicamente, la cota crece a medida que los ficheros son de mayor tamaño y los pequeños picos o valles son debidos principalmente a la cantidad de grupos presentes en un fichero, que es una característica muy variable. Por otra parte, la cota calculada en esta sección tiene un comportamiento muy parecido a la cota para el número de placas necesarias calculada en la sección anterior. Esto se debe a que las variables que rigen ambas cotas son principalmente las mismas, el tamaño del fichero y la cantidad de grupos. Cabe destacar que la cantidad de temperaturas distintas en esta cota no tiene ningún efecto.

Capítulo 4

Análisis de la heurística del estado del arte

Con el fin de plantear posibles mejoras al algoritmo y estudiar la posibilidad de adaptarlo a escenarios más complejos, se ha realizado un estudio exhaustivo del algoritmo, centrando el análisis en el rendimiento de los movimientos implementados. Para que el análisis sea coherente, todas las ejecuciones realizadas tienen, salvo que se indique lo contrario, los mismos parámetros de configuración.

- Temperatura inicial: 100 °C
- Temperatura final: 10^{-9} °C
- Parámetro de enfriamiento (α): 0.9
- Número fijo de iteraciones: 1000

En cuanto a la proporción de los movimientos, ésta se adaptará al estudio concreto que se pretenda realizar en cada caso.

4.1. Solución Inicial

Para la construcción de la solución inicial, los grupos de muestras se ordenan según su temperatura de procesado y se evitan los saltos de placa innecesarios, con el objetivo de intentar minimizar el número total de placas a utilizar. La Figura 4.1 muestra una comparativa entre el número de placas calculado por la solución inicial y la cota inferior presentada en el Capítulo 3. Puede observarse que, en la mayoría de los casos, el resultado obtenido por la solución inicial y la cota coinciden, lo que permite concluir que la solución inicial es buena. Además, atendiendo a los resultados finales del algoritmo, no se mejora este objetivo en ningún caso, si bien es cierto que tampoco se empeora. Esto es importante, pues permite concluir que la metaheurística tiene poca capacidad en este sentido, por lo que todos los cambios que se planteen de cara a la reducción del número de placas a utilizar deben ir enfocados en el cálculo de la solución inicial.

4.2. Intercambio de franjas

El movimiento de intercambio de franjas es el de mayor peso, siendo su proporción de entre 0.8 y 1 en función del problema a resolver. Esto significa que, de cada 100 movimientos realizados, aproximadamente 80 serán intercambios de franjas, lo que lo convierte en el principal foco de atención a la hora de analizar el rendimiento del algoritmo.

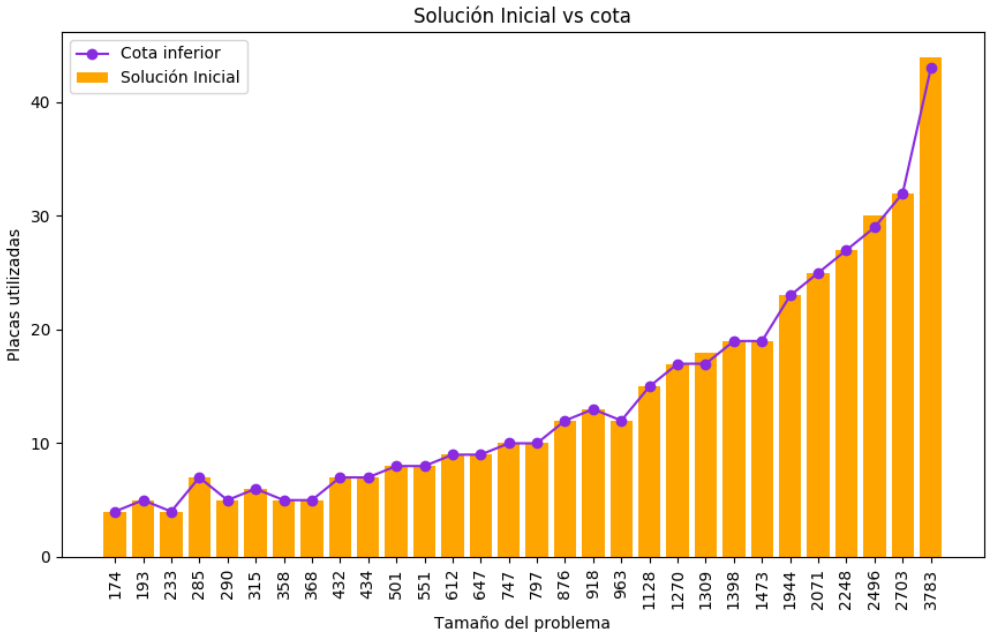


Figura 4.1: Comparativa de solución inicial vs cota inferior.

Para analizar el rendimiento del movimiento de intercambio de franjas, se han ejecutado los 30 ficheros de entrada disponibles, 10 veces cada uno con los parámetros definidos y con una proporción del 100 % al movimiento de intercambio, lo que significa que únicamente se ejecutará este movimiento.

La Tabla 4.1 muestra un resumen del comportamiento del movimiento de intercambio de franjas para todos los ficheros de entrada. En esta prueba, el peso para el movimiento de intercambio es del 100 %, de forma que el total de movimientos se corresponde con movimientos de intercambio. Adicionalmente, cada una de las filas de la tabla corresponde a los valores medios de 10 ejecuciones. Las columnas representan el tamaño del problema en número de muestras, el número de iteraciones del algoritmo ejecutadas, la cantidad de movimientos totales ejecutados (depende directamente del número de iteraciones), la cantidad de movimientos que mejoran la solución, la cantidad de movimientos que empeoran la solución y la cantidad de movimientos que no son factibles.

El dato que arroja la conclusión más importante es el correspondiente a la cantidad de movimientos infactibles, que en ningún caso desciende de un 65 %. Esto significa que, de cada 100 movimientos de intercambio intentados, 65 no pueden ser realizados por no poder cumplir con las restricciones del problema. También se observa que la cantidad de movimientos infactibles asciende a medida que el tamaño del problema es mayor. En consecuencia, la cantidad de movimientos factibles (suma de movimientos que mejoran o empeoran la solución) disminuye. Esta problemática con la infactibilidad explica la razón por la que el algoritmo requiere de más iteraciones para resolver los problemas de mayor tamaño.

Por otra parte, en cuanto a la relación entre los movimientos que mejoran la solución y los que la empeoran, la proporción de mejora es siempre superior.

De la misma forma, la Figura 4.2 muestra un desglose de la infactibilidad del movimiento de intercambio de franjas. Concretamente, la Figura 4.2 a) representa la comparativa entre movimientos factibles e infactibles, donde se observa la forma en la que decrece la proporción de movimientos factibles a medida que crece el tamaño del problema. Por su parte, la Figura 4.2 b) desglosa las causas de la infactibilidad del movimiento, las cuales se han clasificado en dos tipos: asignación y temperatura. Los problemas de temperatura representan aquellos movimientos que no pueden ser llevados a cabo porque,

Tamaño	Iters	Movs. Totales	Movs. Mejora	Movs. Empeoramiento	Movs. Infactibles
174	1000.0	263000.0	74417.2 (28.29 %)	13449.6 (5.11 %)	175133.2 (66.8 %)
193	1000.0	263000.0	52136.3 (19.82 %)	8022.2 (3.05 %)	202841.5 (77.0 %)
233	1000.0	263000.0	52975.7 (20.14 %)	5764.4 (2.19 %)	204259.9 (77.6 %)
285	1000.0	263000.0	36241.0 (13.77 %)	4597.8 (1.74 %)	222161.2 (84.3 %)
290	1000.0	263000.0	44698.7 (16.99 %)	5065.7 (1.92 %)	213235.6 (81.0 %)
315	1000.0	263000.0	56765.7 (21.58 %)	12625.7 (4.80 %)	193608.6 (73.7 %)
358	1000.0	263000.0	41141.0 (15.64 %)	3064.7 (1.16 %)	218794.3 (83.0 %)
368	1000.0	263000.0	76513.5 (29.09 %)	15101.4 (5.74 %)	171385.1 (65.0 %)
432	1000.0	263000.0	40052.6 (15.22 %)	6588.9 (2.50 %)	216358.5 (82.1 %)
434	1000.0	263000.0	34772.4 (13.22 %)	4897.6 (1.86 %)	223330.0 (85.0 %)
501	1000.0	263000.0	29593.5 (11.25 %)	3695.4 (1.40 %)	229711.1 (87.0 %)
551	1000.0	263000.0	30688.8 (11.66 %)	3722.9 (1.41 %)	228588.3 (87.0 %)
612	1000.0	263000.0	29058.6 (11.04 %)	2815.4 (1.07 %)	231126.0 (88.0 %)
647	1000.0	263000.0	26644.1 (10.13 %)	3385.8 (1.28 %)	232970.1 (88.7 %)
747	1000.0	263000.0	23652.4 (8.99 %)	1797.1 (0.68 %)	237550.5 (90.3 %)
797	1000.0	263000.0	31694.6 (12.05 %)	4023.2 (1.52 %)	227282.2 (86.3 %)
876	1000.0	263000.0	22869.7 (8.69 %)	3191.6 (1.21 %)	236938.7 (90.0 %)
918	1000.0	263000.0	20362.4 (7.74 %)	2618.3 (0.99 %)	240019.3 (91.0 %)
963	1000.0	263000.0	20703.3 (7.87 %)	1517.9 (0.57 %)	240778.8 (91.8 %)
1128	1000.0	263000.0	17123.1 (6.51 %)	1890.1 (0.71 %)	243986.8 (93.0 %)
1270	1000.0	263000.0	15536.2 (5.90 %)	1951.6 (0.74 %)	245512.2 (93.0 %)
1309	1000.0	263000.0	14427.1 (5.48 %)	1689.9 (0.64 %)	246883.0 (94.0 %)
1398	1000.0	263000.0	14811.2 (5.63 %)	1734.7 (0.65 %)	246454.1 (94.0 %)
1473	1000.0	263000.0	12994.5 (4.94 %)	1149.3 (0.43 %)	248856.2 (95.0 %)
1944	1000.0	263000.0	11258.6 (4.28 %)	1083.5 (0.41 %)	250657.9 (95.0 %)
2071	1000.0	263000.0	9990.9 (3.79 %)	856.8 (0.32 %)	252152.3 (96.0 %)
2248	1000.0	263000.0	9825.5 (3.73 %)	944.3 (0.35 %)	252230.2 (96.0 %)
2496	1000.0	263000.0	8300.9 (3.15 %)	653.9 (0.24 %)	254045.2 (96.9 %)
2703	1000.0	263000.0	8636.7 (3.28 %)	775.1 (0.29 %)	253588.2 (96.0 %)
3783	1000.0	263000.0	6122.5 (2.32 %)	444.4 (0.16 %)	256433.1 (97.6 %)

Tabla 4.1: Tabla resumen de movimiento de intercambio de franjas.

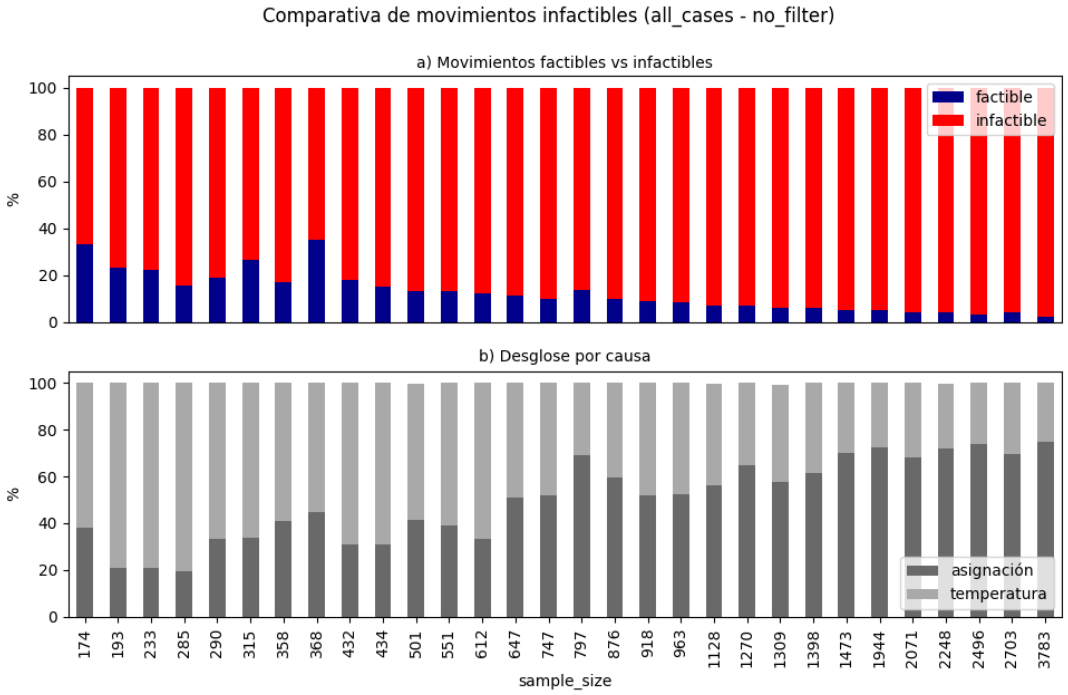


Figura 4.2: Comparativa de movimientos infactibles en intercambio de franjas.

de realizarse el intercambio, no se cumpliría la restricción de la diferencia de temperatura entre franjas adyacentes. Por otra parte, los problemas de asignación son aquellos relacionados con la restricción de presencia de los indicadores de grupo. Estos problemas aparecen cuando las franjas intercambiadas cambian la placa en la que son colocadas y, por tanto, requieren de un indicador faltante.

La figura también muestra cómo la proporción de estas causas es distinta en función del tamaño de los ficheros. Esto se debe a la estructura interna de los problemas, ya que, como se ha visto en el Capítulo 3, en general la mayor concentración de muestras se encuentra en un rango de temperaturas concreto, aproximadamente entre 58 y 62 °C. Por esta razón, en los problemas grandes la probabilidad de que dos franjas escogidas, así como las franjas adyacentes, tengan una temperatura similar aumenta, lo que permite que puedan ser intercambiadas.

En cualquier caso, se concluye que la proporción de movimientos infactibles es muy alta en general, lo que puede traducirse en problemas de eficiencia del algoritmo.

4.3. Agrupamiento de muestras

Según lo indicado en el estado del arte, el peso que se le otorga a este movimiento depende del fichero con el que se trabaja, variando desde el 0 hasta el 20 %. La razón de esta variación reside en el comportamiento interno del movimiento y en cómo afecta a la solución, ya que es un movimiento que sólo se considera fallido cuando encuentra un grupo que no puede ser reagrupado. En el resto de los casos suele mejorar la solución, ya que generalmente es capaz de eliminar un indicador de grupo, con lo que se reduce la cantidad de celdas ocupadas en las placas. En ocasiones, esta mejora implica que ciertas franjas pierdan ocupación, pudiendo pasar de estar completamente llena a ocupada en un porcentaje muy alto. Esto choca con el intento de llenar el mayor número de placas posible, puesto que una placa sólo puede estar llena al 100 % si todas las franjas que la componen también lo están. Esta es la razón por la cual el porcentaje de uso de este movimiento es bajo, sobre todo en los problemas más pequeños. En cambio, en los problemas grandes es interesante asignar un mayor peso a este movimiento, puesto

que aporta flexibilidad a la solución, ya que el hecho de reagrupar muestras puede permitir que más movimientos de intercambio sean factibles por razones de asignación.

4.4. Movimientos de escape

La idea del movimiento de escape es la de proveer al algoritmo de un mecanismo que permita explorar nuevas regiones del espacio de soluciones. Para ello, se ha implementado como una sucesión de multitud de intercambios de franjas tras los cuales se ignora la función objetivo, de modo que se permita total libertad en la exploración. Nótese que se exige en todo momento que los movimientos sí cumplan las restricciones impuestas por el problema.

En algunos problemas, este movimiento tiene un efecto positivo en la solución, obteniendo mejoras en la solución final. Sin embargo, los problemas de eficiencia analizados para el movimiento de intercambio de franjas multiplican su importancia en este movimiento, ya que para conseguir una variación sustancial en la solución que permita recorrer nuevos caminos es necesario ejecutar decenas de miles de intercambios. Esto lleva a que este movimiento sea muy costoso en tiempo e influya negativamente en el rendimiento general del algoritmo.

4.5. Estrategias para intentar acelerar la metaheurística del estado del arte

Tras analizar el funcionamiento del algoritmo planteado en el estado del arte y de detectar fuentes de problemas de rendimiento, en la presente sección se proponen una serie de modificaciones al algoritmo. Dado que el movimiento con mayor peso es el intercambio de franjas, los intentos de mejora se centrarán en él, atendiendo a las causas de fallo detectadas por separado.

4.5.1. Filtrado por temperatura

Los fallos relativos a la restricción de temperatura entre franjas adyacentes son aquellos producidos cuando la temperatura de procesado de alguna de las franjas que se han intentado intercambiar difiere más de 5ºC de sus franjas adyacentes. Para intentar reducir el impacto de estos fallos, se ha definido un nuevo parámetro para el movimiento de intercambio de franjas. Se trata de un umbral definido para ser comparado con la diferencia de temperatura de las franjas que se quieren intercambiar. Si dicha diferencia es menor que el umbral el movimiento sigue adelante y si la diferencia es mayor se seleccionan dos nuevas franjas y se compara la nueva diferencia con el umbral. Este paso se repite hasta que se seleccionen dos franjas que cumplan con el requisito.

La motivación del umbral es intentar prevenir las diferencias de temperatura que las franjas se van a encontrar en los nuevos huecos que se les asignen.

La selección de un valor concreto es compleja, ya que un valor muy bajo haría que el algoritmo pierda flexibilidad y la capacidad de una buena exploración del espacio de soluciones, lo que repercutirá directamente en la calidad de la solución obtenida. Por el contrario, un valor muy alto no variará el comportamiento original del algoritmo y será irrelevante. Finalmente, el valor escogido es de 3, el cual se ha decidido mediante un experimento práctico sobre los datos reales. Para ello se ha ejecutado el algoritmo sobre todos los ficheros utilizando varios umbrales distintos, por ejemplo las Figuras 4.3 y 4.4 presentan resultados para los valores 3 y 4. Puede observarse que la proporción de movimientos factibles e infactibles aumenta en todos los casos, si bien la tendencia descendiente sigue presente. En el caso de los ficheros más grandes, la proporción de fallos debidos a la diferencia de temperaturas es muy pequeño. Es importante destacar el hecho de que los resultados con el umbral con valor 4 son peores que aquellos con umbral con valor 3, lo que refuerza la idea de que un umbral alto tendrá un desempeño similar al algoritmo original.

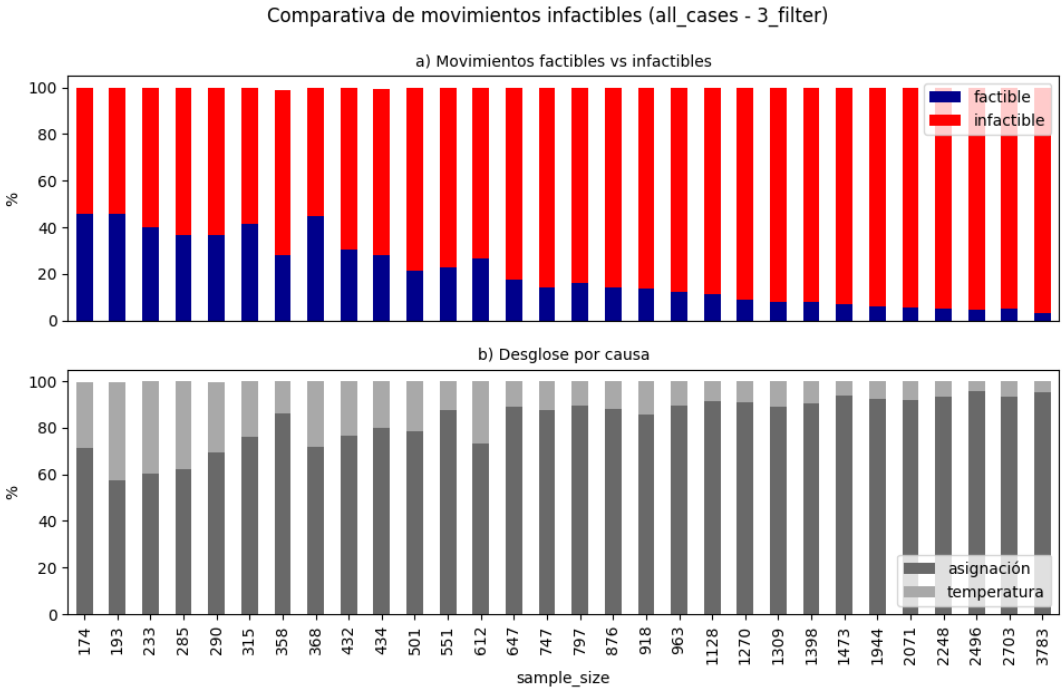


Figura 4.3: Comparativa de movimientos infactibles en intercambio de franjas con un valor de umbral de 3.

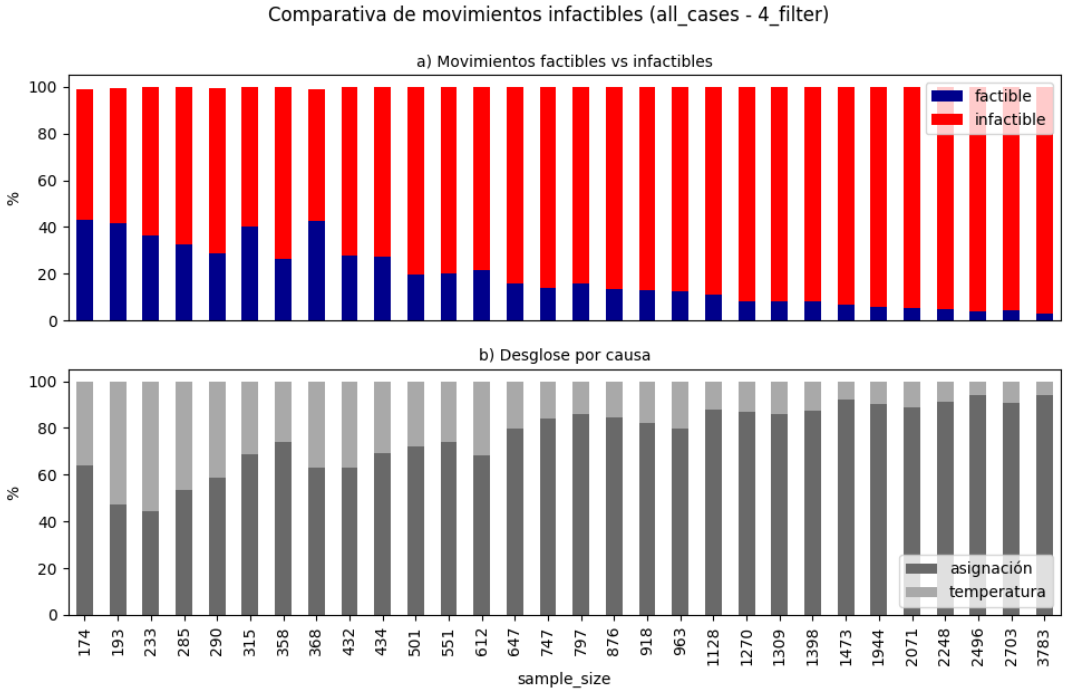


Figura 4.4: Comparativa de movimientos infactibles en intercambio de franjas con un valor de umbral de 4.

4.5.2. Eliminando problemas de asignación

Los fallos de asignación están relacionados con la restricción de presencia de indicadores en las placas, como, por ejemplo, cuando una franja no puede salir de la placa en la que está colocada debido a problemas con los indicadores de los grupos contenidos en ella. De esta forma, si una franja está ocupada por completo y uno de los grupos que contiene no está acompañado por su indicador, esa franja no podrá ser colocada en una nueva placa, ya que no se estaría cumpliendo una de las restricciones del problema.

La solución planteada busca aportar flexibilidad a las franjas para poder moverse libremente sin complicaciones. Para ello, es necesario realizar cambios en la solución inicial, no permitiendo dividir grupos en varias franjas (salvo que el tamaño de los grupos obligue a hacerlo) y habilitando completar franjas hasta 15 huecos para los grupos más grandes, lo que permitiría que estas franjas siempre pudieran ser reasignadas al tener espacio libre suficiente para colocar el indicador de grupo correspondiente.

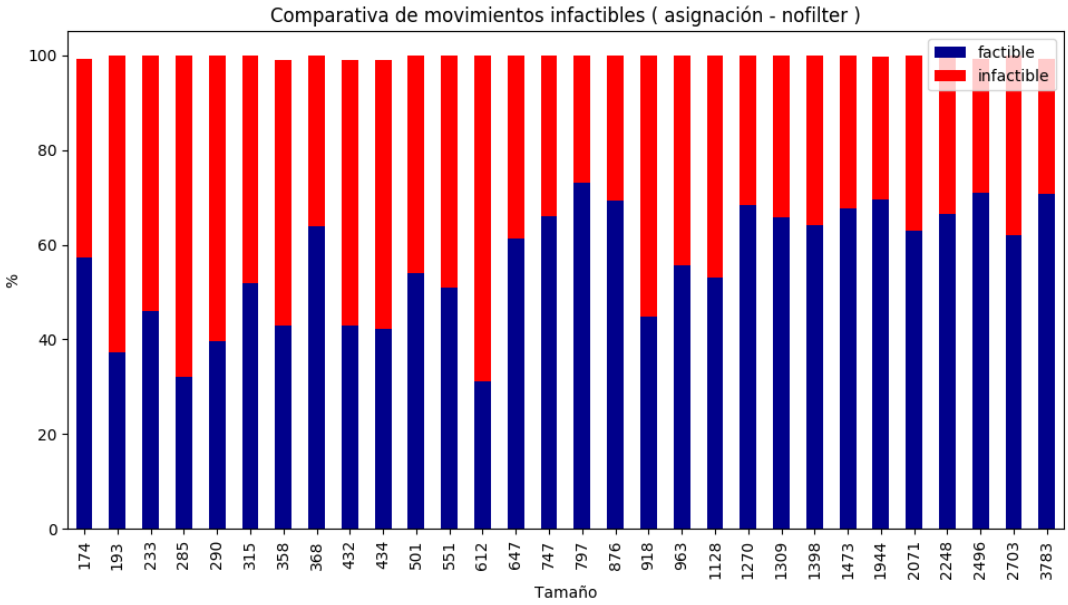


Figura 4.5: Comparativa de movimientos infactibles tras eliminar problemas de asignación.

El resultado es el esperado, consiguiendo eliminar por completo los fallos relativos a la asignación de indicadores. La Figura 4.5 muestra la comparativa de movimientos factibles y no factibles tras realizar esta modificación. Lógicamente, el impacto de esta mejora es mayor en aquellos ficheros donde la proporción de fallos por asignación también lo es, pero la mejora es positiva en general. Es posible combinar las dos propuestas planteadas y cabe suponer que la proporción de movimientos factibles va a ser bastante alta ya que éstas son complementarias. La Figura 4.6 muestra este resultado, donde se observa que, en el peor caso, la proporción de movimientos factibles está en torno al 70 % y, en el mejor caso, en torno al 90 %. Esto significa que ahora, de cada 100 movimientos que se intenten realizar, entre 70 y 90 serán llevados a cabo.

La modificación planteada en esta sección, que ha eliminado satisfactoriamente todos los fallos relativos a la asignación de indicadores, se realiza directamente sobre el cálculo de la solución inicial y se ha demostrado la importancia de que ésta sea de gran calidad, ya que la metaheurística no parece capaz de mejorar el primer objetivo del problema, es decir, reducir el número de placas utilizadas.

La Figura 4.7 muestra la comparativa entre la solución inicial conseguida y la solución inicial original planteada en el estado del arte. Las líneas punteadas marcan los resultados obtenidos por la

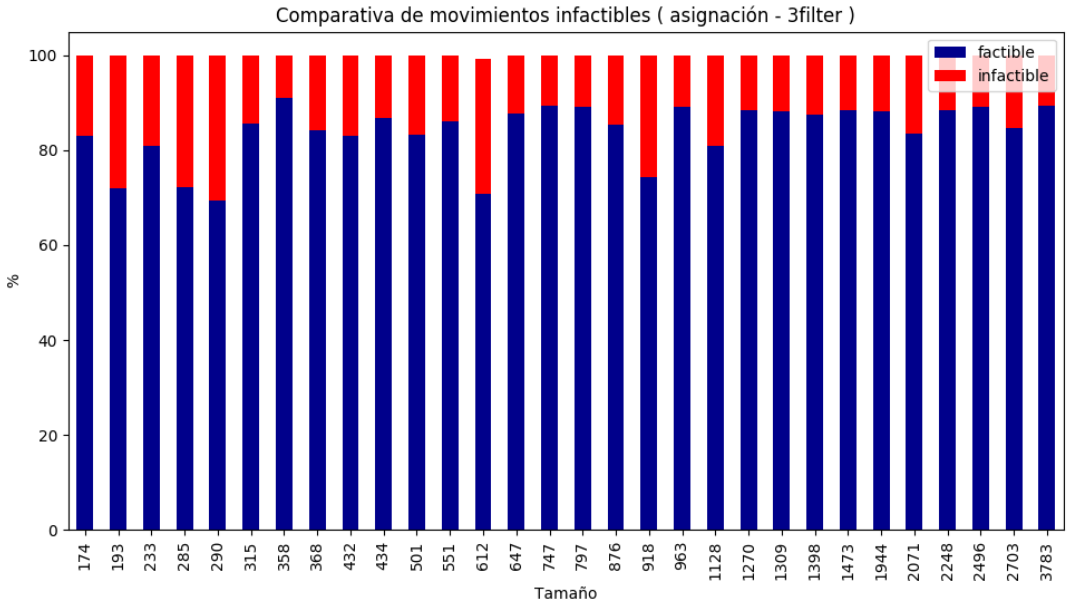


Figura 4.6: Comparativa de movimientos infactibles tras eliminar problemas de asignación y limitar las diferencias de temperatura.

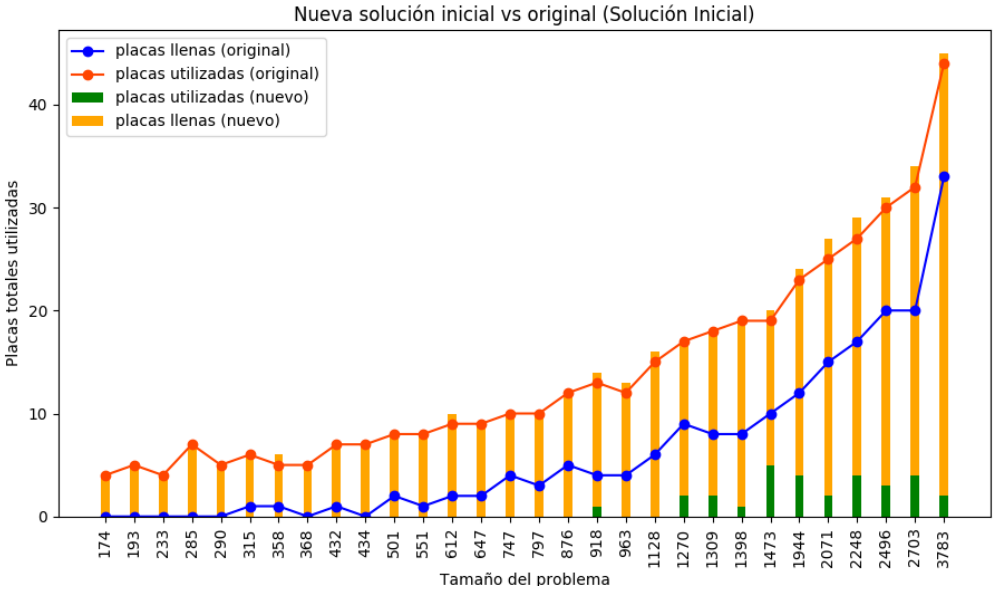


Figura 4.7: Comparativa de la solución inicial propuesta y original.

solución original y las barras los resultados con las modificaciones planteadas. Para que la solución sea aceptable, es crítico que la altura de las barras no superen bajo ningún concepto la línea naranja, ya que eso significa que la cantidad de placas que necesita el algoritmo ha aumentado. Si esta condición se cumpliese, sería deseable que la altura de las barras verdes, que representan las franjas que el algoritmo es capaz de llenar, coincida o supere la línea punteada de color azul. Es fácil de comprobar que no se cumple ninguna de las dos condiciones.

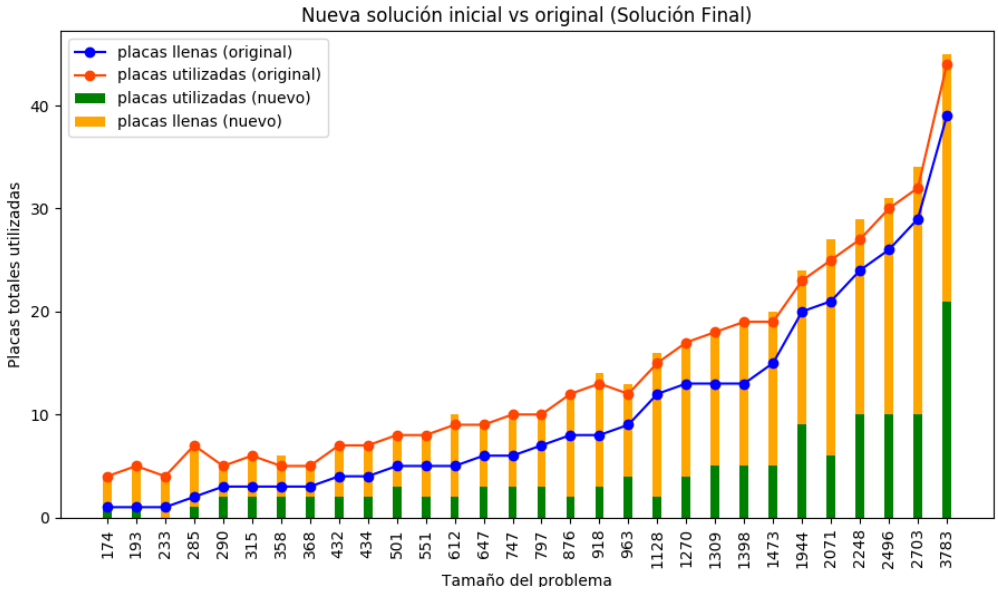


Figura 4.8: Comparativa de la solución final con las propuestas y resultado original.

La Figura 4.8 muestra un análisis similar al anterior, pero esta vez con la solución final aportada por el algoritmo. Se observa que mantiene el número de placas utilizadas pero es capaz de mejorar el resultado en cuanto a número de placas llenas con respecto a la solución inicial. En cualquier caso, el resultado sigue estando lejos del original y de ser aceptable.

Existen dos razones por las cuales esta aproximación da malos resultados. Por una parte, el hecho de no dividir los grupos y no llenar franjas al máximo para reservar huecos para los indicadores aporta gran flexibilidad pero hace las muestras de cada temperatura ocupen muchas más franjas y, por tanto, más placas. Por otra parte, la única forma de llenar una placa es ubicar en ella 6 franjas llenas. Sin embargo, los movimientos diseñados para la metaheurística no son capaces de rellenar huecos en gran medida y su trabajo principal es reordenar las franjas de forma que éstas se coloquen cumpliendo las restricciones y llenando placas.

4.6. Conclusiones

Los resultados de la metaheurística planteada en el estado del arte son buenos. Sin embargo, existen problemas de rendimiento que se acentúan a medida que el tamaño del problema a resolver crece, llegando a requerir de varias horas para calcular la solución. La exploración del espacio de soluciones es costosa, puesto que conseguir movimientos factibles es difícil. Por tanto, evitar mínimos locales es incluso más difícil, lo que supone que mejorar las soluciones pueda resultar realmente costoso en tiempo.

Para intentar mejorar la eficiencia del algoritmo se han planteado dos modificaciones, una para cada una de las principales razones de fallo detectadas. La más prometedora se basa en eliminar

completamente los fallos relacionados con la asignación del indicador de control, que repercute muy positivamente en la proporción de movimientos factibles e infactibles. Sin embargo, esta mejora afecta al desempeño de la solución inicial y se ha visto que en este problema partir de una buena solución inicial es algo crítico.

Capítulo 5

Propuesta de una nueva heurística

5.1. Motivación

El estudio realizado sobre la metaheurística basada en simulated annealing planteada en el estado del arte concluye que el algoritmo, si bien obtiene grandes resultados, tiene problemas de eficiencia, debidos principalmente a la gran cantidad de movimientos infactibles que éste debe intentar para mejorar la solución, sobre todo cuando se trata de evitar el estancamiento en mínimos locales. Las modificaciones propuestas para corregir estos problemas de eficiencia no solventan todas las complicaciones, las cuales residen principalmente en la dificultad de los movimientos diseñados para modificar la solución. Por esta razón, se propone una nueva heurística para la resolución del problema, una que asegure una visión que incluya todo lo aprendido a través del estudio de la estructura interna del problema, así como lo aprendido a partir de los algoritmos propuestos en el estado del arte y las soluciones obtenidas por éstos.

Se trata de una heurística que calcula una solución inicial de partida, sobre la cual se realizan una serie de operaciones de barrido para asegurar la factibilidad y conseguir el mejor valor de la función objetivo posible manteniendo un coste computacional bajo.

5.2. Solución Inicial

El cálculo de una solución inicial parte de un planteamiento similar al del simulated annealing descrito en el estado del arte. El primer paso es realizar un preprocesado de los datos de entrada del problema, ordenando las muestras según su temperatura de procesado y su identificador de grupo de forma ascendente. A continuación, las muestras se colocan de forma secuencial en las franjas, tal y como se explica en el Capítulo 2 con una diferencia, y es que se desechan todas aquellas franjas que no se hayan podido llenar completamente. Concretamente, el hecho de desechar una franja consiste en copiar todo su contenido, eliminándolo de la placa de la solución y utilizando el hueco que deja para colocar nuevas muestras. Este proceso se ilustra en la Figura 5.1. En concreto, en la Figura 5.1 a) puede verse cómo, tras terminar de colocar todas las muestras de temperatura 50ºC, la tercera franja de la placa ha quedado incompleta. De esta forma, todo el contenido de esa franja será copiado y enviado al conjunto de franjas desechadas y el hueco que ocupaba en la placa original se utilizará para continuar con el proceso de llenado normal de las muestras de una nueva temperatura, por ejemplo de 53ºC como muestra la Figura 5.1 b).

Una vez finalizado el proceso, el resultado obtenido será un conjunto de placas totalmente llenas y un posible conjunto de franjas que han sido desechadas y se mantienen al margen para ser ubicadas en pasos posteriores del algoritmo.

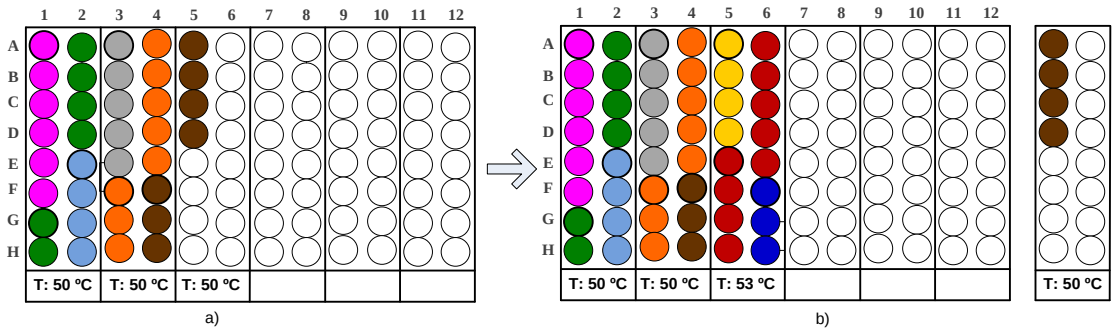


Figura 5.1: Ejemplo de llenado de la solución inicial.

5.3. Operaciones de barrido

Generalmente, la solución inicial calculada no es factible, ya que parte de las muestras de entrada pueden estar almacenadas en franjas que se han marcado como desechadas en primera instancia. Para solucionarlo, el algoritmo realiza una serie de operaciones de barrido a lo largo de toda la solución. Estas operaciones son las siguientes:

5.3.1. Rellenado de franjas con 15 huecos ocupados

Debido al modo en que se rellenan, es posible que una franja no consiga ser ocupada en su totalidad aunque haya muestras suficientes para hacerlo. Esto es debido a la necesidad de colocar el indicador del grupo junto a sus respectivas muestras. Por tanto, cuando se debe colocar un grupo en una franja y ésta no dispone de, como mínimo, dos huecos, se prefiere utilizar una nueva franja para evitar violar las restricciones del problema en el futuro. Esta operación se lleva a cabo utilizando muestras de las franjas que han sido desechadas, buscando aquellas que compartan temperatura con la franja objetivo y buscando grupos que se puedan intercambiar para completarla.

La Figura 5.2 muestra un diagrama del funcionamiento del rellenado de franjas. La tercera franja de la placa tiene un hueco vacío debido a la distribución de los grupos que residen en ella. Sin embargo, la franja puede ser ocupada al 100 % sin coste adicional para el objetivo intercambiando el grupo de color azul por el grupo de color naranja ubicado en la franja desechada.

5.3.2. Rellenado de huecos en las placas

Dado que las franjas que no se pueden llenar por completo son desechadas, en las placas de la solución pueden quedar franjas vacías si la distribución de temperaturas no es continua. Este movimiento busca rellenar esos huecos con las franjas más prometedoras.

La Figura 5.3 muestra un diagrama del rellenado de un hueco de una placa. En este ejemplo, la tercera franja de la placa está vacía. Esto se debe a que no se ha podido llenar una franja con una temperatura que difiera menos de 5 °C de la segunda franja, cuya temperatura de procesado es de 51 °C. Por esta razón, ha sido necesario dejar la tercera franja vacía. Además, se le ha asignado una temperatura de 56 °C, lo que permite asignar una temperatura de procesado de 57 °C a la cuarta franja de la placa. De las dos franjas desechadas a la derecha, es necesario escoger una que será colocada en la placa.

Existen diferentes criterios para escoger el mejor candidato de entre las franjas disponibles, pudiendo priorizar la ocupación de las mismas o la continuidad de sus temperaturas de procesado. Ambos criterios responden a necesidades distintas: un criterio caracterizado como ambicioso, que prioriza la ocupación de las franjas, de forma que se maximice el llenado de la placa, y otro criterio más conservador, el cual atiende a las temperaturas de procesado de las franjas y permitirá que sea más cómodo ubicar las

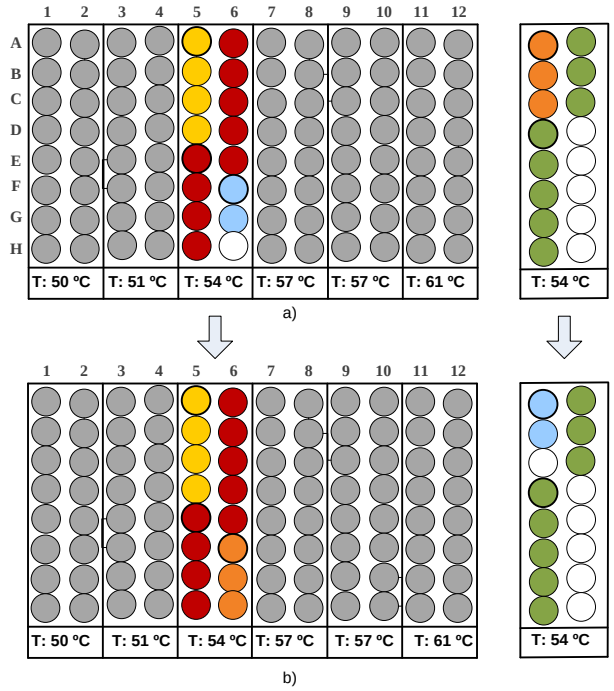


Figura 5.2: Diagrama de rellenado de franjas con 15 huecos ocupados.

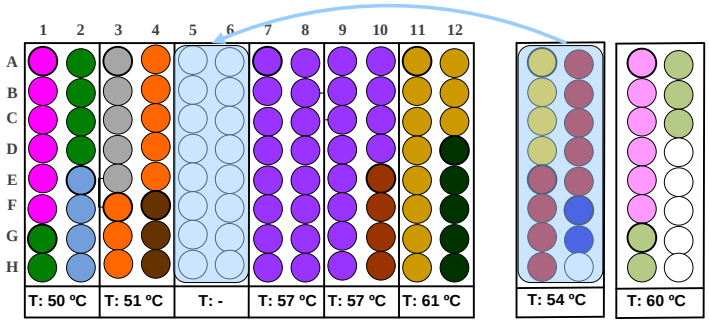


Figura 5.3: Diagrama de rellenado de franjas vacías.

franjas restantes en el último paso del algoritmo. La Figura 5.4 muestra una comparativa entre ambos

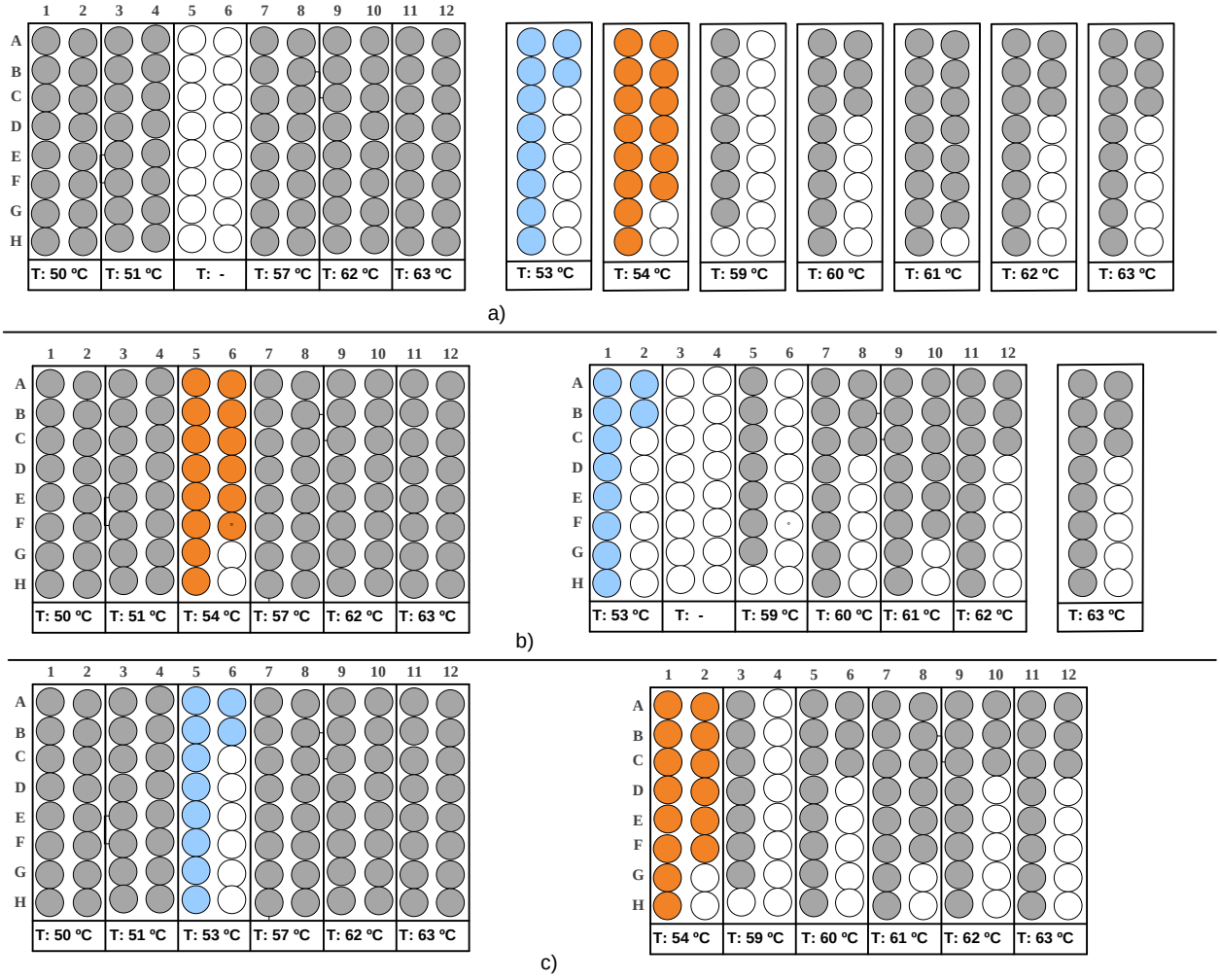


Figura 5.4: Diagrama de ejemplo a la hora de escoger criterios de rellenado.

criterios. En concreto, en la Figura 5.4 a) se muestra la placa con una franja vacía que se pretende rellenar, así como las franjas disponibles para hacerlo, que son aquellas cuyas muestras están marcadas de azul y naranja. La Figura 5.4 b) muestra el primer criterio, uno más ambicioso, según el cual se asigna la franja atendiendo al porcentaje de ocupación de la misma, es decir, se selecciona la franja de color naranja. Esto repercute positivamente en la ocupación total de la placa. Sin embargo, ubicar las franjas desechadas restantes requerirá más espacio, ya que la diferencia de temperaturas de las franjas es superior a 5 °C por tanto, es necesario utilizar más de una placa. Por otra parte, la Figura 5.3 c) muestra el segundo criterio, más conservador, según el cual la franja se asigna al hueco según su temperatura de procesado, es decir, se selecciona la franja de color azul. Esto permite que las franjas restantes puedan ser ubicadas en menos espacio, ya que sus temperaturas de procesado difieren en menos de 5 °C y, por tanto, podrán ir colocadas de forma continua en una sola placa.

5.3.3. Creación de nuevas placas con las franjas desechadas

La última operación está destinada a colocar las franjas restantes que no hayan sido utilizadas en el resto de operaciones, lo que implica crear nuevas placas para añadir a la solución y repartir las franjas restantes de la mejor forma posible.

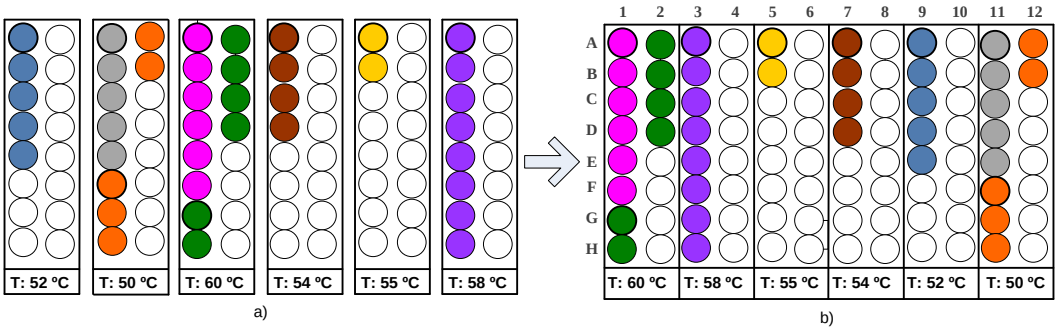


Figura 5.5: Diagrama de creación de nuevas placas con las franjas restantes.

La Figura 5.5 muestra este proceso. En concreto, la Figura 5.5 a) muestra las franjas restantes que deben ser ubicadas en una nueva placa. Para ello se ordenan según su temperatura de procesado y se colocan de forma continua en la nueva placa, dando como resultado la placa mostrada en la Figura 5.5 b). El resultado de esta operación depende directamente de las características de las franjas restantes, así como del resultado de un posible rellenado de huecos en la solución y, de cara a evitar utilizar placas de más, es necesario garantizar que las franjas desechadas tengan la distribución de temperatura lo más continua posible.

5.3.4. Pseudocódigo del algoritmo

El Listado 5.1 muestra el pseudocódigo de la heurística desarrollada. El primer paso es preparar las muestras de forma que estén agrupadas y ordenadas por su temperatura de procesado, luego se genera la solución inicial, obteniendo una lista de placas llenas y una lista de franjas que han sido desechadas al no estar completas. A continuación, se realizan las operaciones de barrido con el objetivo de rellenar tantos huecos como sea posible. Tras cada operación que modifique las placas de la solución siempre se intenta completar las franjas de ocupación 15, ya que tras cada operación es posible que se inserten nuevas franjas. El último paso es asegurarse de que las placas creadas a partir de las franjas desechadas incluyan todos los indicadores de control de grupo que sean necesarios. Todo el código fuente relacionado con el algoritmo se encuentra en el Anexo B.

```
# 1. Procesa la lista de muestras y las agrupa por temperatura
grouped_samples = group_by_temp(samples)

# 2. Calcula la solución inicial, llenando tantas franjas como sea posible,
# el resto las desecha y las almacena en una lista a parte
full_plates, discarded_strips = initial_solution(grouped_samples)

# 3. Intenta completar las franjas de ocupación 15
full_plates = fill_strips(discarded_strips)

# 4. Busca franjas vacías en las placas y las rellena con franjas desechadas
full_plates = fill_empty_slots(discarded_strips)
```

```
# 5. Intenta completar las franjas de ocupación 15
full_plates = fill_strips(discarded_strips)

# 6. Coloca las franjas restantes en nuevas placas
partial_plates = populate_partial_plates(discarded_strips)

# 7. Intenta completar las franjas de ocupación 15
full_plates = fill_strips(discarded_strips)

# 8. Coloca indicadores que puedan faltar
fill_indicators(partial_plates)

# 9. Construye la solución concatenando todas las placa creadas
solution = full_plates + partial_plates
```

Listado 5.1: Pseudocódigo de la heurística.

5.4. Resultados

5.4.1. Placas utilizadas

Durante la ejecución del algoritmo, es necesario seleccionar, de entre las franjas desechadas, los mejores candidatos para rellenar huecos vacíos en las placas. Se han visto los dos criterios diseñados para ello, donde uno es más ambicioso y otro más conservador, desde el punto de vista del número total de placas a utilizar. Lógicamente, esta elección repercute directamente en el desempeño del algoritmo ya que, en ocasiones, el criterio ambicioso es capaz de llenar un mayor número de placas a costa de utilizar más placas de las necesarias. Si bien un resultado de este estilo es inaceptable, en ocasiones, ambos criterios utilizan el mismo número de placas. Es en estas ocasiones donde tiene sentido aprovechar las ventajas de un criterio más ambicioso. La Tabla 5.1 muestra un resumen de aquellos

Tamaño	**Mejor caso**		**Conservadora**		**Ambiciosa**	
	Totales	**Llenas**	**Totales**	**Llenas**	**Totales**	**Llenas**
174	4	2	4	1	**4**	**2**
193	5	2	5	1	**5**	**2**
290	5	3	**5**	**3**	6	3
358	5	2	**5**	**2**	6	2
612	9	6	9	5	**9**	**6**
918	13	9	**13**	**9**	14	9
1270	17	13	17	12	**17**	**13**
2248	27	24	27	23	**27**	**24**
3783	44	40	44	39	**44**	**40**

Tabla 5.1: Comparativa de resultados de las distintas estrategias de la heurística.

resultados donde ambos criterios difieren, escogiendo el mejor en cada caso. Combinar ambos criterios

de forma simultánea permite mejorar la exploración de las soluciones y además, no repercute en el coste computacional del algoritmo, ya que pueden tratarse como problemas independientes ejecutados de forma paralela.

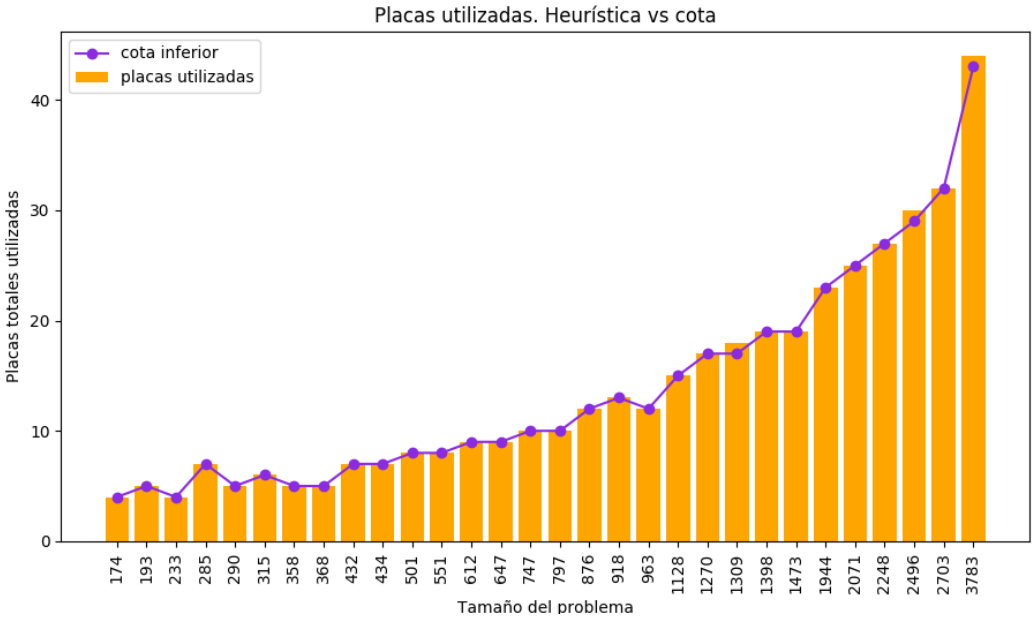


Figura 5.6: Resultados de la heurística propuesta y comparativa con la cota inferior.

En cualquier caso, la correcta evaluación de la calidad de la solución alcanzada pasa por comparar su comportamiento con las cotas propuestas a lo largo del trabajo, así como una comparativa con los resultados obtenidos en el estado del arte. La Figura 5.6 muestra el resultado obtenido por la heurística propuesta para el primer objetivo, el número de placas utilizadas, así como una comparativa con la cota inferior para el número de placas. Puede verse que, en la gran mayoría de los casos, el algoritmo se ajusta a los valores de la cota y, en aquellos dónde no lo hace, es en los mismos ficheros donde la metaheurística del estado del arte tampoco es capaz, lo que parece concluir que en esos casos la cota teórica no es alcanzable. En lo que respecta a este objetivo, la solución es aceptable, ya que no se ha empeorado en ningún caso lo propuesto en el estado del arte.

5.4.2. Ocupación total

El segundo objetivo trata de minimizar el número total de huecos ocupados a lo largo de las placas que componen la solución. Esto se debe a que si bien el número de muestras es estático, la cantidad de indicadores de control no lo son y, por tanto, es importante mantenerlos al mínimo necesario. La Figura 5.7 muestra una comparativa entre la heurística propuesta y la cota inferior calculada para la ocupación total. Los puntos de color rojo indican que la solución obtenida es peor que la cota y su valor refleja la diferencia de huecos ocupados. Se observa que, a medida que el problema crece, la diferencia aumenta. Esto se debe a que generalmente los ficheros más grandes presentan un mayor número de grupos y además estos son de mayor tamaño.

Por otra parte, la Figura 5.8 muestra la diferencia del número total de huecos ocupados entre la heurística propuesta y el estado del arte. Los puntos verdes indican que la nueva heurística ha mejorado los valores del estado del arte y los puntos rojos que ha empeorado. Puede verse cómo, en este caso, la heurística propuesta mejora a medida que el tamaño del problema crece, ya que para el

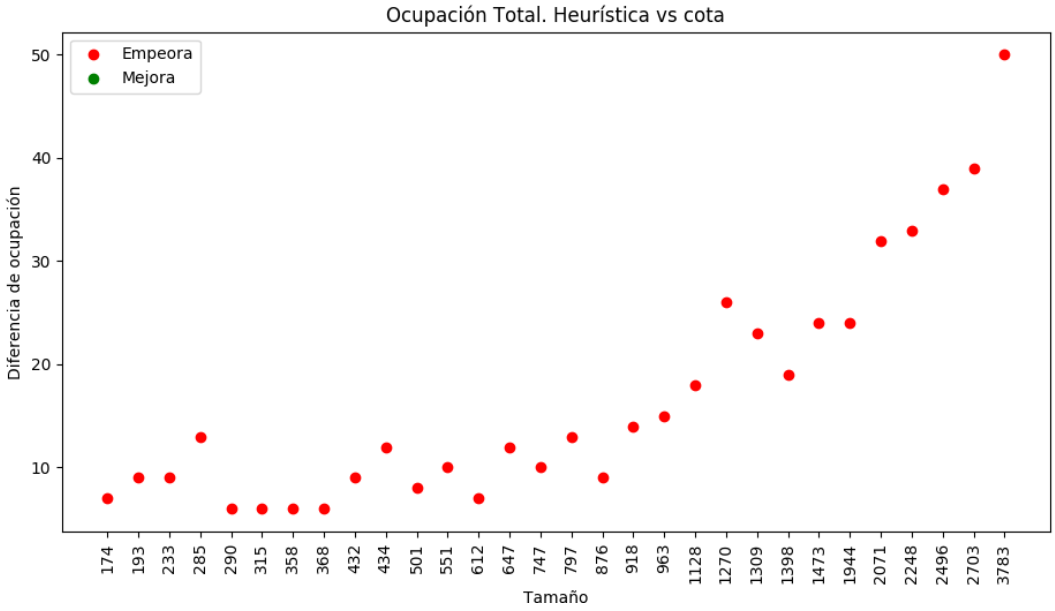


Figura 5.7: Diferencias en la ocupación total entre la heurística propuesta y la cota calculada.

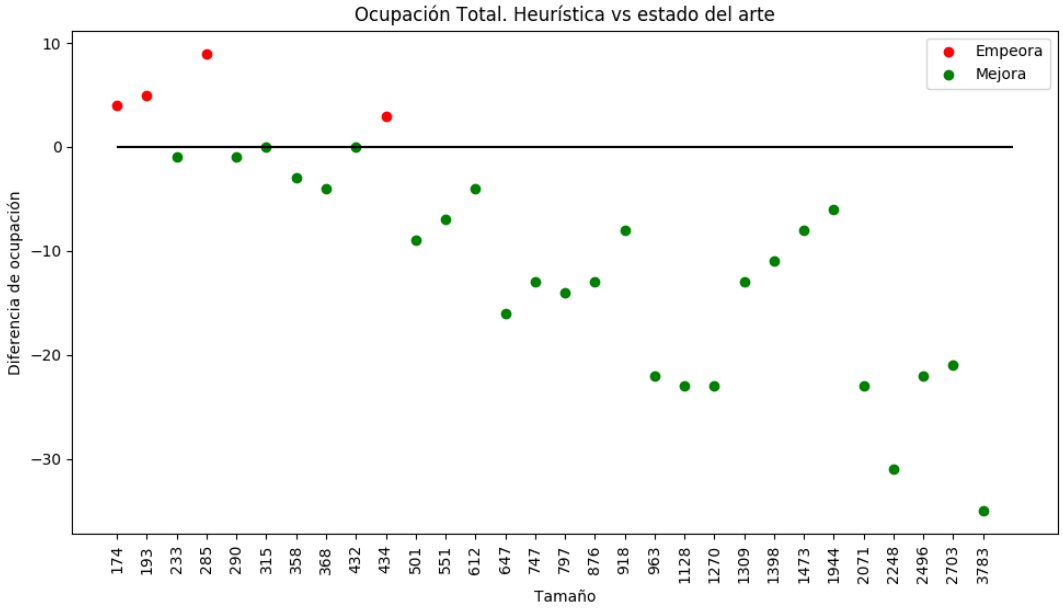


Figura 5.8: Diferencias en la ocupación total entre la heurística propuesta y la metaheurística del estado del arte.

simulated annealing el coste de mejorar la solución crece rápidamente, al necesitar un mayor número de movimientos.

5.4.3. Llenado de placas

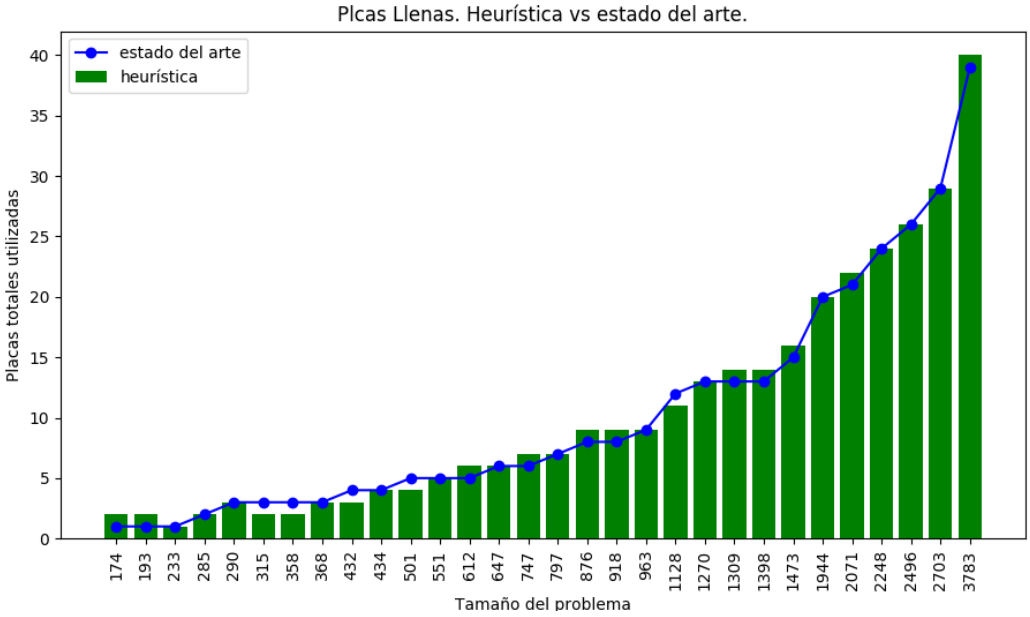


Figura 5.9: Comparativa de llenado de placas entre la heurística propuesta y la metaheurística del estado del arte.

Por último, puede evaluarse la capacidad del algoritmo para el llenado completo de las placas. La Figura 5.9 muestra una comparativa entre la heurística propuesta y los resultados obtenidos en el . La línea de color azul marca el número de placas que la metaheurística del estado del arte ha sido capaz de llenar por completo. Por tanto, la heurística propuesta obtendrá mejores resultados cuando las barras de color verde superen el valor de la línea. La realidad es que en 11 de los 30 casos el algoritmo propuesto mejora la solución, en 5 de 30 la empeora y en los 14 restantes obtiene el mismo resultado. En general, el algoritmo propuesto funciona mejor en los problemas más grandes, como antes, algo razonable debido al comportamiento del simulated annealing y su necesidad de realizar más movimientos cuanto mayor sea el problema a resolver.

5.4.4. Tiempos de ejecución

La Figura 5.10 muestra un resumen del tiempo que el algoritmo necesita para calcular la solución. Cabe destacar que estos tiempos han sido obtenidos ejecutando el algoritmo en una máquina con un procesador Intel(R) Core(TM) i7-3770 con 3.40GHz y 16ĠB de memoria RAM. Puede observarse que es extremadamente rápido, resolviendo el fichero de mayor tamaño en 2 décimas de segundo. Como curiosidad, debido a que la factibilidad de la solución debe calcularse de forma muy detallada y meticulosa, este proceso tarda mucho más en llevarse a cabo que la propia resolución del problema. En cualquier caso, el coste asociado a la comprobación de la factibilidad se ha especificado por separado, puesto que el algoritmo asegura que la solución final va a ser siempre factible y, por tanto, no sería necesario comprobarlo más allá de la correspondiente justificación.

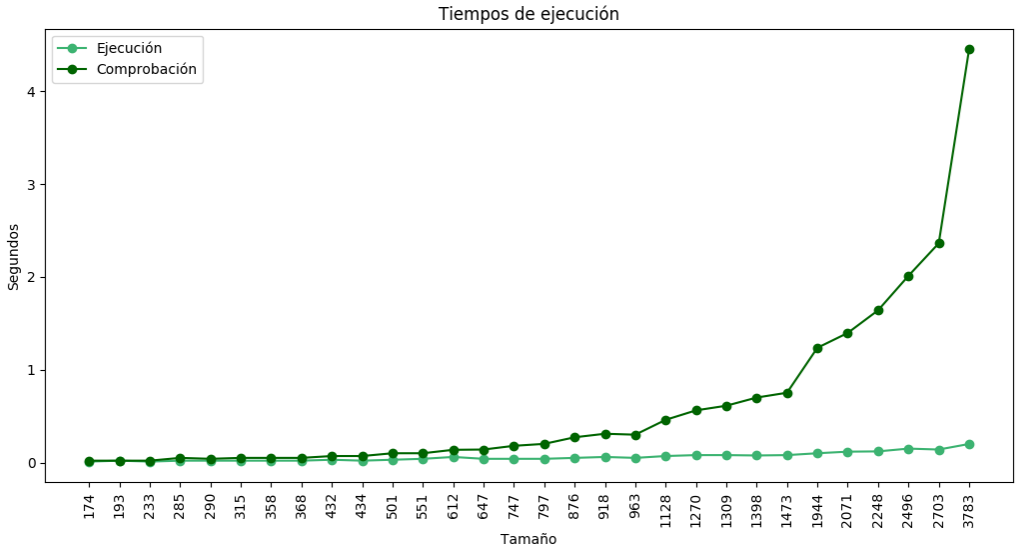


Figura 5.10: Resumen de tiempos de ejecución de la heurística.

5.5. Análisis de mejoras

Dentro de la solución obtenida por la heurística, pueden distinguirse dos partes: por una parte, el conjunto de aquellas placas que están completamente llenas y, por otra, parte el conjunto de las placas que no lo están. Estas últimas están creadas a partir de las franjas desechadas y su organización es aquella que permite colocar todas las franjas en el menor número de placas posible, pero en ningún caso se garantiza que la ocupación de las placas sea la óptima con respecto a los objetivos del problema. Lo interesante es que este conjunto de placas puede verse como un problema en sí mismo, en el cual interesa optimizar la ocupación de las placas que lo componen. Por lo observado en la práctica, se trata de problemas pequeños, ya que en los 30 ficheros disponibles el mayor tamaño de este conjunto es de 6 placas. Con problemas de este tamaño se plantean dos alternativas: por un lado, resolver este nuevo problema de forma exacta y, por otro lado, una metaheurística, como por ejemplo el simulated annealing planteado en el estado del arte, que puede ser realmente eficiente y obtener buenos resultados. Además, como las franjas de estas placas tienden a ser autocontenidas, solamente sería necesario utilizar el movimiento de intercambio de franjas, eliminando problemas de calibración y carga de trabajo adicional a la metaheurística.

Así, por ejemplo, la Figura 5.11 muestra parte de la solución obtenida para uno de los ficheros disponibles, concretamente uno compuesto 1473 muestras divididas en 197 grupos y 16 valores de temperatura. Las placas que se han conseguido llenar completamente no se muestran puesto que no son importantes en este análisis. Puede observarse que, a lo largo de las tres placas, existen ciertas franjas con una ocupación muy alta, por lo que el objetivo será reunir estas franjas de forma que aumenten todo lo posible la ocupación de las placas. Para conseguirlo, es muy importante tener en cuenta que es inaceptable utilizar un número mayor de placas del obtenido en la solución; esto es, no se permitirá utilizar más de las tres placas disponibles. Esta restricción descarta un enfoque sistemático, ya que sería muy costoso evaluar toda la casuística que se pueda presentar.

La Figura 5.12 muestra el resultado obtenido tras reorganizar las franjas de las placas parcialmente llenas utilizando una solución implementada específicamente para este problema. Puede observarse cómo ahora el porcentaje de ocupación de cada una de ellas se adapta mejor a los objetivos del problema, relegando las placas más vacías a las últimas posiciones de la solución. La conclusión principal es que una reorganización que mejore la ocupación de las placas es posible. Por otra parte, es necesario

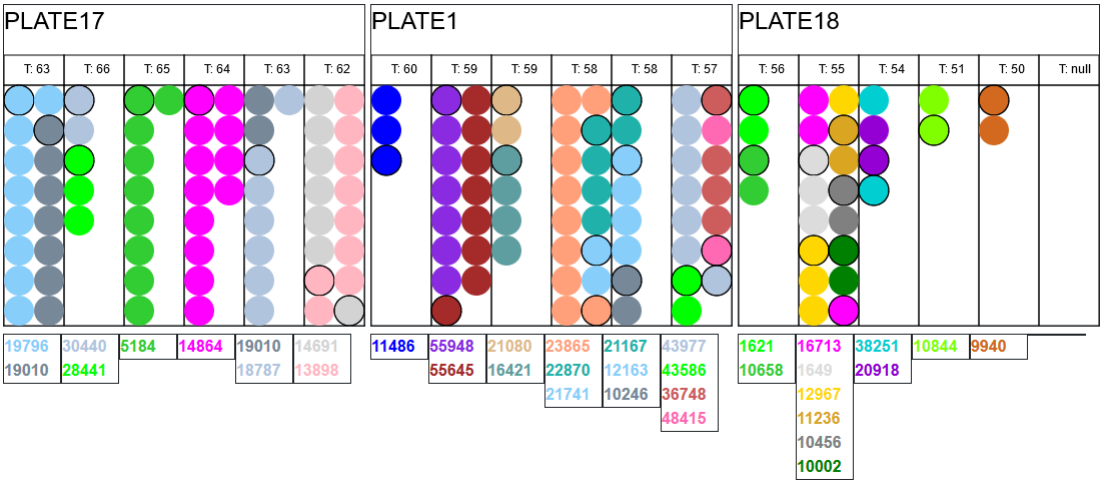


Figura 5.11: Captura de placas parcialmente llenas de un fichero con 1473 muestras.

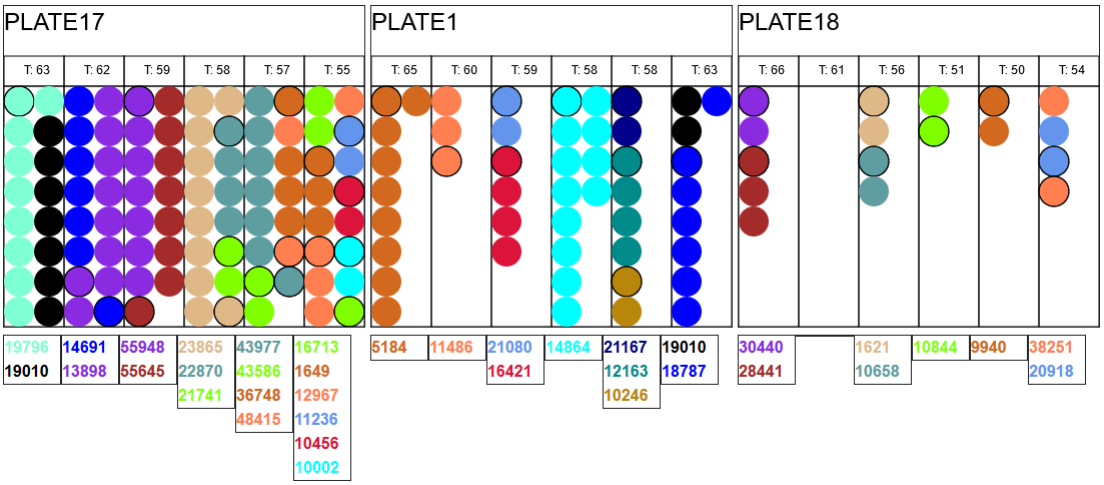


Figura 5.12: Captura de placas reordenadas de cara a mejorar la solución.

plantear una técnica general que permita mejorar cualquier problema similar que pueda ser planteado, ya que es habitual que estos subproblemas tengan unas características muy distintas, siendo habitual que existan grandes huecos entre los valores de temperatura de las franjas a procesar, franjas vacías utilizadas a modo de separador, etc.

5.6. Conclusiones

Se ha propuesto una heurística capaz de resolver el problema extremadamente rápido, resolviendo cualquiera de los ficheros disponibles en décimas de segundo. Es una ganancia considerable, teniendo en cuenta que la metaheurística del estado del arte requiere de entre decenas de segundos hasta horas en los problemas más grandes. Además, la heurística propuesta es capaz de mejorar las soluciones, obteniendo una mejora en la ocupación total de las placas. Esta mejora crece a medida que el tamaño del problema se hace mayor. Además, en ocasiones, es capaz de mejorar el llenado de las placas. Las principales ventajas de la heurística propuesta son las siguientes:

- Coste computacional bajo. La solución es rápida y sencilla de calcular. Además, la complejidad crece de forma lineal con el tamaño del problema.
- No requiere de ningún tipo de calibración ni hay parámetros que estimar. En un entorno donde los problemas a resolver son tan cambiantes, lo normal es que distintos problemas requieran de parámetros con distintos valores. Además, es muy difícil realizar una clasificación de los problemas. El algoritmo propuesto no requiere ningún tipo de procesado en ese sentido.
- Alta estabilidad. Ninguno de los procesos del algoritmo tiene un componente de aleatoriedad, por ello se trata de un algoritmo determinista, con la misma entrada será capaz de calcular siempre la misma solución.
- Está basado en la estructura interna del problema. El algoritmo es capaz de enfrentarse a un nuevo problema con solvencia, ya que tiene en cuenta la estructura y características generales del problema.

Capítulo 6

Aplicación Web

Este capítulo pretende mostrar la aplicación web desarrollada, detallando su funcionamiento, los principales casos de uso disponibles y las operaciones más comunes. El punto de partida es la página principal, donde se muestra un resumen del estado de la aplicación en su conjunto, tal y como muestra la Figura 6.1. Esta página está dividida en tres secciones: la primera permite conocer las tareas pen-

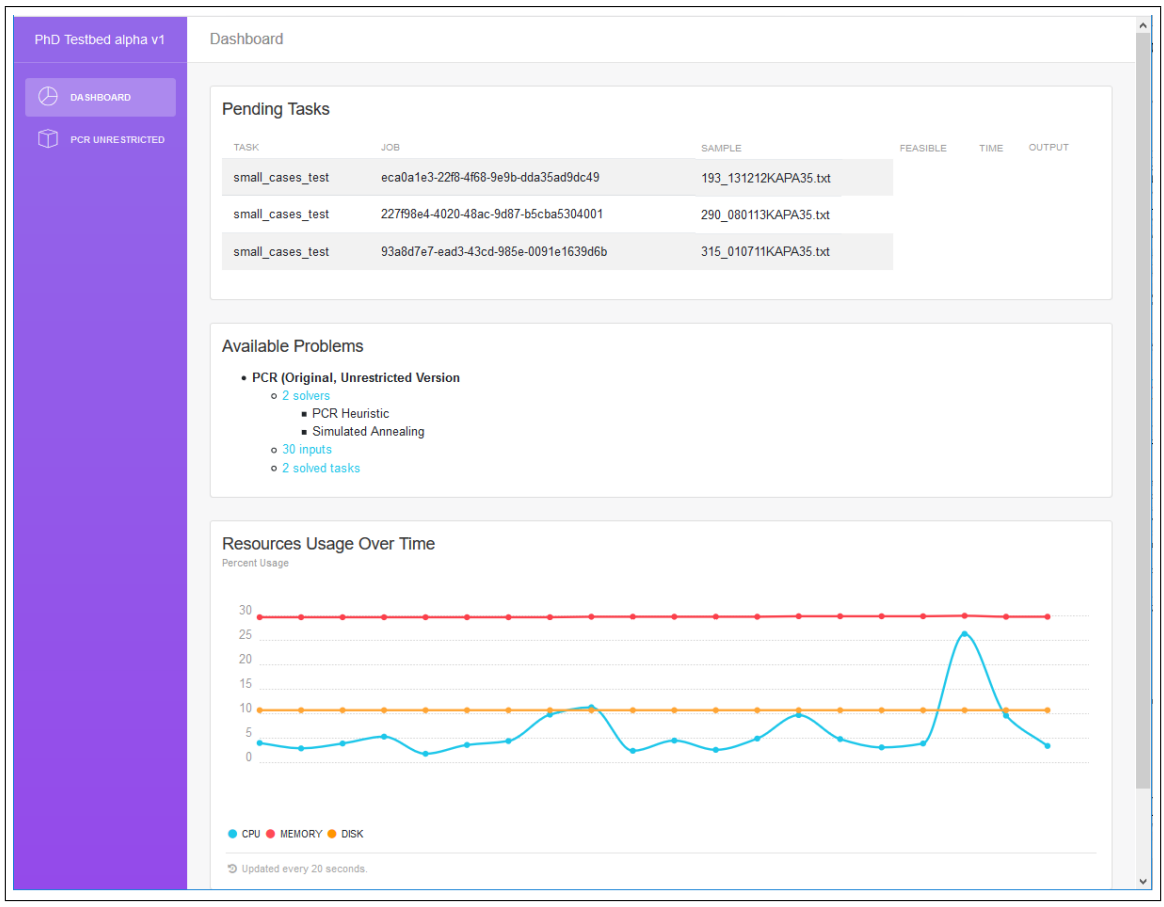


Figura 6.1: Página principal de la aplicación.

dientes de ejecución, esto es útil para rastrear aquellas ejecuciones que requieren de mucho tiempo de

computación. La segunda sección muestra un resumen de los problemas disponibles con algoritmos implementados y sus características. Por ejemplo, del problema PCR estudiado en este trabajo, conocido como original y sin restricciones adicionales, se han implementado dos algoritmos, se dispone de 30 ficheros de datos de entrada y se han registrado dos tareas. Cada uno de estos apartados proporciona un enlace con el que acceder directamente a cada sección. Por último, la página principal muestra un gráfico actualizado en tiempo real del estado de los recursos de la máquina. Esta información es muy útil para realizar un primer análisis sobre el impacto de los algoritmos ejecutados en los recursos disponibles de la máquina.

Desde el menú lateral o en el enlace correspondiente de la sección de problemas disponibles se accede a la página de formularios que permite ejecutar los algoritmos. Esta página se muestra en la Figura 6.2. Los algoritmos implementados tienen su propio formulario, ya que cada uno requiere de

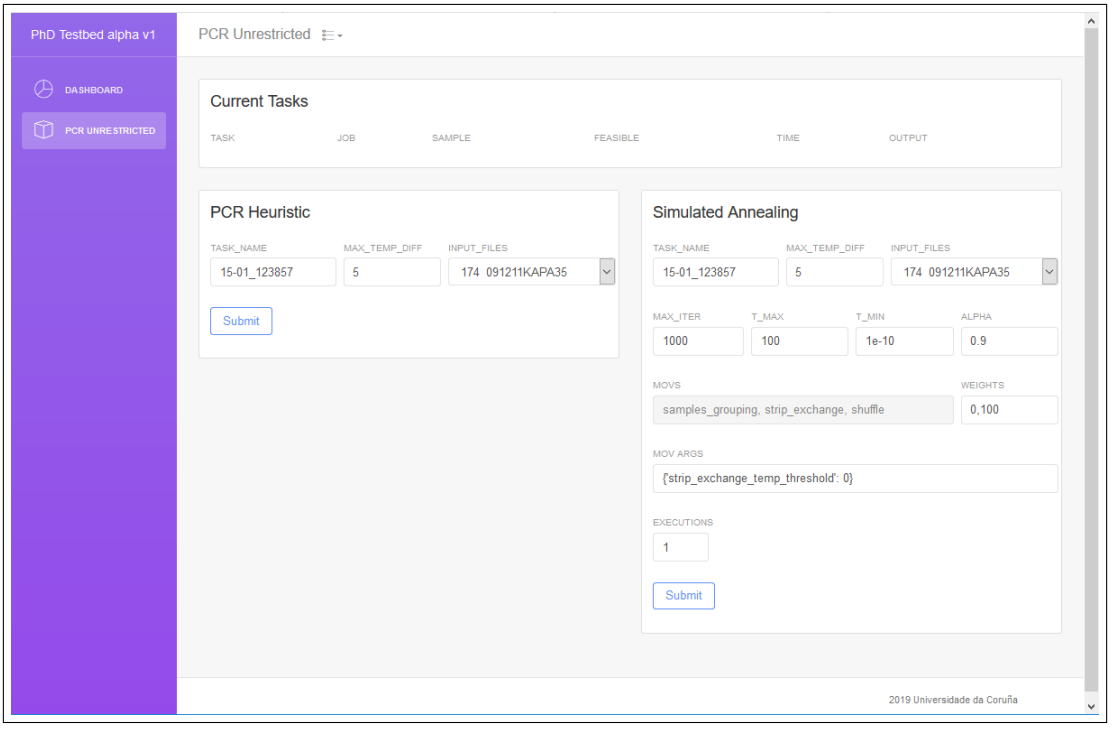


Figura 6.2: Formularios de ejecución de los algoritmos.

parámetros distintos. Por ejemplo, en el caso de la heurística propuesta en este trabajo simplemente se especifica la restricción de diferencia de temperatura entre franjas y el fichero de entrada a realgoritmo. Por otra parte, el simulated annealing requiere de una mayor cantidad de parámetros, como el número de iteraciones total, la temperatura de partida, el coeficiente de enfriamiento o el peso de los distintos movimientos.

Esta página también dispone de una tabla, en la parte superior, de las tareas ejecutadas. Esta tabla añade entradas a medida que se lanzan ejecuciones de los distintos algoritmos, reportando un resumen del resultado a medida que éstas terminan. Por ejemplo, en la Figura 6.3 se observa cómo en la tabla hay una serie de entradas correspondientes a unos trabajos en ejecución, mostrando información como la tarea a la que pertenecen, el indicador único de cada trabajo y el fichero que está siendo resuelto. Cada vez que se termina un trabajo, la entrada correspondiente de la tabla se actualiza e informa de la factibilidad de la solución, el tiempo total consumido y un enlace que lleva directamente a la solución en detalle. Por ejemplo, la Figura 6.4 muestra la tabla en un momento en el que varios de los trabajos se han completado.

Los enlaces proporcionados por la tabla cuando ésta se actualiza debido a que uno de los trabajos

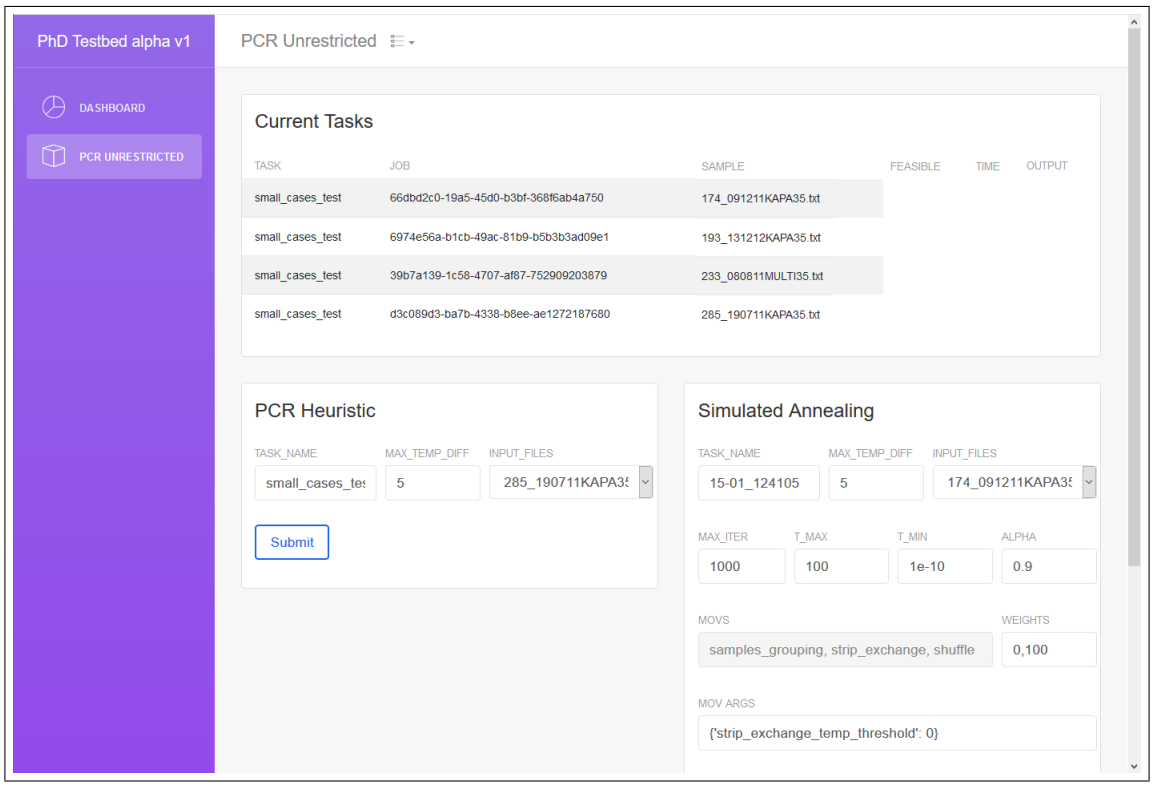


Figura 6.3: Lista de pruebas pendientes de ejecutar.

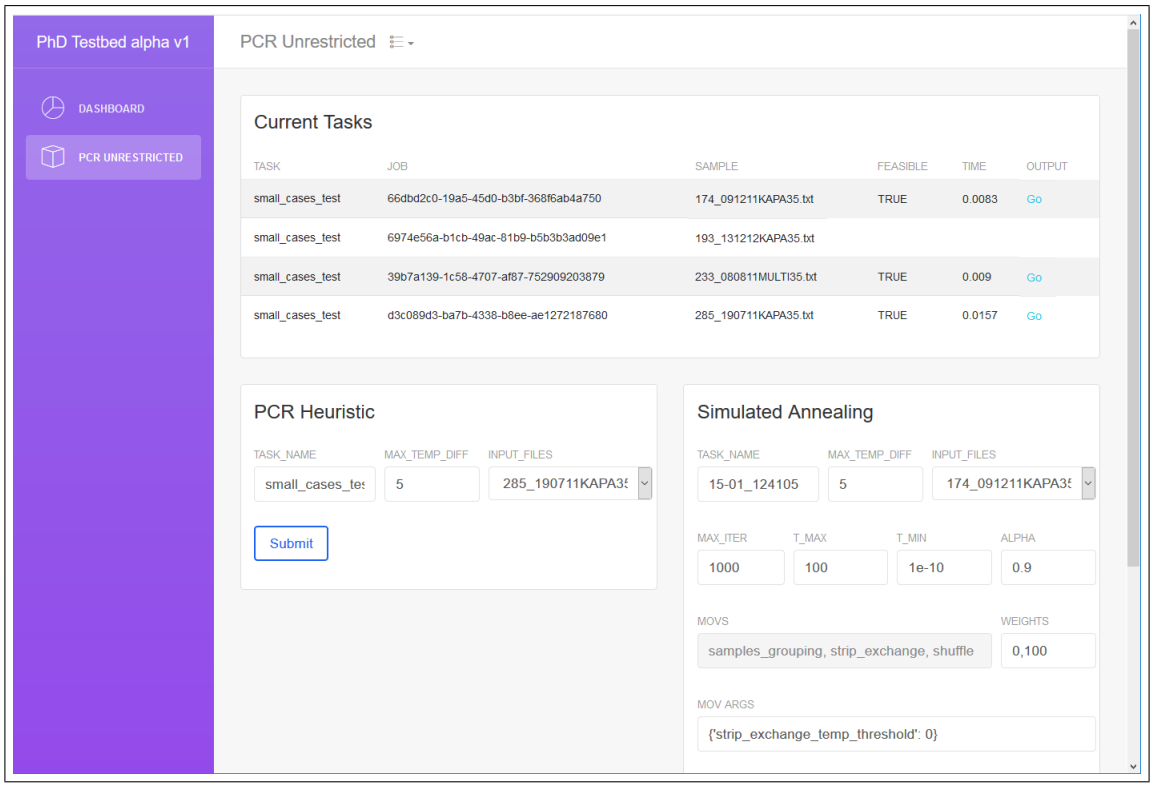


Figura 6.4: Lista de pruebas siendo ejecutadas.

termina se accede a la pantalla que detalla la solución obtenida. La Figura 6.5 muestra un ejemplo

Figura 6.5: Ejemplo de visualización de una solución de un problema PCR.

de una solución. En la parte superior se muestra un resumen del trabajo realizado, resumiendo los parámetros específicos del problema a realgoritmo, los parámetros específicos del algoritmo utilizado (en el caso de la heurística está en blanco pues no requiere de parámetros) y un resumen del valor de los objetivos de la solución. En la parte inferior se detalla la solución de forma gráfica, describiendo las placas necesarias, su contenido y sus características. Cada franja muestra la temperatura de procesado seleccionada, los identificadores de los grupos que contiene y las muestras que la componen, identificando aquellas que se utilizan como identificadores de control.

La página principal también permite acceder a la lista de tareas realizadas. Para cada trabajo realizado se muestra el algoritmo exacto utilizado, el fichero resuelto y los objetivos alcanzados. Las tareas son unidades organizativas que recopilan trabajos realizados, de forma que se puedan agrupar distintos trabajos que compartan alguna característica.

Por último, la aplicación permite acceder a los ficheros de entrada disponibles. Para cada uno de ello,s se dispone de un resumen gráfico de las principales características como el tamaño, distribución de grupos y distribución de temperaturas. La Figura 6.7 muestra una captura del resumen de un fichero de entrada, donde se muestra la distribución de grupos y temperaturas que han servido para realizar los análisis de los ficheros del trabajo.

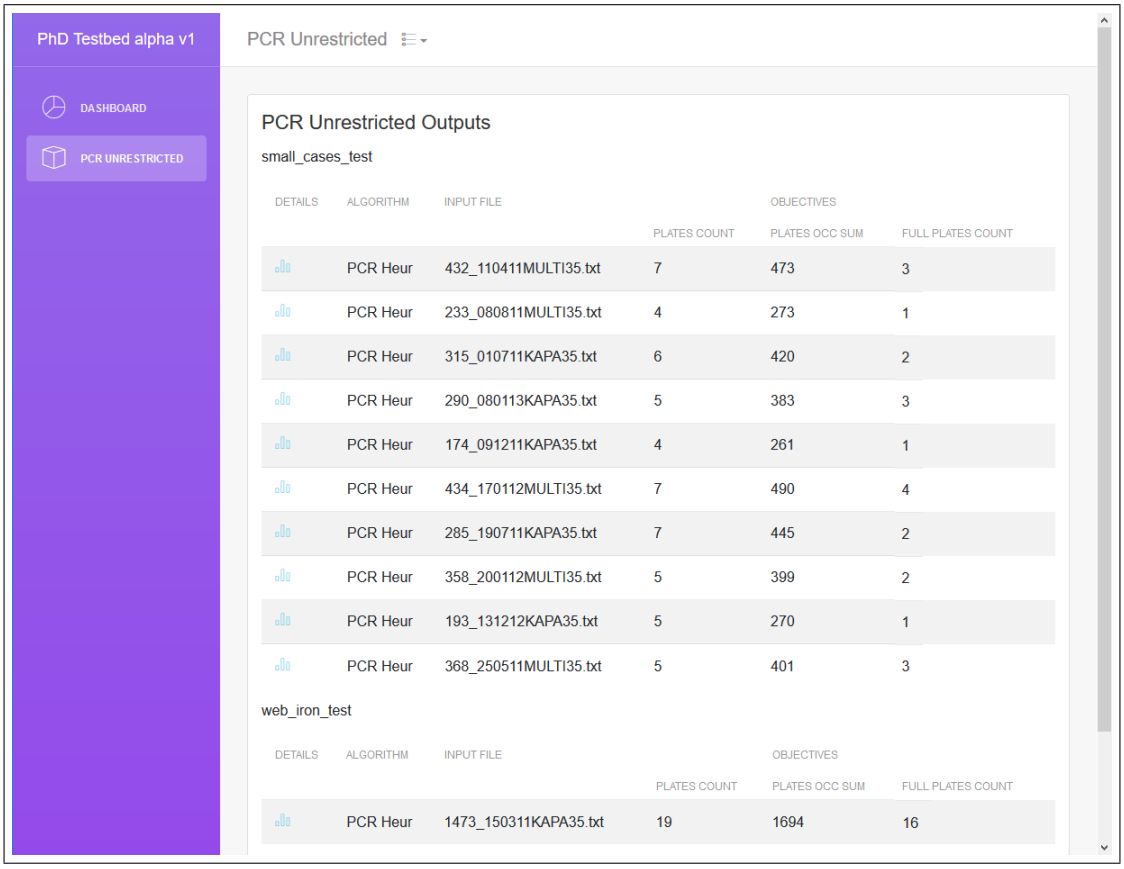


Figura 6.6: Lista de tareas completadas.

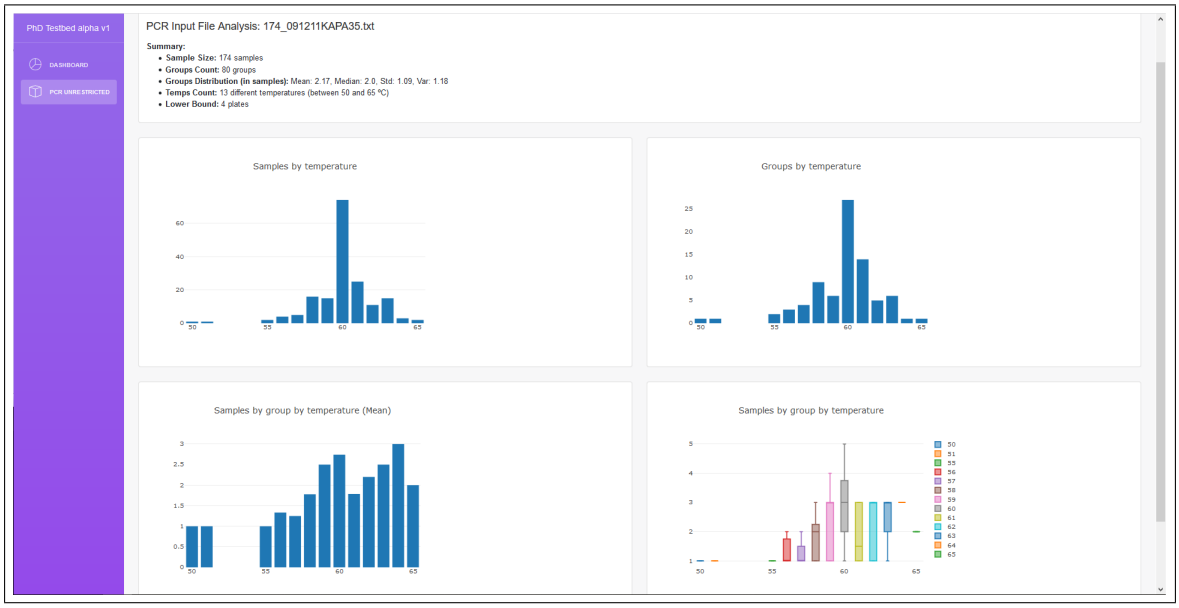


Figura 6.7: Análisis de un fichero de entrada.

Capítulo 7

Implementación del sistema

Este capítulo recoge toda la información relativa a la implementación del software de todo el presente trabajo, abarcando desde la justificación de la elección de cada una de las tecnologías utilizadas hasta una descripción detallada de los retos superados más importantes.

7.1. Python vs R

En el ámbito de la Estadística, el lenguaje de programación R siempre ha sido el principal referente, gracias a un elevado número de librerías y funciones que facilitan el trabajo, así como a su numerosa comunidad de usuarios en el ámbito tanto académico como industrial. Sin embargo, el mundo de las tecnologías de la información y comunicaciones (TIC) crece a pasos agigantados y cada vez es más complicado que lenguajes y tecnologías pensadas para sectores tan específicos mantengan el ritmo de crecimiento que la comunidad exige. Por esta razón, en los últimos tiempos se ha popularizado una corriente que pretende llevar los lenguajes de programación de corte más clásico (como Java o Python, por ejemplo) al ámbito científico. La principal ventaja de esta aproximación es aprovechar el hecho de que estos lenguajes han sido diseñados para realizar ingeniería de software. Por tanto, su diseño, así como su adaptación a los paradigmas principales de programación (estructurada, orientada a objetos y declarativa), son muy limpios y robustos.

El presente trabajo afronta una serie de retos muy distintos entre sí. Por una parte, la implementación de diversos algoritmos así como el análisis estadístico de sus propiedades y los resultados obtenidos. Por otra parte, el diseño e implementación de un sistema distribuido de pruebas y de soporte a la decisión. Tales retos requieren de una infraestructura que permita que el software sea mantenible, escalable, altamente extensible y robusto. Por esta razón se ha decidido optar por Python 3.6 como el lenguaje para desarrollar este trabajo.

7.2. Sistema de pruebas

Se ha desarrollado un sistema de pruebas (*testbed*) completo, distribuido y modular. El objetivo de este sistema es permitir la ejecución de las pruebas necesarias

7.2.1. Arquitectura

La Figura 7.1 muestra el diagrama base de la arquitectura del sistema, presentando los módulos más importantes. El sistema consta de 4 módulos principales: la aplicación web que da acceso al usuario, la base de datos que almacena todos los resultados, el gestor de colas encargado de ejecutar las tareas y, por último, el servicio de mensajería al cual se suscriben los otros módulos para poder comunicarse entre ellos. El objetivo de este diseño es permitir que haya un bajo acoplamiento entre los módulos, al mismo

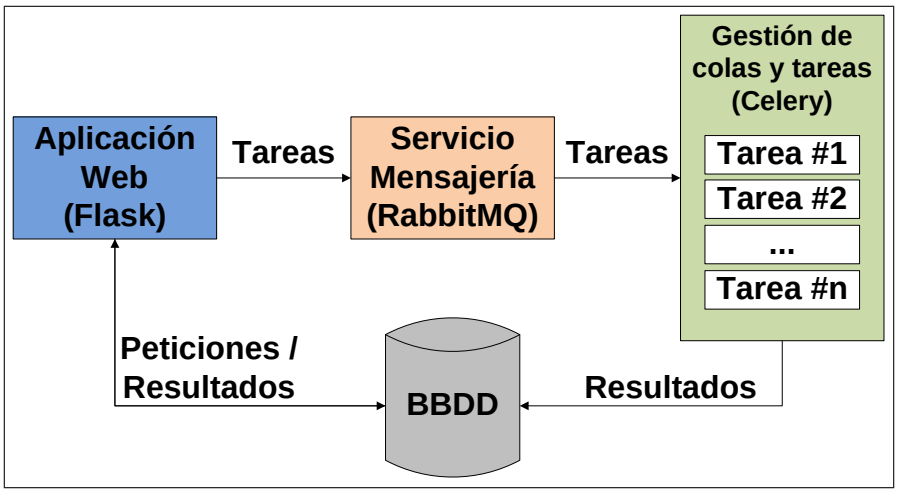


Figura 7.1: Diagrama del sistema desarrollado.

tiempo que se mantiene la escalabilidad del sistema y la centralización de los resultados obtenidos, todo ello accesible de forma sencilla mediante una aplicación web. Adicionalmente, el sistema permite ejecutar pruebas desde la línea de comandos.

Flask (Servicio web)

Flask es una librería implementada en Python que permite crear aplicaciones web ligeras de forma rápida y sencilla. Además, dispone de un sistema de plantillas que permite modificar el aspecto gráfico de la aplicación en conjunto. En este trabajo se ha utilizado Flask para crear una aplicación web que permita al usuario, de forma remota, ejecutar pruebas, ver resultados obtenidos y consultar los gráficos y diagramas generados asociados a dichos resultados. Las ventajas con respecto a una aplicación de escritorio residen especialmente en el carácter multidispositivo de la aplicación, que puede ser accedida desde un ordenador, móvil o tablet sin problemas.

Celery (Gestor de colas y tareas)

Para facilitar la escalabilidad y autonomía del sistema la ejecución de las tareas se delega en una librería que también gestione las colas de trabajo pendiente. El funcionamiento se basa en levantar nodos Celery en las máquinas de cómputo disponibles. Estos nodos son simplemente servicios que se comunican con el servicio de mensajería, reciben las tareas, las ejecutan y devuelven los resultados obtenidos.

RabbitMQ (Broker de mensajería)

RabbitMQ es una implementación del protocolo de mensajería AMQP. El objetivo de este módulo es conectar el resto de módulos entre sí, mediante un sistema de paso de mensajes, lo que permite que el resto de componentes estén desacoplados entre ellos pero manteniéndolos cohesionados.

7.2.2. Diseño de los algoritmos

En general, los algoritmos de optimización tienen una estructura similar compartida por todos ellos. Por ejemplo, para ser ejecutados primero deben cargar información desde una fuente de datos, calcular una solución inicial de partida y una vez ejecutados, será deseable que almacenen los resultados obtenidos. Por esta razón, los algoritmos implementados se han diseñado teniendo en cuenta esta

situación. Para ello, se ha hecho uso extensivo de la herencia presente en el paradigma de orientación a objetos de diseño de software.

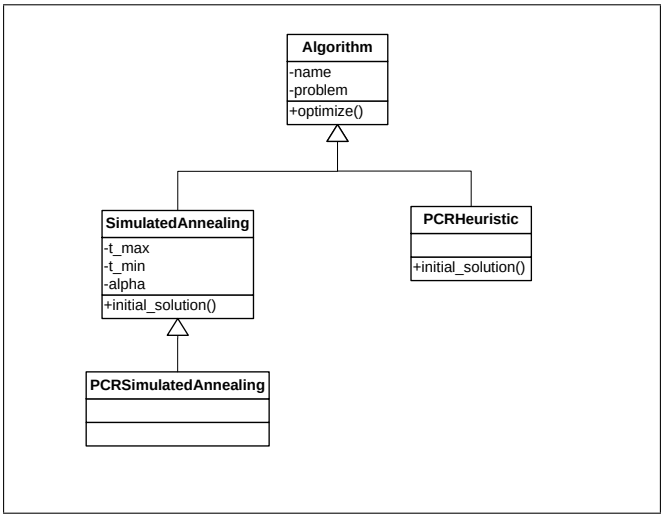


Figura 7.2: Diagrama del sistema desarrollado.

La Figura 7.2 muestra un ejemplo muy básico de la estructura de clases de los dos algoritmos planteados en este trabajo. Ambos parten de la clase padre `Algorithm`, que define las características básicas de cualquier algoritmo implementado. A partir de ahí la especialización difiere, creando una rama para cada algoritmo distinto o incluso para familias enteras de algoritmos, teniendo en cuenta los atributos y comportamientos necesarios en cada caso. Esta aproximación tiene varias ventajas. Por una parte, evita que el contenido compartido por varios algoritmos esté repartido por todos ellos, ya que las propiedades comunes se definen en la clase padre. Por otra parte, facilita enormemente el realizar cambios en los algoritmos o implementar otros nuevos, ya que solo es necesario extender la clase padre correspondiente e implementar aquello que la haga especial.

Capítulo 8

Conclusiones

En el presente trabajo se ha realizado un estudio del problema a resolver, analizando el contexto del mismo y describiendo las necesidades de las empresas que en la actualidad realizan procesos de análisis de ADN. En base a estas necesidades, se ha definido el problema de optimización relacionado, planteado para reducir principalmente los costes de operación. En relación al problema de optimización, se han estudiado las principales soluciones del estado del arte, estudiando y detallando las soluciones alcanzadas, así como el funcionamiento y las principales deficiencias de la metaheurística planteada. Sobre estas deficiencias, se han planteado una serie de mejoras que permitiesen reducir el coste computacional de los cálculos. Sin embargo, debido a un problema de eficiencia relacionado con el diseño interno de la metaheurística, estas mejoras no fueron suficientes, lo que llevó a concluir que era necesario aplicar un nuevo enfoque en la resolución del problema. Para ello se realizó un estudio detallado y completo sobre la estructura interna del problema, el cual permitió desarrollar un marco de comportamiento general que se ha traducido en una serie de contribuciones en forma de cotas que modelan el comportamiento los valores óptimos de los objetivos.

Por otra parte, el conocimiento adquirido a través del estudio del problema ha permitido proponer un nuevo algoritmo para la resolución del problema de optimización. Se trata de una heurística extremadamente rápida, capaz de, como mínimo, igualar los resultados del estado del arte, mejorándolos en muchos casos como, por ejemplo, en la ocupación total de las placas. Por otra parte, debido a su diseño, el desempeño de la heurística puede resentirse si las características habituales de los problemas que debe resolver cambian en gran medida. Por esta razón, se han analizado los principales problemas con lo que se puede topar, planteando soluciones para ellos.

Para la realización del trabajo, debido a que ha sido necesario implementar múltiples algoritmos y se ha visto la necesidad de disponer de un entorno gráfico que permita visualizar soluciones, estudios y pruebas, se ha desarrollado una aplicación web completa que permite realizar los casos de uso comunes de principio a fin. Esta aplicación permite desde visualizar un análisis general de los ficheros de entrada de los problemas, hasta la ejecución paralela y escalable de distintos algoritmos de resolución para los problemas planteados, almacenando los resultados obtenidos, así como dando la posibilidad de visualizarlos.

8.1. Trabajo Futuro

Tomando el estado actual del trabajo como punto de partida y teniendo en cuenta que este trabajo marca el inicio de la etapa predoctoral de su autor, los principales focos donde centrar la atención y las mejoras son los siguientes:

- La solución aportada por la heurística propuesta puede analizarse en dos partes. Por un lado está la primera parte de la solución, que se corresponde con el conjunto de placas llenas por completo. Por otro lado, el conjunto de placas parcialmente llenas. Para la primera parte es importante

analizar si el reparto que se ha realizado es bueno, incluso óptimo. Para la segunda parte es necesario implementar una técnica general de optimización que permita mejorar el porcentaje de ocupación de las mismas.

- Mejoras en el segundo objetivo definido en el problema, concretamente aquel que busca la mínima ocupación total en las placas. En este objetivo la heurística propuesta, si bien mejora los resultados del estado del arte, obtiene unos valores que crecen de forma rápida, por lo que sería importante conseguir ajustar mejor estos resultados a la cota calculada.
- Mejoras en el tercer objetivo, diseñando una estrategia general que permita a la heurística lidiar con las mejoras planteadas en el Capítulo 5, así como con otros casos extremos que puedan presentarse.
- Mejoras en el ajuste de las cotas que modelan el comportamiento del problema, ya que es posible que en algunos casos estas cotas obtengan valores que no son alcanzables en la realidad.
- Mejora de la aplicación web para dar cabida a un mayor número de casos de uso como, por ejemplo, más utilidades dedicadas al análisis descriptivo de las soluciones, dar soporte a la definición de tareas a ejecutar por lotes o habilitar un análisis automatizado del consumo de recursos de los algoritmos implementados.

8.1.1. Casos extremos

Ocupación total de las placas

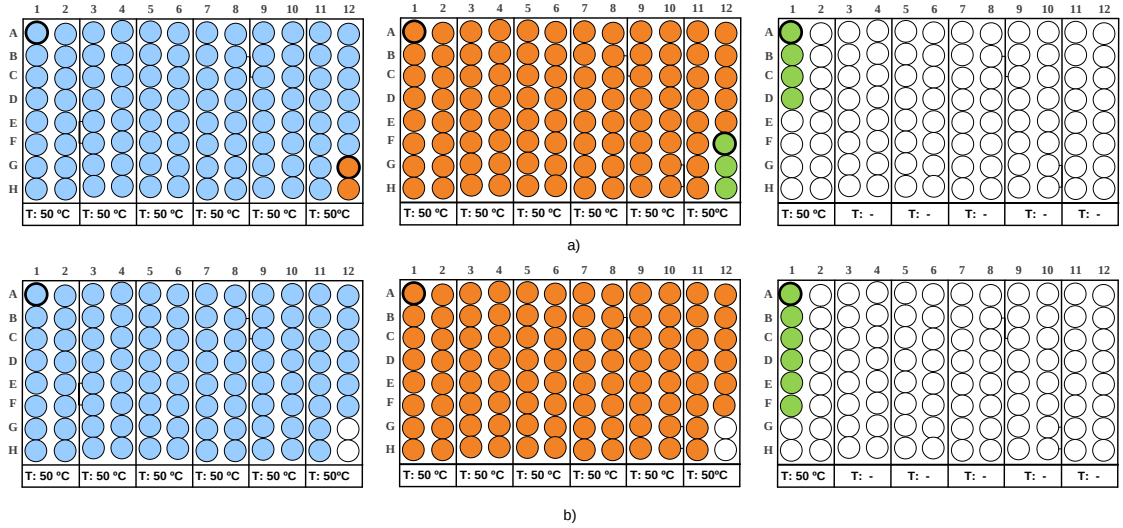


Figura 8.1: Ejemplo de caso extremo debido al reparto de los grupos.

Debido al funcionamiento de la solución inicial de la heurística propuesta, es posible que los grupos de muestras queden repartidos en 2 placas, lo que implicaría colocar 2 indicadores de control, uno en cada placa en la que está presente el grupo.

Por ejemplo, la Figura 8.1 a) muestra la solución obtenida por el algoritmo para un problema conformado por 3 grupos; 2 de ellos de 93 muestras y uno de 5 muestras. Puede verse que, con el objetivo de reducir el número de placas a utilizar, el algoritmo fragmenta los grupos si es necesario. Sin embargo, esta solución utiliza 5 indicadores de control en total, mientras que la Figura 8.1 b) muestra una disposición de los grupos que, ocupando el mismo número de franjas, permite reducir el número de indicadores a 3, reduciendo de esta forma la ocupación total de las placas.

Cabe destacar que no siempre será posible realizar este tipo de reorganizaciones sin perjudicar al número de placas utilizadas, por esta razón, es necesario que la heurística sea capaz de determinar, en cada caso, si es posible minimizar la ocupación total sin utilizar placas adicionales.

Porcentajes de ocupación de las placas

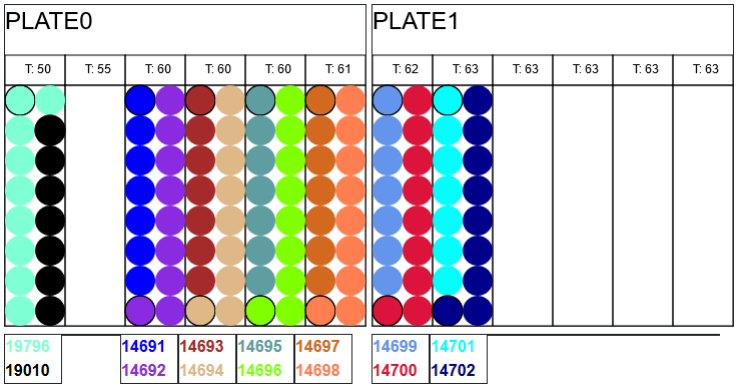


Figura 8.2: Ejemplo de caso extremo debido a la distribución de temperaturas.

En ocasiones, los algoritmos se encuentran con casos muy complejos debido a su estructura que, no siendo la más habitual, es necesario tenerla en cuenta para asegurar una buena adaptabilidad y mantener un buen desempeño en términos generales. En el problema planteado en este trabajo, se ha visto cómo los ficheros de entrada suelen tener unas características similares cuando se trata de la distribución de temperaturas y el reparto de las muestras en ellas. Sin embargo, un cambio en estas características, producido por, por ejemplo, un cambio en las pruebas más habituales que realiza el laboratorio, un cambio en los reactivos utilizados en el proceso PCR o la adopción de nuevas pruebas podría generar casos que no hayan sido planteados hasta el momento. Estos casos podrían afectar al rendimiento del algoritmo si éste no está preparado. Sin embargo, muchos de estos problemas tienen fácil solución.

A continuación se estudiará un tipo de problema que podría darse y que está relacionado con la distribución de temperaturas del fichero a resolver. Suponiendo un fichero de las siguientes características:

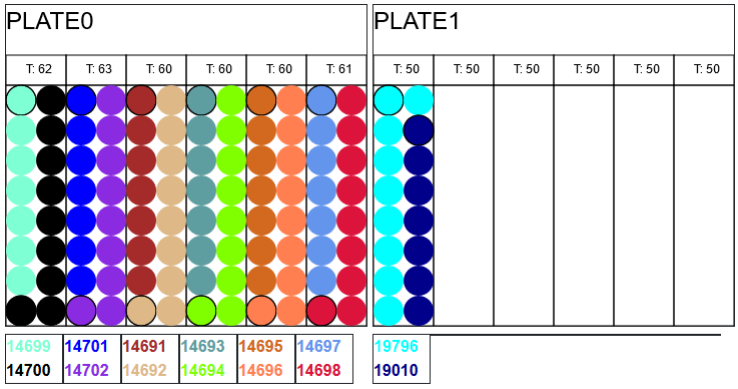


Figura 8.3: Ejemplo de resolución de caso extremo.

- 112 muestras
- 14 grupos
- 7 franjas de temperatura, cuyos valores son: 50 °C, 60 °C, 61 °C, 62 °C y 63 °C.

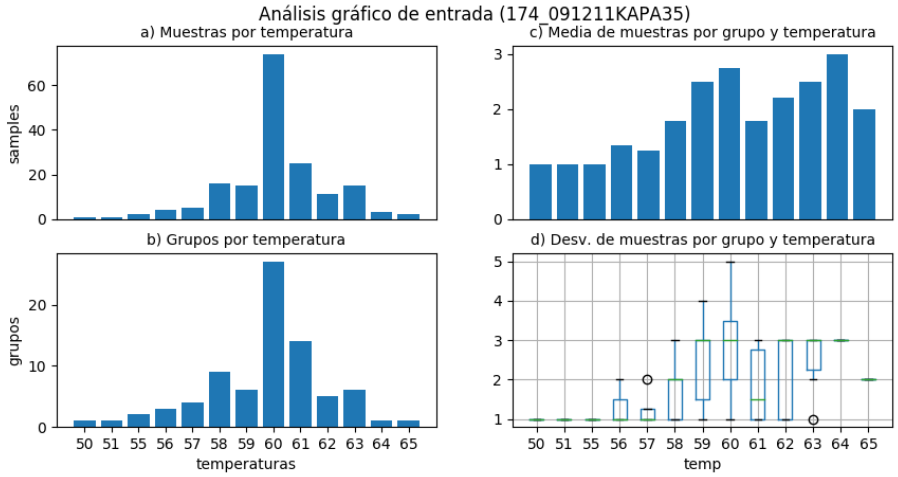
La Figura 8.2 muestra el diagrama resultante de la solución aportada por la heurística para el problema definido. Se trata de un problema pequeño, pero existe un gran salto en los valores de las temperaturas de procesado de las muestras. En un caso así, debido a que la heurística ordena las muestras en función de la temperatura de procesado de los grupos a los que pertenecen y debido a la restricción de la diferencia de temperatura entre franjas adyacentes, la solución que alcanza es la mostrada en la figura. Esta solución es fácilmente mejorable simplemente aislando la franja de menor temperatura, pues es esta franja la causante de la necesidad de crear una separación con las demás.

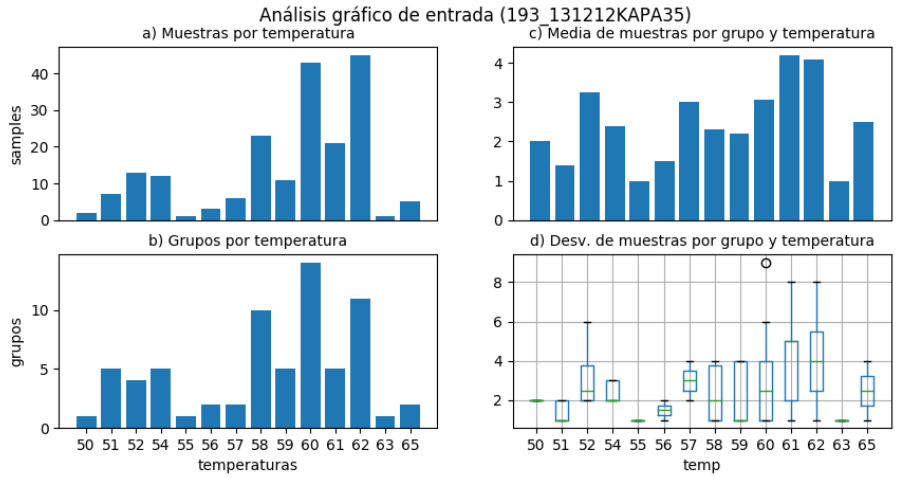
La Figura 8.3 muestra una posible solución al problema. Se ha conseguido que una de las placas esté totalmente completa cuando antes eso no ocurría. Además, ahora la placa de menor ocupación ocupa el último lugar de la solución y también cabe destacar que no se ha perjudicado ningún otro objetivo. Para obtener esta solución puede utilizarse una lógica similar que en el caso anterior, ya que puede verse el problema como uno donde las placas están parcialmente llenas y se está tratando de optimizar el porcentaje de ocupación de las mismas.

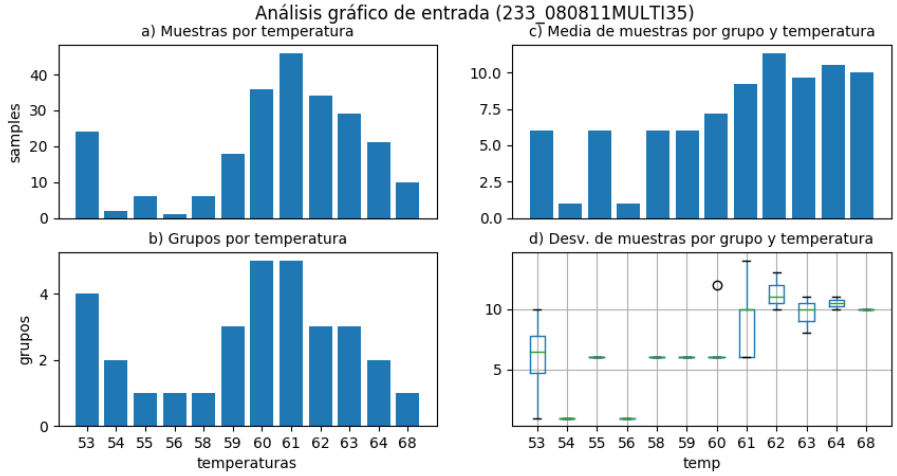
Apéndice A

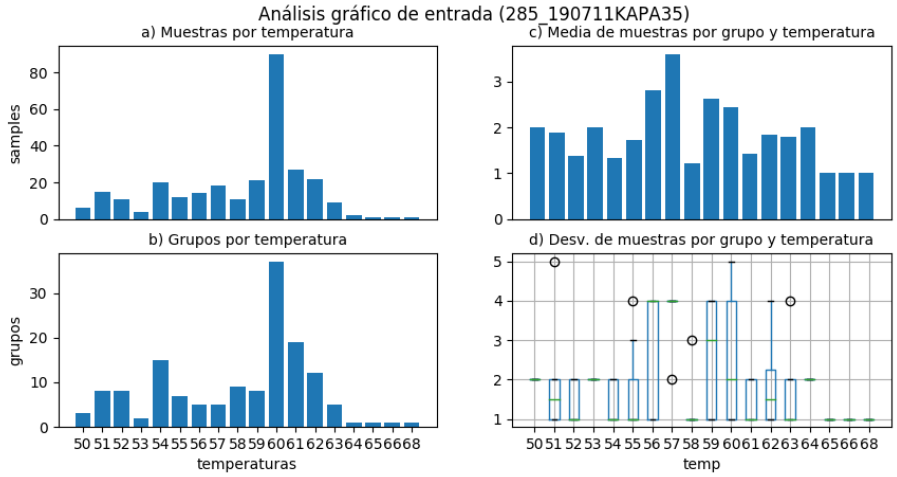
Estudio de ficheros de entrada

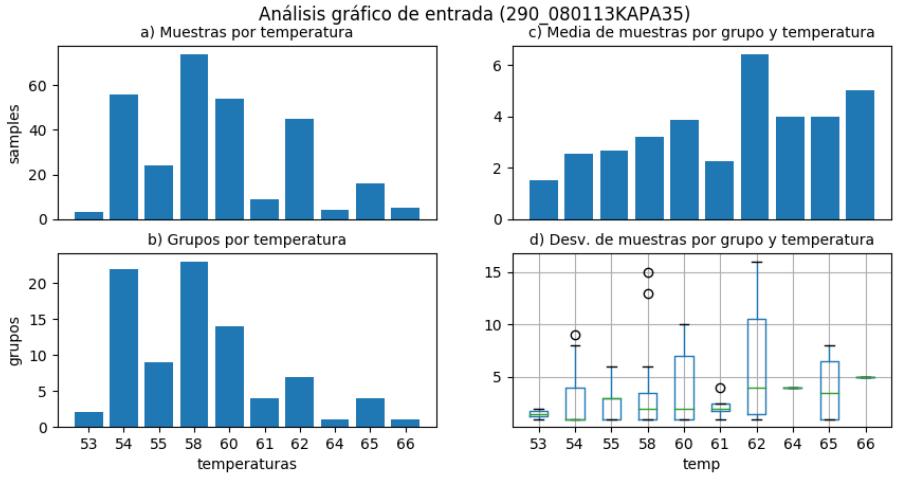
Esta sección presenta todos los gráficos relacionados con el estudio de los ficheros de entrada realizado en el Capítulo 1.

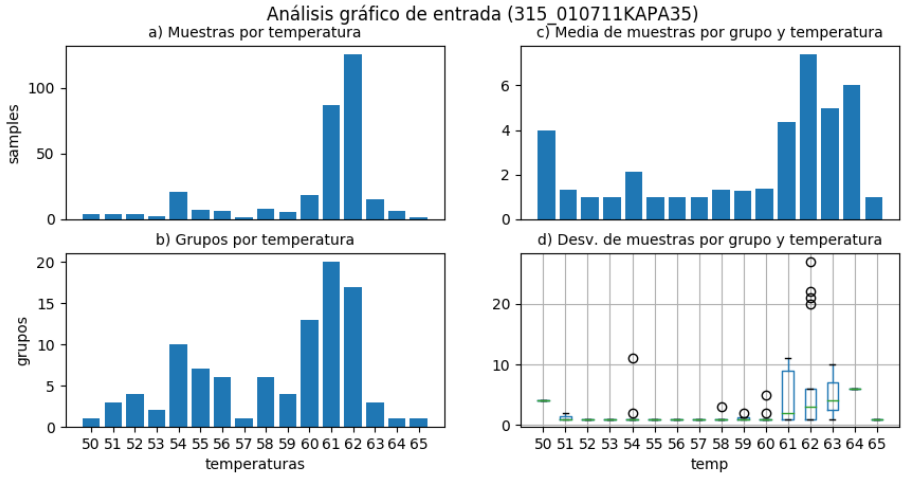


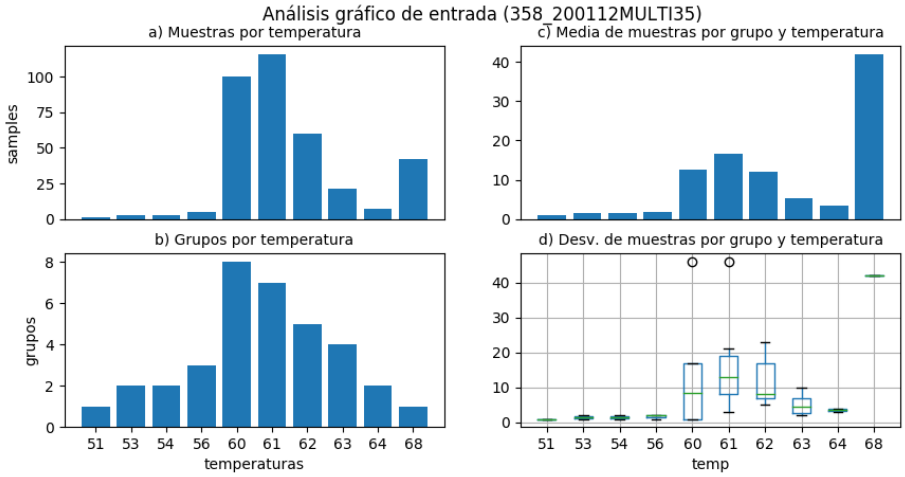


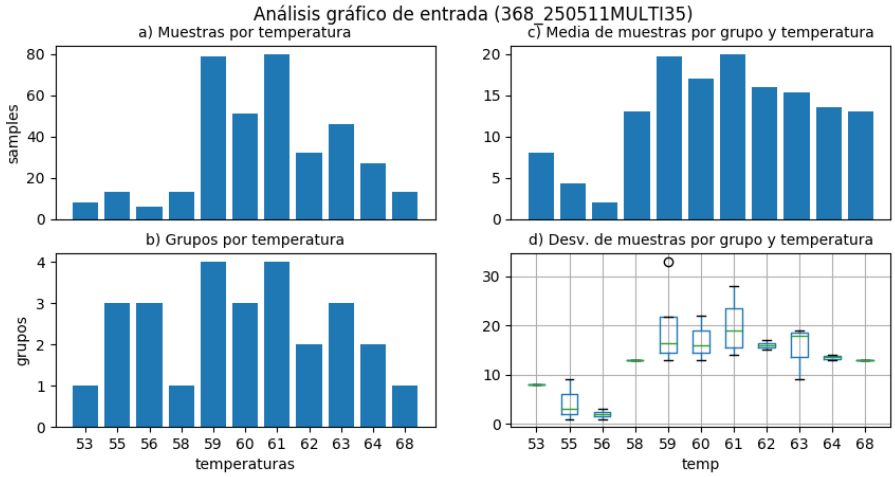

Análisis gráfico de entrada (233_080811MULTI35)
a) Muestras por temperatura
c) Media de muestras por grupo y temperatura
b) Grupos por temperatura
d) Desv. de muestras por grupo y temperatura
samples
grupos
temperaturas
temp

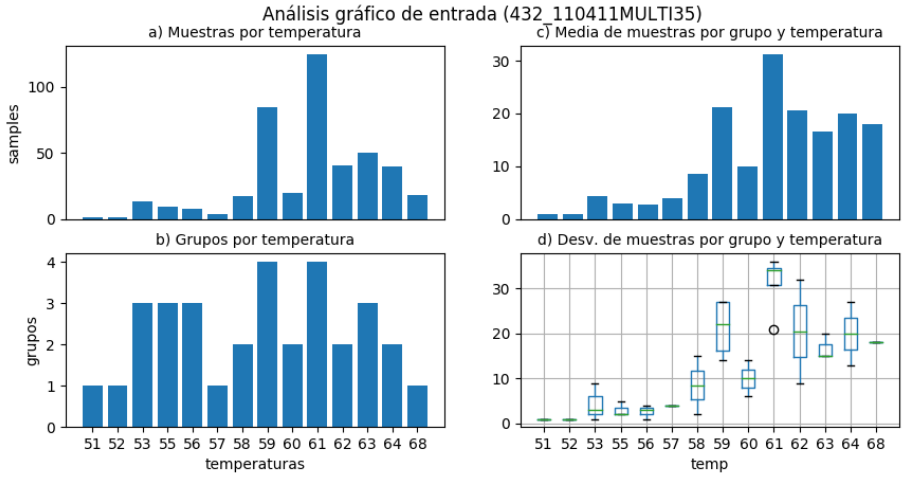

Análisis gráfico de entrada (285_190711KAPA35)
a) Muestras por temperatura
c) Media de muestras por grupo y temperatura
b) Grupos por temperatura
d) Desv. de muestras por grupo y temperatura
samples
grupos
temperaturas
temp

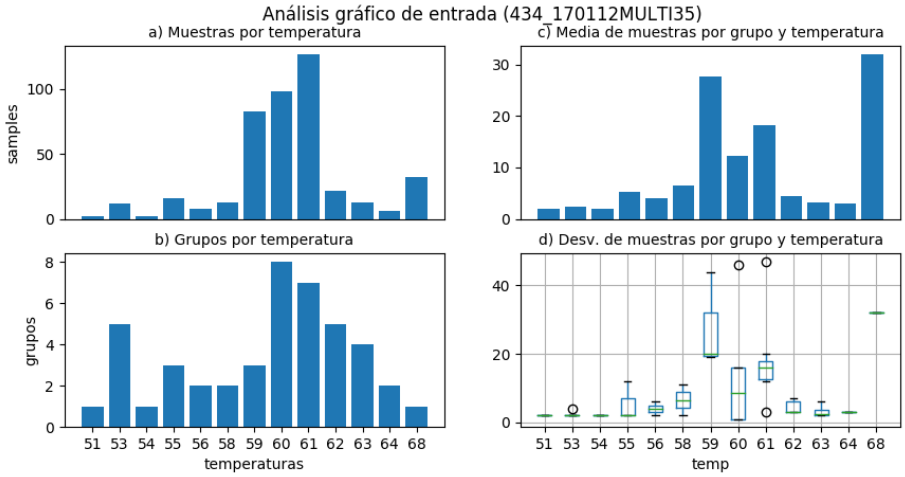

Análisis gráfico de entrada (290_080113KAPA35)
a) Muestras por temperatura
c) Media de muestras por grupo y temperatura
b) Grupos por temperatura
d) Desv. de muestras por grupo y temperatura
samples
grupos
temperaturas
temp

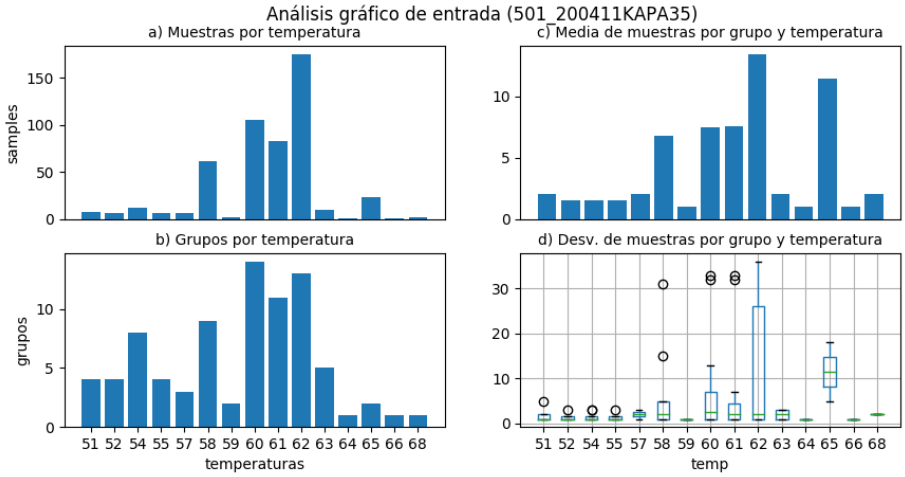

Análisis gráfico de entrada (315_010711KAPA35)
a) Muestras por temperatura
c) Media de muestras por grupo y temperatura
b) Grupos por temperatura
d) Desv. de muestras por grupo y temperatura
samples
grupos
temperaturas
temp

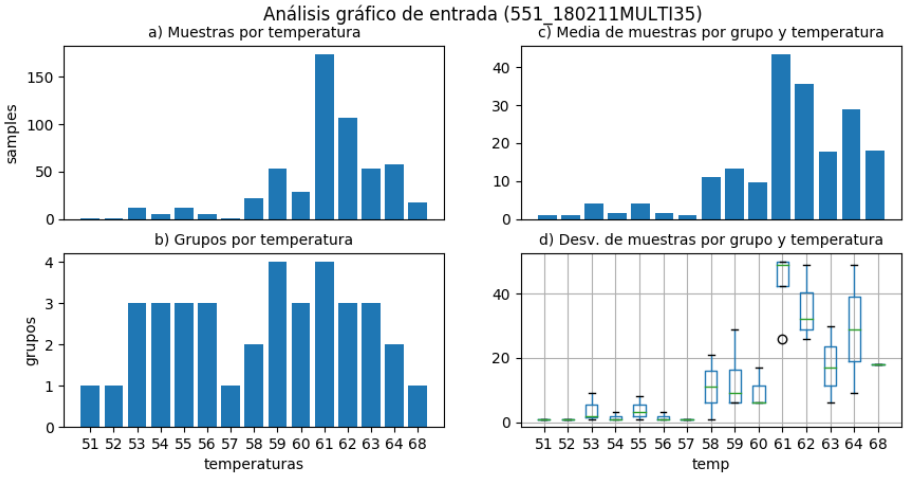

Análisis gráfico de entrada (358_200112MULTI35)
a) Muestras por temperatura
c) Media de muestras por grupo y temperatura
b) Grupos por temperatura
d) Desv. de muestras por grupo y temperatura
samples
grupos
temperaturas
temp

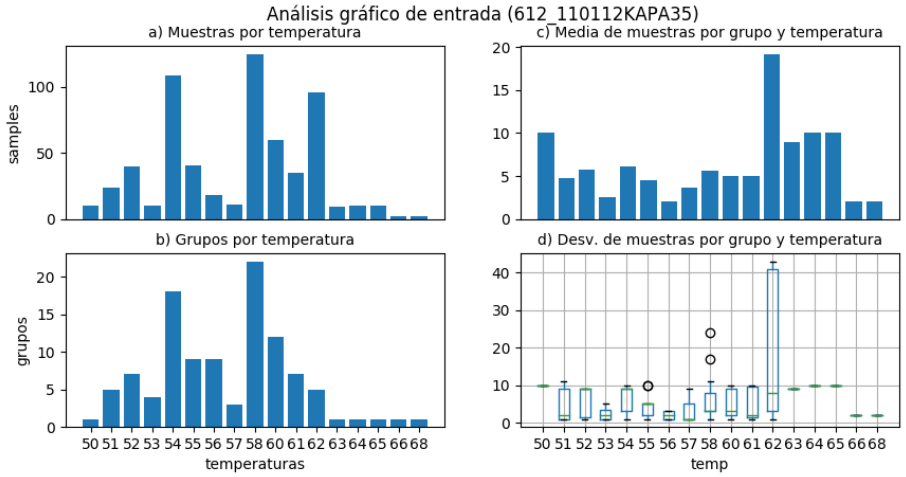

Análisis gráfico de entrada (368_250511MULTI35)
a) Muestras por temperatura
c) Media de muestras por grupo y temperatura
b) Grupos por temperatura
d) Desv. de muestras por grupo y temperatura
samples
grupos
temperaturas
temp

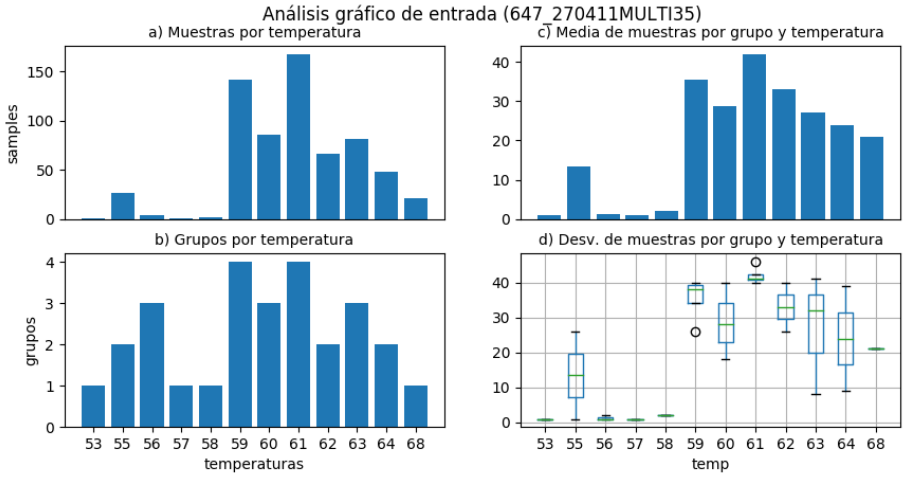

Análisis gráfico de entrada (432_110411MULTI35)
a) Muestras por temperatura
c) Media de muestras por grupo y temperatura
b) Grupos por temperatura
d) Desv. de muestras por grupo y temperatura
samples
grupos
temperaturas
temp
51 52 53 55 56 57 58 59 60 61 62 63 64 68

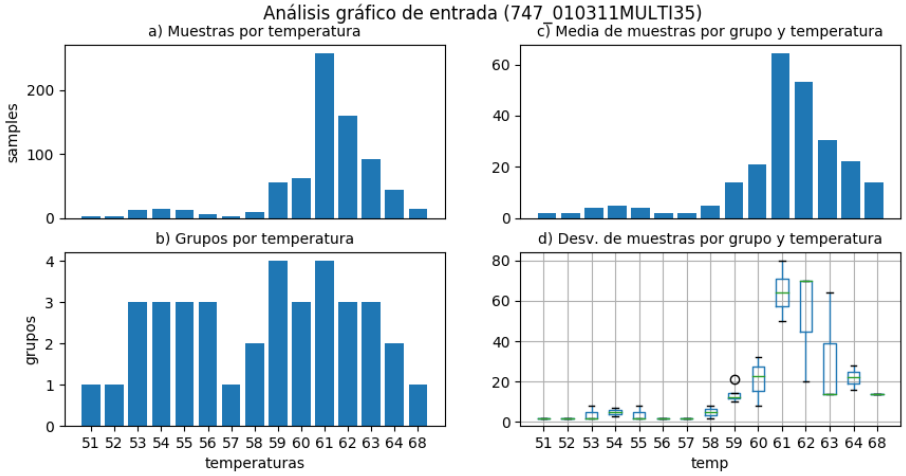

Análisis gráfico de entrada (434_170112MULTI35)
a) Muestras por temperatura
c) Media de muestras por grupo y temperatura
b) Grupos por temperatura
d) Desv. de muestras por grupo y temperatura
samples
grupos
temperaturas
temp
51 53 54 55 56 58 59 60 61 62 63 64 68

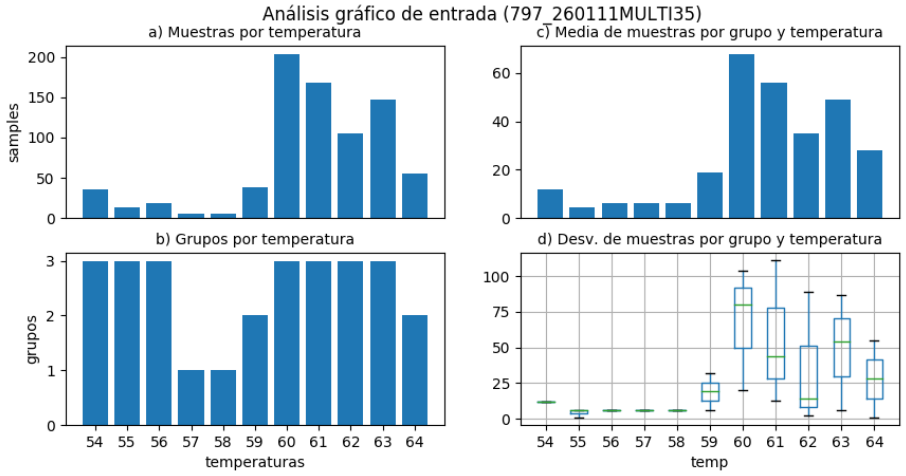

Análisis gráfico de entrada (501_200411KAPA35)
a) Muestras por temperatura
c) Media de muestras por grupo y temperatura
b) Grupos por temperatura
d) Desv. de muestras por grupo y temperatura
samples
grupos
temperaturas
temp
51 52 54 55 57 58 59 60 61 62 63 64 65 66 68

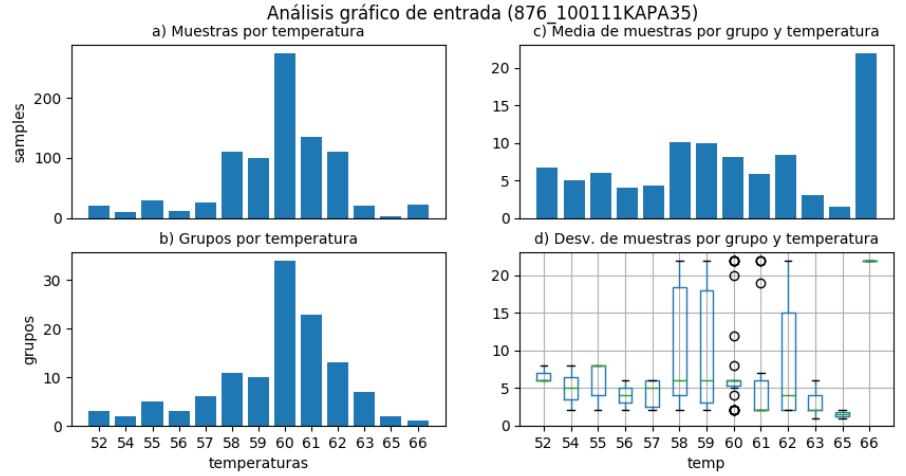

Análisis gráfico de entrada (551_180211MULTI35)
a) Muestras por temperatura
c) Media de muestras por grupo y temperatura
b) Grupos por temperatura
d) Desv. de muestras por grupo y temperatura
samples
grupos
temperaturas
temp

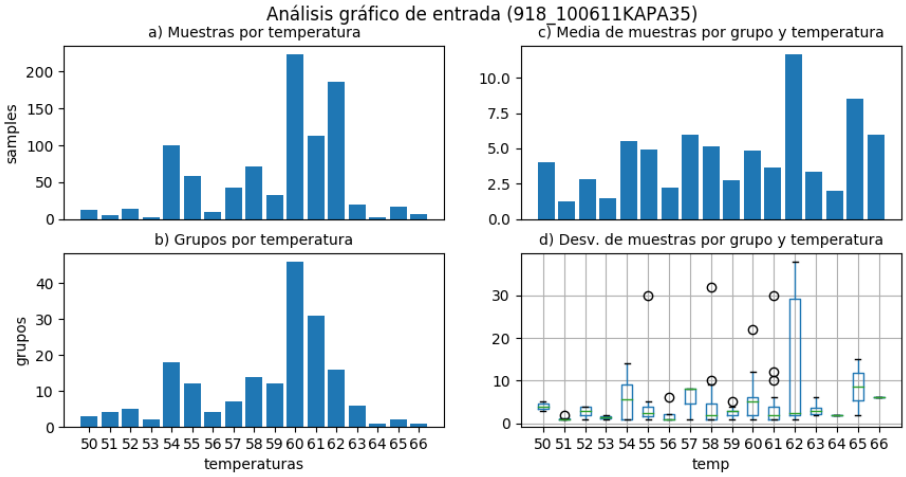

Análisis gráfico de entrada (612_110112KAPA35)
a) Muestras por temperatura
c) Media de muestras por grupo y temperatura
b) Grupos por temperatura
d) Desv. de muestras por grupo y temperatura
samples
grupos
temperaturas
temp

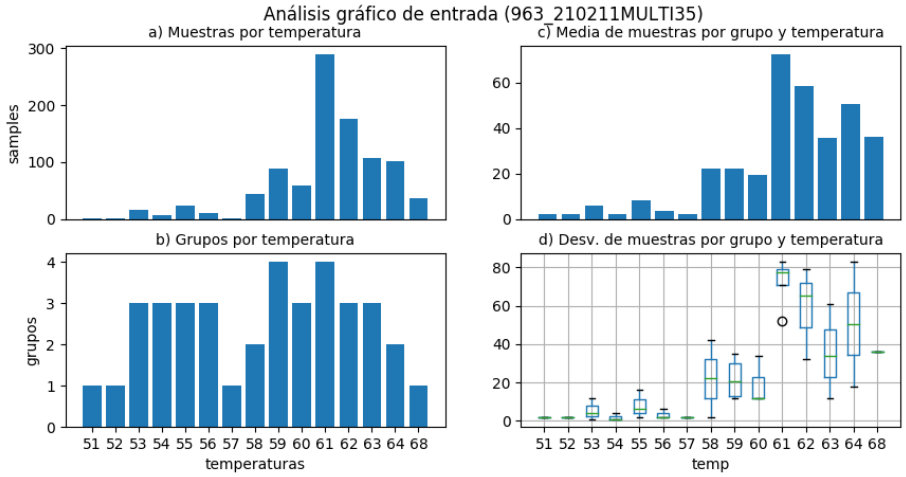

Análisis gráfico de entrada (647_270411MULTI35)
a) Muestras por temperatura
c) Media de muestras por grupo y temperatura
b) Grupos por temperatura
d) Desv. de muestras por grupo y temperatura
samples
grupos
temperaturas
temp

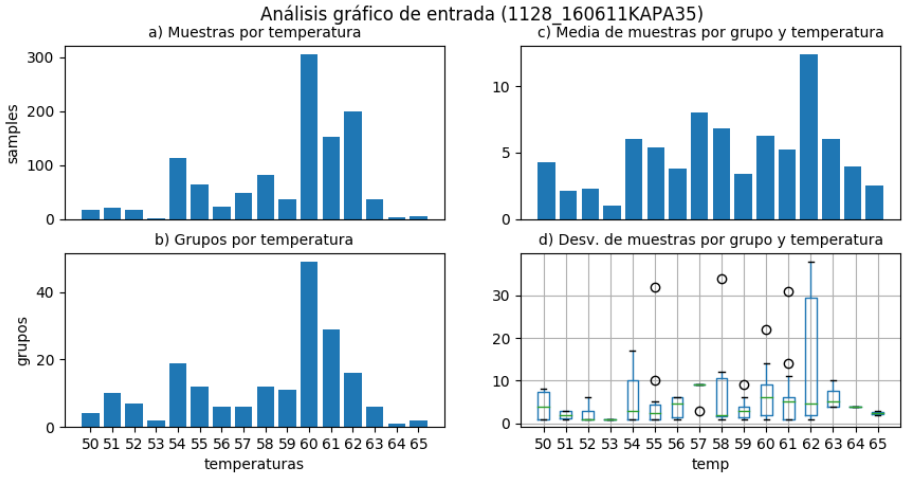

Análisis gráfico de entrada (747_010311MULTI35)
a) Muestras por temperatura
c) Media de muestras por grupo y temperatura
b) Grupos por temperatura
d) Desv. de muestras por grupo y temperatura
samples
grupos
temperaturas
temp

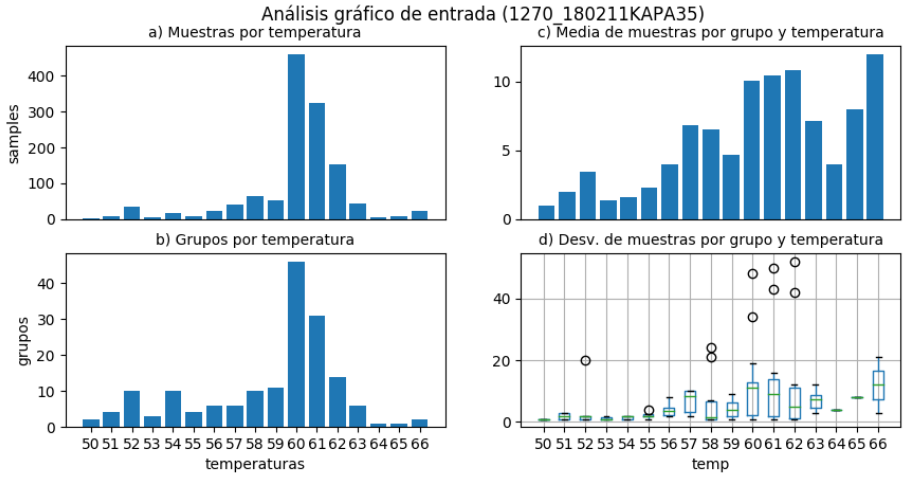

Análisis gráfico de entrada (797_260111MULTI35)
a) Muestras por temperatura
c) Media de muestras por grupo y temperatura
b) Grupos por temperatura
d) Desv. de muestras por grupo y temperatura
samples
grupos
temperaturas
temp

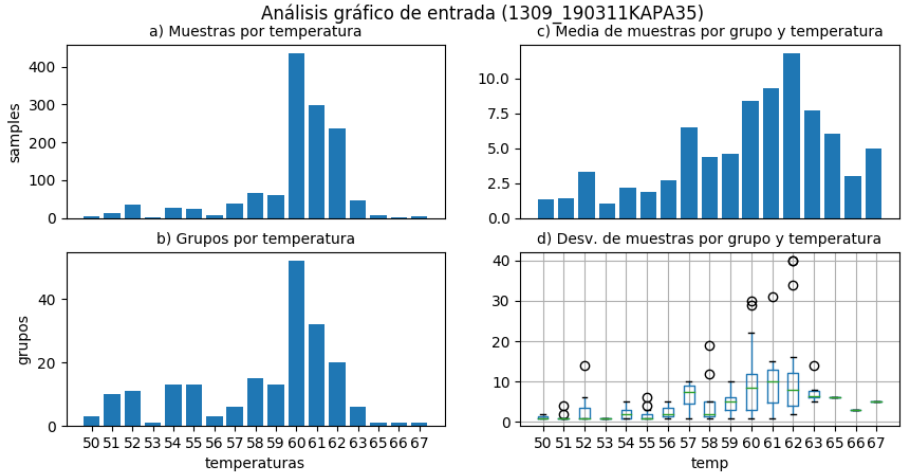

Análisis gráfico de entrada (876_100111KAPA35)
a) Muestras por temperatura
c) Media de muestras por grupo y temperatura
b) Grupos por temperatura
d) Desv. de muestras por grupo y temperatura
samples
grupos
temperaturas
temp

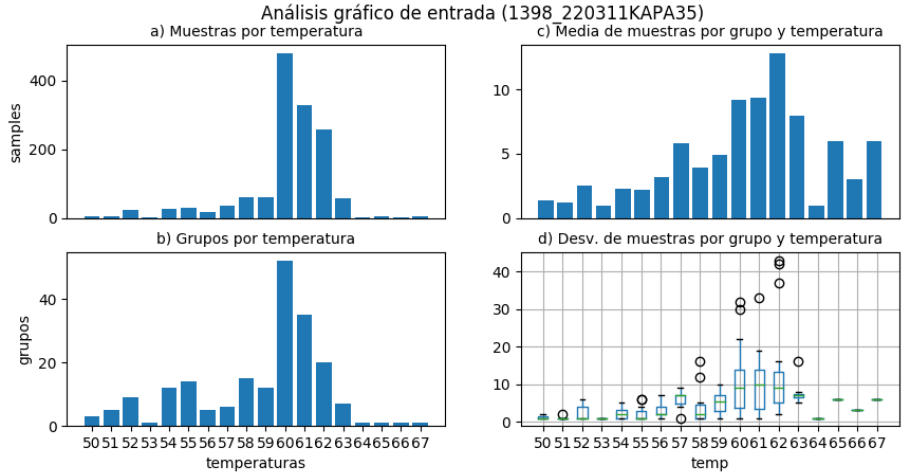

Análisis gráfico de entrada (918_100611KAPA35)
a) Muestras por temperatura
c) Media de muestras por grupo y temperatura
b) Grupos por temperatura
d) Desv. de muestras por grupo y temperatura
samples
grupos
temperaturas
temp

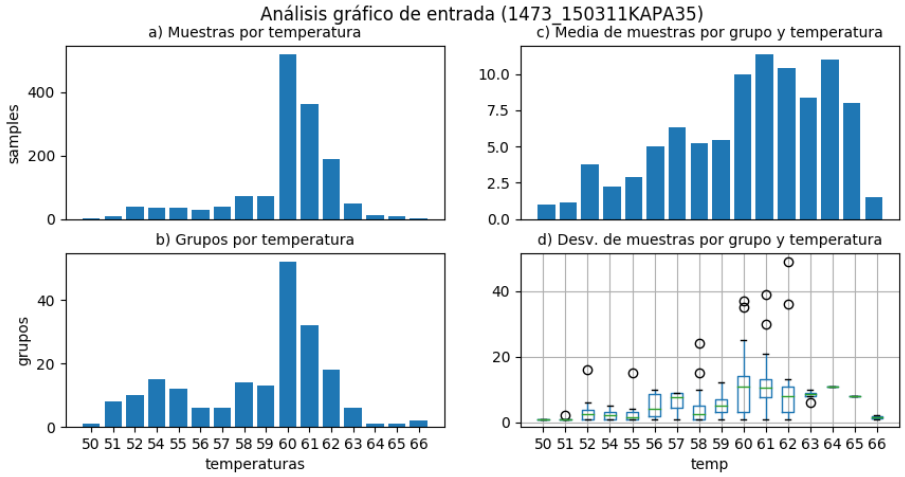

Análisis gráfico de entrada (963_210211MULTI35)
a) Muestras por temperatura
c) Media de muestras por grupo y temperatura
b) Grupos por temperatura
d) Desv. de muestras por grupo y temperatura
samples
grupos
temperaturas
temp

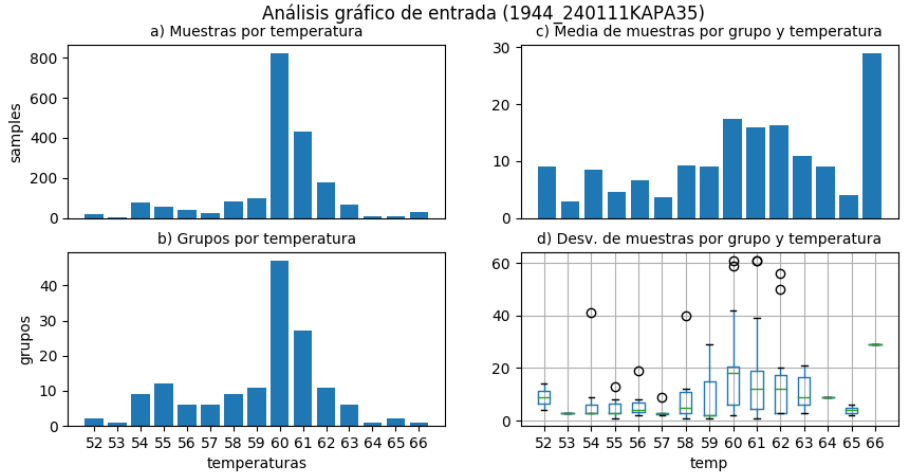

Análisis gráfico de entrada (1128_160611KAPA35)
a) Muestras por temperatura
c) Media de muestras por grupo y temperatura
b) Grupos por temperatura
d) Desv. de muestras por grupo y temperatura
samples
grupos
temperaturas
temp

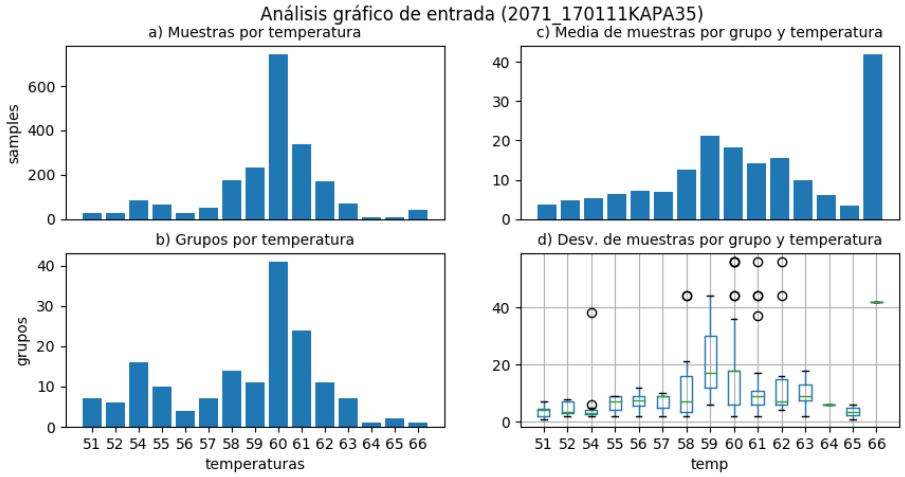

Análisis gráfico de entrada (1270_180211KAPA35)
a) Muestras por temperatura
c) Media de muestras por grupo y temperatura
b) Grupos por temperatura
d) Desv. de muestras por grupo y temperatura
samples
grupos
temperaturas
temp

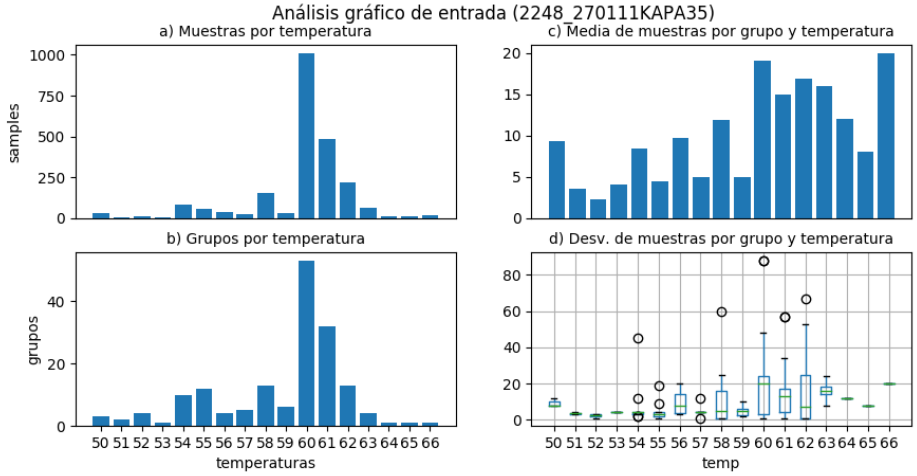

Análisis gráfico de entrada (1309_190311KAPA35)
a) Muestras por temperatura
c) Media de muestras por grupo y temperatura
b) Grupos por temperatura
d) Desv. de muestras por grupo y temperatura
samples
grupos
temperaturas
temp

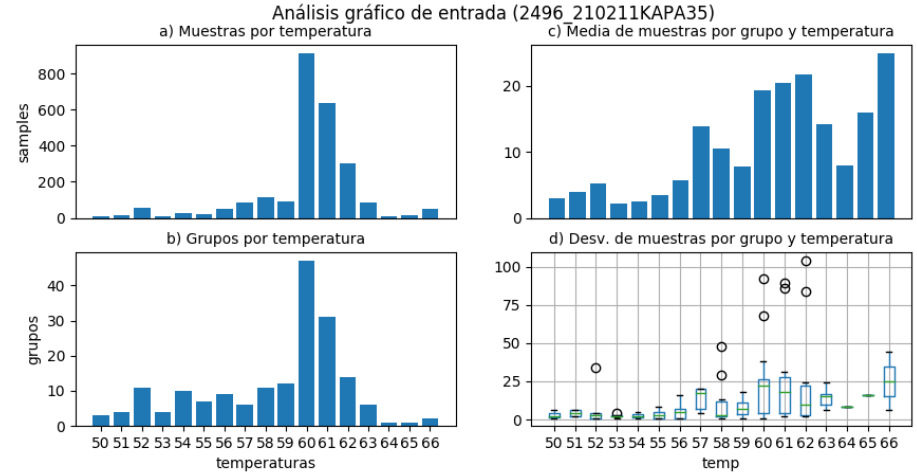

Análisis gráfico de entrada (1398_220311KAPA35)
a) Muestras por temperatura
c) Media de muestras por grupo y temperatura
b) Grupos por temperatura
d) Desv. de muestras por grupo y temperatura
samples
grupos
temperaturas
temp

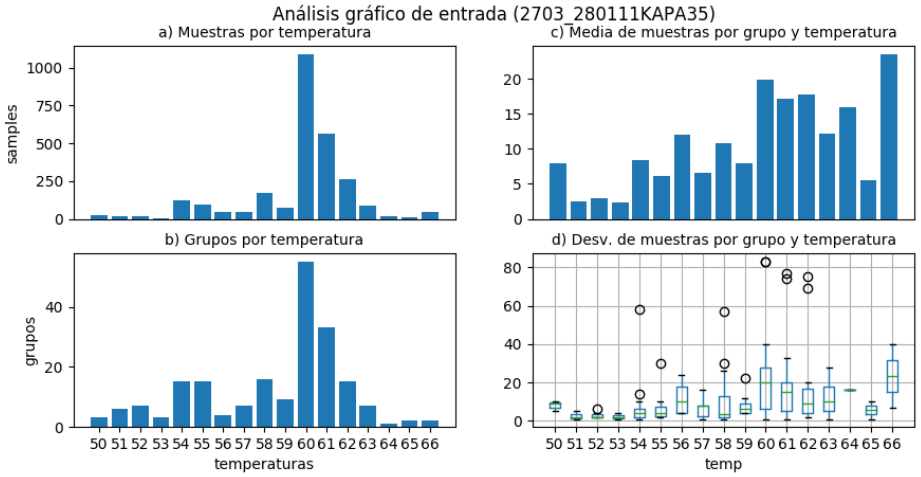

Análisis gráfico de entrada (1473_150311KAPA35)
a) Muestras por temperatura
c) Media de muestras por grupo y temperatura
b) Grupos por temperatura
d) Desv. de muestras por grupo y temperatura
samples
grupos
temperaturas
temp
50 51 52 54 55 56 57 58 59 60 61 62 63 64 65 66

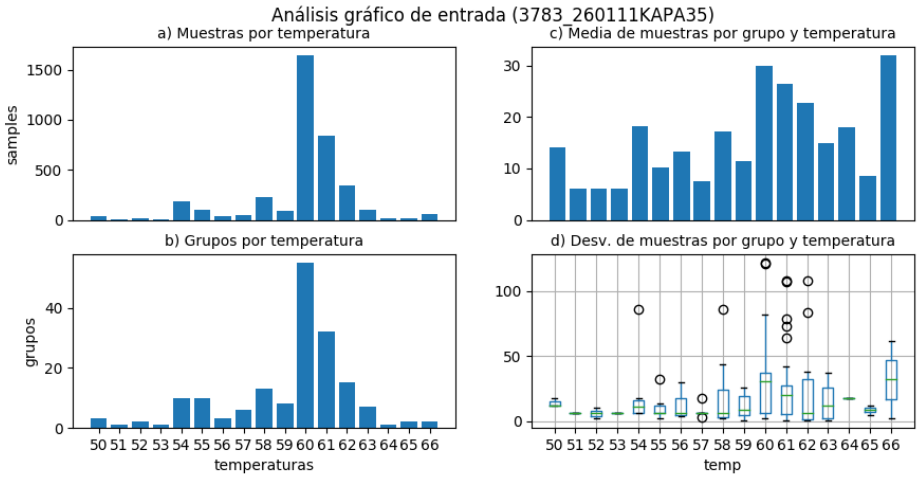

Análisis gráfico de entrada (1944_240111KAPA35)
a) Muestras por temperatura
c) Media de muestras por grupo y temperatura
b) Grupos por temperatura
d) Desv. de muestras por grupo y temperatura
samples
grupos
temperaturas
temp
52 53 54 55 56 57 58 59 60 61 62 63 64 65 66


Análisis gráfico de entrada (2071_170111KAPA35)
a) Muestras por temperatura
c) Media de muestras por grupo y temperatura
b) Grupos por temperatura
d) Desv. de muestras por grupo y temperatura
samples
grupos
temperaturas
temp
51 52 54 55 56 57 58 59 60 61 62 63 64 65 66


Análisis gráfico de entrada (2248_270111KAPA35)
a) Muestras por temperatura
c) Media de muestras por grupo y temperatura
b) Grupos por temperatura
d) Desv. de muestras por grupo y temperatura
samples
grupos
temperaturas
temp


Análisis gráfico de entrada (2496_210211KAPA35)
a) Muestras por temperatura
c) Media de muestras por grupo y temperatura
b) Grupos por temperatura
d) Desv. de muestras por grupo y temperatura
samples
grupos
temperaturas
temp


Análisis gráfico de entrada (2703_280111KAPA35)
a) Muestras por temperatura
c) Media de muestras por grupo y temperatura
b) Grupos por temperatura
d) Desv. de muestras por grupo y temperatura
samples
grupos
temperaturas
temp


Análisis gráfico de entrada (3783_260111KAPA35)
a) Muestras por temperatura
samples
1500
1000
500
0
c) Media de muestras por grupo y temperatura
30
20
10
0
b) Grupos por temperatura
grupos
40
20
0
50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66
temperaturas
d) Desv. de muestras por grupo y temperatura
100
50
0
50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66
temp

Apéndice B

Código fuente de la heurística propuesta

```
# -*- coding: utf-8 -*-

import time
from testbedapp.models.problems.pcr_unrestricted import solution
from testbedapp.models.problems.pcr_base.entity import plate
import testbedapp.models.problems.pcr_base.exceptions.plate
import testbedapp.models.problems.pcr_base.exceptions.strip
import collections
import copy


class Heuristic():
  def __init__(self, problem):
    self.problem = problem
    self._full_plates = []
    self._partial_plates = []
    self._discarded_strips = []
    self._plate_idx = 0

  def group_by_temp(self):
    """
      Coloca las muestras de entrada en un diccionario ordenado por temperatura
    """
    temps = collections.OrderedDict()
    for s in self.problem.samples:
      if s.get_group_temp() not in temps:
        temps[s.get_group_temp()] = []
      temps[s.get_group_temp()].append(s)

    return temps

  def fill_strips(self, limit=15):
    """
      Busca franjas con 15 pocillos ocupados e intenta rellenarlas con otros
    grupos de la misma temperatura que estén en franjas "desechadas"
    """
    for p in self._full_plates:
```

```
    for st in p.strips:
      if st.count_nonempty_wells() == 15:
        for lst in self._discarded_strips:
          if lst.count_nonempty_wells() < limit:
            if lst.get_temp() == st.get_temp():
              candidate = None
              grps = st.get_all_groups()
              for g in grps:
                if g['size'] > 2:
                  for s in g['samples']:
                    if not s.is_indicator():
                      candidate = s
              grps = lst.get_all_groups()
              for g in grps:
                if g['size'] >= 2:
                  if not lst.is_indicator_present(g['id']):
                    continue
                  st.remove_sample(candidate)
                  if not p.is_group_indicator_present(g['id']):
                    st.add_indicator(g['id'], lst.get_temp())
                  for w in g['wells']:
                    if lst.wells[w].is_indicator():
                      continue
                    try:
                      st.assign_sample(lst.wells[w])
                      lst.remove_sample(lst.wells[w])
                    except testbedapp.models.problems.pcr_base.exceptions.
strip.StripNotEnoughWellsError:
                      break
                  lst.assign_sample(candidate)
                  break
              break

def fill_empty_slots(self):
  """
    Busca franjas vacías en las placas de la solución e intenta llenarlos con
   los mejores candidatos de las franjas desechadas

    Con la forma de ordenar de la línea siguiente, el algoritmo puede ser
  capaz de llenar más placas en algunos problemas a base de ocupar más placas.
    # self._discarded_strips = sorted(self._discarded_strips, key=lambda x: (
  x.count_nonempty_wells(), x.get_temp()), reverse=True)
  """
  self._discarded_strips = sorted(self._discarded_strips, key=lambda x: x.
  get_temp(), reverse=True)
  for p in self._full_plates:
    used_temps = []
    for st in p.strips:
      if st.is_empty():
        for lst in self._discarded_strips:
          if p.is_valid_temp_diff(st.strip_id, lst.get_temp()):
            if lst.get_temp() in used_temps:
              continue
            used_temps.append(lst.get_temp())
            lst.strip_id = st.strip_id
            p.strips[st.strip_id] = lst
```

```
            self._discarded_strips.remove(lst)
            grps = lst.get_all_groups()
            for g in grps:
              if not p.is_group_indicator_present(g['id']):
                lst.add_indicator(g['id'], lst.get_temp())
            break

def populate_partial_plates(self):
  """
    Coloca las franjas desechadas restantes en nuevas placas de la mejor
  forma posible
  """
  for lst in self._discarded_strips:
    p = self._partial_plates[-1]
    try:
      dest_st = p.find_empty_strip_for_temp(lst.get_temp())
      lst.strip_id = dest_st.strip_id
      p.strips[dest_st.strip_id] = lst
      st = p.strips[dest_st.strip_id]
    except testbedapp.models.problems.pcr_base.exceptions.plate.
  PlateSlotNotAvailableError:
      p = plate.Plate(self._plate_idx, self.problem.max_temp_diff)
      self._plate_idx += 1
      self._partial_plates.append(p)
      p.strips[0] = lst
      p.strips[0].strip_id = 0
      st = p.strips[0]
    grps = st.get_all_groups()
    for g in grps:
      if not p.is_group_indicator_present(g['id']):
        st.add_indicator(g['id'], st.get_temp())

def fill_indicators(self):
  """
    Coloca los indicadores faltantes en las placas parciales, las cuales está
  n compuestas por franjas desechadas donde no se controla la presencia de
  indicadores
  """
  for p in self._partial_plates:
    for st in p.strips:
      grps = st.get_all_groups()
      for g in grps:
        if not p.is_group_indicator_present(g['id']):
          st.add_indicator(g['id'], st.get_temp())

def initial_solution(self, temps):
  """
    Calcula la solución inicial del algoritmo, obteniendo una lista de placas
   lo más llenas posible y una lista de franjas desechadas
  :param temps: Diccionario ordenado que contiene las muestras de cada
  temperatura
  """
  for temp, samples in temps.items():
    while samples:
      s = samples.pop()
```

```
        p = self._full_plates[-1]
        stlist = p.find_strips_by_temp(int(temp))

        if not stlist:
          try:
            st = p.find_empty_strip_for_temp(int(temp))
          except testbedapp.models.problems.pcr_base.exceptions.plate.
PlateSlotNotAvailableError:
            p = plate.Plate(self._plate_idx, self.problem.max_temp_diff)
            self._plate_idx += 1
            self._full_plates.append(p)
            st = p.strips[0]
            st.set_temp(int(temp))
        else:
          st = stlist[-1]
          if st.count_empty_wells() == 0:
            try:
              st = p.find_empty_strip_for_temp(int(temp))
            except testbedapp.models.problems.pcr_base.exceptions.plate.
PlateSlotNotAvailableError:
              p = plate.Plate(self._plate_idx, self.problem.max_temp_diff)
              self._plate_idx += 1
              self._full_plates.append(p)
              st = p.strips[0]
              st.set_temp(int(temp))

        assigned = False
        if not p.is_group_indicator_present(s.get_group()):
          if st.strip_id == 5:
            if st.count_empty_wells() >= 2:
              st.add_indicator(s.get_group(), int(temp))
              st.assign_sample(s)
              assigned = True

            elif st.count_empty_wells() == 1:
              # Las franjas con 15 muestras se desechan...
              new_st = copy.deepcopy(st)
              self._discarded_strips.append(new_st)

              #.. así como posibles indicadores en franjas anteriores.
              previous_st = p.strips[st.strip_id-1]
              if previous_st.wells[15] is not None and previous_st.wells[15].
is_indicator():
                p.strips[st.strip_id-1].remove_sample(previous_st.wells[15])

              st.reset()
              assigned = False
          else:
            if st.count_empty_wells() >= 2:
              st.add_indicator(s.get_group(), int(temp))
              st.assign_sample(s)
              assigned = True
            elif st.count_empty_wells() == 1:
              st.add_indicator(s.get_group(), int(temp))
              assigned = False
        else:
```

```
            if st.count_empty_wells() >= 1:
              st.assign_sample(s)
              assigned = True
          if not assigned:
            try:
              st = p.find_empty_strip_for_temp(int(temp))
            except testbedapp.models.problems.pcr_base.exceptions.plate.
    PlateSlotNotAvailableError:
              p = plate.Plate(self._plate_idx, self.problem.max_temp_diff)
              self._plate_idx += 1
              self._full_plates.append(p)
              st = p.strips[0]
              st.set_temp(int(temp))
            if not p.is_group_indicator_present(s.get_group()):
              st.add_indicator(s.get_group(), int(temp))
            st.assign_sample(s)

        if not st.is_full():
          new_st = copy.deepcopy(st)
          self._discarded_strips.append(new_st)

          # Si existe un indicador en el último pocillo de la franja anterior, se
     elimina.
          previous_st = p.strips[st.strip_id - 1]
          if previous_st.wells[15] is not None and previous_st.wells[15].
    is_indicator():
            p.strips[st.strip_id - 1].remove_sample(previous_st.wells[15])

          st.reset()

    def optimize(self):
      stats = {}
      start_time = time.time()

      self._full_plates = [plate.Plate(self._plate_idx, self.problem.
      max_temp_diff)]
      self._plate_idx += 1
      self._partial_plates = [plate.Plate(self._plate_idx, self.problem.
      max_temp_diff)]
      self._plate_idx += 1

      # Agrupa por temperaturas
      temps = self.group_by_temp()

      # 1. Llena tantas franjas como sea posible, el resto las desecha y las
      almacena en una lista a parte
      self.initial_solution(temps)
      init_sol = solution.PCRSolution(copy.deepcopy(self._full_plates), self.
      problem.max_temp_diff)
      init_sol.calculate_objectives()

      # 2. Intenta completar las franjas de ocupación 15
      self.fill_strips()

      # 3. Busca franjas vacías en las placas y las rellena con franjas
      desechadas
```

```
    self.fill_empty_slots()

    # 4. Intenta completar las franjas de ocupación 15
    self.fill_strips()

    # 5. Coloca las franjas desechadas restantes en nuevas placas
    self._discarded_strips = sorted(self._discarded_strips, key=lambda x: x.
    get_temp(), reverse=True)
    self.populate_partial_plates()

    # 6. Intenta completar las franjas de ocupación 15
    self.fill_strips(limit=16)

    # 7. Coloca indicadores que puedan faltar
    self.fill_indicators()

    # 8. Filtra posibles placas vacías
    for p in self._full_plates:
      if p.get_occupancy_count() == 0:
        self._full_plates.remove(p)

    for p in self._partial_plates:
      if p.get_occupancy_count() == 0:
        self._partial_plates.remove(p)

    out_solution = solution.PCRSolution([], self.problem.max_temp_diff)
    out_solution.set_plates(self._full_plates + self._partial_plates)
    out_solution.calculate_objectives()

    stats['full_plates'] = [(p.plate_id, st.strip_id, st.get_temp(), st.
    count_empty_wells()) for p in self._full_plates for st in p.strips]
    stats['partial_plates'] = [(p.plate_id, st.strip_id, st.get_temp(), st.
    count_empty_wells()) for p in self._partial_plates for st in p.strips]
    stats['total_time'] = round(time.time() - start_time, 4)
    return {'initial_solution': init_sol, 'solution': out_solution, 'stats':
    stats}
```

Bibliografía

[1] L. Carpente, A. Cerdeira-Pena, S. Lorenzo-Freire, A. S. Places. Optimization in Sanger sequencing. Computers & Operations Research (Under Review).

[2] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi (1983). Optimization by simulated annealing. Science 220: 671-680.

[3] F. Sanger, S. Nicklen, R. Coulson (1977). DNA sequencing with chain-terminating inhibitors. Proceedings of the 410 National Academy of Sciences 74: 5463-5467.