Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería en Organización Industrial

Procesos de Optimización de Fabricación Aditiva

Autor:

Senovilla Minguela, Juan José

Tutor:

López Paredes, Adolfo
Departamento de Organización de Empresas y CIM.

Gordaliza Pastor, Alfonso
Departamento de Estadística e Investigación operativa

Valladolid, junio de 2019.

Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

Resumen

El objeto de este trabajo es presentar un método para incrementar la productividad de las empresas productoras mediante las tecnologías de impresión 3d. El método consta de dos etapas. En la primera se propone un algoritmo que permite optimizar la distribución de las piezas en la superficie de fabricación. Se ha programado con Python. En la segunda se elige la mejor solución desde el punto de vista del retorno y de la productividad en cada orden de fabricación. Esto permitirá a las empresas organizar de forma más eficiente la producción. Este trabajo parte de los desarrollos realizados por la Unidad de Investigación Consolidada INSISOC (UIC086) en el proyecto LONJA3D, y constituye un punto de partida para que los fabricantes puedan mejorar sus ofertas al pujar por los diferentes lotes.

Palabras clave: Impresión 3D, optimización, planificación, producción, Python

Abstract

The purpose of this paper is to present a method to increase the productivity of manufacturing companies through 3D printing technologies. The method consists of two stages. In the first one, an algorithm that allows optimizing the distribution of the pieces on the manufacturing surface is proposed. It has been programmed with Python. In the second one, the best solution is chosen from the point of view of return and productivity in each production order. This will allow companies to organize production more efficiently. This work is based on the developments carried out by the Consolidated Research Unit INSISOC (UIC086) in the LONJA3D project and constitutes a starting point for manufacturers to improve their offers when they are bidding for different batches.

Keywords: 3D printing, optimization, planning, production, Python

Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

Índice general

Índice de ilustraciones

Índice de tablas

1. Introducción

1.1. Alcance y objetivos del proyecto

El objetivo principal de este trabajo es plantear una heurística que ayude en la programación de la producción para fabricantes de piezas producidas mediante fabricación aditiva. Esta heurística buscará optimizar el espacio en las camas de fabricación de las máquinas de impresión 3D, permitiendo al fabricante decidir que lote de piezas producir para obtener el mayor beneficio posible.

Nos centraremos en el caso específico de un fabricante que forma parte de la futura red de proveedores que alojará el proyecto "Lonja3D". Lonja3D es un proyecto desarrollado por el GIR INSISOC que cuenta con financiación regional y desarrolla su actividad en investigación en Industria 4.0. Con este proyecto se pretende solventar el déficit que existe en el desarrollo del mercado de productos fabricados mediante impresión 3D.

Los objetivos fundamentales que se plantean alcanzar mediante este trabajo son los siguientes:

- Profundizar en el conocimiento de la impresión 3D como tecnología de fabricación de productos funcionales.
- Plantear las ventajas y desventajas que presenta la impresión 3D frente a los métodos tradicionales.
- Profundizar en los conocimientos acerca de la planificación y programación de la producción en plantas de fabricación aditiva.
- Redefinir el problema de programación de la producción en plantas de fabricación aditiva.
- Presentar una metodología para resolver el problema mediante el uso de una heurística implementada en lenguaje Python.
- Analizar posibles casos reales de planificación de la producción.
- Validar la eficacia y eficiencia de la heurística propuesta.

Este proyecto recogerá un estudio de la implantación de la impresión 3D en el sector industrial con el objeto de conocer su estado actual y comparar sus características con los sistemas de fabricación tradicionales. Asimismo, se estudiarán los planteamientos que existen hasta día de hoy en planificación de la producción en plantas con fabricación aditiva, para tratar de entender las metodologías de optimización de la producción. De esta forma podremos implementar estas técnicas en nuestra herramienta con la intención de reducir los costes de fabricación, optimizando la capacidad productiva y generando un mayor beneficio. A lo largo del cuerpo del trabajo se desarrollará y explicará detalladamente la heurística que pretende resolver el problema de planificación de la producción. En último lugar se realizarán pruebas para validar la heurística propuesta.

1.2. Justificación

La Cuarta Revolución Industrial, también conocida como Industria 4.0. ya es una realidad, generando en la sociedad actual un cambio de paradigma social y económico. Dentro de

esta revolución podemos destacar tecnologías como inteligencia artificial, Internet de las Cosas (IoT), computación en la nube (Cloud Computing), sistemas ciberfísicos, impresión 3D o vehículos autónomos, entre otras. Todas ellas ofrecerán grandes ventajas a las empresas que sean capaces de integrar estas tecnologías en su estrategia frente a las organizaciones que no se adapten a tiempo al cambio. No solo se tienen que adaptar a los modelos de negocio que existen en la actualidad, sino que también deben ser capaces de detectar las nuevas oportunidades que brindan estas tecnologías y buscar la forma de generar beneficios.

El mercado tanto de bienes como de servicios está cambiando, la industria 4.0. permite realizar una hibridación de ambos, dando lugar a empresas que cuentan con plataformas en la nube desde las cuales los usuarios realizan pedidos personalizados desde sus casas. En la actualidad, cualquier cliente evitará comprar a aquellas empresas que no sean capaces de ofrecer un tiempo de entrega corto del pedido.

La fabricación aditiva, presenta las características idóneas para producir un producto personalizado para el cliente y poder suministrarlo en un período corto de entrega. De esta forma, aquellas organizaciones que apuesten por implementar las tecnologías de impresión 3D para la fabricación de productos personalizados, serán capaces de recortar en los tiempos de entrega y aumentar su competitividad en el mercado.

Este aumento en la competitividad de la empresa viene dado por las ventajas que ofrece la fabricación aditiva. Entre todas ellas, podemos destacar la eficiencia en el uso de material, permitiendo reducir los costes variables que se generan al fabricar una pieza. Otra ventaja destacable es la posibilidad de generar geometrías complejas sin un coste adicional frente a una geometría más simple, siendo esto algo impensable con las tecnologías de fabricación tradicionales. Además, conseguimos reducir tiempos de espera y el stock de los productos, simplificando la cadena de suministro y consiguiendo reducir aún más los costes en los que incurre la empresa.

El objetivo que da pie a nuestro trabajo no es otro que optimizar la capacidad productiva de un fabricante de piezas que utilice tecnologías de fabricación aditiva. Más concretamente trataremos de ayudarle a decidir que lote de piezas ofrece mayor retorno cuando se plantee la planificación de la producción diaria.

1.3. Motivación

La impresión 3D es una tecnología en pleno auge en el sector de la fabricación de piezas funcionales. La evolución de esta tecnología en los últimos años es vertiginosa e implica un cambio inmenso para los sistemas de fabricación. Es importante conocer las nuevas técnicas y como su penetración en la industria puede desplazar a los sistemas tradicionales de fabricación en aquellos ámbitos con gran personalización en el producto.

Es el deber de un ingeniero en organización industrial conocer las posibilidades que brinda la impresión 3D y los cambios que implica en el proceso de fabricación y la logística de las empresas. Además de conocer los nuevos negocios que aparecen basados en esta tecnología puntera.

Actualmente, el mercado de productos fabricados a partir de tecnologías de fabricación aditiva no está suficientemente desarrollado ni cuenta con suficientes proveedores de bienes ni servicios. Cuando una empresa o un particular desea adquirir un producto fabricado mediante impresión 3D debe buscar concienzudamente potenciales proveedores, compartir con ellos su diseño para obtener ofertas, con el riesgo de confidencialidad que esto supone, y comparar con el resto de las ofertas. Otra opción es acudir a *marketplaces*, un Marketplace es una plataforma que hace de unión entre cliente y proveedor, ejemplos representativos de este modelo de negocio son eBay o Amazon. Por último, pueden decantarse por comprar la pieza a un proveedor local, pero es probable que no cuenten con la tecnología necesaria si la pieza es muy compleja y los precios serán poco competitivos.

Como solución a este problema aparece el proyecto Lonja3d, un proyecto financiado por el Fondo Europeo de Desarrollo Regional y gestionado por el GIR INSISOC. En este proyecto se plantea la posibilidad de desarrollar un mercado gestionado, "una lonja", que utilice subastas combinatorias para realizar ofertas de forma colaborativa del lado de la demanda. Un mismo cliente puede realizar pedidos de piezas que se fabriquen con diferentes tecnologías, materiales y requerimientos de calidad. Pujando por una combinación de productos puede obtener un mejor precio que si los pide a diferentes proveedores de forma individual. Por otro lado, la principal ventaja que ofrece la fabricación aditiva a los proveedores es la posibilidad de fabricar de forma simultánea productos con diferentes funcionalidades y de diferentes clientes.

Dentro de este proyecto, los proveedores deberán seleccionar las piezas de los diferentes clientes que desean fabricar y ofrecer un precio. Para ello, buscarán aquellos lotes que les permitan obtener un mayor retorno en forma de beneficio económico. Este proyecto permitirá ampliar el conocimiento en optimización de la fabricación de procesos de fabricación aditiva, ayudando a los proveedores a planificar su producción de forma eficiente.

1.4. Estructura del trabajo

La estructura del trabajo se ha hecho en función de los temas tratados en los diferentes apartados. La memoria se organiza en tres capítulos, quedando excluidos los apartados de introducción y conclusiones.

En el capítulo 2 se hace una breve introducción a la fabricación aditiva y una clasificación de las técnicas existentes en la actualidad. A continuación, se exponen las ventajas y desventajas que presenta la fabricación aditiva frente a tecnologías más tradicionales como *mass cusomization*, moldeo por inyección y técnicas sustractivas. Además, se realiza un breve repaso del estado del arte de la planificación y programación de la producción en plantas de fabricación aditiva y plataformas *Cloud Manufacturing* basadas en tecnologías de impresión 3d. Por último, se presenta el proyecto Lonja3d y se plantea el problema de planificación de producción diaria al que se enfrentarán los fabricantes del proyecto.

En el capítulo 3 se presenta el método utilizado para resolver el problema. El capítulo se compone de cuatro apartados. En los dos primeros apartados se expone la heurística planteada para resolver el problema de asignación de piezas al lote de fabricación. En los

dos apartados siguientes se desarrolla el código implementado en lenguaje Python y se muestra la ejecución del programa.

El capítulo 4 incluye las pruebas realizadas con el fin de analizar el comportamiento del programa presentado frente a posibles casos reales. Se compara la eficacia y eficiencia de la técnica heurística utilizada frente a casos resueltos en la literatura actual que tratan de resolver problemas similares al expuesto en este trabajo. Además, se presentan situaciones donde pueden aparecer dudas sobre la elección del lote a fabricar y se justifica la decisión más acertada.

Los capítulos restantes son la introducción y las conclusiones. En el primero se expondrán los objetivos y alcance del proyecto, la motivación y su justificación. En el capítulo de las conclusiones se revisará la consecución de los objetivos marcados al comienzo de la memoria y se analizarán las lecciones aprendidas. Por último, en este mismo apartado se plantearán futuros trabajos que puedan surgir a partir de este proyecto.

2. Fabricación aditiva

2.1. Introducción a la impresión 3D.

La impresión 3D es una técnica de fabricación aditiva que permite producir un amplio abanico de piezas con estructuras y geometrías complejas a partir de su modelo digital en 3D. El proceso consiste en la adición sucesiva de capas del material una encima de otra hasta completar la pieza deseada. A lo largo de la memoria nos referiremos a fabricación aditiva e impresión 3D como la misma tecnología.

En 1986 Charles Hull desarrolló esta tecnología en el proceso que hoy en día conocemos como estereolitografía (SLA). Se basaba en la fabricación de piezas a partir de un líquido fotopolimérico sobre el que se focaliza un haz de luz ultravioleta. Esta técnica fue solo el comienzo de una nueva forma de producción. Con el tiempo se desarrollaron nuevas tecnologías de fabricación aditiva como la fusión por lecho de polvo, el modelado por deposición fundida y el modelado por haz de electrones entre otras.

En el trabajo de fin de grado de De Antón Heredero, (2018) se explican cada una de las técnicas existentes de fabricación aditiva y se clasifican en función de los siete procesos que considera la Asociación Americana de Ensayo de Materiales (ASTM) tal y como podemos observar en la Tabla 1.

Process categories	Technology	Materials
Vat photopolymerization	SLA	Plastic & Ceramic
	DLP	Plastic
	CDLP	Plastic
Material Extrusion	FDM	Plastic & Composite
Material Jetting	MJ	Plastic & Ceramic
	DOD	Wax
Binder Jetting	BJ	Gypsum, Sand & Metal
Powder bed fusion	SLS	Plastic & Ceramic
	DMLS / SLM	Metal
	EBM	Metal
	MJF	Plastic
Direct Energy Deposition	LENS	Metal
	EBAM	Metal
Sheet Lamination	LOM	Composite & Paper
	UC	Metal

Tabla 1: Clasificación de las tecnologías de fabricación aditiva. Fuente: (De Antón Heredero, 2018)

A continuación, se realiza un pequeño resumen de cada uno de los siete procesos y sus diferencias.

2.1.1. Vat Photopolymerization

Conocido como fotopolimerización, es el proceso de fabricación aditiva en el que se aplica radiación mediante rayos gamma, rayos x, haz de electrones o rayos UV sobre un líquido fotopolimérico. Este líquido se cura capa a capa, haciendo que las moléculas se unan para dar lugar a moléculas de mayor tamaño hasta conformar la pieza final. Dentro de este

proceso de fabricación aditiva nos encontramos con la estereolitografía (SLA), el procesado digital de luz (DLP) y el procesado digital de luz continuo (CDLP).

2.1.2. Material Extrusion

El proceso de extrusión de material es el más popular de los siete procesos de fabricación aditiva. Su funcionamiento es sencillo, realizando una extrusión del material de forma continua hasta formar finas capas que se superponen para dar lugar a la pieza final. La materia prima suele tener forma de filamento y frecuentemente se hace uso de termoplásticos debido a su facilidad de extrusión a altas temperaturas. La técnica más conocida es el modelado por deposición fundida (FDM), también conocido como fabricación mediante Filamento Fundido (FFF).

2.1.3. Material Jetting

Material Jetting o inyección de material es la técnica que permite fabricar piezas con diferentes materiales y altos niveles de detalle y precisión. Se basa en la deposición por lanzamiento controlado de material sobre la cama de fabricación. Simultáneamente se aplica una radiación sobre el material, curándolo o endureciéndolo. Entre las tecnologías más destacadas tenemos la inyección de material (MJ) y el *Drop-On-Demand* (DOD).

2.1.4. Binder Jetting

La metodología de inyección de aglutinante se basa en la impresión por chorro, como ocurría con la técnica de inyección de material, pero el material se inyecta sobre un lecho de polvo. Es una técnica que combina la inyección de material (MJ) con la fusión en lecho de polvo (PBF). El material inyectado suele ser una resina plástica, un polímero o ceras entre otros y recibe el nombre de aglutinante. El material en polvo que se emplea habitualmente es de tipo cerámico o algunos metales como aluminio o acero inoxidable y recibe el nombre de sustrato. Al mezclarse aglutinante y sustrato se solidifican dando lugar a las diferentes capas que conforman la pieza.

2.1.5. Powder Bed Fusion

Los procesos basados en la fusión en lecho de polvo (PBF) producen piezas a partir de la sinterización de partículas de polvo dentro de una cubeta. Para ello, se extiende una capa del material en polvo sobre la cubeta, se aplica un haz de energía sobre la superficie con la forma de la pieza deseada y se extiende una nueva capa, repitiendo este proceso hasta completar la pieza final. Entre todas las tecnologías basadas en lecho de polvo, podemos destacar el sinterizado selectivo por láser (SLS), la fusión selectiva por láser (SLM) y la fusión por haz de electrones (EBM).

2.1.6. Direct Energy Deposition

La técnica de deposición directa de energía (DED) cuenta con diferentes acepciones, como *Laser Solid Forming* (LSF), *Direct Metal Deposition* (DMD) o *Directed Light Fabrication* (DLF). Para fabricar una pieza, se aplica una fuente de energía mediante un haz de electrones o bien con el uso de un láser a la vez que se deposita el material (en polvo o como alambre) sobre la cama de fabricación. Esta tecnología permite realizar reparaciones de piezas de gran tamaño y fabricar piezas en materiales metálicos, con aleaciones o composites de matriz metálica.

2.1.7. Sheet Lamination

La laminación de hojas es una técnica de fabricación aditiva que utiliza como material de fabricación finas láminas sobre las que se realizan cortes mediante un láser o cuchillas para darle la forma deseada. Las láminas, que pueden ser de papel, plástico o metal, se suministran a través de un rollo que las deposita capa a capa.

Las tecnologías más representativas son la fabricación mediante laminado de objetos (LOM) y la consolidación por ultrasonidos (UC).

2.2. Evolución de la fabricación aditiva: del prototipo a la producción de piezas funcionales

La impresión 3D ha evolucionado mucho en los últimos años tanto en técnicas y materiales como en tecnología, dando lugar a una revolución en los procesos de producción y logística de la industria. Las principales industrias que se han visto afectadas por estos cambios han sido la construcción, las fábricas de prototipos, el sector de la biomedicina, la industria aeronáutica y aeroespacial e incluso el sector del automóvil. En particular, en el sector de la construcción se implementó muy lentamente, a pesar ofrecer muchas ventajas como, por ejemplo, reducción de residuos, flexibilidad en los diseños y la automatización.

Continuamente se desarrollan nuevos materiales y técnicas de fabricación aditiva gracias a la expiración de las primeras patentes que se aprobaron. Esto ha permitido a los fabricantes implementar nuevos servicios relacionados con la impresión 3D. Además, los últimos desarrollos en tecnología han permitido reducir los costes de las impresoras de 3D, permitiendo a usuarios adquirirlas en escuelas, en pequeños laboratorios e incluso en sus propias casas.

Inicialmente la impresión 3D estaba orientada a la fabricación de prototipos. Los usuarios de esta tecnología eran básicamente arquitectos y diseñadores que necesitaban producir prototipos con geometrías complejas y un coste reducido. Gracias a las mejoras alcanzadas en el desarrollo de la tecnología en los últimos años se ha conseguido, no solo fabricar prototipos, sino comenzar con la fabricación de piezas funcionales. En el artículo Berman, (2012) se hace referencia a Terry Wholers, gerente de una firma de investigación de mercado especializada en impresión 3D que en 2010 aseguraba que más del 20% de la producción de impresoras 3D estaba realizando productos finales en lugar de prototipos. Wohlers predecía que esto aumentaría al 50% para 2020.

La fabricación de productos personalizados para cada cliente ha sido uno de los grandes desafíos con el que se encontraba toda empresa. Todas ellas incurrían en grandes costes para poder producir unidades a medida para cada cliente. La fabricación aditiva permite solventar este problema, permitiendo a las empresas fabricar un número reducido de piezas personalizadas con costes relativamente bajos.

Si realizamos una comparación de las tecnologías convencionales de fabricación con la impresión 3D, nos encontramos con diferentes ventajas y desventajas para cada una de estas tecnologías. Un análisis detallado se puede encontrar en Berman, (2012), donde se

comparan tecnologías como "*mass customization"*, moldeo por inyección y maquinaria de corte con la impresión 3D.

2.3. Impresión 3D y "mass customization"

En los últimos años, la impresión 3D se ha comparado con las técnicas de fabricación empleadas en "*mass customization*". En ambos casos, se trata de tecnologías de fabricación que permiten producir pequeños lotes de piezas personalizadas con costes relativamente económicos. En cambio, son tecnologías muy diferentes en cuanto a requerimientos de maquinaria y logística.

A diferencia de la impresión 3D, las tecnologías de *mass customization* se basan en la utilización de módulos de piezas montados previamente o estrategias de diferenciación diferida. Un ejemplo claro, es la marca Dell, que permite combinar microprocesadores, gráficas, discos duros y otros componentes del ordenador, para dar lugar a un ordenador personalizado para cada cliente. En cambio, en impresión 3D se parte de materias primas como plásticos, resinas, aleaciones de cromo-níquel y acero inoxidable entre otras.

Un primer punto a favor de la impresión 3D es la necesidad de una cadena de suministro de un número reducido de proveedores. En cambio, en la producción mediante *"mass customization"* se necesita contar con diferentes proveedores que te entreguen los módulos correctos, en el momento preciso, para asegurar que se cuenta con las cantidades exactas en la fecha requerida. Esto obliga a contar con un alto grado de integración en la cadena de suministro y complica la producción de piezas.

Además, mientras que en los sistemas de producción "*mass customization"* se basa en equipos de trabajo, los procesos de fabricación aditiva son automáticos y están basados en el software CAD. Esto significa que, en impresión 3D no es necesario que el empleado esté pendiente del proceso de fabricación, ni si tan siquiera requiere de su intervención hasta que la pieza esté completamente producida. Simplemente se necesita cargar el archivo en el software y esperar a que la impresora haga su trabajo.

A pesar de contar con varias diferencias, también cuenta con alguna similitud, como es el caso de las características económicas. Ambos permiten reducir los riesgos relacionados con stocks, ya que los productos solo se fabrican una vez que la orden del cliente ha sido lanzada y se ha realizado el pago.

Los productos que se pueden fabricar mediante "*masss customization"* se diferencian de los fabricados en impresión 3D en el uso de materiales. Aquellos fabricados por impresión 3D suelen estar compuestos de un solo material, mientras que los fabricados en "*mass customization*", como ordenadores, relojes, zapatos, entre otros, están compuestos por diferentes materiales.

2.4. Impresión 3D y métodos de fabricación mediante moldeo por inyección y métodos substractivos

Si comparamos la impresión 3D con aquellas tecnologías de moldeo por inyección y tecnologías substractivas, encontramos un gran número de ventajas a favor de la fabricación aditiva. Entre las más destacadas podemos destacar la eficiencia en costes y la rapidez.

A diferencia de la fabricación mediante moldes de inyección, que requieren grandes inversiones en moldes, los costes fijos de la impresión 3D son relativamente bajos. La impresión 3D no requiere de herramientas caras, de moldes ni de taladros, siendo muy rentable en la fabricación de pequeños lotes de producción. Esto hace que sea la tecnología perfecta para fabricar pedidos personalizados de clientes y dar servicio a nichos de mercado.

Si comparamos con las tecnologías substractivas, la principal ventaja es el ahorro de material, reduciendo de esta forma la generación de residuos. Las tecnologías substractivas hacen uso de maquinaria como tornos y fresadoras para obtener la forma deseada en la pieza. Para ello, se parte de un bloque metálico o de otro material y se reduce de forma gradual, eliminando el material hasta conseguir la forma final. El ahorro es tan alto, que en algunos casos la fabricación aditiva puede llegar a reducir un 40% el uso de material en comparación con las tecnologías substractivas (Berman, 2012).

Otro punto a favor es el tiempo empleado en fabricar una nueva pieza, siendo inferior en la impresión 3D frente a las otras tecnologías. Esto se consigue gracias al ahorro de tiempo en el set-up de la máquina.

En cuanto al diseño y fabricación de prototipos, el tiempo se reduce considerablemente. El fabricante de zapatos Timberland solía gasta alrededor de 1200 $ (1000 €) y una semana en el diseño de una nueva suela para sus modelos. Mediante la fabricación aditiva, consiguió reducir el tiempo a 90 minutos y el coste a 35 $ (31 €) (Berman, 2012).

Además, con impresión 3D, contamos con la posibilidad de compartir diseños de piezas con otros fabricantes y externalizar la producción, sin la necesidad de que el otro fabricante cuente con un molde determinado o utillajes determinados. Simplemente necesitaran el material y la impresora 3D. No solo podemos compartir el diseño con el fabricante, sino que también hay una gran comunidad de entusiastas dispuestos a ayudar en el diseño de piezas. Entre las páginas más destacadas, se encuentran: https://cults3d.com/, https://www.thingiverse.com/, https://www.youmagine.com/, https://grabcad.com/

La posibilidad de diseñar la pieza de forma interna permite a los ingenieros mecánicos diferenciar el relleno de aquellas zonas que vayan a estar expuestas a mayores esfuerzos mecánicos de otras en las cuales la resistencia estructural pueda ser inferior. Esto no se consigue con tecnologías substractivas ni con moldes. El diseño del relleno también permite controlar otras características de la pieza como es el caso de la distribución de temperaturas. El diseño de conductos internos en las piezas, permiten al aire realizar la función de aislante y redirigir el calor hacia aquellas zonas que no se vean afectadas por las altas temperaturas.

También nos encontramos con puntos en contra de la fabricación aditiva. Por ejemplo, en moldeo por inyección, los costes variables por pieza se reducen con grandes lotes de fabricación, mientras que en impresión 3D esto no ocurre. En cambio, cuando se están realizando prototipos, el coste de fabricar dos variantes de un mismo producto es el mismo con impresión 3D mientras que con otras tecnologías, la inversión es mayor.

Otra desventaja es la falta de precisión, en comparación con las tecnologías substractivas que cuentan con un gran avance tecnológico debido a su gran desarrollo a lo largo de toda la historia. Además, la cantidad de materiales disponibles también es inferior y es muy complicado realizar una misma pieza a partir de materiales diferentes.

Aunque aún no está claro el límite de unidades que sale rentable fabricar con fabricación aditiva frente al moldeo por inyección, algunos autores ya han hecho números. En el artículo de Berman, (2012) se encuentran dos vertientes. La primera sitúa el rango entre 50 y 5000 unidades para piezas fabricadas en plástico. En contrapartida, las fuentes más conservadoras reducen hasta 1000 el número de unidades fabricadas en plástico.

2.5. Definición del problema

La fabricación aditiva está considerada como el centro de la nueva generación de los sistemas de producción. Las características de esta tecnología tienen un impacto no solo en la forma de fabricar las nuevas piezas, sino que afecta a la planificación y programación de la producción, la cadena de suministro y logística. Por lo tanto, es necesario estudiar a fondo la implementación de la maquinaria de fabricación aditiva en las fábricas de la industria 4.0.

Para poder implementar la fabricación aditiva en los procesos de fabricación es necesario un modelo de estimación del tiempo y coste de producción de las diferentes piezas. Wegener, Spierings and Rickenbacher, (2013) investigaron los diferentes modelos de costes existentes y demostraron que mediante una fabricación simultánea de piezas se consigue optimizar la utilización de los recursos reduciendo tanto tiempo como costes. Por otro lado, Zhang and Bernard, (2013) revisaron toda la literatura existente y encontraron que la mayoría de los modelos de estimación de tiempos eran poco preciso y muy complejos para ser usados en la práctica. Piili *et al.*, (2015) encuentra que fabricar piezas de forma simultánea permite reducir los costes hasta entre 81-92% comparado con la fabricación de las piezas de forma separada. En este estudio se plantean dos situaciones extremas, comparando el caso en el cual se fabrica una sola pieza, frente a realizar de forma simultánea tantas piezas como fuera posible. La óptima utilización de la plataforma de fabricación es la principal variable que afecta al proveedor en términos de producción de múltiples piezas. Por ello, es muy importante realizar una buena planificación de las piezas en lotes de fabricación para reducir costes y obtener un mayor beneficio económico.

En la actualidad, hay muy pocos estudios que abarquen la planificación y programación de la producción mediante impresión 3d. Por ejemplo, Chergui, Hadj-Hamou and Vignat, (2018) introducen por primera vez la integración de las características de los procesos de fabricación aditiva como estimación del tiempo de fabricación y localización de las piezas en la cama de fabricación mediante programación de algoritmos. En este estudio se propone una heurística con la finalidad de optimizar la producción de un grupo de piezas que diferentes clientes desean fabricar. Todas las piezas se agrupan y se organizan en diferentes lotes de fabricación que serán asignados a las diferentes impresoras 3D con las que cuenta el proveedor. Se planifica la producción de forma que se realicen varias piezas a la vez en una misma máquina de fabricación aditiva. Para resolver el problema de planificación, se divide en dos etapas, tal y como se puede observar en la Ilustración 1. En la primera, se agrupan las piezas (P) de todos los clientes (C) y se asignan a lotes de producción (J). Estas asignaciones se programan en función de las fechas de entrega, alturas, área de producción y volumen. En la segunda etapa, se trata de optimizar el orden de producción de cada lote en el grupo de máquinas (AM) idénticas que se encuentran en paralelo.

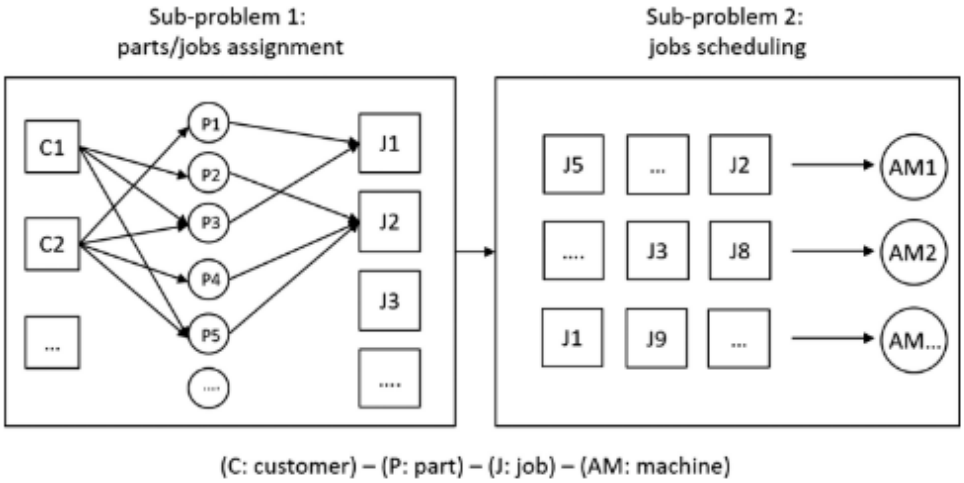


Ilustración 1: Etapas del problema de planificación de la producción. Fuente: (Chergui, Hadj-Hamou and Vignat, 2018)

El objetivo de esta planificación es reducir el retraso total de los pedidos de los clientes. El tiempo de procesado de un lote de piezas está muy influenciado por la altura máxima de las piezas asignadas al lote. Esto se debe a que, en la fabricación de un lote de piezas, no es posible retirar ninguna de ellas hasta que todas están completadas.

Para resolver el problema se implementó en Python una heurística basada en la regla EDD (*earliest due date*), es decir, se programan primero la fabricación de aquellas piezas con fecha de entrega más temprana. Como consecuencia podemos observar que en la localización de las piezas sobre la cama de fabricación de este estudio (Ilustración 2) se deja de lado la optimización del espacio. Esto influirá en los costes de fabricación, impidiendo al fabricante reducir sus costes variables y obtener de esta forma un mayor beneficio. Este estudio deja clara la necesidad de continuar desarrollando métodos de planificación y programación de la producción en fabricación aditiva.

En 2017, Li, Kucukkoc and Zhang, (2017) presentan un artículo en el cual se aborda la planificación de la fabricación en un escenario con pedidos de varios clientes y máquinas con características diferentes (coste unitario, coste de set-up, velocidad de procesado, entre otras). En este caso, el objetivo de la programación de pedidos es minimizar el coste por volumen de material, mientras se cumplen ciertas restricciones. Se presenta un modelo matemático, se programa en CPLEX y se realizan pruebas. A mayores, se muestran dos heurísticas *Best-Fit* (BF) y *Adapted Best-Fit* (ABF) y se explican paso a paso. La única diferencia entre estas dos heurísticas es la regla de decisión que se aplica para seleccionar las piezas de la lista de piezas disponibles.

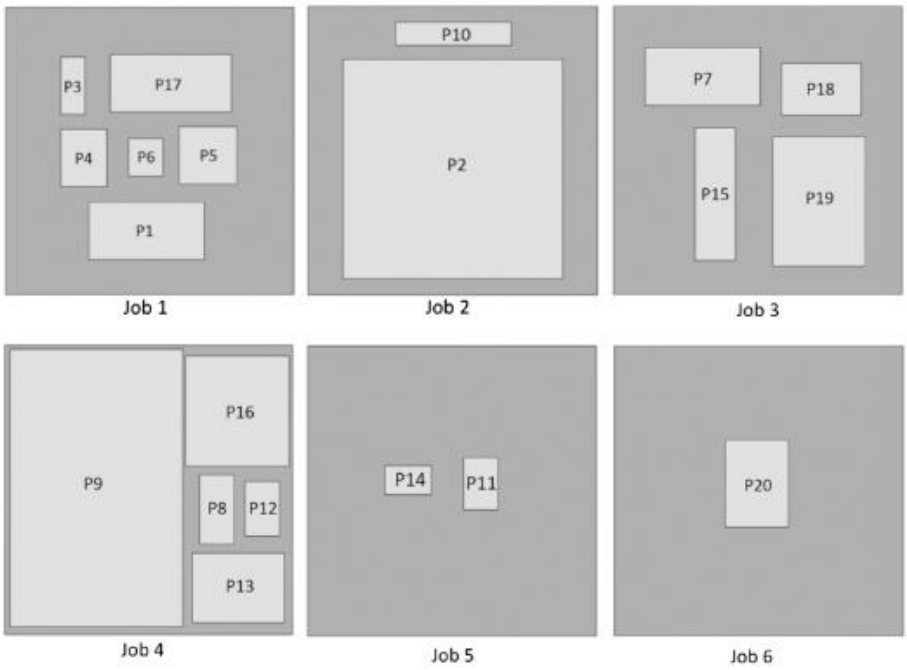


Ilustración 2: Distribución de las piezas en la cama de fabricación. Fuente: (Chergui, Hadj-Hamou and Vignat, 2018)

Como pruebas para conocer la efectividad y eficiencia de las técnicas, se calculó la solución óptima y se compararon los tiempos de computación obtenidos por cada uno de los tres procedimientos, tanto CPLEX como las dos heurísticas BF y ABF. Las conclusiones de este artículo fueron las siguientes:

- Las dos heurísticas propuestas ofrecían resultados factibles en tiempos de computación razonables.
- La necesidad de continuar investigando en técnicas de planificación y programación de recursos en fabricación aditiva.
- La posibilidad de integrar un modelo de distribución de piezas sobre la cama de fabricación para mejorar las heurísticas propuestas.

En este trabajo, trataremos de proponer un modelo que sirva para complementar a estas heurísticas propuestas y dar un mejor resultado en la optimización de la cama de fabricación.

Como ya hemos podido comprobar, hay una clara necesidad en la implantación de métodos que resuelvan la distribución de las piezas en la cama de fabricación. En un artículo escrito por Zhang, Gupta and Bernard, (2016) se realiza un estudio del estado del arte de la optimización de la colocación de varias piezas en la cama de fabricación. La orientación de las piezas es una de las decisiones más importantes que hay que tomar a la hora de fabricar piezas mediante impresión 3D. El coste, la calidad y el tiempo se verán muy afectados por la orientación de la pieza. Hay muchos estudios que resuelven este problema en escenarios con una sola pieza, pero el número de estudios realizados con lotes de piezas es muy reducido.

En el artículo escrito por Zhang *et al.*, (2016) se revisa toda la literatura existente acerca de la optimización de la orientación de una sola pieza. Además, proponen un nuevo método para conseguir la orientación que reduzca el tiempo y el coste de fabricación de la pieza, cumpliendo con los requisitos de calidad. Por último, presentan una herramienta online que forma parte del proyecto europeo "KARMA". Esta herramienta se encuentra en la página web http://karma.aimme.es y permite al usuario obtener una orientación de la pieza que cumpla con sus requerimientos de precisión, tiempo y coste. Por lo tanto, podemos considerar que el problema de orientación de una sola pieza sobre la plataforma de fabricación está resuelto y que contamos con amplia literatura al respecto. Esto nos permitirá simplificar nuestro problema, suponiendo que las piezas traen una orientación impuesta, de forma que solo las rotaremos respecto del eje Z y la trasladaremos en el plano XY.

En cambio, hay muy poca literatura enfocada en resolver el problema de optimización de la orientación y localización sobre la plataforma de fabricación para un lote de piezas. En el artículo escrito por Zhang, Gupta and Bernard, (2016) se clasifican los estudios que existían hasta 2016 y se comentan los más relevantes. Esta clasificación se realiza en función de si se resuelve en 2D o 3D, la representación de las piezas, del algoritmo utilizado, el método empleado para colocar las piezas, los grados de libertad de rotación, la orientación de las piezas y una última variable que llaman "*Production context*". Esta última diferencia aquellos problemas en los cuales todas las piezas caben en la cama de fabricación (*Full placement*) de otros en los cuales no se pueden fabricar todas (*Subset placement*). Podemos observar esta clasificación en la Tabla 2.

Contributor	Placement mode	Part representation	Placement method	Optimization algorithm	Part freedom	Build orientation	Production context
Wodziak et al. [19]	2D nesting 3D packing	Bounding box	Serial	GA	Rotate on XOY by 90°	Considered Fixed	Subset placement
Nyaluke et al. [15]	3D packing	Bounding box	Serial	GA	No rotation	Considered Fixed	Subset placement
Ikonen et al. [8]	3D packing	Original STL model	Parallel	GA	Rotate around three axis by 45°	Not considered	Subset placement
Dickinson [4]	3D packing	Depth map	Parallel	SA	Rotate freely	Not considered	Subset placement
Hur et al. [7]	3D packing	Voxel model	Serial	GA	Rotate on XOY by 90°	Considered Fixed	Full placement
Zhang et al. [22]	3D packing	Bounding box	Parallel	SA	6 pre-set orientations	Not considered	Full placement
Canellidis et al. [2]	2D nesting	Bounding box & Rectangular	Serial	GA	No rotation	Considered Fixed	Subset placement
Gogate and Pande [6]	3D packing	Voxel model	Serial	GA	Rotate on XOY by 90°	Considered Not fixed	Full placement
Canellidis et al. [3]	2D nesting	Polygon (out profile of part)	Serial	GA	No rotation	Considered Fixed	Subset placement
Wu et al. [21]	3D packing	Voxel model	Serial	GA	No rotation	Considered Fixed	Full placement

(Note: *serial placement means parts are packed one by one according to pre-set rules; parallel placement means parts are positioned simultaneously; subset placement means only subset of a part group can be filled into a build volume; full placement means all the parts surely fit the build volume.*)

Tabla 2: Estado del arte de los estudios de optimización del espacio de fabricación en máquinas de fabricación aditiva. Fuente: (Zhang, Gupta and Bernard, 2016)

Algunos de estos métodos son específicos del tipo de técnica de impresión utilizado o de un escenario de producción particular. Por ejemplo, si realizamos la distribución de las piezas en una máquina de sinterizado selectivo por láser, contamos con la posibilidad de superponer unas piezas sobre otras. Las piezas están inmersas en el contenedor de polvo y, por lo tanto, no necesitan soportes. Por lo tanto, la distribución de las piezas debe realizarse en 3D. En nuestro caso, nos centraremos en las técnicas de fabricación aditiva que resuelvan el problema en 2D.

El primer estudio que analizaremos (Canellidis *et al.*, 2006) resuelve el problema en dos etapas. En la primera etapa se orientan las piezas teniendo en cuenta criterios como el cumplimiento de los requerimientos de calidad, el tamaño de la estructura de soporte, el tiemplo empleado en la fabricación y el tamaño del área proyectado. Una vez orientadas, se utiliza un algoritmo genético para optimizar el área de la cama de fabricación disponible. Las piezas se simplifican utilizando su proyección rectangular, tal y como se puede observar en la ilustración 3. Esta simplificación permite reducir la complejidad del problema.

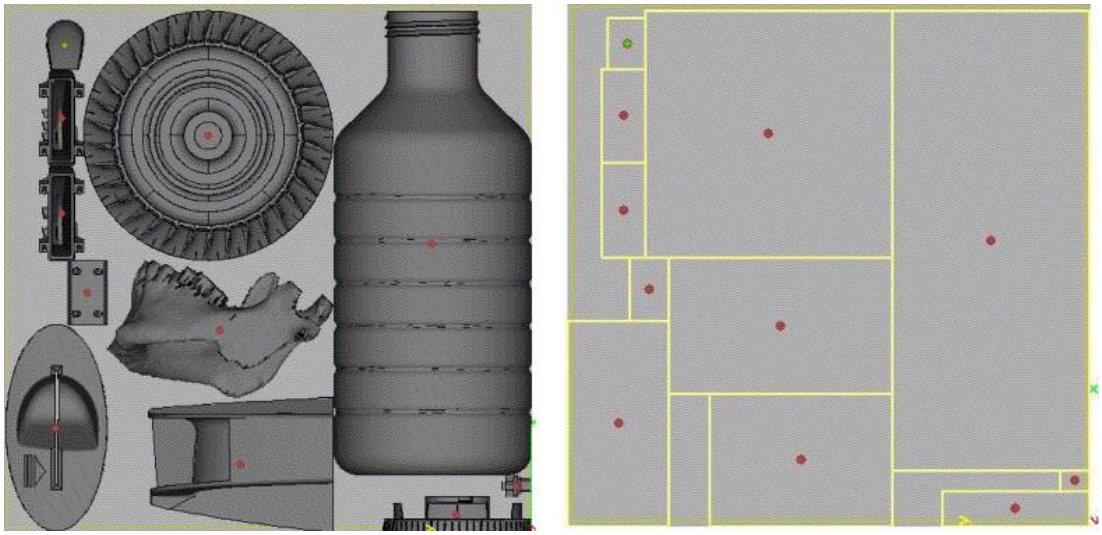


Ilustración 3: Simplificación de piezas a formas rectangulares. Fuente:(Canellidis *et al.*, 2006)

La distribución de las piezas sobre la plataforma se realiza mediante una técnica que denominan *left broder-down border* (LB-DB). Esta técnica trata de colocar las piezas minimizando el largo total requerido posicionando las piezas lo más cercanas al borde inferior izquierdo de la plataforma como sea posible. Los ejes de coordenadas se mueven según se van posicionando las piezas de izquierda a derecha y desde el borde inferior hacia el superior, tal y como se puede observar en la ilustración 4.

Esta técnica se implementó en un programa desarrollado en lenguaje C/C++ y se realizaron pruebas con objetos reales como los presentados en la ilustración 3. Se establecen tres criterios diferentes para la asignación de las piezas: mínimo tiempo empleado en la fabricación de las piezas, mínimo área proyectado y mínima estructura de soporte. En función del criterio se obtiene un porcentaje de área diferente y es responsabilidad del fabricante decidir el criterio más adecuado. Se calcula el porcentaje de área ocupado con cada uno de los criterios y se analizan los resultados. En algunos casos el porcentaje de área ocupado es superior con uno de los criterios y en otros casos se invierten los papeles. Por lo tanto, no se define un criterio que mejore la optimización del área de fabricación. La principal desventaja que presenta el programa es la asignación en paralelo de piezas. El posicionamiento de las piezas se realiza de forma simultánea y no secuencial. Esto permite obtener buenas soluciones en listas de piezas particulares, pero no permite posicionar piezas adicionales a la lista dada inicialmente. Si se desea añadir piezas al lote, el problema debe reconsiderarse desde el principio, adjuntando las piezas nuevas a la lista original. En el problema que presentaremos en este trabajo tendremos en cuenta las lecciones

aprendidas en este artículo para el óptimo posicionamiento de las piezas sobre la plataforma de fabricación.

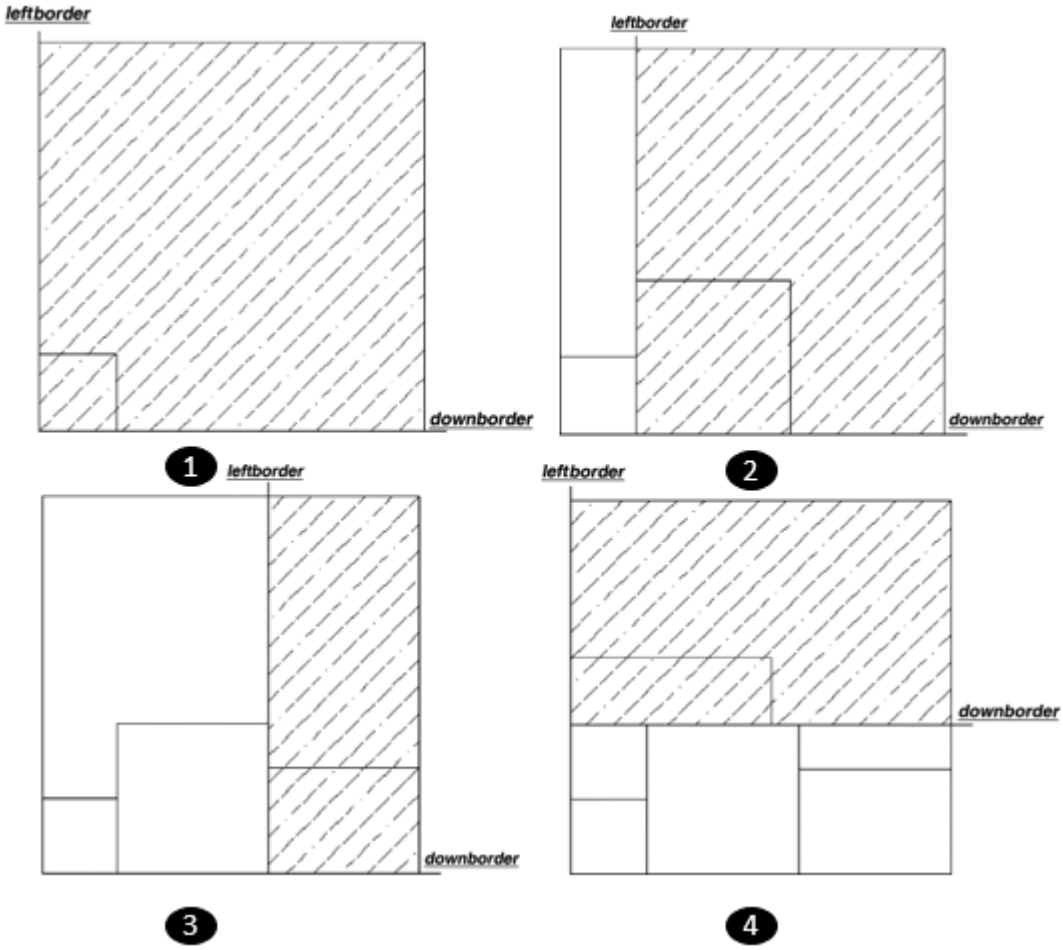


Ilustración 4: Posicionamiento de piezas en la técnica LB-DB. Fuente: (Canellidis et *al.*, 2006)

En 2013, los mismos autores de este artículo mejoran la técnica de optimización de la plataforma de fabricación en 2D y presentan una nueva heurística basada en algoritmos genéticos que mejora la optimización del espacio (Canellidis, Giannatsis and Dedoussis, 2013). La colocación de las piezas sigue utilizando el esquema LB-DB que empleaban en su anterior artículo. Las piezas se colocan de forma simultánea y las figuras se simplifican con su proyección. Como mejora implementan un algoritmo capaz de simplificar las figuras con formas más complejas que un simple rectángulo como en su anterior trabajo. De esta forma, las piezas se asemejan mucho más a la realidad y se consigue un mayor aprovechamiento del espacio de fabricación. En la ilustración 5 se puede observar en color rojo la aproximación que utilizaban con anterioridad y en color verde y líneas a rayas la nueva simplificación de la proyección de las figuras.

Esta técnica es una combinación de la técnica NFP (No-Fit Polygon), empleada para situar piezas con geometrías complejas en el mínimo espacio posible y la técnica LB-DB, explicada

en su anterior artículo *(Canellidis* et al*., 2006)*, que busca optimizar la distribución de las piezas sobre la plataforma de fabricación.

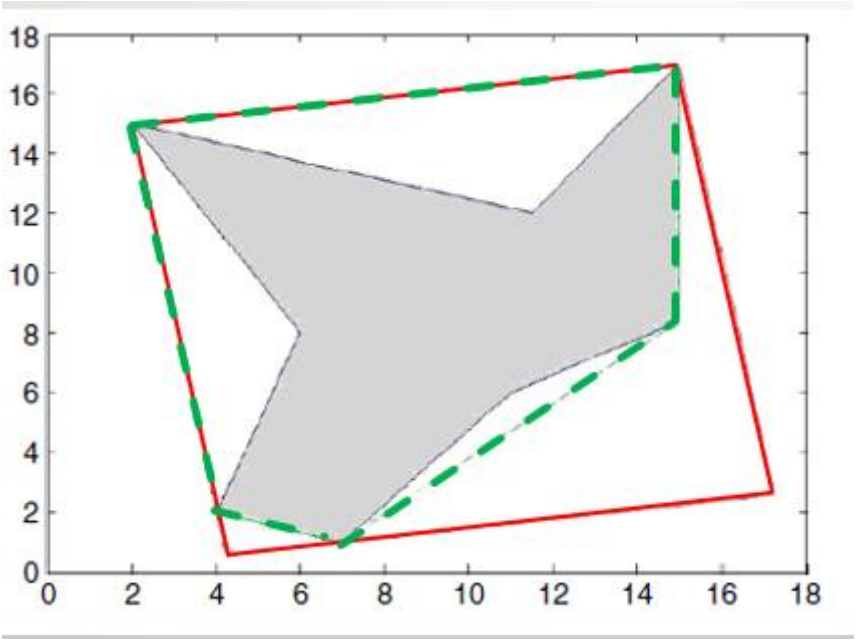


Ilustración 5: Simplificación de las piezas a polígonos. Fuente: (Canellidis, Giannatsis and Dedoussis, 2013)

Cuando se comparan los resultados que se obtienen utilizando la técnica LB-DB de su anterior artículo, la técnica NFP y la combinación de ambas no se encuentran diferencias significativas.

Results of benchmark test cases.

		LBDB	NFP	LBDB and NFP
Albano set	Platform area coverage (%)	82.2	71.6	82.2
	No. parts packed	24	23	24
	Mean comput. time per packing arrangement/generation (s)	1.30	6.00	0.60
	Total computational time (s)	11,554	19,017	5760
Mao set	Platform area coverage (%)	71	71	71
	No. parts packed	20	20	20
	Mean comput. time per packing arrangement/generation (s)	1.18	7.70	0.39
	Total computational time (s)	3115	19,455	928
Blaz set	Platform area coverage (%)	71.4	71.4	71.4
	No. parts packed	20	20	20
	Mean comput. time per packing arrangement/generation (s)	0.98	5.23	0.40
	Total computational time (s)	2652	5230	1372

Tabla 3: Comparación de técnicas de optimización del espacio de fabricación en máquinas de fabricación aditiva. Fuente: (Canellidis, Giannatsis and Dedoussis, 2013)

En la Tabla 3 podemos observar que se obtiene un menor tiempo de computación con la combinación de ambas, pero no se alcanza un mayor porcentaje de área ocupado entre la

técnica LD-DB y la combinación LBDB y NFP. Estos resultados nos indican que podemos reducir las piezas a rectángulos en vez de polígonos más complejos y continuar obteniendo buenos resultados. Por lo tanto, cuando busquemos simplificaciones, reducir las piezas al mínimo rectángulo que contenga la forma de la proyección de la pieza nos permitirá obtener buenas soluciones y reducir notablemente la complejidad de los cálculos que empleemos.

Tal y como comenta Zhou et al., (2018) en su artículo, la impresión 3D como tecnología de rápido desarrollo, se ha convertido en un importante servicio del *Cloud Manufacturing*, gracias a la gran personalización que permite a los usuarios. *Cloud Manufacturing* proporciona un entorno idóneo para integrar de forma eficiente los recursos de las tecnologías de impresión 3D. Básicamente es una plataforma que permite a los clientes realizar sus pedidos de piezas con requerimientos individuales de material, dimensiones y precisión. Se considera que un mismo cliente puede hacer varios pedidos, de la misma forma, un mismo proveedor puede acceder a fabricar varios pedidos de clientes a la vez. La plataforma agrupa las piezas y comprueba que proveedores pueden realizar estos pedidos. El propósito de esta plataforma no se reduce a permitir que los clientes suban sus modelos en 3D a la nube para que los proveedores puedan fabricarlos, sino que se realiza un emparejamiento de cliente-fabricante en función de los atributos de las piezas.

El proceso que se presenta comienza con el cliente realizando el pedido con sus requerimientos personales, para ello, elige el modelo o modelos entre todos los disponibles en la librería que ofrece la plataforma (Ilustración 6). La asignación de los pedidos con los fabricantes se realiza en función del coste, dimensiones del modelo, material de impresión y precisión. Una vez encontrados todos los posibles emparejamientos, se programa su producción. Para ello, se hace uso de vectores de prioridad para cada cliente sobre cada fabricante y viceversa. Las asignaciones se realizan con el objetivo de minimizar el tiempo promedio de entrega del último pedido.

Una plataforma similar se presenta en el proyecto Lonja 3d. Lonja 3d es un proyecto de investigación financiado por el Fondo Europeo de Desarrollo Regional y desarrollado por del GIR INSISOC. En este proyecto se pretende desarrollar un mercado gestionado (una lonja) mediante subastas combinatorias.

En la actualidad, cuando un particular o una empresa desea un producto fabricado mediante impresión 3D debe recurrir a repositorios online o contactar con proveedores locales. Al realizar un pedido de una sola pieza, los costes para el fabricante pueden ser muy elevados y el precio de la pieza aumenta. El proyecto lonja3d tiene como objetivo crear un mercado donde se comerciará con productos fabricados mediante impresión 3D y permitirá a los clientes hacer uso de subastas combinatorias para obtener mejores precios por parte de los fabricantes. Además, los fabricantes pueden optimizar su producción, al fabricar las piezas de diferentes clientes como si formaran parte de un solo pedido, optimizando la capacidad productiva de las máquinas y reduciendo de esta forma los costes individuales de cada pieza.

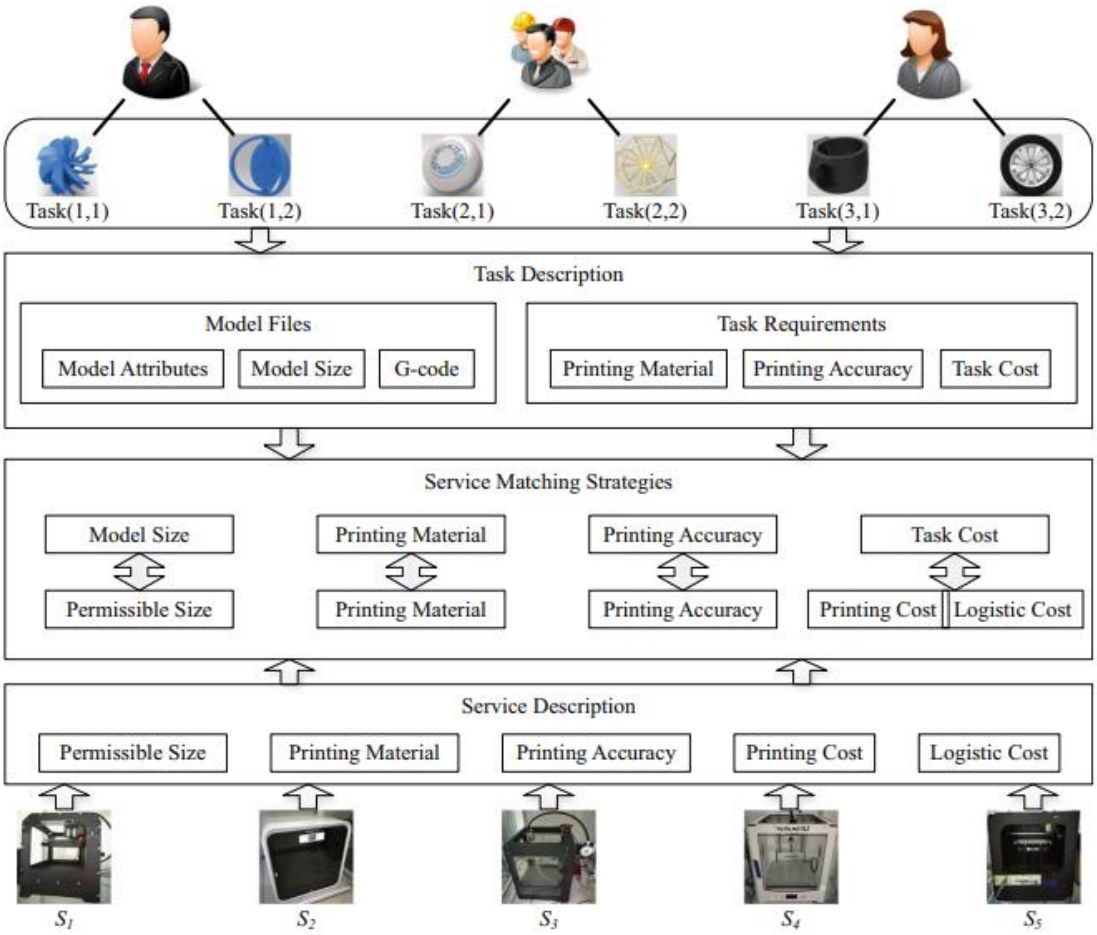


Ilustración 6: Plataforma Cloud Manufacturing. Fuente:(Zhou *et al.*, 2018)

Planteamos el escenario en el cual un grupo de clientes ha realizado diferentes pedidos de piezas y el fabricante ha aceptado la oferta para fabricar estas piezas. En este trabajo ofrecemos una solución al problema con el que se encuentra el fabricante que forma parte de lonja3d. Una vez que los clientes han realizado sus pedidos y estos han sido clasificados y agrupados en función de las impresoras que tengan la capacidad de fabricarlos (material, dimensiones y precisión), el fabricante tendrá que tomar una decisión importante, qué piezas fabricar en cada lote de producción. Suponemos que no todas las piezas caben en la cama de fabricación y habrá que decidir cuales interesa fabricar en primer lugar. Como los tiempos necesarios para producir un lote de piezas son muy altos, el problema que planteamos lo relacionamos con la producción diaria. Es decir, el fabricante tendrá que decidir que lote de piezas va a fabricar durante la producción diaria en una de sus máquinas de impresión 3D y cuales dejará para otros lotes.

El primer problema que tenemos que resolver es el problema de empaquetado. Para simplificar la complejidad del problema, las piezas se reducirán a formas rectangulares (Ilustración 3) que tendremos que posicionar sobre la cama de fabricación. El problema se reduce a encontrar el patrón de empaquetamiento de rectángulos de diferentes tamaños, dentro de un rectángulo de mayor tamaño con el objetivo de desperdiciar el mínimo área.

Este tipo de problema es NP-completo, ya que, al aumentar el número de piezas, las combinaciones posibles aumentan de forma exponencial. Si el número de piezas es n, el tamaño del espacio de soluciones viene dado por:

$$2^n \cdot n!$$

Por lo tanto, si el número de piezas que deseamos posicionar es 25, el tamaño del espacio de soluciones supera al tamaño del problema del agente viajero para un caso con igual número de ciudades, considerando que todas están conectadas entre sí (10^{31}):

$$2^{25} \cdot 25! > 10^{31}$$

Esta es la razón principal por la que necesitamos hacer uso de técnicas heurísticas (Jakobs, 1996). Aunque mediante una heurística no obtenemos la solución óptima al problema, somos capaces de obtener soluciones relativamente buenas y cercanas al óptimo con tiempos de computación cortos.

Para resolver este tipo de problemas, se han propuesto diferentes técnicas metaheurísticas desde que Gilmore y Gomory proponen el primer algoritmo en 1966 (Mirledy Toro, Garcés and Ruiz, 2008). En la tabla 4 se agrupan algunos de los algoritmos presentados hasta la fecha y sus correspondientes autores (Cui, Yao and Cui, 2016).

Cada uno de estos autores ha realizado diferentes pruebas para comprobar la eficiencia y la eficacia de sus algoritmos, comparando tanto los porcentajes de ocupación obtenidos, como los tiempos de computación empleados con el resto de los planteamientos anteriores. Yaodong Cui en su artículo (Cui, Yao and Cui, 2016) presenta una heurística híbrida que funciona en dos etapas. En la primera etapa se emplea una heurística basada en generación de columnas para resolver un conjunto de problemas residuales. Como solución de esta primera fase se obtienen un conjunto de patrones factibles. En la segunda fase se aplica un algoritmo de programación lineal aplicado a los patrones obtenidos en la primera etapa. Se establece un tiempo límite de computación y se guarda la mejor solución obtenida en ese intervalo de tiempo. Con esta heurística híbrida se consigue superar al resto de modelos existentes hasta la fecha en cuanto a calidad de la solución obtenida. En cambio, los tiempos de computación son superiores a los presentados por Polyakovsky and M'Hallah, (2009), Fleszar and Charalambous, (2011), Fleszar, (2013) y Cui, Cui and Tang, (2015). En este trabajo presentaremos una heurística diferente a las utilizadas hasta la fecha y compararemos nuestra eficiencia con la batería de problemas que se presenta en alguno de estos artículos.

Una vez optimizada la superficie de fabricación, el fabricante deberá elegir entre todos los lotes posibles aquel que ofrezca un mayor beneficio. A pesar de haber resuelto el problema en 2D, las piezas cuentan con una altura y un porcentaje de relleno. El beneficio generado por cada pieza dependerá en gran medida de estas dos variables y será necesario establecer un criterio de decisión entre lotes con porcentajes de ocupación del área de fabricación similares.

Autor	Técnica	Año
Christofides y Withlock	Búsqueda en árbol	1977
Wang	Desarrollo incremental	1983
Vasko	Desarrollo incremental	1989
Oliveira y Ferreira	Desarrollo incremental	1990
Lai y Chan	Recocido simulado	1997
Parada	Recocido simulado con codificación de árbol binario	1998
Lodi	Búsqueda Tabú	1999
Leung	Algoritmo evolutivo	2001
Beasley	Algoritmo genético	2003
Yaodong Cui	Algoritmo exacto	2007
Eliana Mirledy	Algoritmo híbrido constructivo de búsqueda en vecindad variable y recocido simulado	2008
Filipe Alvelos	Búsqueda de vecindad variable descendente	2009
Sergey Polyakovsky	Heurística basada en agentes	2009
Chan	Estructura de vecindad estocástica	2011
Yaodong Cui	Agrupación secuencial	2013
Krzysztof Fleszar	Finite-Best-Strip	2013
Filipe Alvelos	Heurística híbrida basada en generación por columnas	2014
Yi-Ping Cui	Heurística secuencial	2015
Yaodong Cui	Heurística híbrida en dos fases	2016

Tabla 4: Algoritmos empleados en la resolución del problema de empaquetado bidimensional. Fuente: Elaboración propia.

3. Resolución del problema

En primer lugar, vamos a presentar las consideraciones iniciales que tenemos en cuenta para simplificar la situación y poder ofrecer una solución factible en el menor tiempo posible.

- La localización de las piezas se resuelve en 2D, en la base formada por los ejes X e Y. Esto se plantea de esta forma porque consideramos que no es posible superponer piezas.
- Las piezas se simplificarán a rectángulos, siendo esta forma la proyección de la pieza sobre el plano XY. Estos rectángulos serán algo mayores que la propia proyección para evitar el contacto entre piezas, de forma que no se comprometa la calidad en la pieza final. El volumen de cada pieza será un cubo (Ilustración 7).
- Las piezas cuentan con una orientación fija, de forma que no se comprometa la calidad final de cada una de ellas. Solo podremos rotarlas respecto al eje Z y trasladarlas en los ejes X e Y.
- El cálculo de la cantidad de material se aproximará por el producto del volumen del cubo que representa cada pieza multiplicado por el porcentaje de relleno.

La resolución del problema planteado se realizará en dos etapas, una primera que llamaremos “problema de empaquetado” y una segunda etapa que denominaremos como “elección del ganador”. En la primera etapa, calcularemos posibles lotes de fabricación con su correspondiente distribución en la cama de fabricación, buscando optimizar la capacidad productiva de la máquina. En la segunda, decidiremos cual de todos los lotes reporta mayores beneficios al fabricante.

3.1. Problema de empaquetado

Una vez que contamos con el conjunto de piezas compatibles para fabricar de forma simultánea en una impresora 3d, es decir, aquellas piezas que se fabriquen con el mismo material, requerimientos de calidad alcanzables por la máquina y dimensiones inferiores al espacio de fabricación, definiremos los parámetros de cada una de las piezas. Cada pieza vendrá definida por 6 parámetros:

- Nombre (P_i): servirá para identificar y diferenciar cada una de las piezas.
- Relleno (r_i): porcentaje de relleno de una pieza, medido en tanto por uno. Por ejemplo, 0 indica que la pieza está conformada por la cáscara vacía. Un 0.2 indica un relleno del 20% y el valor 1 representa una pieza completamente maciza.
- Largo (l_i): dimensión de la pieza en el eje X. Medida en mm.
- Ancho (w_i): dimensión de la pieza en el eje Y. Medida en mm.
- Alto (h_i): dimensión de la pieza en el eje Z. Medida en mm.
- Asignación: variable Booleana que nos permitirá saber si la pieza está o no asignada al lote de fabricación. Todas las piezas comienzan con esta variable en “False”, es decir, disponibles.
- Coordenada X (cX): indicará la posición de la coordenada de la esquina superior izquierda de la pieza sobre el eje X (Ilustración 8)

- Coordenada Y (cY): indicará la posición de la coordenada de la esquina superior izquierda de la pieza sobre el eje Y (Ilustración 8).

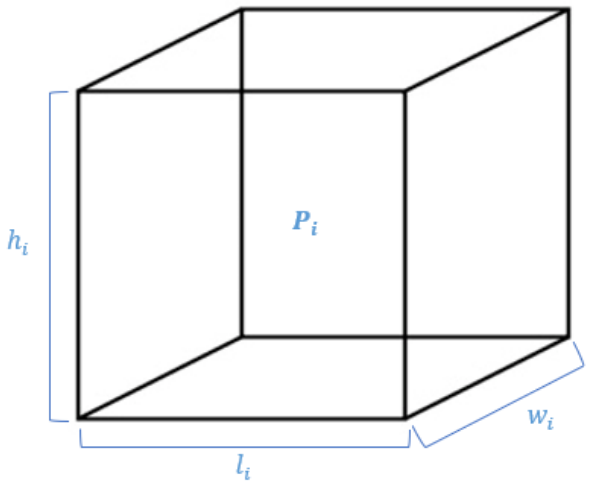


Ilustración 7: Dimensiones de la representación de las piezas. Fuente: Elaboración propia.

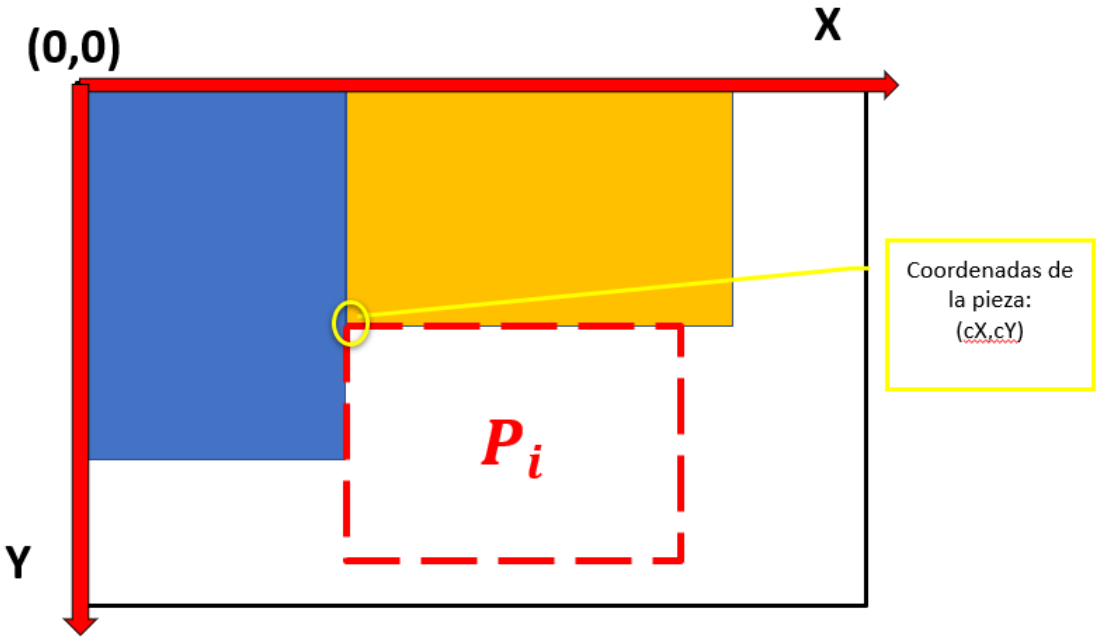


Ilustración 8: Coordenadas de la pieza Pi. Fuente: Elaboración propia.

A partir de estos parámetros podemos calcular:

- El área de cada pieza (a_i):

$$a_i = l_i \times w_i$$

- El volumen (v_i):

$$v_i = l_i \times w_i \times h_i$$

- La cantidad de material necesario para su fabricación (m_i):

$$m_i = v_i \times r_i$$

Además, definiremos nuestra área de fabricación en 3D, mediante el ancho y el largo de la cama de fabricación de la impresora 3D que ofrezca el proveedor (Ilustración 9). Cada área vendrá definida por 4 parámetros:

- Nombre (A_j): servirá para identificar el área de fabricación de cada impresora.
- Largo (L_j): dimensión del área de fabricación en el eje X. Medida en mm.
- Ancho (W_j): dimensión del área de fabricación en el eje Y. Medida en mm.
- Alto (H_j): dimensión del área de fabricación en el eje Z. Medida en mm.
- Coordenada X (CX): indicará la posición de la coordenada de la esquina superior izquierda del área sobre el eje X.
- Coordenada Y (CY): indicará la posición de la coordenada de la esquina superior izquierda del área sobre el eje Y.

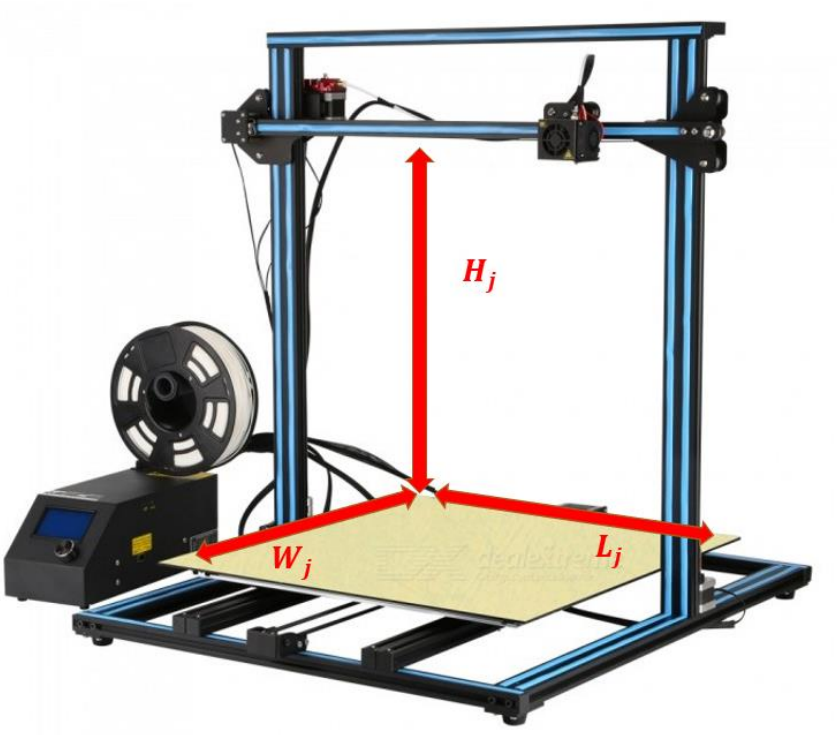


Ilustración 9: Dimensiones de la impresora 3D. Fuente: Elaboración propia.

En primer lugar, vamos a explicar los términos que usaremos a lo largo del proceso de asignación:

- Lista de piezas disponibles (LPD): lista de piezas que no se han asignado a ningún área/subárea de fabricación.

- Lista de áreas disponibles (LAD): lista de todas las áreas que no han sido utilizados y en los cuales, a priori, caben piezas de la lista de piezas disponibles.

Partimos del conjunto de piezas compatibles.

Ejemplo:

P1	P2	P3	P4	P5

Esta lista se ordena de forma aleatoria generando la primera LPD. Con la reordenación aleatoria conseguiremos generar lotes con porcentajes de ocupación diferentes, distinto número de piezas y diferentes distribuciones de piezas sobra la plataforma de fabricación. Dentro de la literatura de optimización del espacio en problemas de empaquetado no hay ningún estudio que localice las piezas da la forma que explicaremos en este trabajo. Por lo tanto, no contamos con criterio de ordenación que mejore los resultados obtenidos. Mediante una reordenación aleatoria conseguimos obtener una gran cantidad de lotes diferentes con un tiempo de computación ínfimo.

Ejemplo: LPD

P2	P4	P1	P5	P3

Además, contaremos con la LAD, compuesta por el área de fabricación de la impresora 3D que hayamos introducido en el programa.

Ejemplo: LAD

A1				

El programa se encargará de asignar las piezas del lote al área de fabricación de la siguiente forma:

Se comprueba si la primera pieza de la LPD, P2 en nuestro ejemplo, está asignada al lote de fabricación. Si es así, pasará a la siguiente pieza de la LPD. En caso contrario, comprobaremos si cabe en la primera área de LAD. Para ello, se compara en primer lugar el ancho de la pieza (w_i) y el ancho del área de fabricación (W_j) (ilustración 10).

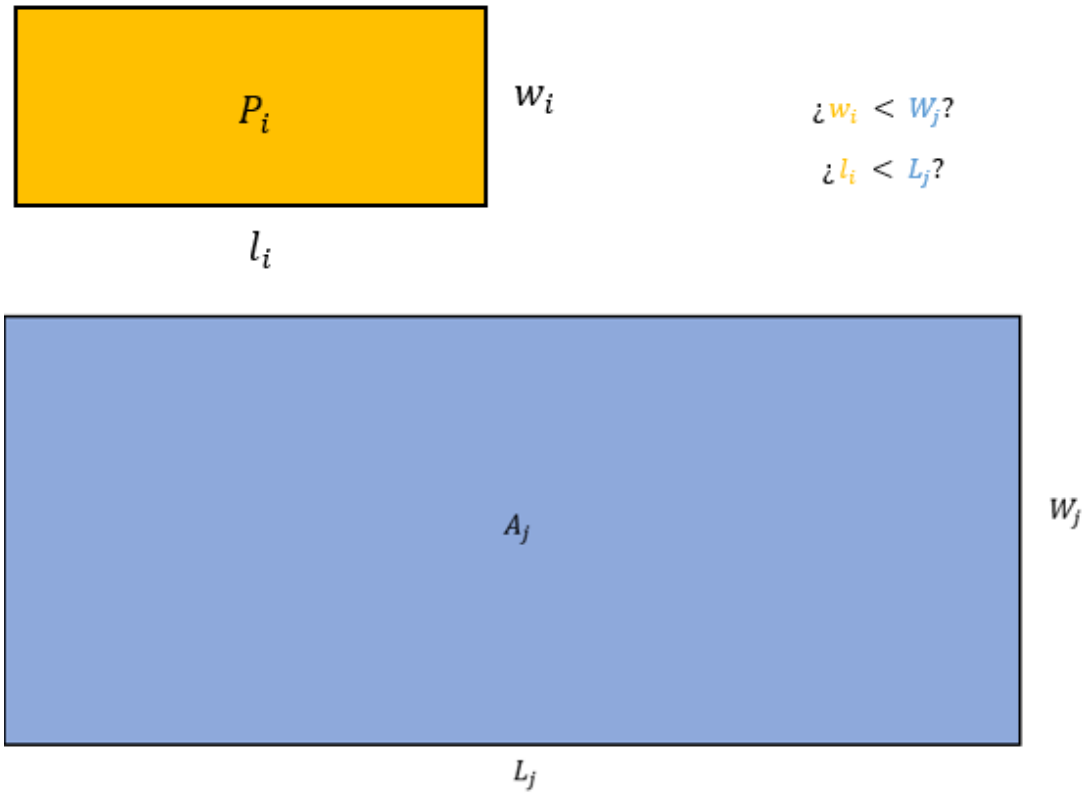


Ilustración 10: Comparación de dimensiones Pieza-Área. Fuente: Elaboración propia.

Como resultado podemos encontrarnos con dos casos diferentes:

1) El ancho de la pieza es menor que el ancho del área de fabricación. En este primer caso, se procederá a comprobar si el largo de la pieza es menor al del área de fabricación. De nuevo nos encontramos con dos nuevas posibilidades:
    a. El largo de la pieza es menor que el largo del área de fabricación (Ilustración 11). Si esto ocurre, se asigna la pieza al lote de fabricación.

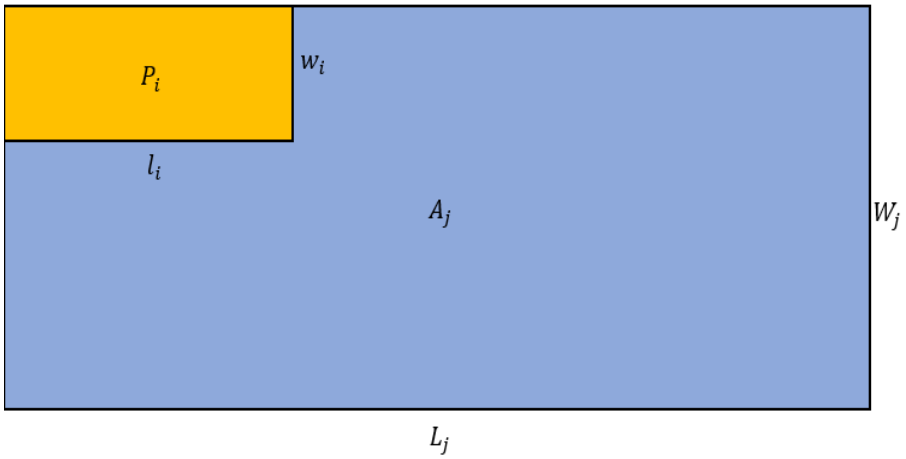


Ilustración 11: Posicionamiento de la pieza Pi sobre el área Aj. Fuente: Elaboración propia.

Además, crearemos dos nuevas áreas:
El primero (A_{j+1}) se creará a partir de la siguiente fórmula:

$$A_{j+1} = (l_i, W_j - w_i)$$

Coincidiendo el largo del área con el largo de la pieza asignada al lote. El ancho coincidirá con la resta del ancho total del área de fabricación y el ancho de la pieza asignada (ilustración 12).

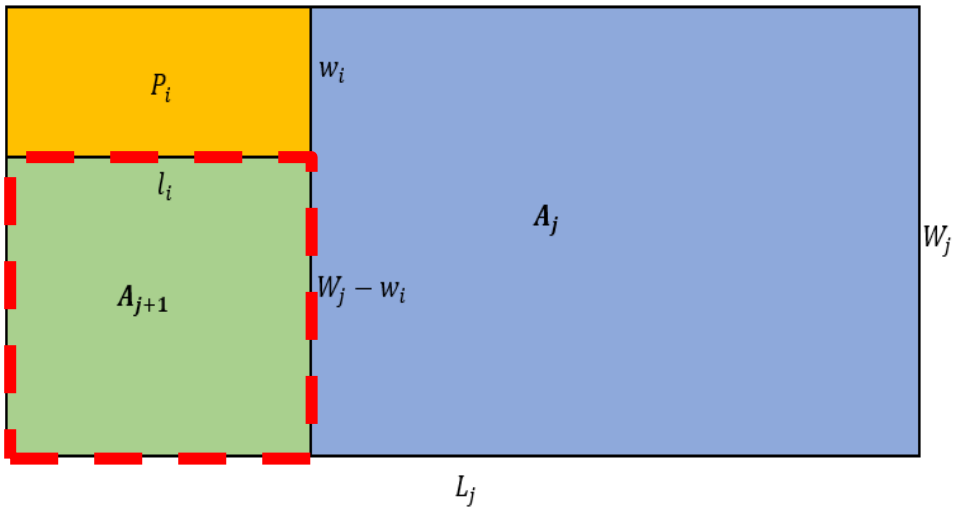


Ilustración 12: Dimensiones de la nueva área A_{j+1}. Fuente: Elaboración propia.

El segundo (A_{j+2}) se creará a partir de la siguiente fórmula:

$$A_{j+2} = (L_j - l_i, W_j)$$

Coincidiendo el ancho del área con el ancho del área en el que la pieza ha sido asignada al lote. El largo coincidirá con la resta del largo total del área de fabricación y el largo de la pieza asignada (Ilustración 13).

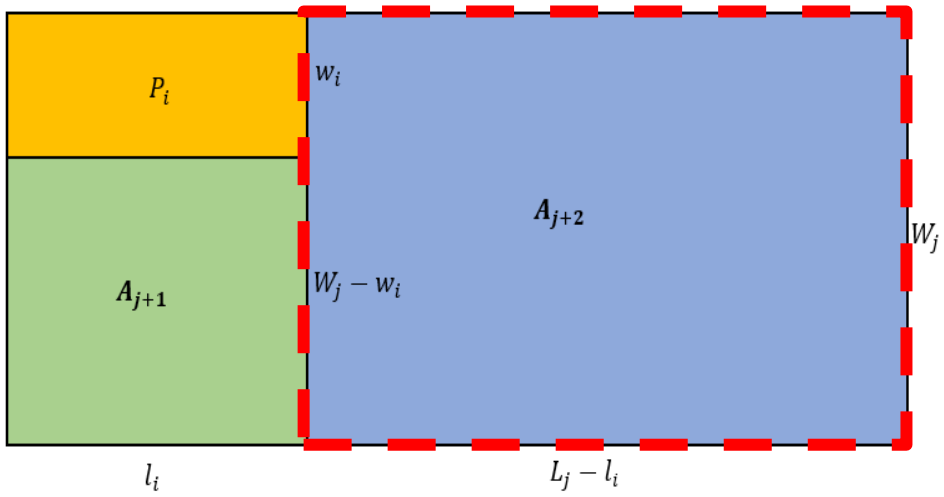


Ilustración 13: Dimensiones de la nueva área A_{j+2}. Fuente: Elaboración propia.

Estas dos nuevas áreas se incluirán en la LAD, mientras que el área A_j, a partir del cual se han creado, se eliminará de la lista. Para definir la posición de las nuevas áreas creados, se asignarán unas coordenadas a partir de las coordenadas del área inicial (Ilustración 14).

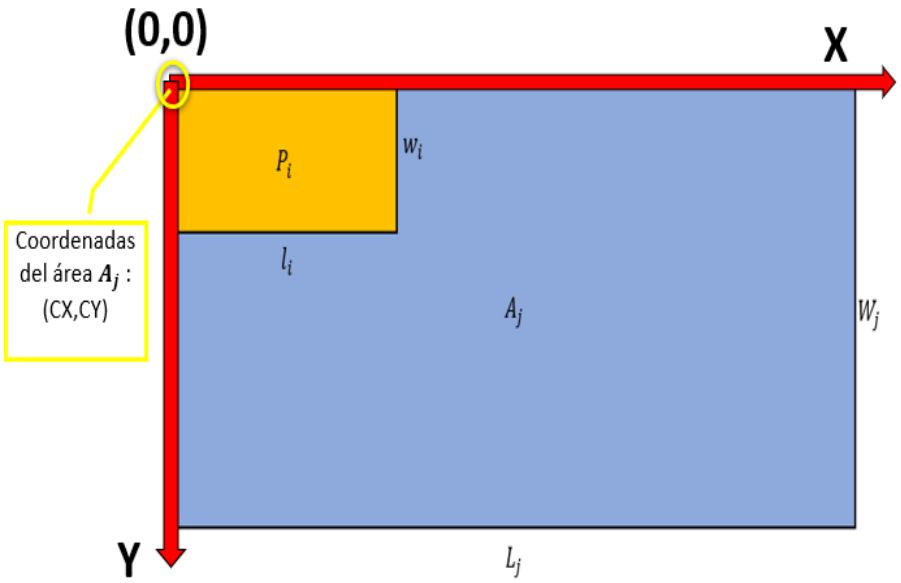


Ilustración 14: Coordenadas del área Aj. Fuente: Elaboración propia.

La coordenada de A_{j+1} en el eje X coincidirá con la coordenada X del área a partir del cual se ha creado. Mientras que la coordenada en el eje Y será la coordenada en el eje Y del área a partir del cual se ha creado más el ancho de la pieza asignada (w_i), tal y como se puede observar en la Ilustración 15.

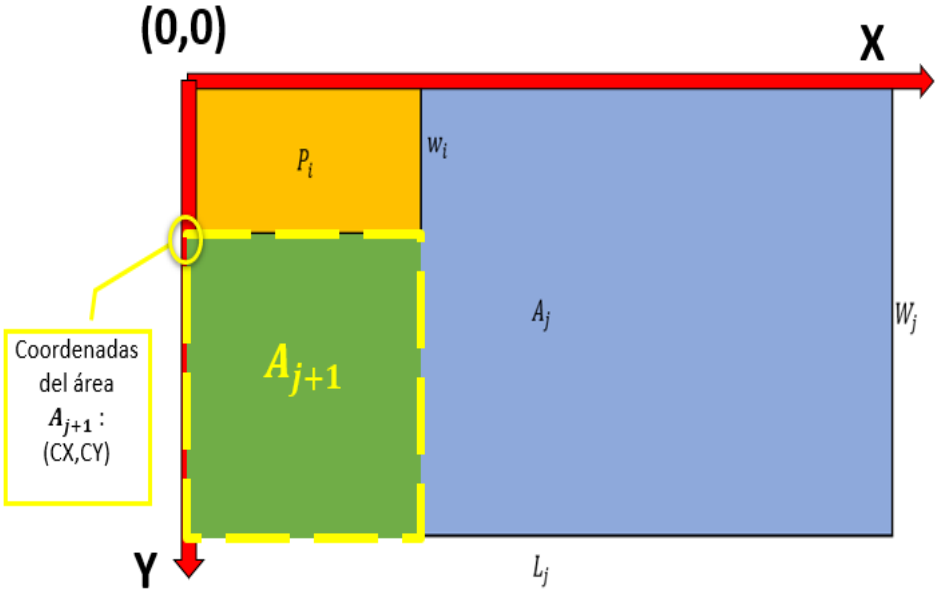


Ilustración 15: Coordenadas de la nueva área A_{j+1}. Fuente: Elaboración propia.

La coordenada de A_{j+2} en el eje X será la suma de la coordenada X del área a partir del cual se ha creado y el largo de la pieza asignada (l_i). Mientras que la coordenada en el eje Y coincidirá con la coordenada Y del área a partir del cual se ha creado, tal y como se muestra en la Ilustración 16.

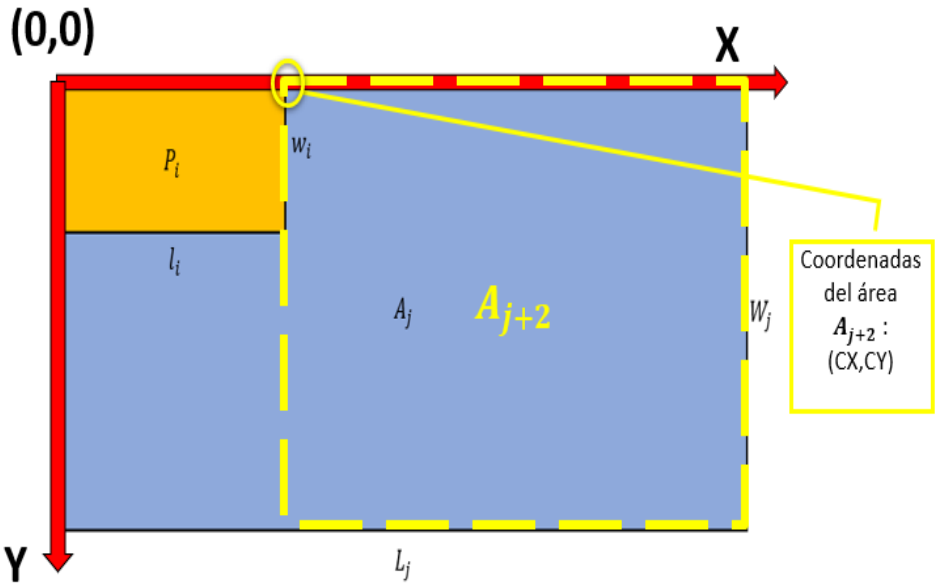


Ilustración 16: Coordenadas de la nueva área A_{j+2}. Fuente: Elaboración propia.

b. En el caso de que el largo de la pieza sea mayor que el largo del área de fabricación, reorientamos la pieza (Ilustración 17). Para ello simplemente cambiamos el valor del ancho por el largo y viceversa.

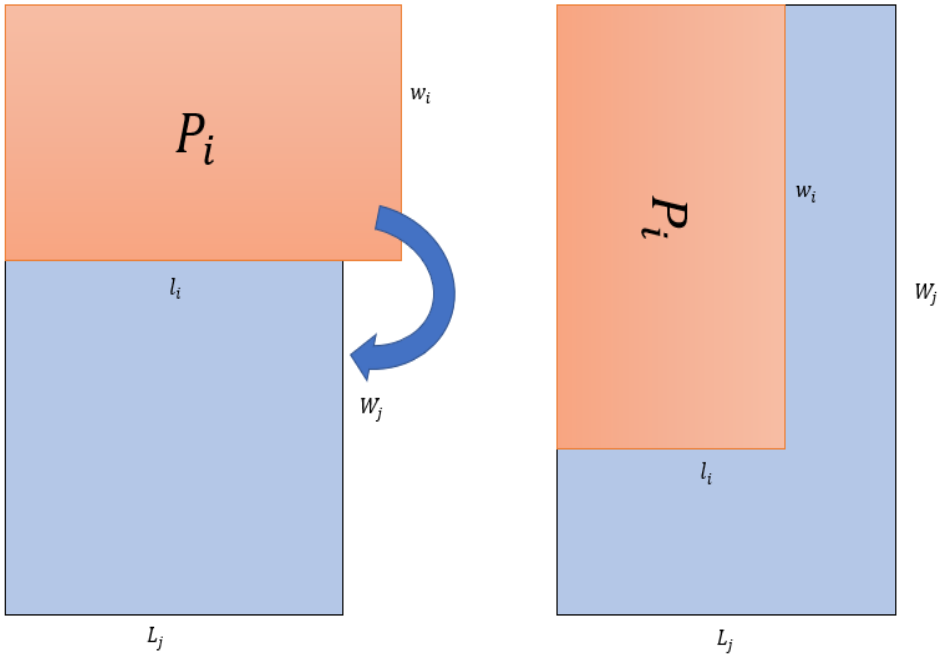


Ilustración 17: Rotación de la pieza Pi para el caso $l_i > L_j$. Fuente: Elaboración propia.

Una vez realizado el cambio, comprobamos si el largo de la pieza es menor que el del área de fabricación.

i. Si esto no se cumple, la pieza no cabe en ninguna de sus orientaciones, pasamos a comprobar si cabe en la siguiente área de la LAD.
ii. Si el largo de la pieza es menor que el del área de fabricación, tendremos que comprobar si el ancho es menor. Dos nuevas situaciones:
    1. Si se cumple, asignamos la pieza al lote de fabricación y pasaremos a la siguiente pieza. Además, tendremos que crear dos nuevas áreas, de la misma forma que se indicó en el paso 1. a) y eliminar el área correspondiente de la LAD.
    2. Si no se cumple, pasamos a comprobar si cabe en la siguiente área de la LAD.

2) La otra posibilidad que debemos contemplar es que el ancho de la pieza sea mayor que el ancho del área de fabricación. En este caso, reorientamos la pieza cambiando el valor del ancho por el largo y viceversa (Ilustración 18).

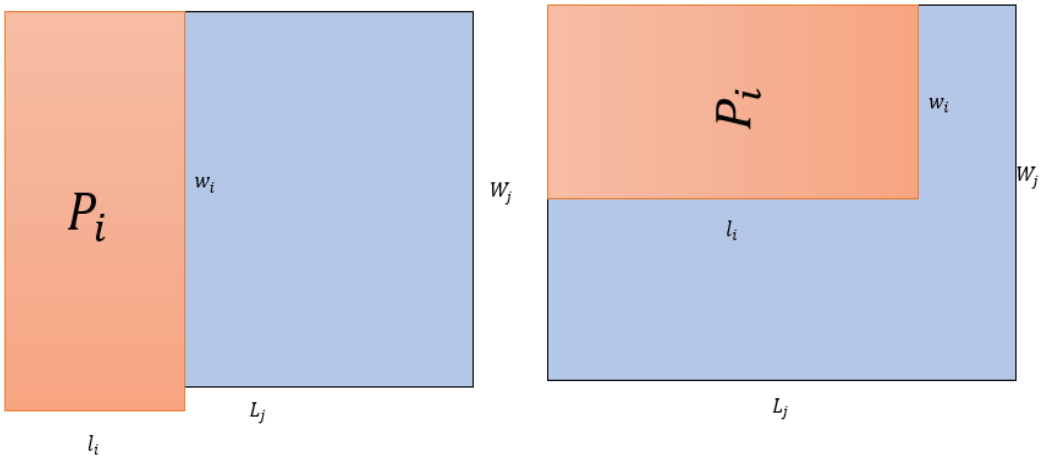


Ilustración 18: Rotación de la pieza Pi para el caso $w_i > W_j$. Fuente: Elaboración propia

Una vez reorientada, comprobaremos si el nuevo ancho de la pieza es menor que el ancho del área de fabricación. De nuevo tenemos posibilidades:

a. Si esto no se cumple, la pieza no cabe en ninguna de sus orientaciones y pasaremos a comprobar si cabe en la siguiente área de la LAD.
b. Si el ancho de la pieza es menor que el del área de fabricación, tendremos que comprobar si el largo es menor, encontrándonos ante dos nuevas situaciones:
    i. Si se cumple, asignamos la pieza al lote de fabricación y pasaremos a la siguiente pieza. Además, tendremos que crear dos nuevas áreas, de la misma forma que se indicó en el paso 1. a) y eliminar el área correspondiente de la LAD.
    ii. Si no se cumple comprobaremos si cabe en la siguiente área de la LAD.

Repetiremos esta secuencia para cada una de las piezas de nuestra LPD, con sus dos orientaciones posibles en cada una de las áreas que se vayan creando a lo largo de la simulación. La simulación se dará por finalizada una vez que todas las piezas hayan sido asignadas al área de fabricación, o bien cuando las piezas sin asignar no quepan en el área restante.

3.2. Elección del ganador

Cada simulación llevará asignada una variable que nos ayudará a decantarnos por el lote ganador del problema de empaquetado. Esta variable es la cantidad de material empleado para fabricar el lote de piezas respecto al área de fabricación disponible en la impresora 3D. Para calcular la cantidad de material utilizada en una pieza, simplificaremos el volumen de la pieza por un cubo. Siendo el volumen total (v_i):

$$v_i = l_i \times w_i \times h_i$$

Para calcular la cantidad de material necesario para su fabricación (m_i) simplemente multiplicamos el volumen del cubo por el porcentaje de relleno (r_i):

$$m_i = v_i \times r_i$$

Realizamos el sumatorio de las cantidades de material utilizado para cada pieza asignada al lote de fabricación y obtenemos la cantidad total empleada en ese lote, medida en volumen (mm^3).

Se realizarán tantas simulaciones como se crea conveniente y una vez completadas todas, el programa se detendrá automáticamente. Por pantalla aparecerá aquella simulación que haya empleado una mayor cantidad de material, mostrando el número de la simulación, la lista de piezas asignadas al lote, las piezas sin asignar, el porcentaje de área ocupado y la cantidad de material empleado.

Esta herramienta permite al fabricante decidir que lote de fabricación le aporta un mayor beneficio entre todas las posibilidades. Aquel lote con mayor utilización de material será el que produciremos. Nos basamos en que, a mayor cantidad de material utilizado en la producción de un mismo lote, mayor beneficio recibirá el fabricante. Para justificar esta decisión, consideramos que como consecuencias de elegir el lote con mayor cantidad de material empleado tenemos:

- Reducción en el tiempo total de fabricación. Por ejemplo, en las técnicas de fabricación aditiva con lecho en polvo, la impresora extiende capas completas de material en polvo sin importar si el láser incide sobre una pequeña porción o una más grande. Por lo tanto, al asignar lotes con mayor cantidad de material, estamos aprovechando el tiempo empleado en extender estas capas de material durante la producción de piezas.
- Aumento en la cantidad de material. Siendo el coste más significativo en la impresión 3d (De Antón Heredero, 2018). Con el aumento de la cantidad de material empleado en el lote se aumenta el retorno generado.
- Optimización del consumo de energía. De la misma forma que ocurre con la reducción del tiempo de fabricación, al optimizar el tiempo de procesado, estamos optimizando el consumo de energía y, por tanto, reduciendo el coste de las piezas.

Esto repercutirá sobre los costes finales de cada pieza y permitirá al fabricante optimizar la producción, reduciendo los costes individuales de cada pieza y aumentando el margen de beneficio.

3.3. Implementación en Python

3.3.1. Introducción al Lenguaje Python

Para realizar las simulaciones y obtener una solución factible en el menor tiempo posible, se ha decidido realizar un programa en Python.

Python es un lenguaje de programación creado por Guido van Rossum a principios de los años noventa. Su nombre proviene del conocido programa de televisión de comedia británica de los años setenta llamado "*Monty Python's Flying Circus".* Python ofrece un entorno de

desarrollo integrado (IDE) que recibe el nombre de IDLE, haciendo honor a Eric Idle, un miembro del grupo del programa. Un IDE es un paquete de herramientas de software que permite desarrollar programas. Entre estas herramientas se encuentra un editor para crear nuevos programas y editarlos, un traductor que permite ejecutar los programas y un depurador para encontrar errores mientras se ejecuta el programa. Además, Python cuenta con un intérprete interactivo, conocido como Shell de Python, que permite al usuario interactuar con el programa.

Una gran ventaja de este lenguaje frente al resto es la amplia biblioteca de módulos. Python cuenta con módulos integrados que proporcionan una funcionalidad específica. Algunos ejemplos son el módulo *math*, que ofrece algunas funciones matemáticas más complejas que las implementadas en el propio "core" (núcleo) de Python, el módulo *random* que permite generar números aleatorios con una programación muy sencilla, entre otros.

Python es un lenguaje que está en pleno auge, gracias a que ofrece muchas ventajas, por ejemplo:

- Simplicidad de código, no es necesario definir el tipo de cada dato.
- El código está muy estructurado y, por lo tanto, es muy legible por otros usuarios.
- Cuenta con una gran cantidad de librerías.
- Su portabilidad tanto en Mac, Linux, como en Windows.
- Hay una gran comunidad de programadores que aportan módulos y librerías de forma continua.

Como inconveniente, podemos destacar que, al ser un lenguaje interpretado, su ejecución es más lenta que otros lenguajes. Al realizar un programa tan corto como el nuestro, esto no supondrá un problema.

Estas características hacen de Python un lenguaje de programación muy utilizado por empresas de gran calibre como: Youtube (Google), Quora, Yahoo, NASA, IBM, Mozilla, Instagram y Dropbox. Entre las áreas que destaca se encuentran:

- Desarrollo web
- Backend para Web Services y APIs Rest
- Integración con Web services SOAP
- Machine learning
- Inteligencia Artificial
- Simulación
- Infraestructura y DevOps
- Unit Testing

Entre todas áreas se encuentra nuestro caso, la simulación. Pero, la razón principal por la cual utilizamos Python es que se trata de un lenguaje orientado a objetos. La Programación Orientada a Objetos (POO) es un paradigma de programación que usa objetos y sus interacciones para diseñar aplicaciones y programas de ordenador. Es una forma especial de programar que permite expresar de una forma más cercana las cosas de la vida real que otros tipos de programación. La POO se basa en objetos, tal y como son nuestras piezas con

características definidas como su largo, ancho, alto, relleno, etc. Estas características formarán parte de lo que definiremos como una clase, y cada objeto (instancia de una clase), es decir, cada pieza contará con un valor diferente para el alto, ancho, largo, etc.

Además, cuenta con listas que nos permiten almacenar diferentes tipos de datos e incluso objetos. De esta forma, podemos crear una lista en la cual vamos introduciendo todas las piezas. Las listas en Python se recorren con muy pocas líneas de código, simplificando en gran medida el código que necesitamos para programar la heurística que resuelve el problema de empaquetado explicado antes. Además, acceder a las propiedades de cada una de las piezas es muy sencillo, pudiendo realizar comparaciones entre las dimensiones de las piezas y las diferentes áreas sin necesitar muchas líneas de código como ocurriría en otros lenguajes que no están orientados a objetos.

3.3.2. Inicio del Programa

A continuación, vamos a explicar la programación utilizada para implementar la heurística propuesta.

En primer lugar, añadimos la librería *random* para poder generar de forma aleatoria el orden de la lista de piezas y realizar tantas simulaciones como creamos convenientes. También usaremos esta librería para asignar colores aleatorios a las piezas y diferenciarlos.

Además, añadiremos *tkinter* que es un "*binding*" (adaptación de una librería al lenguaje Python) que nos permitirá representar todas las piezas y su colocación en el área de fabricación. Para poder extraer datos desde un archivo Excel y guardar los resultados, importamos una nueva librería: *openpyxl*. De esta librería necesitaremos la función *load_workbook* que nos permitirá extraer sus datos y modificarlos. Para agregar todas estas librerías y funciones, tenemos las primeras líneas de código (Ilustración 19).

```
from tkinter import *
import random
from openpyxl import load_workbook
```

Ilustración 19: Importación de librerías e Python. Fuente: Elaboración propia.

Una vez añadida la librería de Excel, tenemos que indicar el nombre del archivo que vamos a utilizar para guardar los resultados. El nombre del archivo se asigna como una cadena de caracteres a la variable **FILE_PATH.** Es necesario declarar a qué hoja del Excel queremos acceder. Declararemos la variable **HojaSimulaciones** y la utilizaremos para guardar el nombre de la hoja con aquellos lotes de piezas que nos interesen. Definimos la variable **SHEET** y guardamos en ella el nombre de la hoja donde estará la solución que ofrezca el mayor porcentaje de área ocupado y aquella que cuente con la mayor cantidad de material empleado. Una vez creadas estas variables, llamamos a la función "*load_workbook*" y le indicamos el archivo que deseamos abrir y el modo de apertura. Con el comando "*read_only = False*" indicamos que no solo queremos acceder a los datos, sino que también deseamos modificar el archivo. Creamos las variables **sheet** y **hoja** como hojas del archivo Excel en Python (Ilustración 20).

```
print("Introduzca el archivo en el que desea guardar los resultados: ")
FILE_PATH=input("Nombre del archivo(Ej: nombre.xlsx): ")
SHEET=input("Hoja de la solución ganadora: ")
HojaSimulaciones=input("Hoja para guardar todos los posibles lotes: ")
wb = load_workbook(FILE_PATH, read_only=False)
sheet = wb[SHEET]
hoja = wb[HojaSimulaciones]
```

Ilustración 20: Código Python para la introducción de datos en Excel. Fuente: Elaboración propia.

A continuación, inicializamos las clases de los objetos que vamos a usar a lo largo del programa. Una clase especifica el conjunto de variables de instancia (características propias de la clase) y métodos (funciones propias de la clase) que están agrupados para definir un tipo de objeto.

La clase **CArea** que nos permitirá generar objetos que cuenten con las siguientes variables de instancia del objeto (características que definen el objeto): nombre, largo, ancho, alto, área y volumen. Las variables se inicializan en el método *__init__* , dando un valor inicial a cada una de ellas. Mediante el parámetro *self._* seguido del identificador de cada una de las características del área, conseguimos acceder a la instancia del objeto al que pertenece el método.

- Nombre: identificado como **strN**
- Largo: identificado como **dL**
- Ancho: identificado como **dW**
- Alto: identificado como **dH**
- Coordenada X: nos permitirá conocer la posición de la esquina superior izquierda del área en el eje X respecto a la esquina superior izquierda de la ventana. Lo identificamos como **cX**
- Coordenada Y: nos permitirá conocer la posición de la esquina superior izquierda del área en el eje Y respecto a la esquina superior izquierda. Lo identificamos como **cY**
- Área: identificado como **dA**
- Volumen: identificado como **dV**

Todas ellas se crean con un valor inicial, siendo una cadena de caracteres vacía en el caso del nombre y el valor cero para las variables que representan dimensiones o coordenadas (Ilustración 21).

Otra clase que crearemos será **ABuffer** (Ilustración 22), básicamente es el contenedor en el cual introduciremos todas las áreas que vayamos creando a lo largo de cada simulación. Este contenedor se definirá como una lista. Además, definiremos dos funciones propias de esta clase que nos permitirán añadir nuevas áreas a la lista (*add*) y eliminar objetos de la lista (*pop*).

```python
class CArea:
    listLabels=["nombre","largo","ancho","alto","Coordenada X","Coordenada Y","area","volumen"]
    def __init__(self,
                 strN  = "",
                 dL    = 0,
                 dW    = 0,
                 dH    = 0,
                 cX    = 0,
                 cY    =0):
        self._strN = strN
        self._dL   = dL
        self._dW   = dW
        self._dH   = dH
        self._cX   =cX
        self._cY   =cY
        self._dA   = dL * dW
        self._dV   = dL * dW * dH
    def to_list(self): return [self._strN, self._dL, self._dW, self._dH, self._cX, self._cY, self._dA, self._dV]
```

Ilustración 21: Definición de la clase CArea en Python. Fuente: Elaboración propia.

```
class ABuffer:
        def __init__(self):
                self._listData = [ ]
        # Agregar un nuevo elemento.
        #
        def add(self, xObj): self._listData.append(xObj)
        #-----------------------------------------------------------------------

        # Extrae el elemento en la posición uI del contenedor.
        def pop(self, uI):
                if uI > len(self._listData): return None

                return self._listData.pop(uI)
        #-----------------------------------------------------------------------
#-------------------------------------------------------------------------------
```

Ilustración 22: Definición de la clase ABuffer en Python. Fuente: Elaboración propia.

De la misma forma, crearemos la clase **CPart** (Ilustración 23) para definir las piezas con sus variables de instancia: nombre, relleno, largo, ancho, alto, asignación, área y volumen.

- Nombre: identificado como **strN**
- Relleno: identificado como **dRelleno**
- Largo: identificado como **dL**
- Ancho: identificado como **dW**
- Alto: identificado como **dH**
- Asignación: identificado como **dAsig**
- Coordenada X: nos permitirá conocer la posición de la esquina superior izquierda de la pieza en el eje X respecto a la esquina superior izquierda de la ventana. Lo identificamos como **cX**
- Coordenada Y: nos permitirá conocer la posición de la esquina superior izquierda de la pieza en el eje Y respecto a la esquina superior izquierda. Lo identificamos como **cY**
- Área: identificado como **dA**
- Volumen: identificado como **dV**

En este caso, creamos un valor inicial para nombre que corresponde a una cadena de caracteres vacía, un valor *"False"* para la variable Booleana asignación (indicando que las piezas comienzan sin asignarse a ningún lote) y un valor cero para el resto de las variables que se corresponden con dimensiones o coordenadas.

Al igual que hemos creado el contenedor para las áreas, definimos **CBuffer** para listar todas las piezas (Ilustración 24). Contará con las mismas funciones internas que la clase **ABuffer** pero referidas a esta nueva clase.

```python
class CPart:
        listLabels = ["nombre","relleno","largo","ancho","alto","asignacion","Coordenada X","Coordenada Y","area","volumen"]

        def __init__(self,
                         strN    = "",
                        dRelleno = 0.0,
                         dL      = 0,
                         dW      = 0,
                         dH      = 0,
                         dAsig   = False,
                         cX      = 0,
                         cY      = 0):
                self._strN      = strN
                self._dRelleno  = dRelleno
                self._dL        = dL
                self._dW        = dW
                self._dH        = dH
                self._dAsig     = dAsig
                self._cX        =cX
                self._cY        =cY
                self._dA        = dL * dW
                self._dV        = dL * dW * dH
        #-----------------------------------------------------------------------------------------------

        def to_list(self): return [self._strN, self._dL, self._dW, self._dH, self._dAsig, self._dRelleno,self._cX,self._cY, self._dA, self._dV]
        #-----------------------------------------------------------------------------------------------
```

Ilustración 23: Definición de la clase CPart en Python. Fuente: Elaboración propia.

```
class CBuffer:
        def __init__(self):
                self._listData = [ ]

        # Agregar un nuevo elemento.
        #
        def add(self, xObj): self._listData.append(xObj)
        #---------------------------------------------------------------------

        # Extrae el elemento en la posición uI del contenedor.
        def pop(self, uI):
                if uI > len(self._listData): return None

                return self._listData.pop(uI)
        #---------------------------------------------------------------------

        def sort(self, dA):

                return self._listData.sort(dA)
```

Ilustración 24: Definición de la clase CBuffer en Python. Fuente: Elaboración propia.

Comenzamos con la definición de diferentes variables que usaremos a lo largo del programa.

En primer lugar, inicializamos la librería *tkinter* que habíamos importado al comienzo del programa con el comando **Tk().** Esto nos permitirá realizar la representación de las piezas.

Continuamos con la definición de **contador,** una variable la función de contador. Comenzará tomando el valor cero y la utilizaremos para guardar el valor de la simulación a la que nos lleguemos hasta realizar tantas simulaciones como el usuario desee realizar.

La variable **Max_Volumen_Mat** nos permitirá identificar aquel lote que cuente con un mayor uso de material. Siendo esta la variable que hará al fabricante decantarse por un lote u otro en la segunda etapa de la heurística (elección del ganador). La variable **Max_Porc_Ocup** servirá para guardar el porcentaje de ocupación del área de fabricación de aquel lote que cuenta con mayor uso de material.

La variable **Maxim_Porc_Ocup** servirá para identificar aquel lote que cuente con un mayor porcentaje de ocupación del área de fabricación. La variable **Maxim_Volumen_Mat** nos permitirá guardar el valor del uso de material de aquel lote que cuenta con mayor porcentaje de ocupación.

Ahora creamos dos objetos: **xBuffer** y **nArea.** Siendo **xBuffer** (objeto de la clase **ABuffer**) la lista en la que guardaremos todos lo áreas que se vayan creando a lo largo de la simulación y **nArea** un objeto de la clase **CArea**. Estos objetos contarán con las variables de instancia y funciones que hemos definido en las primeras líneas de código.

Todas las variables se inicializan con un número muy reducido de líneas de código, tal y como se muestra en la Ilustración 25.

```
# AQUÍ COMIEZA TODO EL CÓDIGO REFERENTE A LAS SIMULACIONES:
master = Tk()
contador=0
Max_Porc_Ocup=0
Max_Volumen_Mat=0
xBuffer = CBuffer()
nArea=CArea()
Maxim_Porc_Ocup=0
Maxim_Volumen_Mat=0
```

Ilustración 25: Inicialización de primeras variables del programa. Fuente: Elaboración propia.

3.3.3. Entradas del programa

A continuación, pedimos al usuario que introduzca los siguientes datos del área de fabricación: nombre, largo, ancho y alto (Ilustración 26).

```
print("Introduce las dimensiones del área de fabricación de la impresora 3d: "

nArea._strN=input("Introduce el nombre del área: ")
nArea._dL=int(input("Introduce el largo del área: "))
nArea._dW=int(input("Introduce el ancho del área: "))
nArea._dH=int(input("Introduce el alto del área: "))
```

Ilustración 26: Definición del área de fabricación en Python. Fuente: Elaboración propia.

Ahora es el turno de las piezas. En primer lugar, se pregunta al usuario como desea introducir los datos de las piezas que desea fabricar. Puede elegir entre introducir a mano cada una de las piezas o introducir los datos desde un archivo Excel. La opción que elija el usuario se guardará en una variable que llamaremos **Opcion.** El usuario tendrá que introducir 1 para teclear los datos o bien 2 para introducirlos desde el archivo Excel (Ilustración 27).

```
print("\nPara introducir las piezas, elija una opción: ")
print("1 - Datos a mano. ")
print("2 - A partir de archivo Excel. ")
Opcion=int(input("¿Cómo desea introducir los datos? "))
```

Ilustración 27: Definición de la variable Opcion en Python. Fuente: Elaboración propia.

En el primer caso, se pregunta por el número de piezas que el fabricante tiene como pedido. A continuación, se crea un bucle para pedir al usuario que introduzca el nombre, relleno, largo, ancho y alto de cada pieza. Para asignar estos valores a la pieza, se crea el objeto **nPieza** que contará con las variables de instancia y funciones definidas al principio del programa en la clase **CPart**.

El porcentaje de relleno se introducirá en tanto por uno. En caso de que el valor introducido por el usuario sea negativo o superior a 1, se repetirá la pregunta hasta que se introduzca un valor válido.

Se realizarán comprobaciones para cada una de las dimensiones que introduce el usuario. En caso de que se introduzcan números negativos o bien dimensiones que superen las dimensiones del espacio de fabricación, supondremos que el usuario se ha confundido y le

volveremos a pedir los datos. Las dimensiones se transformarán a un dato del tipo *float* (coma flotante), permitiendo al usuario introducir decimales en las dimensiones de las piezas.

Una vez que se han introducido todos los datos, se utiliza la función *add*, definida para la clase **ABuffer**, para añadir cada una de las piezas a la lista **xBuffer**.

Si el usuario ha decidido introducir todas las piezas mediante un archivo de Excel, el archivo deberá contener una hoja con la misma estructura que se muestra en la Ilustración 29.

En primer lugar, se pide el nombre del archivo y la hoja donde se encuentran los datos. Estos nombres se guardan en las variables **archivo** y **datos** respectivamente. Creamos cada una de las piezas como un objeto y asignamos los datos de cada fila a una nueva pieza. A continuación, se añade cada una de estas piezas a la lista de piezas **xBuffer** (Ilustración 30).

El siguiente paso, será definir el número de simulaciones, es decir, el número de posibles lotes que queremos generar. El usuario será el encargado de decidir el número de simulaciones que desea realizar. Cuanto mayor sea, mayores son las probabilidades de obtener una solución cercana al óptimo pero mayor es el tiempo empleado para alcanzar esta solución.

Como vamos a generar un gran número de lotes, vamos a establecer un primer filtro mediante el porcentaje de área ocupado. De esta forma, sólo guardaremos aquellos lotes que superen el porcentaje de área ocupado que indique el usuario. Este dato lo guardaremos en una nueva variable: **opc**. La definición de esta variable y el código que se explica en los siguientes párrafos de este apartado se muestran en la Ilustración 31.

Reiniciamos la lista de piezas, cambiando la variable booleana asignación a "*False*" para indicar que la pieza no está asignada. Esto es necesario porque al finalizar cada simulación, las piezas que se hayan asignado al lote de fabricación tendrán asignado el valor "*True*" en esta variable. Además, reiniciaremos los valores de las coordenadas de las piezas (**x._cX** y **x._cY**) para evitar problemas en la localización de las piezas sobra la cama de fabricación. La lista de áreas se actualiza, definiendo **yBuffer** (objeto de la clase **ABuffer**) como una lista en la cual no aparezca ningún área que se haya creado en la simulación anterior. Esta lista contendrá únicamente el área de fabricación introducido por el usuario. Se crea un nuevo objeto de la clase **CArea** y se le asignan los valores de las variables **nArea._strN, nArea._dL, nArea._dW** y **nArea._dH** que hemos guardado al inicio del programa (Ilustración 28). Los valores de **nArea._cX** y **nArea._cY** comienzan en 0, para definir el punto (0,0) del eje de coordenadas en el plano XY. Para adjuntar el área de fabricación se utiliza la función *add*, interna de la clase **ABuffer**, que ya hemos explicado antes.

Partiremos de una lista formada siempre por las mismas piezas, pero ordenadas de forma aleatoria para cada simulación. El orden de las piezas se modifica con la función *random.shuffle* aplicada sobre la variable **xBuffer._listData.**

Por último, cuando se inicia una simulación, se incrementa una unidad el valor del contador que definimos en las primeras líneas del programa.

```
if Opcion == 1:
    piezas=int(input("\n¿Cuantas piezas desea fabricar? "))
    for n in range(piezas):
        nPieza=CPart()
        print("Introduce los parámetros de la pieza número ",n+1,": \n")
        nPieza._strN=input("Introduce el nombre de la pieza: ")
        nPieza._dRelleno=float(input("Introduce el % de relleno de la pieza (0 será el 0% y 1 se corresponde al 100%): "))
        while nPieza._dRelleno>1 or nPieza._dRelleno<=0:
            print("Valor no válido.")
            nPieza._dRelleno=float(input("Introduce el % de relleno de la pieza (0 será el 0% y 1 se corresponde al 100%): "))

        nPieza._dL=float(input("Introduce el largo de la pieza: "))
        while nPieza._dL > nArea._dL and nPieza._dW or nPieza._dL <= 0:
            if nPieza._dL > nArea._dL or nPieza._dL > nArea._dW:
                print("El largo de la pieza supera las dimensiones del área de fabricación [",nArea._dL,",",nArea._dW,"].")
            else:
                print("El valor introducido no es válido.")

        nPieza._dL=float(input("Introduce un valor válido para el largo: "))

    nPieza._dW=float(input("Introduce el ancho de la pieza: "))
    while nPieza._dW>nArea._dW  and nPieza._dL or nPieza._dW<=0:
        if nPieza._dW>nArea._dW or nPieza._dW > nArea._dL:
            print("El ancho de la pieza supera las dimensiones del área de fabricación [",nArea._dL,",",nArea._dW,"]." )
        else:
            print("El valor introducido no es válido.")

        nPieza._dW=float(input("Introduce un valor válido para el ancho: "))

    nPieza._dH=float(input("Introduce el alto de la pieza: "))
    while nPieza._dH>nArea._dH or nPieza._dH<=0:
        if nPieza._dH>nArea._dH:
            print("El alto de la pieza supera el alto del área de fabricación(",nArea._dH,").")
        else:
            print("El valor introducido no es válido.")

        nPieza._dH=float(input("Introduce un valor válido para el alto: "))

    xBuffer.add(CPart(nPieza._strN,nPieza._dRelleno,nPieza._dL,nPieza._dW,nPieza._dH))
```

Ilustración 28: Definición de las propiedades de las piezas del pedido en Python. Fuente: Elaboración propia.

	A	B	C	D	E	F	G	H	I
1	Nombre	Relleno	Largo	Ancho	Altura	Area	Volumen		
2	P1	0,1	100	150	100	15000	1500000		
3	P2	0,2	100	150	100	15000	1500000		
4	P3	0,3	200	300	100	60000	6000000		
5	P4	0,5	100	250	100	25000	2500000		
6	P5	0,2	300	200	100	60000	6000000		
7	P6	0,3	100	150	100	15000	1500000		
8	P7	0,15	250	150	100	37500	3750000		
9									
10									
11									
12									
13									

Ilustración 29: Archivo de datos del pedido en Excel. Fuente: Elaboración propia.

```
else:
        print("Introduzca el nombre del archivo acabado en .xlsx (Ej: ejemplo.xlsx) ")
        archivo=input("Archivo: ")
        datos=input("Nombre de la hoja donde se encuentran los datos: ")
        workbook = load_workbook(archivo, read_only=False)
        pagina = workbook[datos]
        for Nombre, Relleno, Largo, Ancho, Altura, Area, Volumen in pagina.iter_rows(min_row=2):
                nPieza=CPart()
                nPieza._strN=Nombre.value
                nPieza._dRelleno=float(Relleno.value)
                nPieza._dL=Largo.value
                nPieza._dW=Ancho.value
                nPieza._dH=Altura.value
                nPieza._dA=Area.value
                nPieza._dV=Volumen.value
                xBuffer.add(CPart(nPieza._strN,nPieza._dRelleno,nPieza._dL,nPieza._dW,nPieza._dH))
```

Ilustración 30: Introducción de los datos del pedido mediante archivo Excel. Elaboración propia.

```
simulaciones=int(input("¿cuantas simulaciones quieres hacer?"))
print("¿Qué lotes quiere guardar?")
opc=int(input("Introduzca el % mínimo de área ocupado: "))

for contador in range(simulaciones):

        for x in xBuffer._listData:
                x._dAsig=False
                x._cX=0
                x._cY=0

        yBuffer= ABuffer()
        nArea._cX=0
        nArea._cY=0
        yBuffer.add(CArea(nArea._strN,nArea._dL,nArea._dW,nArea._dH,nArea._cX,nArea._cY))
        Num_Piezas=0
        random.shuffle(xBuffer._listData)
        contador=contador+1
```

Ilustración 31: Comienzo de una simulación en Python. Fuente: Elaboración propia.

3.3.4. Asignación de las piezas al lote da fabricación

Para cada simulación, se ejecutará un bucle que recorra todas las piezas de la lista y para cada una de ellas, recorrerá todas las áreas de la lista de áreas. Si la pieza estuviera ya

asignada, se pasaría a la siguiente y si no está asignada (variable booleana asignación en False), comprobamos que el ancho y el largo de la pieza es menor que el ancho del área correspondiente. Si no se cumple una de estas condiciones, rotamos la pieza, intercambiando los valores del ancho y largo. Para ello, se define la variable **NewL**, en la cual guardaremos el valor del largo de la pieza para poder realizar el cambio. Una vez intercambiados, se repite la comprobación de ancho y largo. Cuando se cumpla tanto la condición de ancho como la de largo, la pieza se asigna al área de fabricación y se realizan las siguientes operaciones:

- Creamos una primera área como un objeto de la clase **CArea** mediante la fórmula:
  $$A_{j+1} = (l_i, W_j - w_i)$$
  Donde l_i se corresponde con **i._dL**, W_j será **j._dW** y w_i aparece como **i._dW.**
  El valor del largo se guarda en la variable **NewLongArea1**, mientras que el valor del ancho se guarda en la variable **NewWidthArea1**.
- De la misma forma, creamos una segunda área mediante la fórmula:
  $$A_{j+2} = (L_j - l_i, W_j)$$
  Donde l_i se corresponde con **i._dL**, W_j será **j._dW** y L_j aparece como **j._dL.**
  El valor del largo se guarda en la variable **NewLongArea2**, mientras que el valor del ancho se guarda en la variable **NewWidthArea2**.
- Creamos dos nuevas variables para cada área, **NewCoordX1** y **NewCoordY1** para la primera área que creamos (A_{j+1}) y **NewCoordX2** y **NewCoordY2** para el segundo área (A_{j+2}). En estas variables guardaremos las coordenadas de cada nueva área que se cree a partir de las coordenadas del área al que se asigna la pieza (Ilustración 33) La coordenada de A_{j+1} en el eje X coincidirá con la coordenada X del área a partir del cual se ha creado. Mientras que la coordenada en el eje Y será la coordenada Y del área a partir del cual se ha creado más el ancho de la pieza asignada, tal y como se muestra en la Ilustración 33. La coordenada de A_{j+2} en el eje X será la suma de la coordenada en este eje del área a partir del cual se ha creado y el largo de la pieza asignada. Mientras que la coordenada en el eje Y coincidirá con la coordenada en el eje Y del área a partir del cual se ha creado, tal y como se muestra en la Ilustración 35.
- Se añaden las dos nuevas áreas a la lista **yBuffer**, mediante la función *add*.
- Como la pieza ha sido asignada al lote de producción, tenemos que actualizar la variable booleana asignación, representado por **i._dAsig,** y cambiar su valor a “True”.
- Las coordenadas de la pieza coincidirán con las que tenía el área donde se ha asignado, por lo tanto, igualamos el valor de **i._cX** y **i._cY** a **j._cX** y **j._cY** respectivamente.
- Por último, guardamos el índice del área en el cual hemos colocado la pieza en la variable **m.** Esto nos permite identificarla y eliminarla de la lista con la función **pop.**

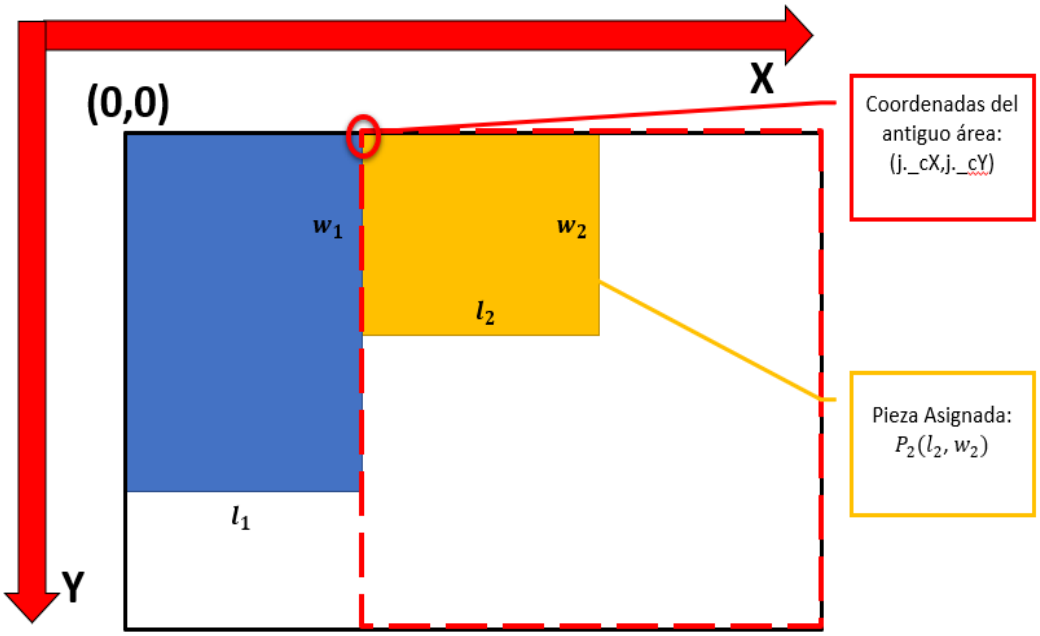


Ilustración 32: Coordenadas del área de fabricación Aj. Fuente: Elaboración propia.

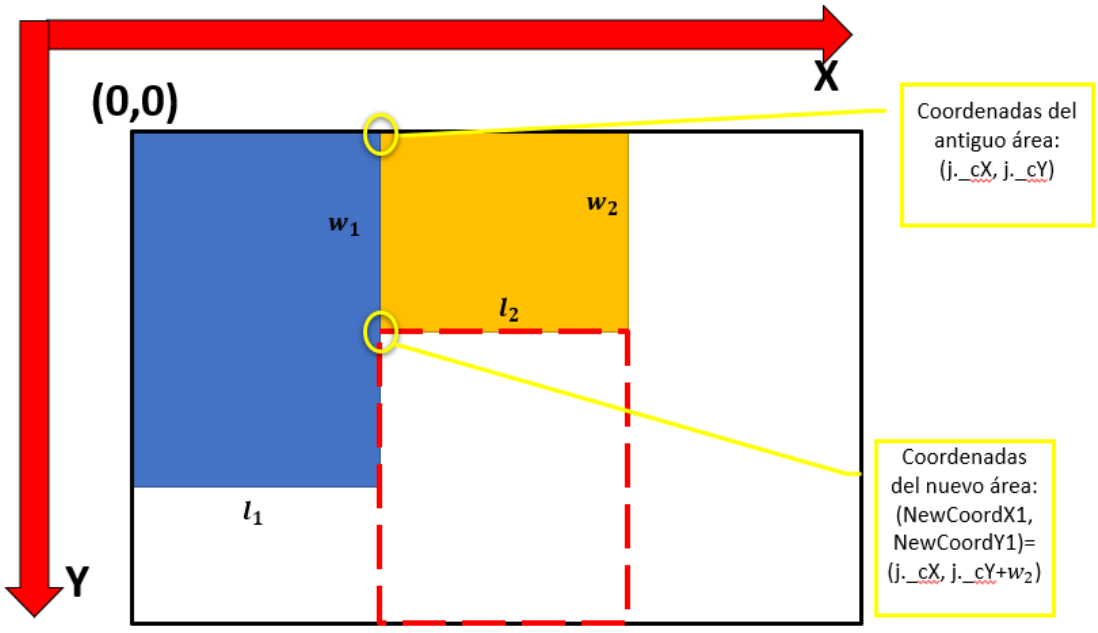


Ilustración 33: Coordenadas de la nueva área (A_{j+1}). Fuente: Elaboración propia.

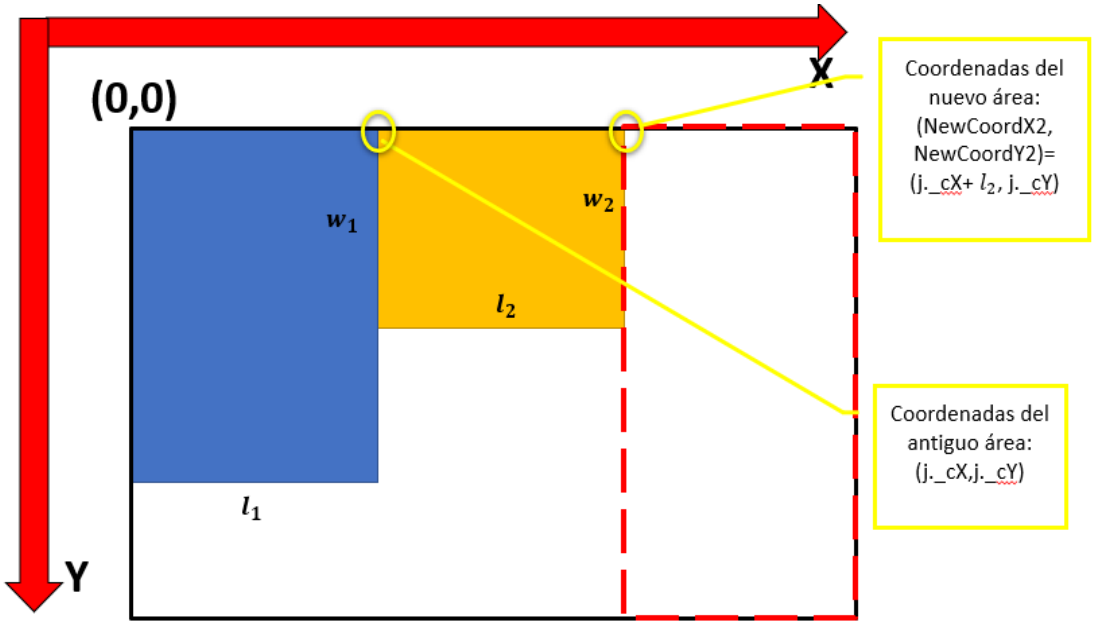
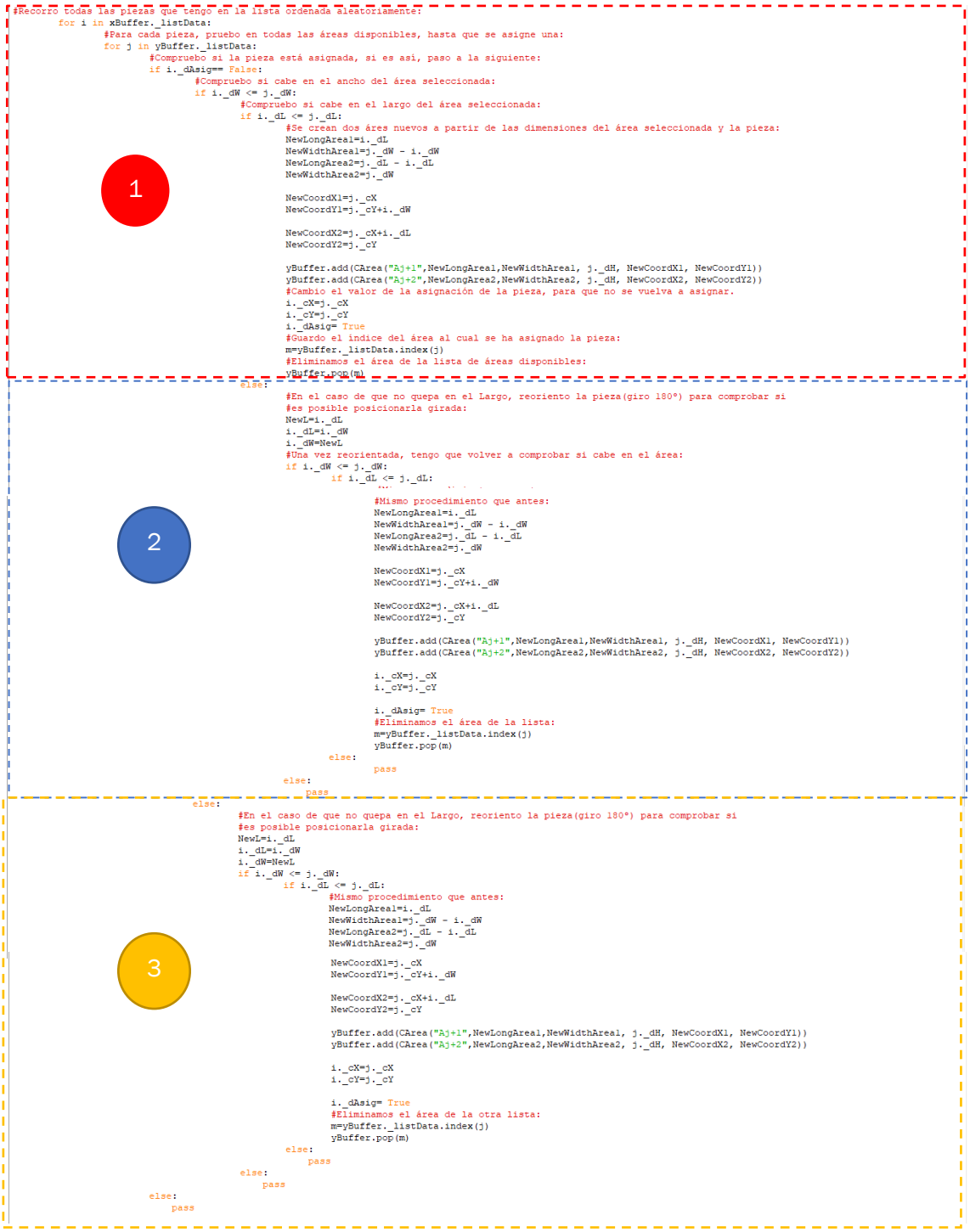


Ilustración 34: Coordenadas de la nueva área (A_{j+2}). Fuente: Elaboración propia.

En la Ilustración 35 se puede observar la indentación de los bucles que componen este núcleo central del programa. A continuación, se divide la imagen en tres secciones (Ilustración 36, Ilustración 37 e Ilustración 38) para poder observar con mayor detalle el código.




```
#Recorro todas las piezas que tengo en la lista ordenada aleatoriamente:
        for i in xBuffer._listData:
                #Para cada pieza, pruebo en todas las áreas disponibles, hasta que se asigne una:
                for j in yBuffer._listData:
                        #Compruebo si la pieza está asignada, si es así, paso a la siguiente:
                        if i._dAsig== False:
                                #Compruebo si cabe en el ancho del área seleccionada:
                                if i._dW <= j._dW:
                                        #Compruebo si cabe en el largo del área seleccionada:
                                        if i._dL <= j._dL:
                                                #Se crean dos áres nuevos a partir de las dimensiones del área seleccionada y la pieza:
                                                NewLongArea1=i._dL
                                                NewWidthArea1=j._dW - i._dW
                                                NewLongArea2=j._dL - i._dL
                                                NewWidthArea2=j._dW

                                                NewCoordX1=j._cX
                                                NewCoordY1=j._cY+i._dW

                                                NewCoordX2=j._cX+i._dL
                                                NewCoordY2=j._cY

                                                yBuffer.add(CArea("Aj+1",NewLongArea1,NewWidthArea1, j._dH, NewCoordX1, NewCoordY1))
                                                yBuffer.add(CArea("Aj+2",NewLongArea2,NewWidthArea2, j._dH, NewCoordX2, NewCoordY2))
                                                #Cambio el valor de la asignación de la pieza, para que no se vuelva a asignar.
                                                i._cX=j._cX
                                                i._cY=j._cY
                                                i._dAsig= True
                                                #Guardo el índice del área al cual se ha asignado la pieza:
                                                m=yBuffer._listData.index(j)
                                                #Eliminamos el área de la lista de áreas disponibles:
                                                yBuffer.pop(m)
                                        else:
                                                #En el caso de que no quepa en el Largo, reoriento la pieza(giro 180º) para comprobar si
                                                #es posible posicionarla girada:
                                                NewL=i._dL
                                                i._dL=i._dW
                                                i._dW=NewL
                                                #Una vez reorientada, tengo que volver a comprobar si cabe en el área:
                                                if i._dW <= j._dW:
                                                        if i._dL <= j._dL:

                                                                #Mismo procedimiento que antes:
                                                                NewLongArea1=i._dL
                                                                NewWidthArea1=j._dW - i._dW
                                                                NewLongArea2=j._dL - i._dL
                                                                NewWidthArea2=j._dW

                                                                NewCoordX1=j._cX
                                                                NewCoordY1=j._cY+i._dW

                                                                NewCoordX2=j._cX+i._dL
                                                                NewCoordY2=j._cY

                                                                yBuffer.add(CArea("Aj+1",NewLongArea1,NewWidthArea1, j._dH, NewCoordX1, NewCoordY1))
                                                                yBuffer.add(CArea("Aj+2",NewLongArea2,NewWidthArea2, j._dH, NewCoordX2, NewCoordY2))

                                                                i._cX=j._cX
                                                                i._cY=j._cY

                                                                i._dAsig= True
                                                                #Eliminamos el área de la lista:
                                                                m=yBuffer._listData.index(j)
                                                                yBuffer.pop(m)
                                                        else:
                                                                pass
                                                else:
                                                        pass
                                else:
                                        #En el caso de que no quepa en el Largo, reoriento la pieza(giro 180º) para comprobar si
                                        #es posible posicionarla girada:
                                        NewL=i._dL
                                        i._dL=i._dW
                                        i._dW=NewL
                                        if i._dW <= j._dW:
                                                if i._dL <= j._dL:
                                                        #Mismo procedimiento que antes:
                                                        NewLongArea1=i._dL
                                                        NewWidthArea1=j._dW - i._dW
                                                        NewLongArea2=j._dL - i._dL
                                                        NewWidthArea2=j._dW

                                                        NewCoordX1=j._cX
                                                        NewCoordY1=j._cY+i._dW

                                                        NewCoordX2=j._cX+i._dL
                                                        NewCoordY2=j._cY

                                                        yBuffer.add(CArea("Aj+1",NewLongArea1,NewWidthArea1, j._dH, NewCoordX1, NewCoordY1))
                                                        yBuffer.add(CArea("Aj+2",NewLongArea2,NewWidthArea2, j._dH, NewCoordX2, NewCoordY2))

                                                        i._cX=j._cX
                                                        i._cY=j._cY

                                                        i._dAsig= True
                                                        #Eliminamos el área de la otra lista:
                                                        m=yBuffer._listData.index(j)
                                                        yBuffer.pop(m)
                                                else:
                                                    pass
                                        else:
                                            pass
                        else:
                            pass
```

Ilustración 35: Núcleo central del programa en Python. Bucles de asignación de piezas a lotes. Fuente: Elaboración propia.

```
#Recorro todas las piezas que tengo en la lista ordenada aleatoriamente:
    for i in xBuffer._listData:
        #Para cada pieza, pruebo en todas las áreas disponibles, hasta que se asigne una:
        for j in yBuffer._listData:
            #Compruebo si la pieza está asignada, si es así, paso a la siguiente:
            if i._dAsig== False:
                #Compruebo si cabe en el ancho del área seleccionada:
                if i._dW <= j._dW:
                    #Compruebo si cabe en el largo del área seleccionada:
                    if i._dL <= j._dL:
                        #Se crean dos áres nuevos a partir de las dimensiones del área seleccionada y la pieza:
                        NewLongArea1=i._dL
                        NewWidthArea1=j._dW - i._dW
                        NewLongArea2=j._dL - i._dL
                        NewWidthArea2=j._dW

                        NewCoordX1=j._cX
                        NewCoordY1=j._cY+i._dW

                        NewCoordX2=j._cX+i._dL
                        NewCoordY2=j._cY

                        yBuffer.add(CArea("Aj+1",NewLongArea1,NewWidthArea1, j._dH, NewCoordX1, NewCoordY1))
                        yBuffer.add(CArea("Aj+2",NewLongArea2,NewWidthArea2, j._dH, NewCoordX2, NewCoordY2))
                        #Cambio el valor de la asignación de la pieza, para que no se vuelva a asignar.
                        i._cX=j._cX
                        i._cY=j._cY
                        i._dAsig= True
                        #Guardo el índice del área al cual se ha asignado la pieza:
                        m=yBuffer._listData.index(j)
                        #Eliminamos el área de la lista de áreas disponibles:
                        yBuffer.pop(m)
```

Ilustración 36: Sección 1 del núcleo central del programa de Python. Fuente: Elaboración propia.

```
else:
    #En el caso de que no quepa en el Largo, reoriento la pieza(giro 180°) para comprobar si
    #es posible posicionarla girada:
    NewL=i._dL
    i._dL=i._dW
    i._dW=NewL
    #Una vez reorientada, tengo que volver a comprobar si cabe en el área:
    if i._dW <= j._dW:
        if i._dL <= j._dL:
            #Mismo procedimiento que antes:
            NewLongArea1=i._dL
            NewWidthArea1=j._dW - i._dW
            NewLongArea2=j._dL - i._dL
            NewWidthArea2=j._dW

            NewCoordX1=j._cX
            NewCoordY1=j._cY+i._dW

            NewCoordX2=j._cX+i._dL
            NewCoordY2=j._cY

            yBuffer.add(CArea("Aj+1",NewLongArea1,NewWidthArea1, j._dH, NewCoordX1, NewCoordY1))
            yBuffer.add(CArea("Aj+2",NewLongArea2,NewWidthArea2, j._dH, NewCoordX2, NewCoordY2))

            i._cX=j._cX
            i._cY=j._cY

            i._dAsig= True
            #Eliminamos el área de la lista:
            m=yBuffer._listData.index(j)
            yBuffer.pop(m)
        else:
            pass
    else:
        pass
```

Ilustración 37: Sección 2 del núcleo central del programa de Python. Fuente: Elaboración propia.

```
else:
    #En el caso de que no quepa en el Largo, reoriento la pieza(giro 180°) para comprobar si
    #es posible posicionarla girada:
    NewL=i._dL
    i._dL=i._dW
    i._dW=NewL
    if i._dW <= j._dW:
        if i._dL <= j._dL:
            #Mismo procedimiento que antes:
            NewLongArea1=i._dL
            NewWidthArea1=j._dW - i._dW
            NewLongArea2=j._dL - i._dL
            NewWidthArea2=j._dW

            NewCoordX1=j._cX
            NewCoordY1=j._cY+i._dW

            NewCoordX2=j._cX+i._dL
            NewCoordY2=j._cY

            yBuffer.add(CArea("Aj+1",NewLongArea1,NewWidthArea1, j._dH, NewCoordX1, NewCoordY1))
            yBuffer.add(CArea("Aj+2",NewLongArea2,NewWidthArea2, j._dH, NewCoordX2, NewCoordY2))

            i._cX=j._cX
            i._cY=j._cY

            i._dAsig= True
            #Eliminamos el área de la otra lista:
            m=yBuffer._listData.index(j)
            yBuffer.pop(m)
        else:
            pass
    else:
        pass
else:
    pass
```

Ilustración 38: Sección 3 del núcleo central del programa de Python. Fuente: Elaboración propia.

Para cada simulación se acaba creando un lote de producción. Cada lote llevará asignado un valor de área ocupado que nos permitirá saber en qué medida estamos aprovechando el área de fabricación que tenemos disponible. Es una variable muy importante ya que, a mayor ocupación del área de fabricación, menor será el coste asociado a cada pieza que se fabrica en el lote. Definimos la variable **AreaOcupado** y la inicializamos con valor 0. Para cada simulación, realizamos un bucle que recorre la lista de piezas **xBuffer._listData** y cada vez que se encuentra con una pieza con la variable booleana en "True" suma el área de la pieza al valor de **AreaOcupado**.

Para calcular el área que está libre, definimos la variable **AreaLibre** y la inicializamos con valor 0. Recorremos con un bucle la lista de áreas disponibles y sumamos el valor de cada área individual a la variable **AreaLibre**. A continuación, se calcula el porcentaje de área ocupado, como el cociente entre el área ocupado y el área de fabricación que inicialmente estaba disponible.

Además, se calcula el volumen de material utilizado y se guarda en la variable **VolumenOcupado** que se inicializa con valor 0. De nuevo utilizamos el bucle que recorre la lista de piezas comprobando si está o no asignada al lote de fabricación para sumar el volumen de material de cada pieza cuando la variable booleana se encuentre en *"True"*. El volumen de material de cada pieza (m_i) se calcula con:

$$m_i = l_i \times w_i \times h_i \times r_i$$

En el programa de Python, l_i corresponde a **i._dL**, w_i se identifica con **i._dW**, h_i será la variable **i._dH** y por último $r_i$corresponde a **i._dRelleno.** El código referente al cálculo de **AreaOcupado, AreaLibre** y **VolumenOcupado** se muestra en la Ilustración 39.

En el comienzo del programa, utilizamos la variable **opc** para eliminar aquellos lotes que no cumplieron con un mínimo de porcentaje de ocupación. El valor guardado en esta variable lo comparamos con **Porc_Ocup** y sólo guardamos aquellos lotes que cumplan esta condición. Los lotes se guardan en un archivo Excel mediante el uso de la función *append* aplicada a la variable **hoja** (Ilustración 40) que hemos creado al inicio. Esta función nos permite escribir datos en una fila como si fuera una lista. En nuestro caso, la utilizaremos para escribir en Excel lo siguiente:

- Número de la simulación
- Los datos de cada pieza: nombre, relleno, largo, ancho, alto, si ha sido asignada, coordenada X, coordenada Y, área, volumen y cantidad de material.
- Los datos de cada simulación: porcentaje de área ocupado y cantidad de material total.
- El número de piezas que compone cada lote.

Toda la programación explicada hasta ahora corresponde a la resolución de la primera etapa que hemos denominado problema de empaquetado. Una vez que se han finalizado las simulaciones, podemos pasar a la segunda etapa, la elección del ganador. Para elegir un lote entre todos los que se han simulado, tendremos en cuenta la cantidad de material que hemos empleado en la fabricación del lote.

En primer lugar, hemos definido la variable **Max_Volumen_Mat** (se inicia con valor 0) para guardar el mayor valor de la variable **VolumenOcupado** que aparezca entre todas las simulaciones. En cuanto un lote obtenga un valor de esta variable superior al guardado en **Max_Volumen_Mat**, se ejecutarán las siguientes operaciones:

- Se guarda su valor de área ocupado en una nueva variable **Max_Porc_Ocup.**
- Se crea **MaxBuffer** como un objeto de la clase **CBuffer**. Nos servirá para guardar las piezas asignadas al lote con mayor volumen de material asociado.
- Se crea **NoAsign** como un objeto de la clase **Cbuffer**. Nos servirá para guardar las piezas que no se han asignado al lote de mayor volumen de material asociado.
- Se crea la variable **Sim_Max** y se igual al valor del contador para guardar el nº de la simulación.

El código referente a estas operaciones se muestra en la Ilustración 41.

```
#Calculo el área que he ocupado con las piezas asignadas a este lote, y su cantidad de material:
AreaOcupado=0
VolumenOcupado=0.0
for i in xBuffer._listData:
       if i._dAsig==True:
                Num_Piezas=Num_Piezas+1
                AreaOcupado=AreaOcupado+(i._dL*i._dW)
                VolumenOcupado=VolumenOcupado+(i._dL*i._dW*i._dH*i._dRelleno)

#Calculo el área que me queda libre tras asignar todas las piezas del lote:
AreaLibre=0

for j in yBuffer._listData:
        Areas=j._dL*j._dW
        AreaLibre=AreaLibre+Areas

Porc_Ocup=((AreaOcupado/(AreaLibre+AreaOcupado))*100)
```

Ilustración 39: Cálculo del área ocupado y volumen de material del lote en Python. Fuente: Elaboración propia.

La otra variable de interés para el fabricante es el porcentaje de área ocupado. Para facilitar que el fabricante realice una comparación rápida entre aquel lote con mayor porcentaje de ocupación del área de fabricación y el lote con mayor cantidad de material empleado, se guardarán los datos del lote con mayor porcentaje de área ocupado.

En primer lugar, hemos definido la variable **Maxim_Porc_Ocup** (se inicia con valor 0) para guardar el mayor valor de la variable **Porc_Ocup** que aparezca entre todos los lotes. En cuanto un lote obtenga un valor de esta variable superior al guardado en **Maxim_Porc_Ocup**, se ejecutarán las siguientes operaciones (Ilustración 42):

- Se guarda su valor de área ocupado en una nueva variable **Maxim_Porc_Ocup.**
- Se crea **Max_P_Buffer** como un objeto de la clase **CBuffer**. Nos servirá para guardar las piezas asignadas al lote con mayor porcentaje de área ocupadoa.
- Se crea **NoAsign_P** como un objeto de la clase **Cbuffer**. Nos servirá para guardar las piezas que no se han asignado a este lote.
- Se crea la variable **S_Max** y se igual al valor del contador para guardar el nº de la simulación.

```
#Escribo en el archivo en Excel los datos de Porcentje de ocupación y cantidad de material de esta simulación:

if Porc_Ocup > opc:
        hoja.append([" ","SIMULACIÓN Nº ",contador])
        for i in xBuffer._listData:
                hoja.append([i._strN,i._dRelleno,i._dL,i._dW,i._dH,i._dAsig,i._cX,i._cY,i._dA,i._dV,i._dL*i._dW*i._dH*i._dRelleno])

        hoja.append(["Porcentaje ocupación:",Porc_Ocup,"Cantidad de material:",VolumenOcupado])
        hoja.append(["Número de piezas asignadas al lote: ",Num_Piezas])
```

Ilustración 40: Filtro de porcentaje de área ocupado en Python. Fuente: Elaboración propia.

```
#Lote con mayor cantidad de material empleado:

if VolumenOcupado > Max_Volumen_Mat:
        Max_Porc_Ocup=Porc_Ocup
        Max_Volumen_Mat=VolumenOcupado
        MaxBuffer = CBuffer()
        NoAsign = CBuffer()
        #Guardo el valor de la simulación en la cual tenemos mayor cantidad de material utilizada:
        Sim_Max=contador
        #Guardo las piezas asignadas al lote con mayor cantidad de material utilizado y su orden:
        for i in xBuffer._listData:
            if i._dAsig==True:
                MaxBuffer.add(CPart(i._strN,i._dRelleno,i._dL,i._dW,i._dH,i._dAsig,i._cX,i._cY))
            else:
                NoAsign.add(CPart(i._strN,i._dRelleno,i._dL,i._dW,i._dH,i._dAsig,i._cX,i._cY))
```

Ilustración 41: Lote de mayor cantidad de material empleado. Fuente: Elaboración propia.

```
#Lote con mayor porcentaje de área ocupado:
if Porc_Ocup>Maxim_Porc_Ocup:
    Maxim_Porc_Ocup=Porc_Ocup
    Maxim_Volumen_Mat=VolumenOcupado
    Max_P_Buffer = CBuffer()
    NoAsign_P = CBuffer()
    #Guardo el valor de la simulación en la cual tenemos mayor porcentaje de área ocupado:
    S_Max=contador
    #Guardo las piezas asignadas al lote y su orden:
    for i in xBuffer._listData:
        if i._dAsig==True:
            Max_P_Buffer.add(CPart(i._strN,i._dRelleno,i._dL,i._dW,i._dH,i._dAsig,i._cX,i._cY))
        else:
            NoAsign_P.add(CPart(i._strN,i._dRelleno,i._dL,i._dW,i._dH,i._dAsig,i._cX,i._cY))
```

Ilustración 42: Lote con mayor porcentaje de área ocupado. Fuente: Elaboración propia.

3.3.5. Salidas del programa

Una vez que hemos terminado de resolver los dos subproblemas, solo tenemos que mostrar por pantalla el lote que ha sido seleccionado para fabricar en la primera orden de producción. Se muestran las piezas asignadas al lote y aquellas que no han sido asignadas, la simulación en la cual se ha obtenido el lote óptimo (**Sim_Max**), el porcentaje de área disponible que abarcamos (**Max_Porc_Ocup**) y el volumen de material asociado al lote (**Max_Volumen_Mat**). Este código se muestra en la Ilustración 43.

Por último, vamos a realizar la representación de la colocación de todas las piezas asignadas al lote ganador. En primer lugar, le pedimos al usuario que introduzca un valor de redimensionado. Este valor lo guardaremos en la variable **k** y nos permitirá realizar una representación acorde a la resolución de la pantalla de cada ordenador. Por ejemplo, si mi área de fabricación es de 2000x2000 y mi pantalla cuenta con una resolución de 1000x1000, el factor de redimensionado será 0,5. Con valores superiores, el tamaño de la ventana será superior a nuestra resolución.

Mediante la variable **w**, creamos la ventana con el uso de la librería *Tkinter* que inicializamos con el comando *master= Tk()* al inicio del programa.

Para dibujar el área de fabricación utilizamos la función *create_rectangle*, perteneciente a la librería *tkinter*. Como parámetros le pasamos la coordenada de la esquina superior izquierda (0,0) y las dimensiones que estén guardadas en las variables **nArea._dL** y **nArea._dW** que servirán como coordenada de la esquina inferior derecha. Estos valores los multiplicamos por el factor de redimensionado guardado en la variable **k**.

Se crea una lista de colores mediante la variable **col** y le añadimos algunos colores como azul, rojo y naranja, entre otros. Ordenamos de forma aleatoria la lista y le asignamos a la variable **a** el primer color de la lista. A continuación, sacamos por pantalla el nombre de cada pieza, su posición y sus dimensiones.

Dibujamos las piezas utilizando de nuevo la función **create_rectangle**. Como parámetros le pasaremos las coordenadas de la esquina superior izquierda, guardadas en las variables **i._cX, i._cY** y las coordenadas de la esquina inferior derecha cuyo valor coincidirá con la suma de esta coordenada inicial y el valor del largo o ancho respectivamente (i._cX+i._dL) y (i._cY+i._dW).

Para identificar las piezas, se coloca su nombre en el centro de cada una de ellas mediante el comando **create_text** y la posición del centro de cada pieza.

Todo el código referente a la generación de la imagen se muestra en la Ilustración 44.

```
print("\n\n  ----------------------  FIN DEL PROGRAMA  ----------------------  ")
print("\nEn la simulación número",Sim_Max," se encontró la solución con mayor utilización de material. ")
print("\nEl porcentaje de ocupación es: ",Max_Porc_Ocup,"%")
print("Cuenta con un volumen de material utilizado de: ",Max_Volumen_Mat," mm^3")
print("Está asociado al lote: ")
print("\n[", end="")
for i in MaxBuffer._listData:
    print(i._strN, end=" ")
print("]")
print("\n")
print("Piezas sin asignar: ")
print("[", end="")
for i in NoAsign._listData:
    print(i._strN, end=" ")
print("]")
```

Ilustración 43: Lote ganador. Fuente: Elaboración propia.

```
k=float(input("Factor redimensionado: "))
w = Canvas(master, width = nArea._dL*k, height = nArea._dW*k)
master.title("Distribución de las piezas")
w.pack()

#Creamos un rectangulo de las dimensiones del área de fabricación:
w.create_rectangle(0,0,nArea._dL*k, nArea._dW*k, fill="White")
#Creamos una lista de colores para rellenar los rectángulos para diferenciar las piezas:
col=["blue","red","orange","green","pink","yellow","brown","purple","grey"]

print("\n")

for i in Max_P_Buffer._listData:
    #Cada pieza llevará asignado un color de forma aleatoria:
    random.shuffle(col)
    a=col[0]
    #Sacamos por pantalla la lista de piezas con sus coordenadas, sus dimensiones y el color
    print("Pieza ---- Coord X ---- Largo ---- Ancho ---- Color")
    print(i._strN," ------- ",i._cX," ------ ", i._cY," ---- ",i._dL," ---- ",i._dW," ---- ",a,end="\n ")

    #Creamos cada rectángulo y le asignamos el nombre de la pieza:
    w.create_rectangle(i._cX*k,i._cY*k,i._cX*k+i._dL*k, i._cY*k+i._dW*k, fill=a)
    w.create_text(i._cX*k+((i._dL*k)/2),i._cY*k+((i._dW*k)/2),text=i._strN)
```

Ilustración 44: Representación de las piezas del lote ganador mediante librería de Python Tkinter. Fuente: Elaboración propia.

Guardamos los datos del lote ganador en la hoja que se haya guardado en la variable **sheet** de nuestro archivo Excel. Para ello, recorremos la lista **MaxBuffer._listData** y mediante la función "*append*" vamos escribiendo fila a fila los datos de cada pieza que ha sido asignada al lote. A continuación, hacemos lo mismo con las piezas que no han sido asignadas al lote. Escribimos el porcentaje de ocupación (**Max_Porc_Ocup**) y la cantidad de material empleada (**Max_Volumen_Mat**), y la simulación en la que ha aparecido (**Sim_Max**).

Por último, guardamos los datos del lote con mayor porcentaje de área ocupado en la misma hoja. Para ello, recorremos la lista **Max_P_Buffer._listData** y mediante la función "*append*" vamos escribiendo fila a fila los datos de cada pieza que ha sido asignada al lote. A continuación, hacemos lo mismo con las piezas que no han sido asignadas al lote. Escribimos el porcentaje de ocupación (**Maxim_Porc_Ocup**) y la cantidad de material empleada (**Maxim_Volumen_Mat**), y la simulación en la que ha aparecido (**S_Max**) y guardamos el archivo Excel de la variable **FILE_PATH** mediante la función "*sabe*".

Estas últimas líneas de código se pueden observar en la Ilustración 45.

```
#Ahora vamos a guardar los resultados de las mejores simulaciones en un Excel:

sheet.append(["Lote con mayor cantidad de material empleado: "])
for i in MaxBuffer._listData:
        sheet.append([i._strN,i._dRelleno,i._dL,i._dW,i._dH,i._dAsig,i._cX,i._cY,i._dA,i._dV])

for i in NoAsign._listData:
        sheet.append([i._strN,i._dRelleno,i._dL,i._dW,i._dH,i._dAsig,i._cX,i._cY,i._dA,i._dV])

sheet.append(["Porcentaje ocupación:",Max_Porc_Ocup,"Cantidad de material:",Max_Volumen_Mat])
sheet.append(["Simulación nº: ",Sim_Max])

sheet.append(["Lote con mayor porcentaje área ocupado: "])
for i in Max_P_Buffer._listData:
        sheet.append([i._strN,i._dRelleno,i._dL,i._dW,i._dH,i._dAsig,i._cX,i._cY,i._dA,i._dV])

for i in NoAsign_P._listData:
        sheet.append([i._strN,i._dRelleno,i._dL,i._dW,i._dH,i._dAsig,i._cX,i._cY,i._dA,i._dV])

sheet.append(["Porcentaje ocupación:",Maxim_Porc_Ocup,"Cantidad de material:",Maxim_Volumen_Mat])
sheet.append(["Simulación nº: ",S_Max])

wb.save(FILE_PATH)
```

Ilustración 45: Guardado de soluciones en Excel mediante Python. Fuente: Elaboración propia.

3.4. Ejecución del programa en Python

Una vez explicada la estructura interna del programa, podemos proceder a ejecutarlo. Para interactuar con Python, contamos con el intérprete interactivo conocido como *Shell*. Una vez ejecutamos el programa escrito desde el IDLE, aparece la pantalla del *Shell* donde introduciremos los datos de nuestro problema. En primer lugar, nos pedirá el nombre del archivo Excel donde guardaremos los lotes generados, la solución con mayor porcentaje de área ocupado y aquella con mayor cantidad de material empleado (Ilustración 46).

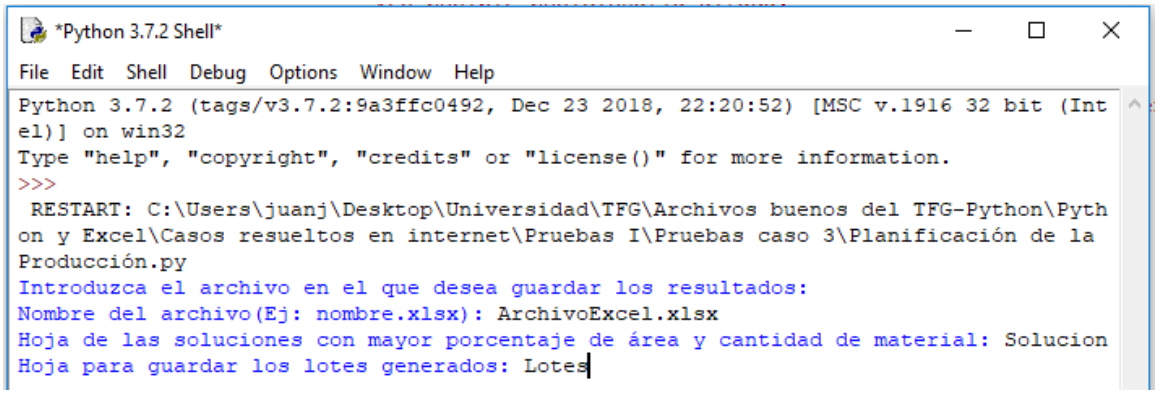


Ilustración 46: Primeras entradas del Shell de Python. Fuente: Elaboración propia.

A continuación, debemos introducir los datos de nuestra impresora 3d: un identificador, el largo, el ancho y el alto (Ilustración 47).

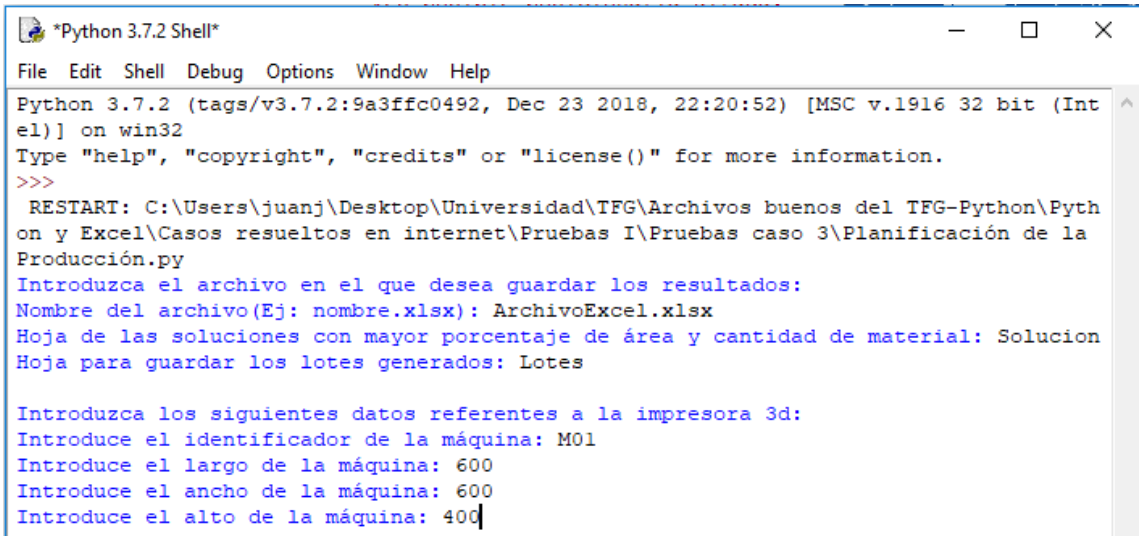


Ilustración 47: Entradas de la máquina de impresión 3d. Fuente: Elaboración propia.

Una vez introduzcamos tanto el identificador como las dimensiones del espacio de fabricación (largo, ancho y alto), nos preguntará cómo deseamos introducir los datos de las piezas (Ilustración 48). Si deseamos hacerlo de forma manual o bien mediante un archivo en Excel.

En el caso de elegir la primera opción, tendremos que indicar cuantas piezas tiene el pedido.

Para cada una de las piezas nos pedirá su identificador, el porcentaje de relleno (en tanto por uno) y sus dimensiones (largo, ancho y alto). Esto se puede observar en la Ilustración 49.

Si elegimos la segunda opción, tendremos que introducir el nombre del archivo y la hoja que guarda los datos de las piezas (Ilustración 50). El archivo de Excel puede coincidir con el archivo donde deseamos guardar tanto las mejores soluciones como el resto de los lotes.

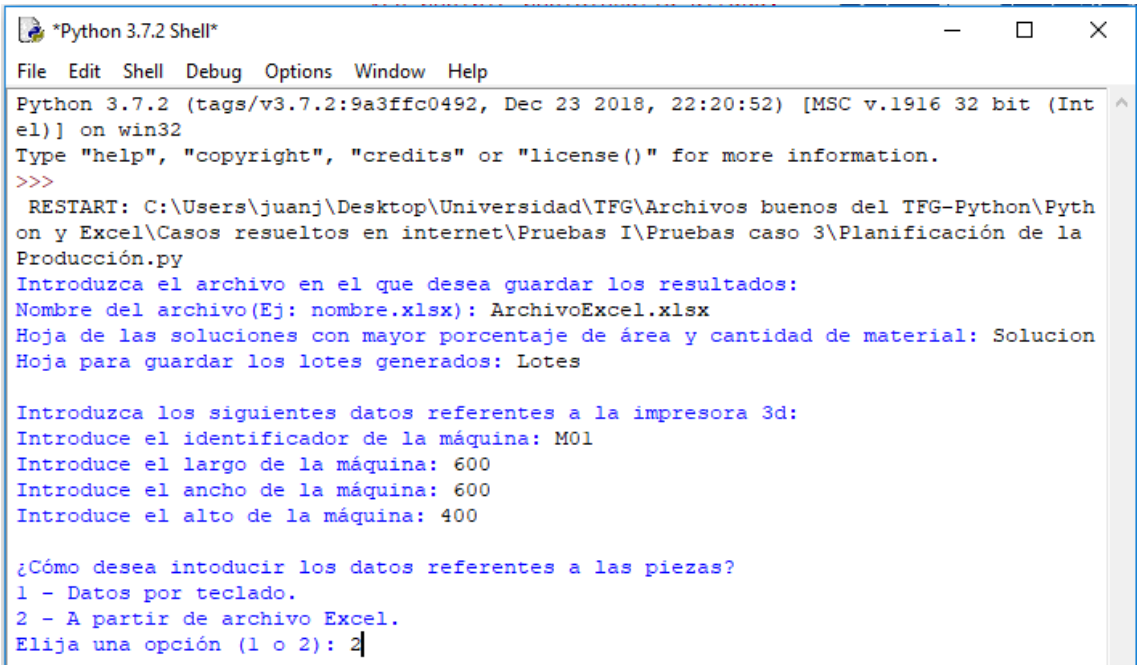


Ilustración 48: Selección del método de introducción de los datos de las piezas. Fuente: Elaboración propia.

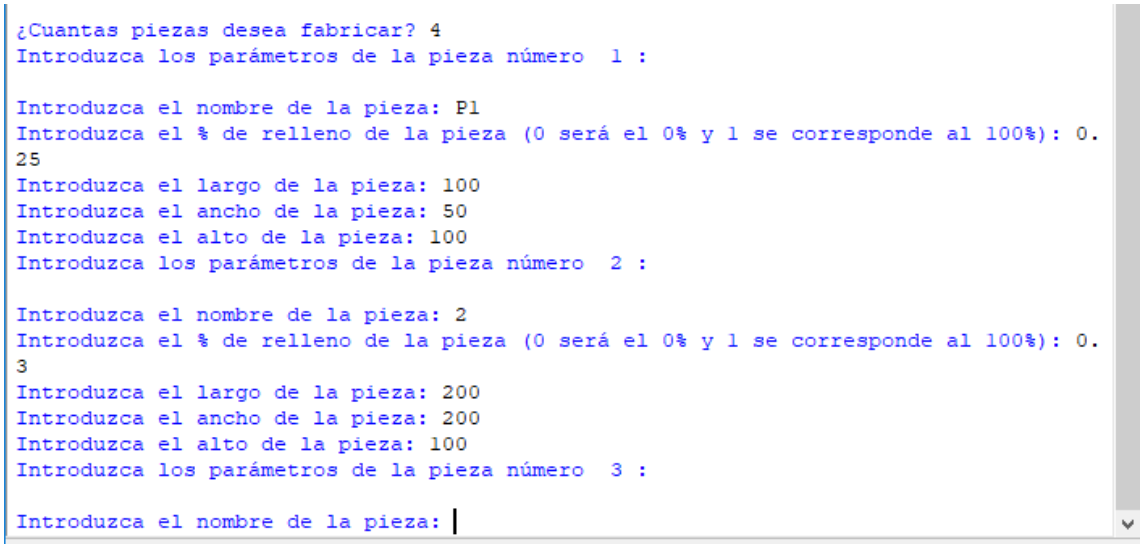


Ilustración 49: Introducción de los datos de las piezas por teclado. Fuente: Elaboración propia.

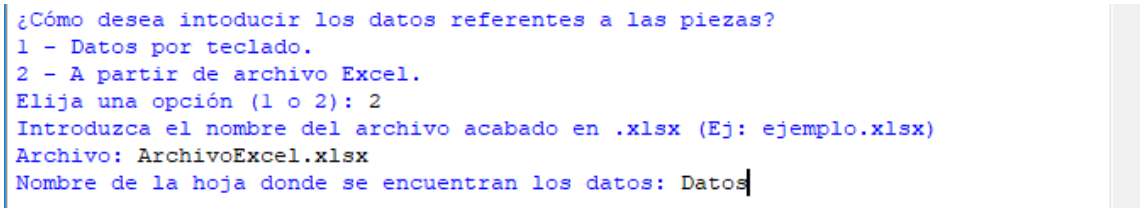


Ilustración 50: Introducción de los datos de las piezas mediante un archivo Excel. Fuente: Elaboración propia.

Para que los datos se lean de forma correcta, es necesario que las columnas del archivo Excel tengan los mismos nombres que en la siguiente imagen. En caso contrario, la lectura no será correcta y tendremos que volver a ejecutar el programa. Los datos deben estar estructurados por columnas tal y como se muestra en le Ilustración 51.

	A	B	C	D	E	F	G
1	Nombre	Relleno	Largo	Ancho	Altura	Area	Volumen
2	P1	0,3	200	200	200	40000	8000000
3	P2	0,3	200	200	200	40000	8000000
4	P3	0,3	200	200	200	40000	8000000
5	P4	0,35	150	200	200	30000	6000000
6	P5	0,35	150	200	200	30000	6000000
7	P6	0,35	150	200	200	30000	6000000
8	P7	0,35	150	200	200	30000	6000000
9	P8	0,25	400	200	200	80000	16000000
10	P9	0,2	100	200	200	20000	4000000
11	P10	0,2	100	200	200	20000	4000000
12	P11	0,3	300	100	200	30000	6000000
13	P12	0,3	300	100	200	30000	6000000
14	P13	0,3	250	250	200	62500	12500000

Ilustración 51: Archivo Excel con los datos de las piezas. Fuente: Elaboración propia.

Cuando hayamos terminado de introducir todas las piezas, tendremos que decidir el número de simulaciones que deseamos realizar (Ilustración 52). Cuanto mayor sea el número de piezas, mayor será el número de posibles combinaciones y, por lo tanto, mayor el número de simulaciones para poder obtener un resultado cercano al óptimo. Por otro lado, cuantas más simulaciones realicemos, mayor tiempo de ejecución. Buscaremos un balance entre las simulaciones y el tiempo.

Con el fin de evitar que se guarden lotes que no sean relevantes, utilizaremos un filtro de porcentaje de ocupación. De esta forma, sólo guardaremos aquellos lotes que abarquen un mínimo del área total disponible. El porcentaje se introducirá en tanto por ciento. Es decir, 0 implica un 0% y 100 el 100%.

```
¿Cómo desea intoducir los datos referentes a las piezas?
1 - Datos por teclado.
2 - A partir de archivo Excel.
Elija una opción (1 o 2): 2
Introduzca el nombre del archivo acabado en .xlsx (Ej: ejemplo.xlsx)
Archivo: ArchivoExcel.xlsx
Nombre de la hoja donde se encuentran los datos: Datos
¿Cuantas simulaciones desa realizar?10000
¿Qué lotes quiere guardar?
Introduzca el % mínimo de área ocupado: 90
```

Ilustración 52: Filtro de los lotes. Fuente: Elaboración propia.

Una vez ejecutamos el programa, debemos esperar mientras se realizan los cálculos programados y la asignación de las piezas a los diferentes lotes. Una vez finalizada la ejecución, por pantalla aparecerá el lote con mayor utilización de material, las piezas asignadas y aquellas que se han quedado fuera del lote (Ilustración 53).

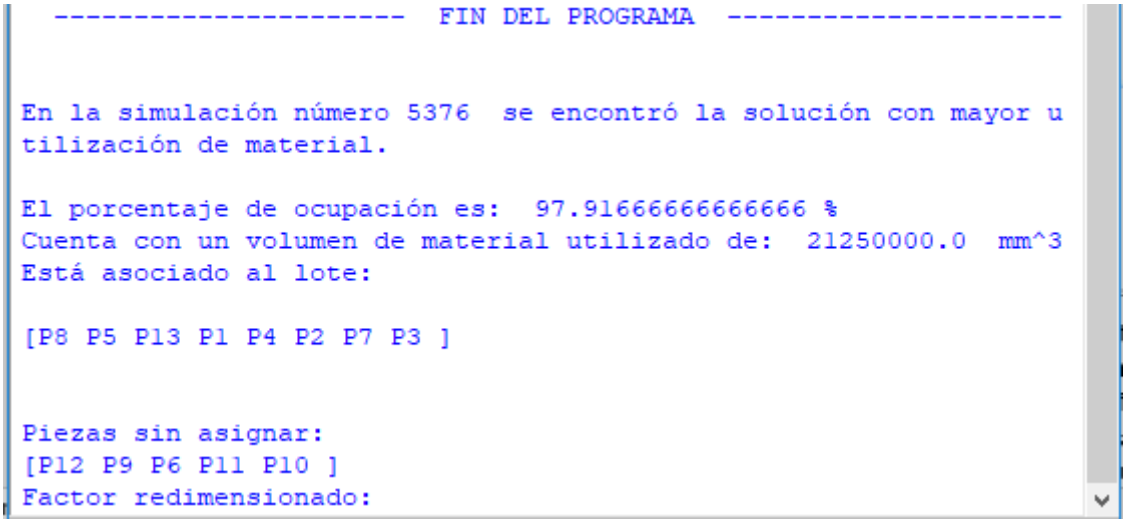


Ilustración 53: Lote ganador, su porcentaje de ocupación y la cantidad de material empleado. Fuente: Elaboración propia.

Por último, tendremos que introducir como queremos ver la representación por pantalla. Para ello introduciremos un factor de redimensionado. La imagen de la distribución de las piezas sobre la plataforma de fabricación aparecerá con una resolución del tamaño del área de fabricación de la impresora 3d. Si introducimos un área de dimensiones 600x600, la imagen contará con una resolución de 600x600. Cada ordenador cuenta con una pantalla con diferente resolución, por lo que será necesario redimensionar la imagen que se genera. Por ejemplo, si nuestra resolución es 1200x1200, el factor de redimensionado será como mucho 2. Si introducimos un valor superior, la imagen será más grande que la pantalla y no nos permitirá ver la distribución completa. Una vez introducido este factor, se mostrará una tabla con los datos de las piezas del lote, su posición (Coordenadas X e Y), sus dimensiones y el color asignado, tal y como se muestra en el ejemplo de la Ilustración 54.

Además, se abre una ventana nueva con la representación de las piezas del lote con mayor cantidad de material empleado (Ilustración 55). Este sería el lote que deberíamos producir en el caso de buscar generar el mayor beneficio posible optimizando la capacidad productiva de nuestra impresora 3d.

Las salidas del programa se guardan en el archivo Excel que hayamos indicado al inicio. En nuestro caso, hemos estructurado el archivo Excel en tres hojas. En la primera hoja, llamada “Solución”, tenemos el lote con mayor porcentaje de ocupación (Ilustración 56) y el lote con mayor cantidad de material asignado (Ilustración 57). Las piezas que están asignadas a estos lotes tienen en la columna Asignación un fondo verde y aparece la palabra VERDADERO. Aquellas que no están asignadas, tienen en esta misma columna un fondo rojo y aparece la palabra FALSO. Además, tenemos información tanto del porcentaje de área ocupado como de la cantidad de material empleada en cada lote.

```
Pieza ---- Coord X ---- Coord Y ---- Largo ---- Ancho ---- Color
P8  -------  0  ------  0  ----  400  ----  200  ----  orange
 Pieza ---- Coord X ---- Coord Y ---- Largo ---- Ancho ---- Color
P5  -------  0  ------  200  ----  150  ----  200  ----  grey
 Pieza ---- Coord X ---- Coord Y ---- Largo ---- Ancho ---- Color
P13  -------  150  ------  200  ----  250  ----  250  ----  pink
 Pieza ---- Coord X ---- Coord Y ---- Largo ---- Ancho ---- Color
P1  -------  400  ------  0  ----  200  ----  200  ----  blue
 Pieza ---- Coord X ---- Coord Y ---- Largo ---- Ancho ---- Color
P4  -------  0  ------  400  ----  150  ----  200  ----  brown
 Pieza ---- Coord X ---- Coord Y ---- Largo ---- Ancho ---- Color
P2  -------  400  ------  200  ----  200  ----  200  ----  pink
 Pieza ---- Coord X ---- Coord Y ---- Largo ---- Ancho ---- Color
P7  -------  150  ------  450  ----  200  ----  150  ----  grey
 Pieza ---- Coord X ---- Coord Y ---- Largo ---- Ancho ---- Color
P3  -------  400  ------  400  ----  200  ----  200  ----  yellow

>>>
```

Ilustración 54: Salida final del programa. Fuente: Elaboración propia.

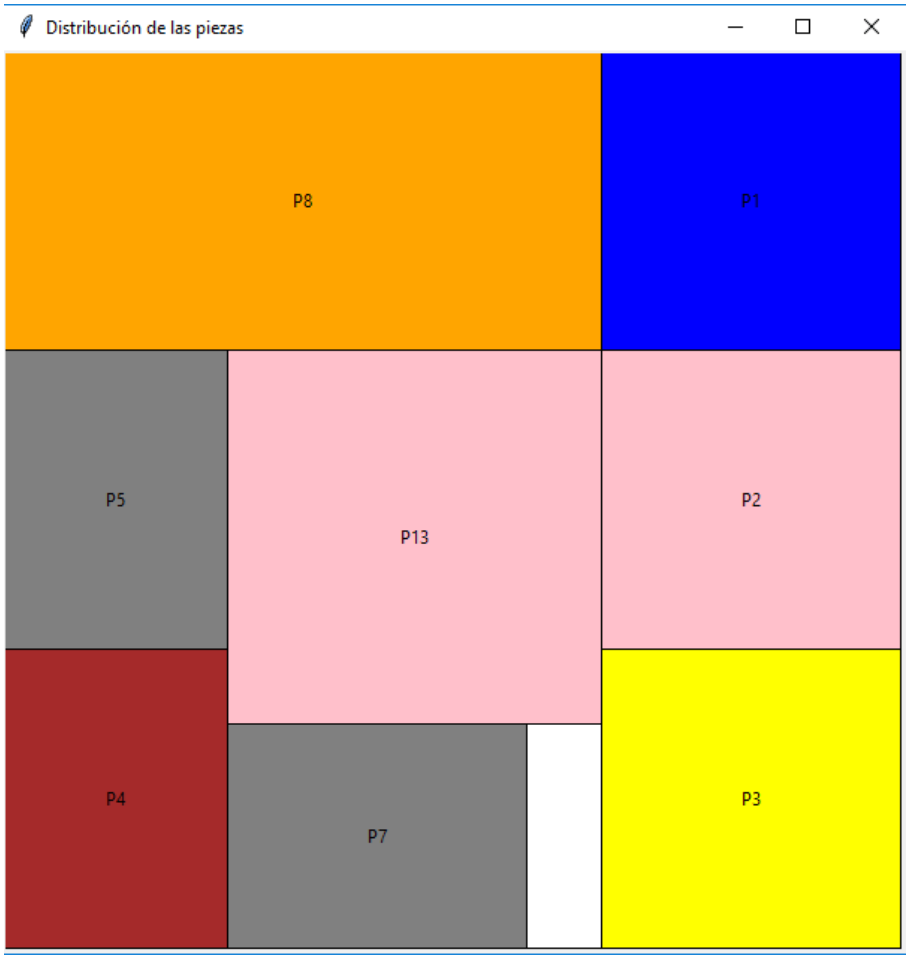


Ilustración 55: Representación de la distribución de las piezas asignadas al lote con mayor cantidad de material empleado. Fuente: Elaboración propia.

	Identificador	% Relleno	Largo	Ancho	Alto	Asignación	Coordenada X	Coordenada Y	Área	Volumen
2	Lote con mayor cantidad de material empleado:									
3	P8	0,25	400	200	200	VERDADERO	0	0	80000	16000000
4	P5	0,35	150	200	200	VERDADERO	0	200	30000	6000000
5	P13	0,3	250	250	200	VERDADERO	150	200	62500	12500000
6	P1	0,3	200	200	200	VERDADERO	400	0	40000	8000000
7	P4	0,35	150	200	200	VERDADERO	0	400	30000	6000000
8	P2	0,3	200	200	200	VERDADERO	400	200	40000	8000000
9	P7	0,35	200	150	200	VERDADERO	150	450	30000	6000000
10	P3	0,3	200	200	200	VERDADERO	400	400	40000	8000000
11	P12	0,3	300	100	200	FALSO	0	0	30000	6000000
12	P9	0,2	200	100	200	FALSO	0	0	20000	4000000
13	P6	0,35	200	150	200	FALSO	0	0	30000	6000000
14	P11	0,3	100	300	200	FALSO	0	0	30000	6000000
15	P10	0,2	200	100	200	FALSO	0	0	20000	4000000
16	Porcentaje ocupación:	97,91666667	Cantidad de material:	21250000						
17	Simulación nº:	5376								

Ilustración 56: Lote guardado en Excel con mayor cantidad de material empleado. Fuente: Elaboración propia.

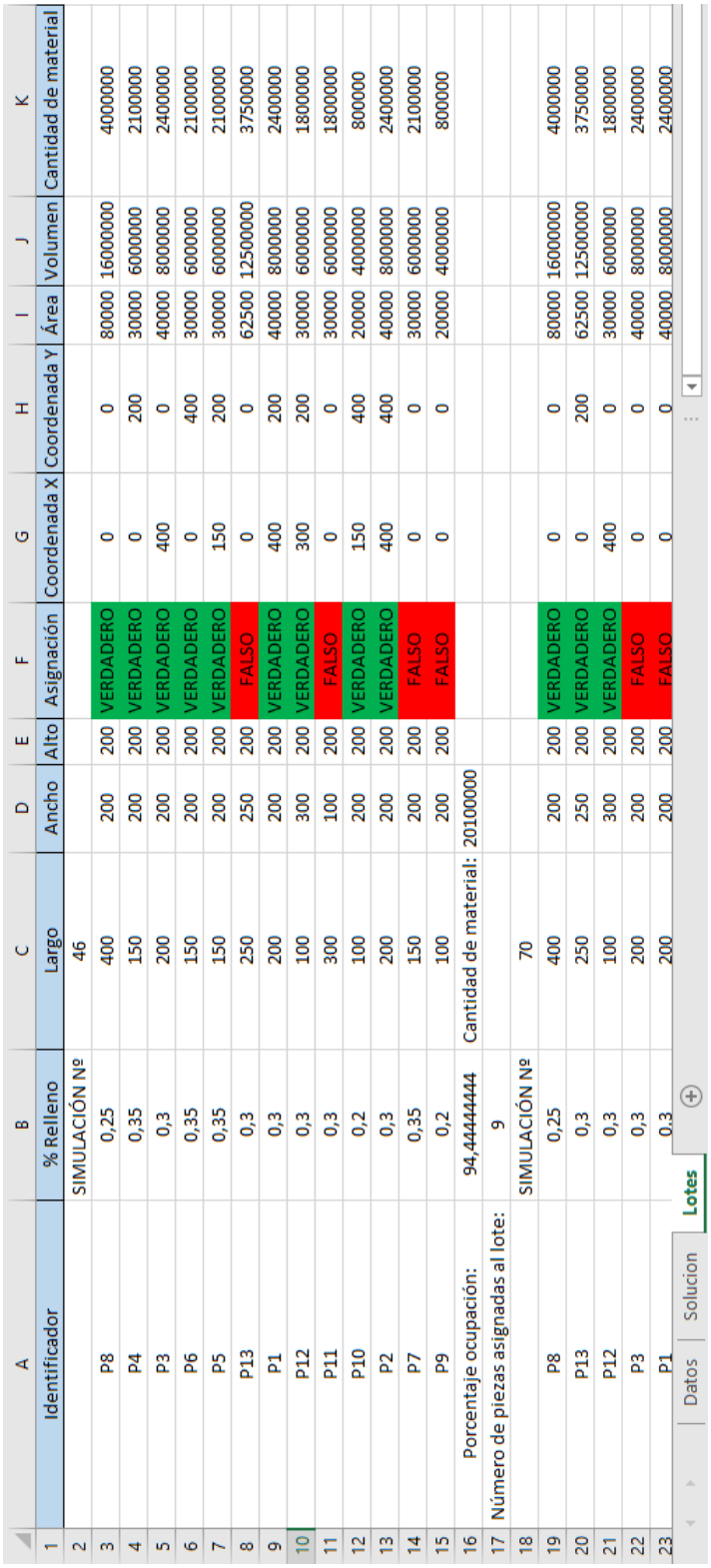
	A	B	C	D	E	F	G	H	I	J
1	Identificador	% Relleno	Largo	Ancho	Alto	Asignación	Coordenada X	Coordenada Y	Área	Volumen
18	Lote con mayor porcentaje área ocupado:									
19	P8	0,25	400	200	200	VERDADERO	0	0	80000	16000000
20	P5	0,35	150	200	200	VERDADERO	0	200	30000	6000000
21	P3	0,3	200	200	200	VERDADERO	400	0	40000	8000000
22	P7	0,35	150	200	200	VERDADERO	0	400	30000	6000000
23	P10	0,2	100	200	200	VERDADERO	150	200	20000	4000000
24	P2	0,3	200	200	200	VERDADERO	400	200	40000	8000000
25	P1	0,3	200	200	200	VERDADERO	400	400	40000	8000000
26	P4	0,35	150	200	200	VERDADERO	250	200	30000	6000000
27	P9	0,2	100	200	200	VERDADERO	150	400	20000	4000000
28	P6	0,35	150	200	200	VERDADERO	250	400	30000	6000000
29	P11	0,3	300	100	200	FALSO	0	0	30000	6000000
30	P12	0,3	300	100	200	FALSO	0	0	30000	6000000
31	P13	0,3	250	250	200	FALSO	0	0	62500	12500000
32	Porcentaje ocupación:	100	Cantidad de material:	21200000						
33	Simulación nº:	4223								

Ilustración 57: Lote con mayor porcentaje de ocupación guardado en Excel. Fuente: Elaboración propia.

En la segunda hoja, llamada “Lotes” en el ejemplo que se muestra en la Ilustración 58, podemos observar los resultados de cada uno de los lotes que superan el filtro de porcentaje de área ocupado. De la misma forma que en la primera hoja, contamos con los datos de cada pieza y además la cantidad de material que se utiliza para fabricar cada una de ellas.

Con el fin de facilitar la identificación de aquellos lotes con mayor porcentaje de ocupación del área disponible y mayor cantidad de material empleado, se resaltan con un fondo azul los 10 mejores valores de cada columna (Ilustración 59).

Por último, se ha creado una tercera hoja más estructurada. En ella podemos observar todos los lotes y ordenarlos en función del porcentaje de ocupación, de la cantidad de material o el número de piezas asignadas al lote. En esta hoja, podemos detectar de un simple vistazo que piezas están asignadas a cada lote. Tal y como se puede observar en la Ilustración 60, debajo de cada tipo de pieza aparece un 1 en fondo verde en los casos que la pieza está asignada al lote y un cero con fondo blanco en aquellas piezas que no están asignadas.



	A	B	C	D	E	F	G	H	I	J	K
1	Identificador	% Relleno	Largo	Ancho	Alto	Asignación	Coordenada X	Coordenada Y	Área	Volumen	Cantidad de material
2		SIMULACIÓN Nº	46								
3	P8	0,25	400	200	200	VERDADERO	0	0	80000	16000000	4000000
4	P4	0,35	150	200	200	VERDADERO	0	200	30000	6000000	2100000
5	P3	0,3	200	200	200	VERDADERO	400	0	40000	8000000	2400000
6	P6	0,35	150	200	200	VERDADERO	0	400	30000	6000000	2100000
7	P5	0,35	150	200	200	VERDADERO	150	200	30000	6000000	2100000
8	P13	0,3	250	250	200	FALSO	0	0	62500	12500000	3750000
9	P1	0,3	200	200	200	VERDADERO	400	200	40000	8000000	2400000
10	P12	0,3	100	300	200	VERDADERO	300	200	30000	6000000	1800000
11	P11	0,3	300	100	200	FALSO	0	0	30000	6000000	1800000
12	P10	0,2	100	200	200	VERDADERO	150	400	20000	4000000	800000
13	P2	0,3	200	200	200	VERDADERO	400	400	40000	8000000	2400000
14	P7	0,35	150	200	200	FALSO	0	0	30000	6000000	2100000
15	P9	0,2	100	200	200	FALSO	0	0	20000	4000000	800000
16	Porcentaje ocupación:	94,44444444	Cantidad de material:	20100000							
17	Número de piezas asignadas al lote:	9									
18		SIMULACIÓN Nº	70								
19	P8	0,25	400	200	200	VERDADERO	0	0	80000	16000000	4000000
20	P13	0,3	250	250	200	VERDADERO	0	200	62500	12500000	3750000
21	P12	0,3	100	300	200	VERDADERO	400	0	30000	6000000	1800000
22	P3	0,3	200	200	200	FALSO	0	0	40000	8000000	2400000
23	P1	0,3	200	200	200	FALSO	0	0	40000	8000000	2400000

Datos | Solucion | Lotes

Ilustración 58: Lotes guardados en Excel tras la ejecución del programa. Fuente: Elaboración propia.

	A	B	C	D	E	F	G	H	I	J	K
1	Identificador	% Relleno	Largo	Ancho	Alto	Asignación	Coordenada X	Coordenada Y	Área	Volumen	Cantidad de material
4546		SIMULACIÓN Nº	5376								
4547	P8	0,25	400	200	200	VERDADERO	0	0	80000	16000000	4000000
4548	P5	0,35	150	200	200	VERDADERO	0	200	30000	6000000	2100000
4549	P13	0,3	250	250	200	VERDADERO	150	200	62500	12500000	3750000
4550	P1	0,3	200	200	200	VERDADERO	400	0	40000	8000000	2400000
4551	P4	0,35	150	200	200	VERDADERO	0	400	30000	6000000	2100000
4552	P2	0,3	200	200	200	VERDADERO	400	200	40000	8000000	2400000
4553	P7	0,35	200	150	200	VERDADERO	150	450	30000	6000000	2100000
4554	P12	0,3	300	100	200	FALSO	0	0	30000	6000000	1800000
4555	P3	0,3	200	200	200	VERDADERO	400	400	40000	8000000	2400000
4556	P9	0,2	200	100	200	FALSO	0	0	20000	4000000	800000
4557	P6	0,35	200	150	200	FALSO	0	0	30000	6000000	2100000
4558	P11	0,3	100	300	200	FALSO	0	0	30000	6000000	1800000
4559	P10	0,2	200	100	200	FALSO	0	0	20000	4000000	800000
4560	Porcentaje ocupación:	97,91666667	Cantidad de material:	21250000							
4561	Número de piezas asignadas al lote:	8									

Ilustración 59: Lotes resaltados en archivo Excel. Fuente: Elaboración propia.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	Num piezas:	13															
2	Lote	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	p13	Porcentaje ocupación:	Cantidad material	Nº Piezas
3	1	1	1	1	1	1	1	0	1	0	1	0	1	0	94,44444444	20100000	9
4	2	0	0	0	1	0	1	1	1	1	1	1	1	1	92,36111111	19250000	9
5	3	1	0	1	1	1	1	1	0	1	1	1	0	1	92,36111111	20350000	10
6	4	0	1	0	1	1	1	1	1	1	1	1	1	0	94,44444444	20000000	10
7	5	1	1	0	1	1	1	1	0	1	1	1	0	1	92,36111111	20350000	10
8	6	1	1	1	0	0	0	1	1	1	1	1	1	0	91,66666667	18500000	9
9	7	1	1	1	1	1	1	1	0	1	1	0	0	1	95,13888889	20950000	10
10	8	0	0	1	1	1	1	0	1	1	1	0	1	1	95,13888889	19850000	9
11	9	1	1	1	1	0	1	1	1	1	1	0	0	0	91,66666667	19100000	9
12	10	0	0	1	1	1	1	1	1	1	1	1	1	0	94,44444444	20000000	10
13	11	1	1	1	0	0	1	0	1	1	1	1	1	0	91,66666667	18500000	9
14	12	1	1	1	0	0	0	1	1	1	1	1	1	0	91,66666667	18500000	9
15	13	1	1	1	0	0	1	0	1	1	1	1	1	0	91,66666667	18500000	9
16	14	0	0	0	1	0	1	1	1	1	1	1	1	1	92,36111111	19250000	9
17	15	1	1	1	1	1	1	1	1	0	1	0	0	0	94,44444444	20400000	9
18	16	1	1	1	1	0	1	0	1	1	1	1	0	0	91,66666667	18800000	9
19	17	1	0	1	1	1	1	1	0	1	1	1	0	1	92,36111111	20350000	10
20	18	0	1	0	1	1	1	1	1	1	1	1	1	0	94,44444444	20000000	10
21	19	0	1	0	1	1	1	1	1	0	0	0	1	1	92,36111111	20350000	8
22	20	1	1	1	1	1	1	1	0	1	1	1	1	0	94,44444444	20800000	11

Ilustración 60: Lotes estructurados en el archivo Excel. Fuente: Elaboración propia.

4. Resultados

El objetivo de este apartado es probar la eficacia y la eficiencia de nuestro algoritmo. Se realizarán diferentes casos de estudio tanto del problema de empaquetado como de la etapa de elección del ganador.

4.1. Problema de empaquetado

La asignación de las piezas al lote de fabricación comienza por una primera etapa que hemos denominado "problema de empaquetado". Este problema ha sido estudiado durante años y en diferentes artículos podemos encontrar casos de estudio que nos permitirán comparar la eficacia y la eficiencia de sus procedimientos con nuestra propuesta. En concreto, haremos uso de los casos presentados en (Mirledy Toro and Granada Echeverri, 2007) y (Cui, 2005).

Además, presentaremos nuevos casos de estudio para evaluar el desempeño de nuestro programa ante diferentes situaciones que podrían aparecer en la vida real.

4.1.1. Caso 1: Cambio en el orden de las piezas

El procedimiento de asignación de piezas que presentamos crea nuevas áreas según se van asignando piezas al lote de fabricación. Por lo tanto, hay una gran relación entre la pieza que se asigna y las dimensiones de las nuevas áreas. Esto nos plantea una primera pregunta, ¿obtendremos mejores soluciones ordenando la lista inicial de mayor a menor área o de menor a mayor área?

Para dar una respuesta planteamos un primer caso con 15 piezas de diferentes dimensiones. A continuación, se muestra la tabla 5 con las propiedades de las 15 piezas, sin un orden especial.

Nombre	Relleno	Largo	Ancho	Altura	Área	Volumen
P1	0,3	200	200	200	40000	8000000
P2	0,3	200	200	200	40000	8000000
P3	0,3	200	200	200	40000	8000000
P4	0,35	150	200	200	30000	6000000
P5	0,35	150	200	200	30000	6000000
P6	0,35	150	200	200	30000	6000000
P7	0,35	150	200	200	30000	6000000
P8	0,25	400	200	200	80000	16000000
P9	0,2	100	200	200	20000	4000000
P10	0,2	100	200	200	20000	4000000
P11	0,2	100	200	200	20000	4000000
P12	0,2	300	150	200	45000	9000000
P13	0,2	100	50	200	5000	1000000
P14	0,2	50	100	200	5000	1000000
P15	0,2	250	250	200	62500	12500000

Tabla 5: Piezas del caso 1. Fuente: Elaboración propia.

Trataremos de distribuir estas piezas en un área de 600x600. Es decir, no todas las piezas van a caber en el área y comprobaremos si obtenemos mejores soluciones comenzando a asignar piezas con mayor área en primer lugar o en orden inverso.

En la tabla 6 mostramos las piezas ordenadas de mayor a menor área.

Nombre	Relleno	Largo	Ancho	Altura	Área	Volumen
P8	0,25	400	200	200	80000	16000000
P15	0,2	250	250	200	62500	12500000
P12	0,2	300	150	200	45000	9000000
P1	0,3	200	200	200	40000	8000000
P2	0,3	200	200	200	40000	8000000
P3	0,3	200	200	200	40000	8000000
P4	0,35	150	200	200	30000	6000000
P5	0,35	150	200	200	30000	6000000
P6	0,35	150	200	200	30000	6000000
P7	0,35	150	200	200	30000	6000000
P9	0,2	100	200	200	20000	4000000
P10	0,2	100	200	200	20000	4000000
P11	0,2	100	200	200	20000	4000000
P13	0,2	100	50	200	5000	1000000
P14	0,2	50	100	200	5000	1000000

Tabla 6: Piezas del caso 1 ordenadas de mayor a menor área. Fuente: Elaboración propia.

A continuación, procedemos a ejecutar el programa y mostramos los resultados obtenidos. En la tabla 7 tenemos un 1 con fondo verde debajo de la casilla de las piezas que están asignadas al área de fabricación. Contamos con un 88,19 % del área de fabricación ocupado mediante la colocación de las piezas P4, P5, P6, P7, P8, P12, P13, P14 y P15 tal y como se puede observar en la Ilustración 61.

P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	% Ocupación	Nº Piezas
0	0	0	1	1	1	1	1	0	0	0	1	1	1	1	88,1944	9

Tabla 7: Lote de piezas asignadas de mayor a menor área. Fuente: Elaboración propia.

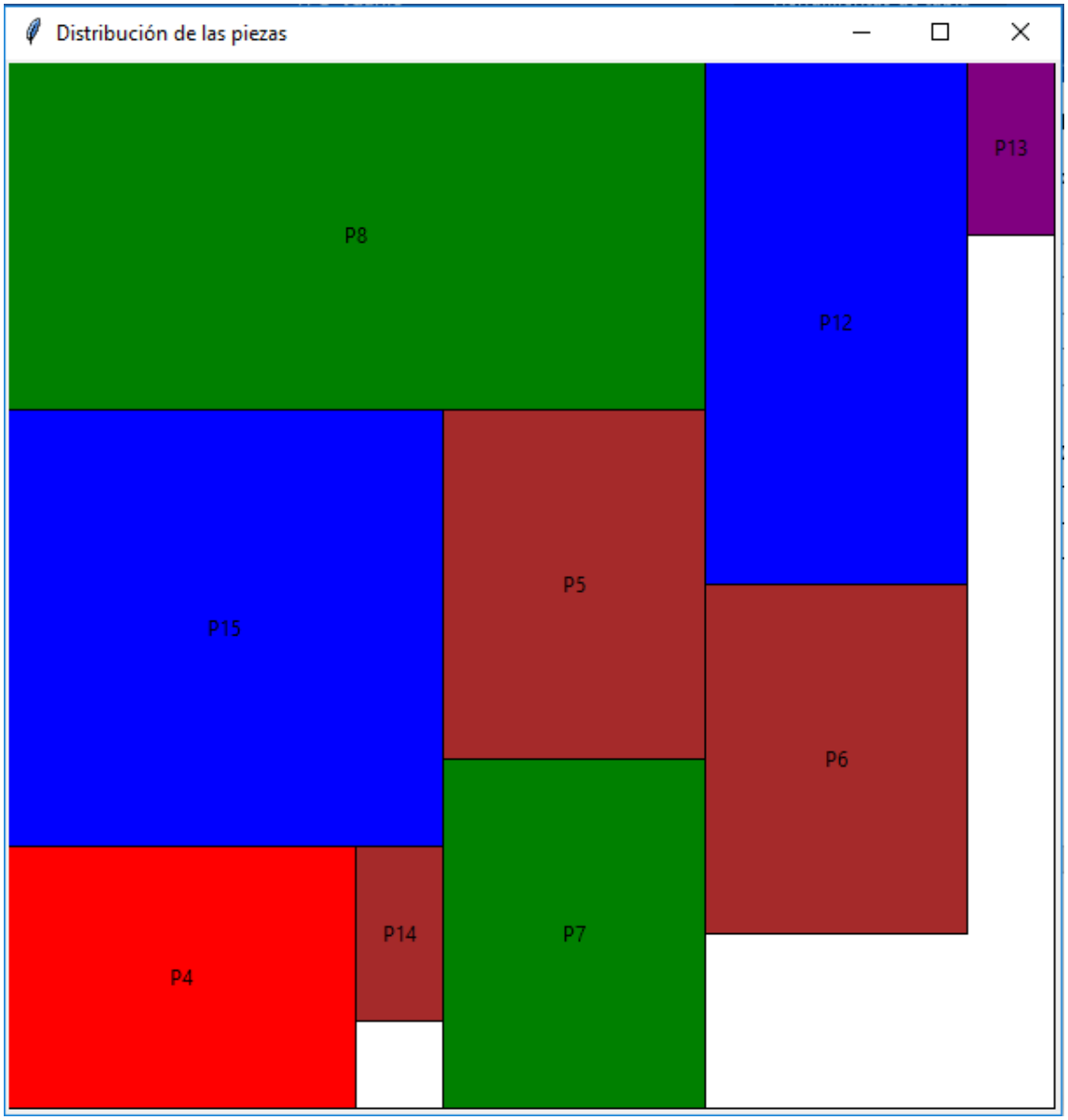


Ilustración 61: Distribución de piezas del lote ordenado de mayor a menor área. Fuente: Elaboración propia.

Ordenamos las piezas de menor a mayor área (Tabla 8) y procedemos a ejecutar el programa para poder comparar los resultados obtenidos.

La tabla 9 muestra el resultado de las piezas asignadas al área de fabricación para este segundo caso. Contamos con un 44,44 % del área de fabricación ocupado mediante la colocación de las piezas P4, P5, P6, P9, P10, P11, P13 y P14, tal y como se puede observar en la Ilustración 62.

Nombre	Relleno	Largo	Ancho	Altura	Área	Volumen
P13	0,2	100	50	200	5000	1000000
P14	0,2	50	100	200	5000	1000000
P9	0,2	100	200	200	20000	4000000

P10	0,2	100	200	200	20000	4000000
P11	0,2	100	200	200	20000	4000000
P4	0,35	150	200	200	30000	6000000
P5	0,35	150	200	200	30000	6000000
P6	0,35	150	200	200	30000	6000000
P7	0,35	150	200	200	30000	6000000
P1	0,3	200	200	200	40000	8000000
P2	0,3	200	200	200	40000	8000000
P3	0,3	200	200	200	40000	8000000
P12	0,2	300	150	200	45000	9000000
P15	0,2	250	250	200	62500	12500000
P8	0,25	400	200	200	80000	16000000

Tabla 8: Piezas del caso 1 ordenadas de menor a mayor área. Fuente: Elaboración propia.

P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	% ocupación	Nº Piezas
0	0	0	1	1	1	0	0	1	1	1	0	1	1	0	44,4444	8

Tabla 9: Lote de piezas asignadas de menor a mayor área. Fuente: Elaboración propia.

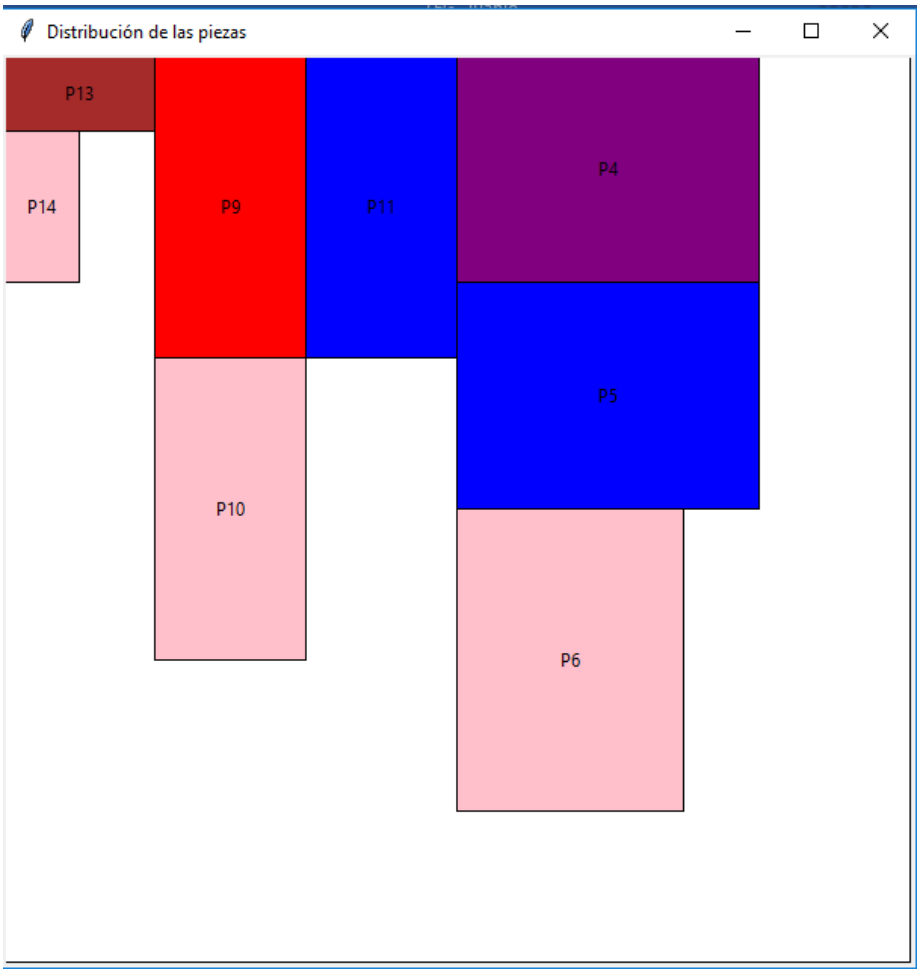


Ilustración 62: Distribución de piezas del lote ordenado de menor a mayor área. Fuente: Elaboración propia.

Salta a la vista que el programa se comporta mucho mejor realizando la asignación de piezas de mayor a menor área que en el sentido contrario. En el primer caso contábamos con 9 piezas distribuidas en el área inicial, ocupando un 88,19 % del área total. En cambio, en el segundo caso se colocan hasta 8 piezas, pero el porcentaje de ocupación baja a 44,44%, es decir, un 43,75% menos. La causa principal de esta bajada es la forma de creación de áreas que hemos implementado en el programa. Al generar el área de debajo con el ancho de la pieza más pequeña se está creando un área con una gran diferencia entre el largo y el ancho. De esta forma desaprovechamos una gran cantidad de espacio que en la colocación de las piezas de mayor a menor área no se pierde.

Estas dos pruebas sirven para demostrar que nuestro programa se comporta mejor con un orden de piezas de mayor a menor área. Esto nos servirá para obtener una solución buena inicial si deseamos implementar técnicas de búsqueda de óptimos locales en futuros trabajos.

La cuestión que nos planteamos ahora es si podemos obtener alguna solución mejor que esta con algún otro orden en la lista de piezas inicial. Para ello, vamos a ejecutar el programa reordenando la lista de piezas de forma aleatoria y realizaremos un gran número de simulaciones hasta obtener una solución cercana al óptimo.

Si utilizamos nuestro programa para resolver el caso planteado en la Tabla 10 y realizamos un total de 200.000 simulaciones conseguimos obtener un resultado óptimo, con un 100% de ocupación del área de fabricación. Las piezas asignadas a este lote son P1, P2, P3, P4, P6, P7, P8, P9, P10, P11, P13 y P14 y su distribución se puede observar en la Ilustración 63.

P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	% Ocupación	Nº Piezas
1	1	1	1	0	1	1	1	1	1	1	0	1	1	0	100	12

Tabla 10: Piezas asignadas en la solución óptima del caso 1. Fuente: Elaboración propia.

Esta solución es óptima y si observamos la Ilustración 63 no parece que siga ningún patrón establecido como los que hemos probado en las primeras pruebas. La solución se obtiene gracias a la similitud en las dimensiones de las piezas. El área total se divide en dos regiones, una primera formada por las piezas P3 y P8 y otra región formada por P1, P11, P10, P4, P14, P6, P2, P9, P7 y P13. La segunda región a su vez se divide en tres subáreas en las que encajan perfectamente las piezas.

No se observa ningún patrón de asignación de las piezas que nos permita maximizar el porcentaje de ocupación y, por lo tanto, el resto de las pruebas se ejecutarán reordenando la lista de piezas inicial de forma aleatoria.

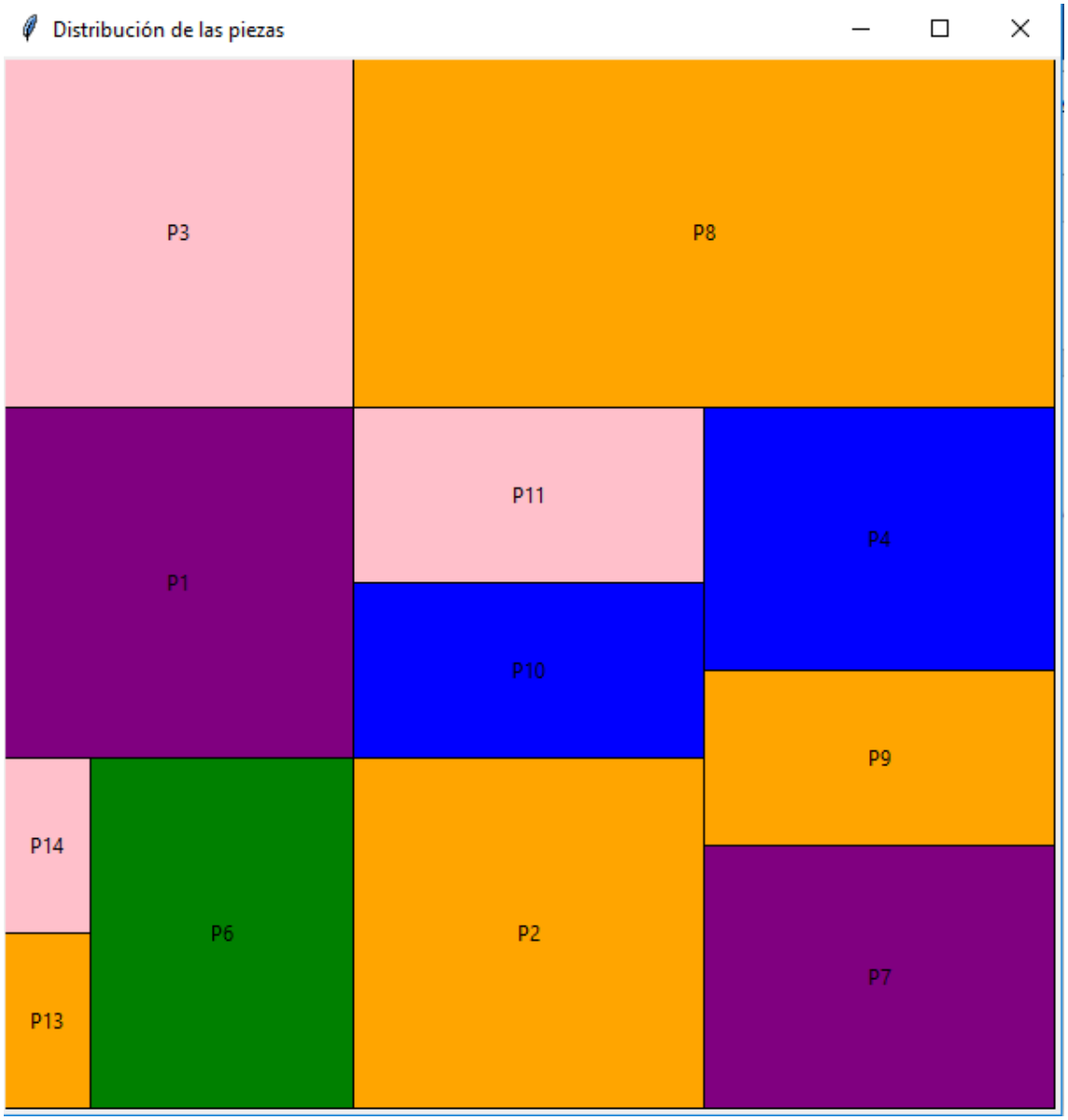


Ilustración 63: Distribución de las piezas en la solución óptima del caso 1.

4.1.2. Caso 2: Pruebas realizadas con número reducido de piezas (33 y 40)

Aunque la literatura relacionada con el problema de empaquetado es muy extensa, son pocos los casos en los que encontramos las dimensiones de las piezas y el área en el que se distribuyen. Dentro de estos casos, nos encontramos con los que presentan Eliana Mirledy y Mauricio Granada en su artículo "Problema de empaquetamiento rectangular bidimensional tipo guillotina resuelto por algoritmos genéticos" (Mirledy Toro and Granada Echeverri, 2007). En este apartado probaremos nuestro algoritmo con las mismas piezas que usaron en sus pruebas y compararemos los resultados. En el primer caso, que denominaremos como caso 2.1., se presentan 33 piezas entre las que se encuentran 6 tipos de piezas diferentes. En el segundo caso, que denominaremos caso 2.2, se presentan un total de 40 piezas, con 10 tipos de piezas diferentes.

4.1.2.1. Caso 2.1: Primera prueba con 33 piezas

En el primer caso nos encontramos con 33 piezas con las dimensiones descritas en la tabla 11. El área de fabricación cuenta con unas dimensiones de 24x38.

Nombre	Largo	Ancho	Área	Cantidad
p1	12	19	228	2
p2	14	19	266	2
p3	2	4	8	25
p4	3	38	114	1
p5	5	19	95	2
p6	2	2	4	1

Tabla 11: Piezas del caso 2.1. Fuente: Elaboración propia.

Este problema cuenta con un espacio de soluciones de $2^{33} \cdot 33! = 7{,}4589 \times 10^{46}$. Para resolverlo, utilizaron el algoritmo genético de Chu-Beasly, que consiste en mantener constante el tamaño de la población de alternativas de solución, de manera que en cada iteración se reemplaza una alternativa de la población usando un eficiente mecanismo de modificación de esta, pero teniendo en cuenta que no se admiten configuraciones repetidas dentro de la población. En cada iteración la población es reemplazada sistemáticamente por un único descendiente generado. Esta estrategia tiene la ventaja de permitir encontrar múltiples soluciones y además conservar la diversidad del conjunto de alternativas.

Se utilizó una población de 10 individuos y se generó aleatoriamente el vector inicial de los individuos y de las secciones. El caso se corrió un total de 20 veces, obteniendo en el 99% de los casos la solución de la Ilustración 64.

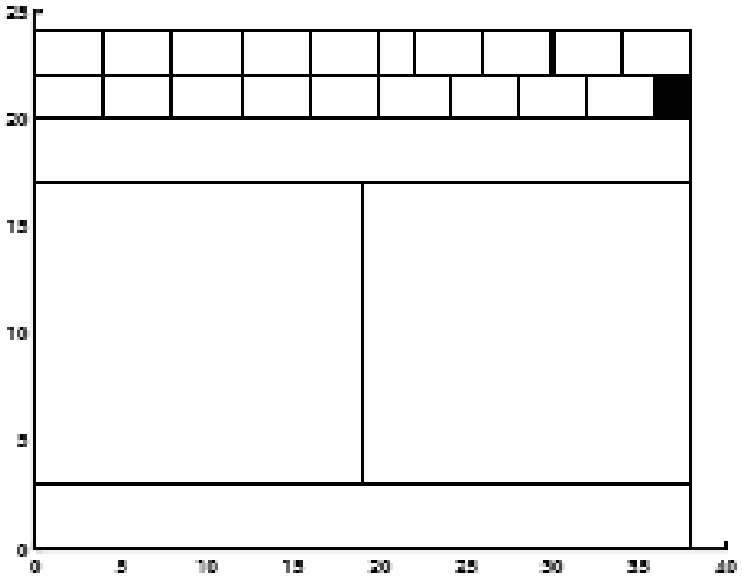


Ilustración 64: Solución al caso 2.1. de Eliana Mirledy y Mauricio Granada. Fuente:(Mirledy Toro and Granada Echeverri, 2007)

Las zonas con fondo negro son los espacios libres, dando lugar a un porcentaje de utilización del material disponible del 99,5%. Este resultado se obtuvo con un equipo con procesador Intel Pentium de 1.6 GHz en un tiempo de ejecución de 20 segundos.

Este será el tiempo límite que vamos a utilizar para correr nuestro programa. En 20 segundos podemos realizar aproximadamente 12000 simulaciones. De las cuales guardaremos tan solo aquellas que igualen o superen el 99,5% de ocupación del área disponible.

De las 12000 simulaciones, se obtuvieron 7 veces soluciones con un 99,5% de ocupación o más. El tiempo empleado en obtener la solución óptima fue de 20 segundos y se consiguió obtener dos combinaciones diferentes con un 100% de ocupación. La primera solución se muestra en la tabla 12.

Solución 1 del caso 2.1			
Tipo	Disponibles	Asignadas	Sin asignar
P1	2	2	0
P2	2	1	1
P3	25	9	16
P4	1	1	0
P5	2	0	2
P6	1	1	0

Tabla 12: Primera solución óptima del caso 2.1. Fuente: Elaboración propia.

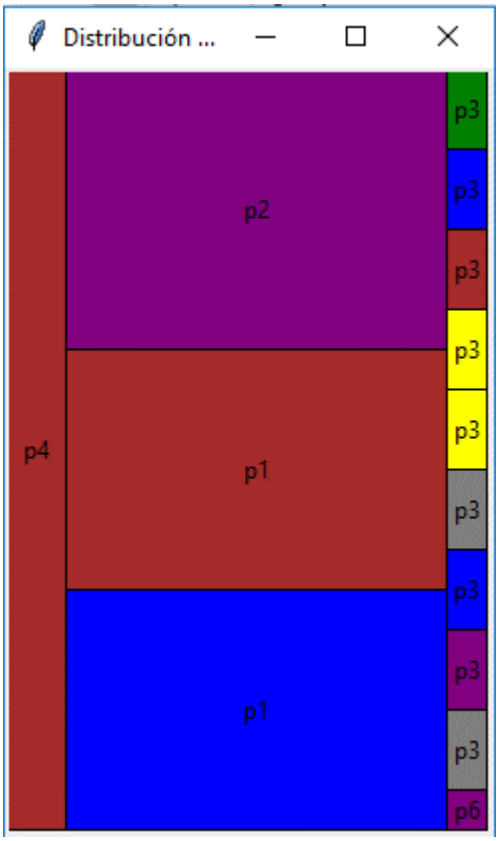


Ilustración 65: Distribución de las piezas en la primera solución óptima del caso 2.1

La primera de ellas utiliza un total de 14 piezas, siendo dos del tipo P1, una del tipo P2, nueve del tipo P3, una del P4 y la última de P6. Esta combinación permite ocupar el 100% del espacio con la distribución que se puede apreciar en la Ilustración 65.

Solución 2 del caso 2.1			
Tipo	Disponibles	Asignadas	Sin asignar
P1	2	2	0
P2	2	1	1
P3	25	0	25
P4	1	0	1
P5	2	2	0
P6	1	0	1

Tabla 13: Segunda solución óptima del caso 2.1. Fuente: Elaboración propia.

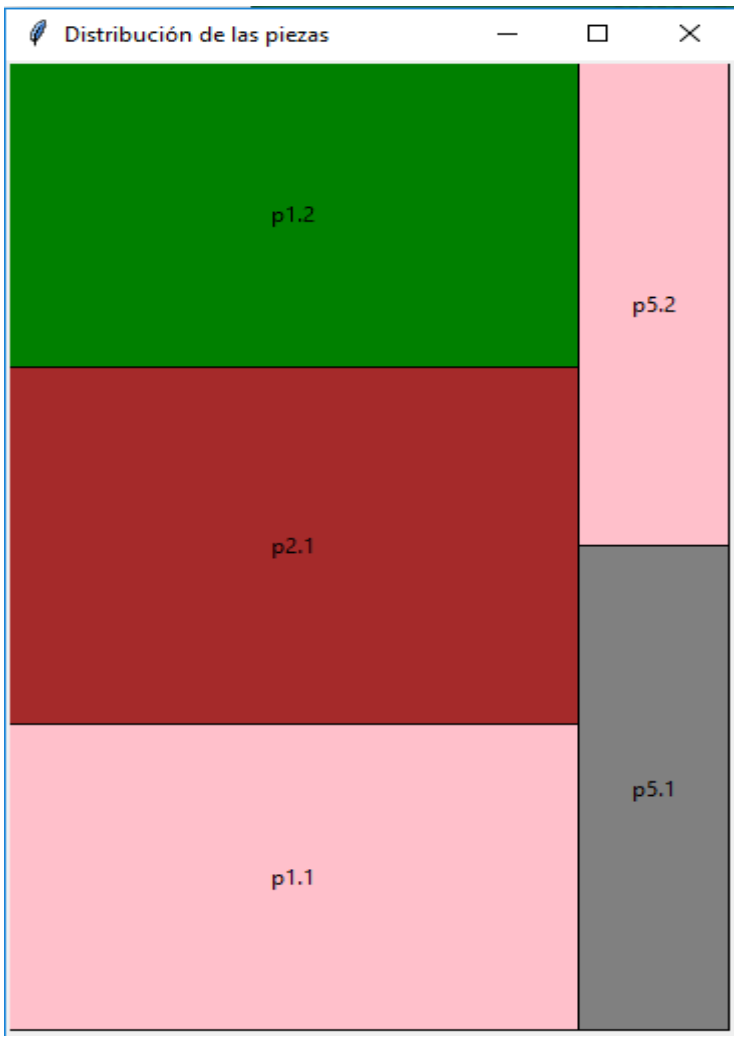


Ilustración 66: Distribución de las piezas en la segunda solución óptima del caso 2.1. Fuente: Elaboración propia.

La segunda combinación de piezas que permite ocupar el 100% del espacio está formada por 5 únicas piezas, dos del tipo P1, una del tipo P2 y dos más del tipo P5. Esta solución se muestra en la tabla 13 y la Ilustración 66.

Ambas soluciones optimizan mejor el espacio de fabricación que el algoritmo presentado por Eliana Mirledy y Mauricio Granada. Su algoritmo permitía ocupar un 99,5% del espacio disponible pero no encontraba la solución óptima del 100% tal y como conseguimos en nuestro caso.

Como nuestro programa parte de una lista de piezas ordenadas de forma aleatoria en cada simulación, se realizan un total de 100000 simulaciones y se analizan los resultados. De esta forma podremos sacar un promedio tanto de las simulaciones necesarias para encontrar la mejor solución como del tiempo necesario.

En 41 ocasiones se obtuvo un resultado igual o mejor que el obtenido por Eliana y Mauricio. Esto quiere decir que en promedio cada 2439 simulaciones, obtenemos una distribución de piezas igual o mejor que la suya. El tiempo empleado en realizar las 100000 simulaciones es de 2 minutos y 41 segundos, es decir 161 segundos. Por lo tanto, podemos aproximar el tiempo empleado en realizar unas 1000 simulaciones a 1,61 segundos.

Con estos datos podemos aproximar el tiempo que necesitamos para obtener una solución igual o mejor que la suya con una simple regla de tres:

$$\frac{1{,}61\ seg}{1000\ simulaciones} = \frac{t}{2439\ simulaciones}$$

$$t = 3{,}93\ segundos$$

Es decir, frente a los 20 segundos empleados por su algoritmo, conseguimos rebajar el tiempo en 16 segundos y obtener incluso una solución mejor en algunos casos.

Es importante destacar varias diferencias entre su planteamiento del problema y nuestro caso. En nuestro caso el programa permite girar las piezas, mientras que ellos no valoran esta posibilidad. Esto influye en la complejidad del problema, aumentando el espacio de soluciones. A su vez, permite obtener soluciones imposibles de alcanzar con su técnica.

4.1.2.2. Caso 2.2: Segunda prueba con 40 piezas

En este segundo caso, aumenta tanto el número de piezas como el número de clases. Contamos con un total de 40 piezas, siete más que en el caso anterior y 10 tipos diferentes, es decir, 4 más que en el caso 2.1.

El área disponible para situar las piezas es de 10x10. Las dimensiones de las piezas se muestran en la tabla 14.

Este problema cuenta con un espacio de soluciones de $2^{40} \cdot 40! = 8{,}971 \times 10^{59}$. Para resolverlo, utilizaron el algoritmo genético de Chu-Beasly, comentado antes. Se utilizó una población de 10 individuos y se generó aleatoriamente el vector inicial de los individuos y de las secciones. El caso se corrió un total de 20 veces, obteniendo en el 99% de los casos la solución de la Ilustración 67.

Nombre	Largo	Ancho	Área	Cantidad
p1	3	2	6	4
p2	7	2	14	4
p3	4	2	8	4
p4	6	2	12	4
p5	9	1	9	4
p6	8	4	32	4
p7	4	1	4	4
p8	1	10	10	4
p9	3	7	21	4
p10	4	5	20	4

Tabla 14: Piezas del caso 2.2. Fuente: Elaboración propia.

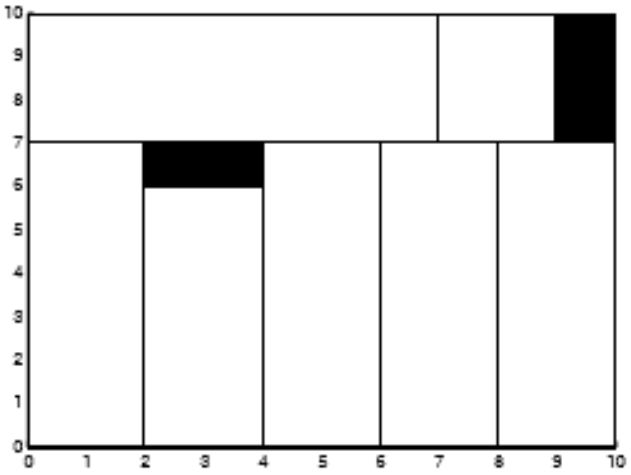


Ilustración 67: Solución al caso 2.2. de Eliana Mirledy y Mauricio Granada. Fuente: (Mirledy Toro and Granada Echeverri, 2007)

De la misma forma que en el caso anterior, la zona con fondo negro representa el espacio que no se consigue aprovechar con su algoritmo. En este caso, el porcentaje de ocupación es del 95%, siendo inferior al obtenido en el primer caso. Este resultado se obtuvo con un equipo con procesador Intel Pentium de 1.6 GHz en un tiempo de ejecución de 28 segundos.

Al igual que hicimos en el caso 2.1., utilizaremos un tiempo de ejecución de 28 segundos, es decir, podemos realizar aproximadamente 17000 simulaciones. Del total, guardaremos tan solo aquellas que superen el 95% de ocupación del área disponible.

De las 17000 simulaciones, en 5923 se ha conseguido un resultado igual o superior al obtenido por Eliana y Mauricio. Dentro de estos resultados, contamos con distribuciones que ocupan el 96, 97, 98, 99 e incluso el 100% del espacio disponible. Con la distribución de las 6 piezas de la tabla 15, obtenemos un 100% del espacio ocupado, tal y como se puede observar en la Ilustración 68.

Solución 1 del Caso 2.2			
Tipo	Disponibles	Asignadas	Sin asignar
P1	4	1	3
P2	4	1	3
P3	4	2	2
P4	4	0	4
P5	4	0	4
P6	4	2	2
P7	4	0	4
P8	4	0	4
P9	4	0	4
P10	4	0	4

Tabla 15: Solución óptima del caso 2.2. Fuente: Elaboración propia.

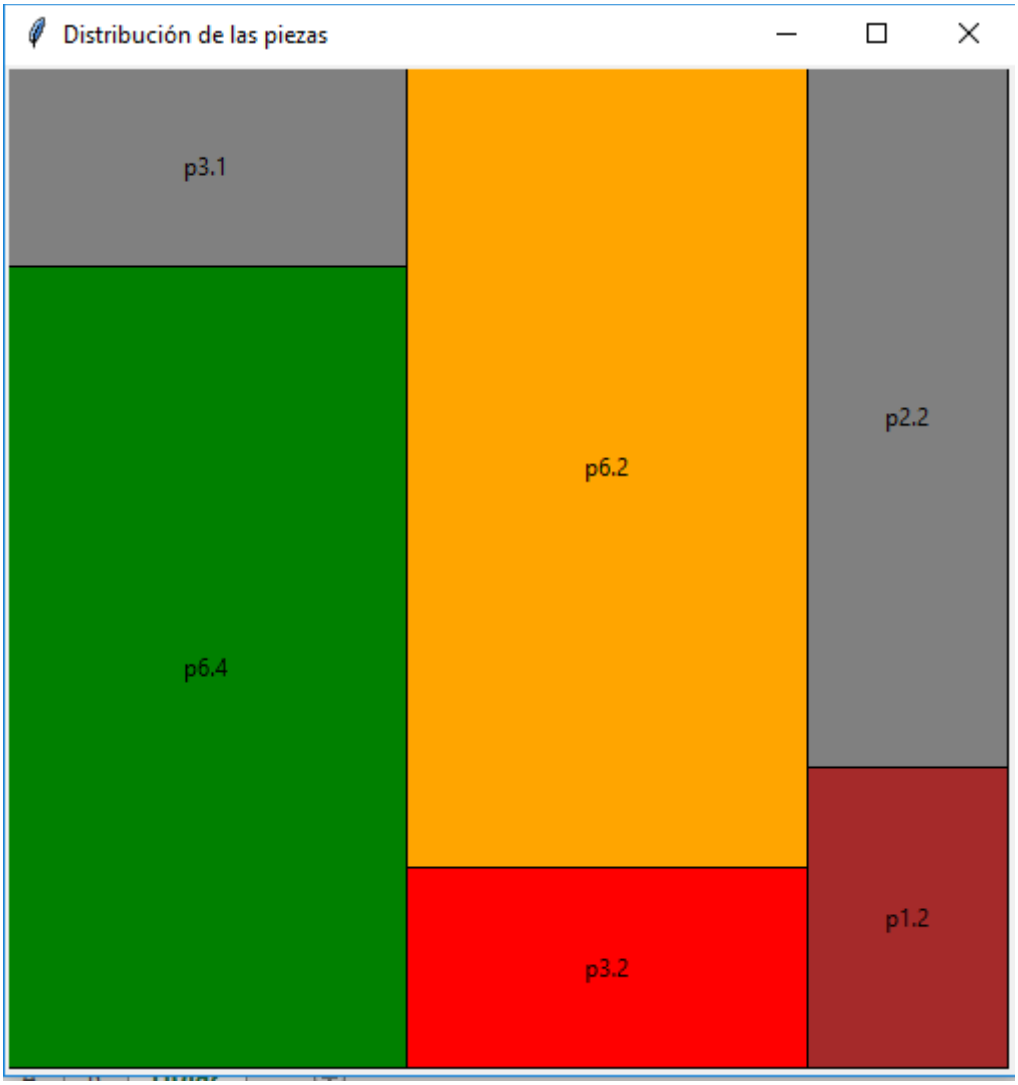


Ilustración 68: Distribución de las piezas en la solución óptima del caso 2.2. Fuente: Elaboración propia.

Los resultados obtenidos dejan claro que nuestro programa se comporta mucho mejor en este escenario. De nuevo realizamos 100000 simulaciones y promediamos tanto el tiempo como el número de veces que obtenemos una solución igual o superior a la suya. El tiempo empleado coincide con el empelado en el caso 2.1, siendo de 161 segundos. Por lo tanto, cada simulación tarda en 1,61 segundos en realizarse. En cuanto al resultado obtenido, en 34837 simulaciones se obtuvo un resultado superior al 95%. Esto quiere decir que en una de cada tres simulaciones se obtiene un resultado que supera el 95% de ocupación.

Por lo tanto, podemos promediar el tiempo empleado en obtener una solución en 4,83 segundos. Es decir, el tiempo que necesitamos para obtener una solución mejor que la de Eliana y Mauricio es de aproximadamente 5 segundos. Siendo 23 segundos más rápido que el tiempo empleado por ellos. Además, las soluciones son incluso mejores que las suyas, llegando a ofrecer distribuciones que ocupan el 100% del área disponible.

Al igual que ocurría en el caso anterior, al contar con la posibilidad de girar las piezas nuestro programa presenta un espacio de soluciones superior al expuesto en el artículo.

Como conclusiones de las pruebas realizadas con pedidos de entre 30 y 40 piezas podemos destacar las siguientes:

- El programa obtiene soluciones de gran calidad, superando incluso las reportadas por algunos autores en sus estudios. Es capaz de obtener la solución óptima en situaciones donde es factible alcanzarla. Dando lugar a distribuciones de piezas que abarcan el 100% del área disponible.
- Los tiempos de computación son bajos, permitiendo obtener una solución cercana al óptimo en cuestión de minutos. Esto permitirá a los fabricantes poder tomar decisiones de forma rápida y comenzar cuanto antes con la producción diaria.

4.1.3. Caso 3: Pruebas con número grande de piezas (94 y 106)

Si buscamos casos con un mayor número de piezas, encontramos 50 casos de estudio que se repiten en las pruebas de diferentes artículos. Cada uno de los casos presenta 20 tipos de piezas rectangulares y un total de 100 piezas aproximadamente. En el artículo presentado por Yaodong Cui, (Cui, 2005), encontramos las dimensiones de las piezas de los cinco primeros casos. Compararemos nuestros resultados con la mejor solución obtenida hasta la publicación de Mirledy Toro, Garcés and Ruiz, (2008). A pesar de no ser artículos recientes, son los únicos que hemos encontrado con comparación del porcentaje de ocupación en problemas de empaquetado y un número alto de piezas. Como nuestro principal objetivo es obtener soluciones con altos porcentajes de ocupación, estos casos nos sirven para comprobar si nuestra heurística cuenta con un buen comportamiento ante casos con un alto número de piezas. Tanto las piezas como las dimensiones del espacio donde se deben colocar aparecen en la tabla 16.

En esta tabla contamos con el valor del largo y ancho del área, L y W respectivamente. Las piezas vienen dadas por el largo l_i, ancho w_i y su demanda d_i. En los siguientes apartados se tabulan los datos caso a caso para poder observar mejor las dimensiones de cada una de las piezas.

Los resultados que se obtuvieron en los primeros 25 casos se pueden observar en la tabla 17. De estos 25 casos presentados, solo contamos con datos de las piezas de los cinco primeros y realizaremos pruebas con 2 de ellos.

ID	$L \times W$	$l_i \times w_i \times d_i$
1	2002 × 1001	53 × 67 × 1, 135 × 194 × 1, 119 × 407 × 7, 160 × 382 × 5, 194 × 410 × 8, 205 × 86 × 6, 297 × 319 × 3, 449 × 414 × 8, 81 × 223 × 2, 195 × 78 × 9, 400 × 172 × 4, 429 × 216 × 1, 170 × 217 × 4, 386 × 275 × 8, 51 × 324 × 4, 424 × 77 × 9, 92 × 416 × 4, 240 × 111 × 5, 383 × 160 × 8, 269 × 203 × 9
2	2449 × 1201	442 × 136 × 9, 300 × 64 × 10, 194 × 378 × 6, 103 × 301 × 4, 218 × 252 × 8, 310 × 89 × 10, 230 × 349 × 6, 352 × 341 × 10, 341 × 392 × 5, 316 × 113 × 7, 134 × 82 × 2, 144 × 334 × 3, 254 × 233 × 7, 111 × 167 × 1, 357 × 75 × 1, 426 × 399 × 2, 419 × 352 × 3, 193 × 311 × 6, 347 × 198 × 6, 243 × 190 × 1
3	2275 × 1191	132 × 209 × 4, 382 × 313 × 10, 129 × 97 × 6, 448 × 123 × 6, 247 × 299 × 6, 80 × 51 × 3, 195 × 239 × 7, 91 × 287 × 9, 251 × 77 × 5, 317 × 196 × 1, 397 × 115 × 75, 169 × 293 × 1, 70 × 260 × 10, 360 × 141 × 6, 389 × 305 × 5, 381 × 322 × 9, 51 × 336 × 6, 235 × 335 × 1, 287 × 128 × 1, 365 × 234 × 4
4	2222 × 1116	305 × 279 × 6, 343 × 384 × 5, 103 × 376 × 5, 268 × 210 × 3, 441 × 86 × 6, 302 × 93 × 2, 236 × 54 × 6, 120 × 173 × 5, 95 × 150 × 10, 215 × 392 × 5, 339 × 246 × 10, 73 × 267 × 2, 73 × 80 × 9, 96 × 124 × 9, 87 × 390 × 7, 160 × 445 × 1, 109 × 187 × 3, 231 × 58 × 9, 340 × 118 × 4, 446 × 237 × 3
5	2239 × 1036	271 × 428 × 5, 234 × 68 × 7, 225 × 327 × 7, 233 × 351 × 3, 396 × 255 × 1, 361 × 77 × 4, 252 × 239 × 4, 369 × 380 × 7, 207 × 144 × 1, 439 × 293 × 2, 164 × 116 × 2, 145 × 268 × 2, 84 × 245 × 6, 385 × 117 × 9, 445 × 390 × 6, 417 × 204 × 7, 329 × 150 × 4, 60 × 94 × 6, 191 × 78 × 1, 441 × 205 × 10

Tabla 16: Dimensiones de las piezas y áreas del caso 3. Fuente:(Cui, 2005)

Caso de Prueba		*Respuesta de la Literatura*		*Respuesta del algoritmo propuesto*		*Diferencia*
Caso	*Área disponible*	*Área utilizada*	*Porcentaje de uso*	*Área utilizada*	*Porcentaje de uso*	
1	2004002	1964584	0,9803	1990174,386	0,9931	25590,3862
2	2941249	2867218	0,9748	2887424,143	0,9817	20206,1433
3	2709525	2656454	0,9804	2679178,320	0,9888	22724,3200
4	2479752	2370496	0,9559	2443299,646	0,9853	72803,6456
5	2319604	2281600	0,9836	2275995,445	0,9812	-5604,5552
6	2976190	2918503	0,9806	2931249,531	0,9849	12746,5310
7	2265650	2227422	0,9831	2240954,415	0,9891	13532,4150
8	2272792	2238658	0,9849	2247791,288	0,9890	9133,2880
9	2764384	2690038	0,9731	2717389,472	0,9830	27351,4720
10	2624232	2574750	0,9414	2579620,056	0,9830	4870,0560
11	3235950	3115873	0,9628	3162817,530	0,9774	46944,5300
12	2921333	2837743	0,9713	2856771,541	0,9779	19028,5407
13	2553588	2501400	0,9795	2525753,891	0,9891	24353,8908
14	2436480	2343169	0,9617	2350959,552	0,9649	7790,5520
15	3072540	3017184	0,9819	3031675,218	0,9867	14491,2180
16	2174340	2102756	0,9670	2136723,918	0,9827	33967,9180
17	2401595	2276323	0,9478	2317299,016	0,9649	40976,0155
18	2827455	2736919	0,9679	2751961,952	0,9733	15042,9515
19	2181600	2145051	0,9832	2149966,80	0,9855	4915,8000
20	3157786	2994717	0,9483	3050737,055	0,9661	56020,0546
21	2953340	2900376	0,9820	2892501,196	0,9794	-7874,8040
22	3204790	3066344	0,9568	3086853,728	0,9632	20509,7280
23	2763800	2676037	0,9682	2686966,36	0,9722	10929,3600
24	2408892	2287330	0,9495	2292060,738	0,9515	4730,7380
25	3163699	3037774	0,9601	3108967,007	0,9827	71193,0073

Tabla 17: Resultados del caso 3 obtenidos por Eliana Mirledy y Alejandro Garcés. Fuente: (Mirledy Toro, Garcés and Ruiz, 2008).

4.1.3.1. Caso 3.1: Prueba con 106 piezas

En primer lugar, comenzamos con el caso de la tabla 16 que tiene ID=1. Las dimensiones del área disponible son de 2002x1001. Las 106 piezas están distribuidas en 20 clases diferentes y la cantidad de cada una de ellas se puede observar en la tabla 18.

Nombre	Largo	Ancho	Área	Cantidad
p1	53	67	3551	1
p2	135	194	26190	1
p3	119	407	48433	7
p4	160	382	61120	5
p5	194	410	79540	8
p6	205	86	17630	6
p7	297	319	94743	3
p8	449	414	185886	8
p9	81	223	18063	2
p10	195	78	15210	9
p11	400	172	68800	4
p12	429	216	92664	1
p13	170	217	36890	4
p14	386	275	106150	8
p15	51	324	16524	4
p16	424	77	32648	9
p17	92	416	38272	4
p18	240	111	26640	5
p19	383	160	61280	8
p20	269	203	54607	9

Tabla 18: Piezas del caso 3.1. Fuente: Elaboración propia.

Este problema cuenta con un espacio de soluciones de $2^{106} \cdot 106! = 9{,}2997 \cdot 10^{201}$. Para resolverlo, utilizaron una propuesta que emplea la codificación de árbol binario y un algoritmo dividido en tres etapas que trabajan con estrategias individuales inspiradas en algoritmos de vecindad variable, recocido simulado y técnicas constructivas para lograr la solución del problema. Al trabajar con un espacio de soluciones tan complejo, su propuesta es mucho más compleja que la nuestra. Por ello, no buscaremos encontrar su solución, pero si acercarnos lo máximo posible a su resultado. Su algoritmo consigue un porcentaje de ocupación del 99,31 % y no se detalla el tiempo empleado para obtenerla. En nuestro caso al tratarse con un espacio de soluciones del orden de 10^{201} utilizaremos 100000 simulaciones. Guardaremos aquellas que superen el 95% de ocupación del área de fabricación.

En ninguna de las soluciones se consiguió superar el 95% del porcentaje de área ocupado. La mejor solución ocupaba el 94,03% del espacio disponible. Para conseguirlo, fueron necesarios 14 minutos de tiempo de ejecución del programa. La distribución obtenida se componía de las piezas de la tabla 19 y se muestra en la Ilustración 69.

Solución del Caso 3.1			
Tipo	Disponibles	Asignadas	Sin asignar
P1	1	1	0
P2	1	1	0
P3	7	0	7
P4	5	3	2
P5	8	0	8
P6	6	1	5
P7	3	1	2
P8	8	4	4
P9	2	1	1
P10	9	7	2
P11	4	1	3
P12	1	0	1
P13	4	0	4
P14	8	2	6
P15	4	3	1
P16	9	2	7
P17	4	1	3
P18	5	0	5
P19	8	3	5
P20	9	1	8

Tabla 19: Piezas de la solución al caso 3.1. Fuente: Elaboración propia.

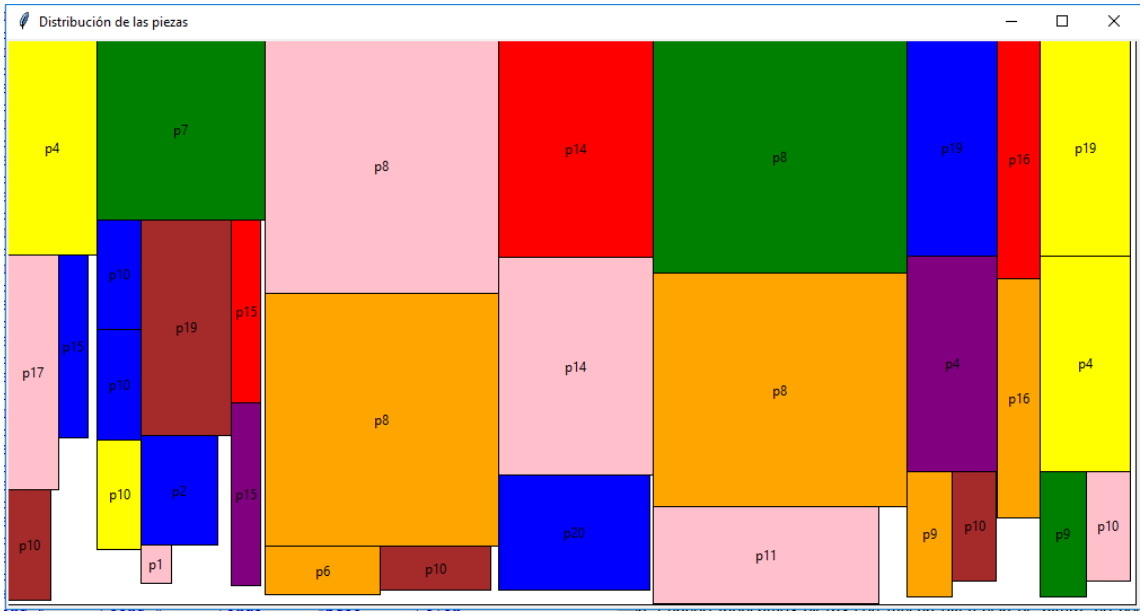


Ilustración 69: Distribución de las piezas de la solución al caso 3.1. Fuente: Elaboración propia.

Contamos con un total de 33 piezas asignadas, dando lugar a un 94,03% de ocupación que, si lo comparamos con los resultados de la tabla 17, donde se obtuvieron un 98,03% y 99,31% de ocupación, es un 4-5% menor que los mejores resultados. Esto implica que nuestro programa no se comporta tan bien como los otros cuando el espacio de soluciones es tan grande como en este caso. Además, el número de simulaciones necesario para obtener estos resultados es muy alto, dando lugar a tiempos de computación de casi quince minutos.

4.1.3.2. Caso 3.2: Prueba con 94 piezas

En el caso anterior, los resultados obtenidos muestran que nuestro programa no se comporta demasiado bien ante un número tan alto de piezas. Para comprobar esta afirmación, vamos a someterlo a una segunda prueba con un número alto de piezas. Esta vez, las piezas serán las descritas en el caso con ID = 5 de la tabla 16. Contamos con un total de 94 piezas de 20 tipos diferentes (Tabla 20). El área es superior al del caso anterior, siendo sus dimensiones 2239x1036.

Nombre	Largo	Ancho	Área	Cantidad
P1	271	428	115988	5
P2	234	68	15912	7
P3	225	327	73575	7
P4	233	351	81783	3
P5	396	255	100980	1
P6	361	77	27797	4
P7	252	239	60228	4
P8	369	380	140220	7
P9	207	144	29808	1
P10	439	293	128627	2
P11	164	116	19024	2
P12	145	268	38860	2
P13	84	245	20580	6
P14	385	117	45045	9
P15	445	390	173550	6
P16	417	204	85068	7
P17	329	150	49350	4
P18	60	94	5640	6
P19	191	78	14898	1
P20	441	205	90405	10

Tabla 20: Piezas del caso 3.2. Fuente: Elaboración propia.

En este caso, tenemos 12 piezas menos que en el caso anterior, reduciéndose el espacio de soluciones a $2^{94} \cdot 94! = 2{,}15 \cdot 10^{174}$. A pesar de ser inferior al caso anterior, sigue siendo un espacio de soluciones muy grande. Por ello, realizaremos un total de 100000 simulaciones y buscaremos soluciones que superen el 95% de ocupación.

Solución del caso 3.2.			
Nombre	Disponibles	Asignadas	Sin asignar
P1	5	2	3
P2	7	7	0
P3	7	2	5
P4	3	0	3
P5	1	0	1
P6	4	3	1
P7	4	1	3
P8	7	2	5
P9	1	1	0
P10	2	0	2
P11	2	2	0
P12	2	0	2
P13	6	5	1
P14	9	1	8
P15	6	2	4
P16	7	1	6
P17	4	0	4
P18	6	6	0
P19	1	1	0
P20	10	6	4

Tabla 21: Solución del caso 3.2. Fuente: Elaboración propia.

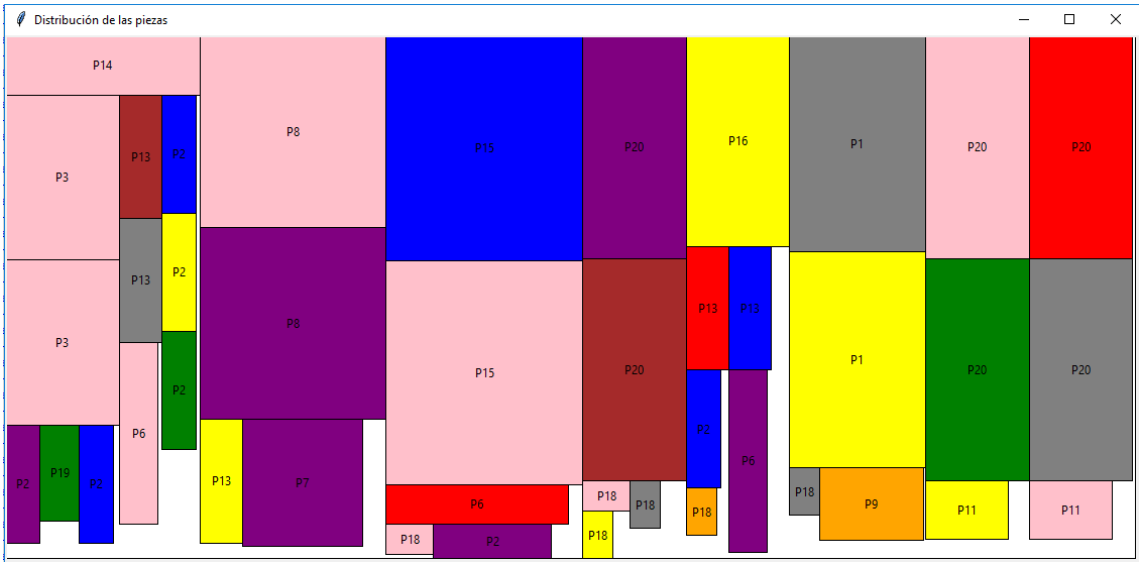


Ilustración 70: Distribución de las piezas de la solución 3.2. Fuente: Elaboración propia.

Contamos con un total de 42 piezas distribuidas a lo largo del área disponible. Con la distribución de la ilustración 70 conseguimos ocupar un 92,85%.

Contamos con un total de 42 piezas asignadas, dando lugar a un 92,85% de ocupación que, si lo comparamos con los resultados de la tabla 17, donde se obtuvieron un 98,06% y 98,49% de ocupación, es un 5-6 % menor que los mejores resultados. Esto implica que nuestro programa no se comporta tan bien como los otros cuando el espacio de soluciones es tan grande como en este caso. Además, el número de simulaciones necesario para obtener estos resultados es muy alto, dando lugar a tiempos de computación de casi 15 minutos.

Como conclusiones podemos destacar las siguientes:

- El programa no consigue los resultados obtenidos por otras técnicas propuestas cuando nos enfrentamos a casos que rondan las 100 piezas. A pesar de ello, los porcentajes de ocupación obtenidos son altos en todos los casos, superando el 90% en todas las pruebas realizadas.
- El tiempo de computación necesario para obtener buenos resultados ronda los 15 minutos. Esto no resulta un problema cuando lo que tratamos es de planificar la producción diaria.

4.2. Selección del ganador

En la segunda etapa de nuestro problema, nos encontramos con los diferentes lotes obtenidos en la resolución del problema de empaquetado. Entre todos los lotes posibles, elegiremos aquellos que superen el porcentaje de ocupación mínimo que deseemos. Con este primer filtro, obtenemos los lotes de partida para la selección del ganador. En este apartado planteamos diferentes escenarios que nos pueden aparecer en casos reales y daremos indicaciones para elegir un lote antes que otro.

Las piezas fabricadas por impresión 3D permiten a los fabricantes definir la estructura interna, decidiendo tanto el tipo de relleno que llevará cada pieza como su porcentaje. Además, se puede decidir el porcentaje de relleno, en función de las características estructurales que deseamos tener en nuestra pieza. Contamos desde la estructura más simple basada en rectángulos hasta patrones de panel de abeja. Algunos de los tipos más utilizados se pueden observar en la Ilustración 71.

Podemos obtener tanto piezas formadas por la mera cáscara, es decir, con un 0% de relleno, como piezas macizas (Ilustración 72). El porcentaje de relleno dependerá mucho del uso de la pieza, no necesitará el mismo relleno una pieza que sirva como adorno que una pieza funcional.

Otro factor importante será la altura de la pieza. Cuanto mayor sea la altura, mayor es el tiempo requerido para fabricarla, además de requerir una mayor cantidad de material. Por lo tanto, si empleamos mayor cantidad de material, el retorno que obtiene el fabricante es mayor. Al igual que con el porcentaje de relleno, habrá que estudiar el efecto de fabricar piezas altas frente a otras con menor altura. Aunque en algunas situaciones el porcentaje de ocupación de la cama de fabricación sea menor, la cantidad de material total podría ser superior debido a la asignación de piezas muy altas.

En el caso de la impresión 3D, la diferencia de alturas entre las piezas que se fabrican de forma simultánea es un factor que debemos tener muy en cuenta. Al fabricar piezas con mucha diferencia de altura de forma simultánea, no podemos retirar aquellas que tienen

menor altura hasta que todas las piezas estén completamente elaboradas. Este factor se tendrá en cuenta y se estudiarán casos en los que pueda servir de criterio final.

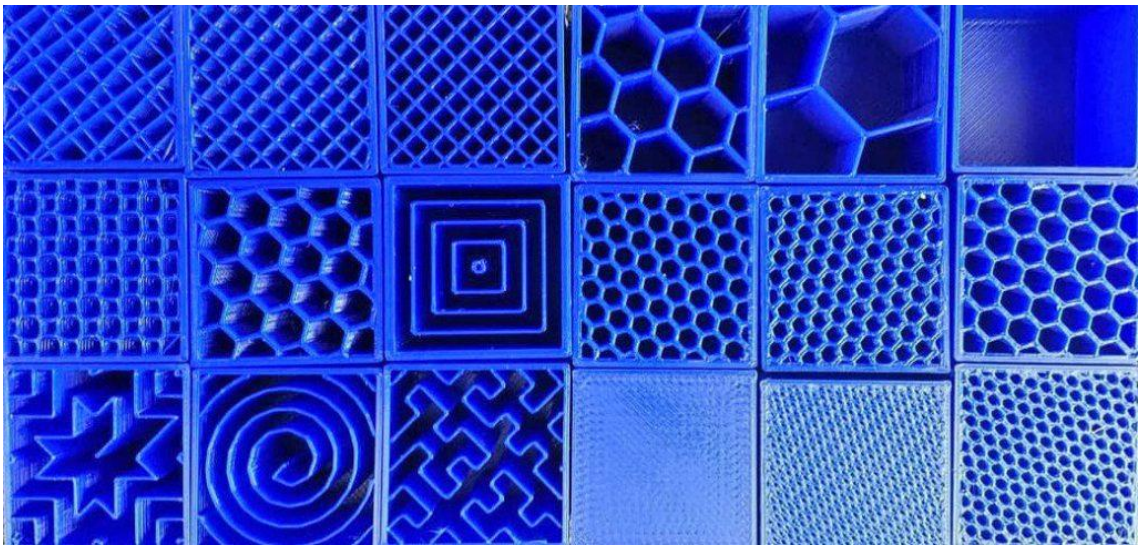


Ilustración 71: Estructura interna de las piezas fabricadas por impresión 3d. Fuente: (LifeHacks3D, 2019)

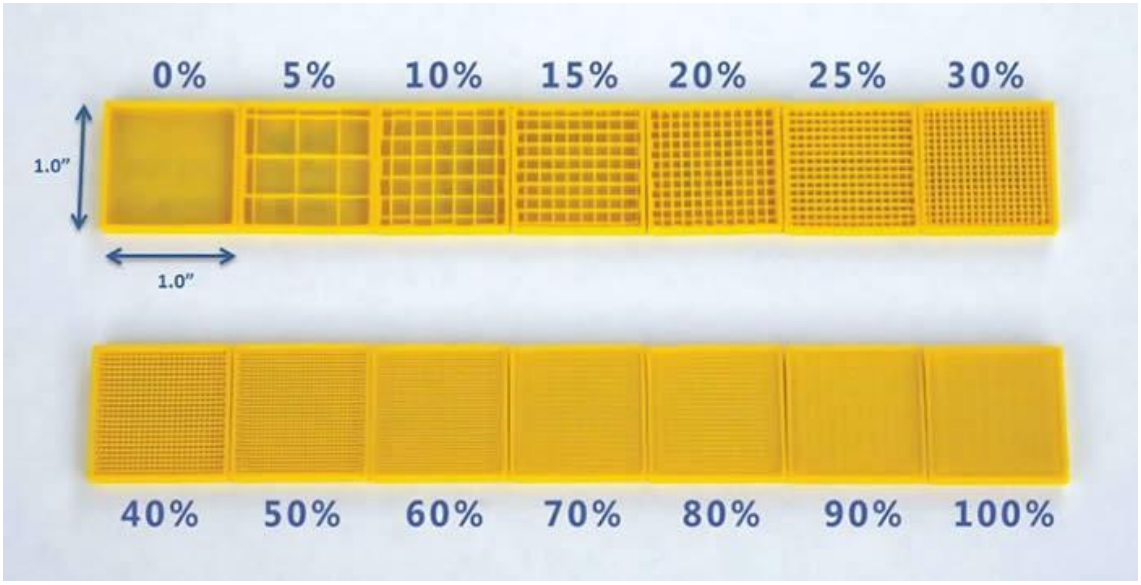


Ilustración 72: Porcentaje de relleno de diferentes piezas impresión 3D. Fuente: (3D Works, 2019)

4.2.1. Prueba con diferentes porcentajes de relleno de las piezas

En primer lugar, utilizaremos las piezas del caso 1. Los datos de las piezas aparecen en la tabla 5. El área disponible tendrá unas dimensiones de 600x600 y la altura suficiente para que todas las piezas se puedan fabricar. Realizaremos 10000 simulaciones y guardaremos aquellos que superen el 90% de ocupación. En primer lugar, mantendremos la altura igual en todas las piezas y veremos los diferentes lotes que obtenemos.

Dentro de las soluciones obtenidas, seleccionamos los 8 lotes con mayor porcentaje de ocupación y los ordenamos de mayor a menor tal y como se puede observar en la tabla 22.

p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	p13	p14	p15	% ocupación:	Cantidad material	Nº Piezas
1	1	1	1	1	0	1	1	1	1	1	0	1	1	0	100	20300000	12
1	1	1	1	1	1	0	1	1	1	1	0	1	1	0	100	20300000	12
1	1	1	0	1	1	1	1	0	0	0	0	0	1	1	99,31	20200000	9
1	0	1	1	1	1	0	1	1	1	0	0	1	0	1	99,31	19400000	10
0	1	0	0	1	1	0	1	1	1	1	1	1	1	1	99,31	17700000	11
1	1	1	1	1	1	1	0	1	1	1	1	1	1	0	98,61	20200000	13
1	1	1	1	1	1	1	0	1	1	0	0	1	1	1	97,92	20100000	12
1	1	1	1	1	1	1	1	0	0	1	0	1	1	0	97,22	20800000	11

Tabla 22: Lotes de piezas con mayor porcentaje de ocupación. Fuente: Elaboración propia.

Cada fila representa un lote diferente y aquellas casillas con un 1 y fondo verde indican que la pieza está asignada al lote. Aquellas casillas que aparecen con un 0 y fondo blanco indican que la pieza no está asignada al lote correspondiente. Si nos fijamos en la cantidad de material, podemos observar que el lote con mayor porcentaje de ocupación no es el que mayor cantidad de material utiliza, ni el que cuenta con un mayor número de piezas asignadas.

Por lo tanto, una vez que entramos a considerar la tercera dimensión, el porcentaje de relleno de las piezas hará que no siempre el lote con mayor porcentaje de ocupación sea la opción que nos reporte mayor beneficio. Si reordenamos la tabla en función de la cantidad de material obtenemos los resultados que se muestran en la tabla 23.

p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	p13	p14	p15	% ocupación	Cantidad material	Nº Piezas
1	1	1	1	1	1	1	1	0	0	1	0	1	1	0	97,22	20800000	11
1	1	1	1	1	0	1	1	1	1	1	0	1	1	0	100	20300000	12
1	1	1	1	1	1	0	1	1	1	1	0	1	1	0	100	20300000	12
1	1	1	0	1	1	1	1	0	0	0	0	0	1	1	99,31	20200000	9
1	1	1	1	1	1	1	0	1	1	1	1	1	1	0	98,61	20200000	13
1	1	1	1	1	1	1	0	1	1	0	0	1	1	1	97,92	20100000	12
1	0	1	1	1	1	0	1	1	1	0	0	1	0	1	99,31	19400000	10
0	1	0	0	1	1	0	1	1	1	1	1	1	1	1	99,31	17700000	11

Tabla 23: Lotes de piezas con mayor cantidad de material utilizado. Fuente: Elaboración propia.

El lote que mayor cantidad de material emplea es aquel formado por las piezas p1, p2, p3, p4, p5, p6, p7, p8, p11, p13 y p14 (11 en total) y cuenta con una cantidad de 20.800.000 mm^3 de material. Este será el lote que más beneficio nos reporte y ocupará tan solo un 97,22 % del espacio disponible. Estando por encima de este valor de ocupación el resto de los lotes presentados en la tabla.

Si lo comparamos con el lote del 100%, se diferencia en 3 piezas. En el lote con 100% de ocupación, se fabricarían las piezas p9 y p10 y se dejaría de fabricar la p6. El porcentaje de relleno de las piezas p9 y p10 es del 20% mientras que el de la pieza p6 es del 40%, es decir, justo el doble. Las dimensiones de la pieza p6 son 150x200x200 que multiplicado por 40%

de relleno dan un total de 2100000 mm³ de material. En cambio, las piezas p9 y p10 tienen cada una unas dimensiones de 100x200x200 que multiplicado por 20% de relleno da un total de 800000 mm³ de material (1600000 mm³ entre las dos). Esta diferencia en el relleno consigue que a pesar de tratarse de un área menor el de la pieza p6 frente a la suma de áreas de p9 y p10, sea superior en términos de volumen de material.

Con este ejemplo queda claro que el aprovechamiento de la cama de fabricación es importante pero que no será el factor decisorio. En cambio, si lo será la cantidad de material y por ello las piezas con alto valor de porcentaje de relleno tendrán más valor que aquellas con un menor porcentaje de relleno a igualdad de dimensiones.

4.2.2. Pruebas con diferentes alturas de las piezas

Para realizar las pruebas de este apartado vamos a reutilizar las piezas de la tabla 5, pero esta vez mantendremos el porcentaje de relleno constante y variaremos las alturas de las piezas. El objetivo de estas pruebas es comprobar el efecto de la altura en la elección del ganador. Las piezas con sus nuevas alturas y porcentajes de relleno se muestran en la tabla 24.

Nombre	Relleno	Largo	Ancho	Altura	Área	Volumen
P1	1,0	200	200	100	40000	4000000
P2	1,0	200	200	100	40000	4000000
P3	1,0	200	200	200	40000	8000000
P4	1,0	150	200	100	30000	3000000
P5	1,0	150	200	150	30000	4500000
P6	1,0	150	200	150	30000	4500000
P7	1,0	150	200	100	30000	3000000
P8	1,0	400	200	400	80000	32000000
P9	1,0	100	200	300	20000	6000000
P10	1,0	100	200	300	20000	6000000
P11	1,0	100	200	100	20000	2000000
P12	1,0	300	150	150	45000	6750000
P13	1,0	100	50	100	5000	500000
P14	1,0	50	100	100	5000	500000
P15	1,0	250	250	200	62500	12500000

Tabla 24: Piezas de la prueba con diferentes alturas. Fuente: Elaboración propia.

Partimos de estas piezas y un área disponible de 600x600 con una altura de 400, superior a la pieza más alta de la tabla. Al igual que en el caso anterior, realizaremos 10000 simulaciones y guardaremos aquellos lotes que superen el 90% de ocupación.

Dentro de las soluciones obtenidas, seleccionamos los 9 lotes con mayor cantidad de material empleado y los ordenamos de mayor a menor en función del porcentaje de ocupación, tal y como se puede observar en la tabla 25.

p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	p13	p14	p15	% ocupación	Cantidad material	Nº Piezas
1	1	1	0	1	1	1	1	1	1	1	0	1	1	0	100	75000000	12
0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	98,61	76750000	12
0	1	1	1	1	0	0	1	1	1	1	0	1	1	1	97,92	79000000	11
0	0	1	0	1	1	1	1	1	1	1	0	1	1	1	95,14	79500000	11
1	0	1	0	0	0	0	1	1	1	1	1	1	1	1	93,75	78250000	10
0	1	1	0	1	1	0	1	1	1	0	0	1	1	1	92,36	78500000	10
0	1	1	0	1	0	1	1	1	1	0	0	1	1	1	92,36	77000000	10
0	0	1	0	0	1	0	1	1	1	1	1	1	1	1	90,97	78750000	10
0	0	1	1	0	0	0	1	1	1	1	1	1	1	1	90,97	77250000	10

Tabla 25: Lotes de piezas con mayor porcentaje de ocupación. Fuente: Elaboración propia.

Se puede observar que los lotes con mayor porcentaje de ocupación no son los que utilizan mayor cantidad de material. A pesar de haber sido capaces de optimizar al 100% el espacio de fabricación, el lote que permite alcanzar esta ocupación no es el más rentable a la hora de fabricarlo. En la tabla 26, reordenamos los lotes en función de la cantidad de material para poder sacar conclusiones más claras y comparar en función de este factor.

p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	p13	p14	p15	% ocupación	Cantidad material	Nº Piezas
0	0	1	0	1	1	1	1	1	1	1	0	1	1	1	95,14	79500000	11
0	1	1	1	1	0	0	1	1	1	1	0	1	1	1	97,92	79000000	11
0	0	1	0	0	1	0	1	1	1	1	1	1	1	1	90,97	78750000	10
0	1	1	0	1	1	0	1	1	1	0	0	1	1	1	92,36	78500000	10
1	0	1	0	0	0	0	1	1	1	1	1	1	1	1	93,75	78250000	10
0	0	1	1	0	0	0	1	1	1	1	1	1	1	1	90,97	77250000	10
0	1	1	0	1	0	1	1	1	1	0	0	1	1	1	92,36	77000000	10
0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	98,61	76750000	12
1	1	1	0	1	1	1	1	1	1	1	0	1	1	0	100	75000000	12

Tabla 26: Lotes de piezas con mayor cantidad de material utilizado. Fuente: Elaboración propia.

El lote que cuenta con menor cantidad de material es aquel que consigue alcanzar un 100% del área disponible. La cantidad de material empleada es de 75000000 mm^3 que frente a los 79500000 mm^3 del primer lote de la tabla, es 4500000 mm^3 menos. En cambio, el porcentaje de ocupación es un 4,86% superior. Si analizamos el lote pieza a pieza, en el lote con 100% ocupación contamos con las piezas p1 y p2 que no están en el lote con 95% de ocupación. La pieza p15 ocupa el lugar de estas en el lote de 95%. Si comparamos las dimensiones de estas piezas, p1 y p2 cuentan con unas dimensiones de 200x200 y una atura de 100, mientras que p15 cuenta con una base de 250x250 y una altura de 200. Aunque el espacio de la base formada por el conjunto de piezas p1 y p2 es superior al abarcado por la p15, cuando calculamos volúmenes, p1 y p2 ocupan un total de 8000000 mm^3 frente a los 12500000 m^3 de la p15 por si sola.

El objetivo de este apartado era demostrar que la altura de las piezas influye en el subproblema de selección del ganador. Queda demostrado que es una variable a tener en cuenta y que piezas con gran altura pueden reportar mayores beneficios que otras que cuenten con un área superior.

5. Conclusiones

5.1. Objetivos alcanzados

Al comienzo del trabajo nos planteamos una serie de objetivos. En este momento, estamos en condiciones de revisar si hemos conseguido cumplir estos objetivos y evaluar el éxito del trabajo.

El objetivo primordial de este trabajo era proponer una solución al problema de planificación de la producción diaria de los fabricantes que formarán parte del proyecto Lonja3D. Para ello, se ha desarrollado un programa en Python que permitirá a los fabricantes detectar el lote que reportará mayores beneficios.

En el primer bloque del trabajo se ha realizado un breve análisis de las tecnologías de impresión 3d existentes. Se han presentado sus ventajas y desventajas frente a las tecnologías de producción más tradicionales. Este análisis nos ha permitido comprender la importancia de la impresión 3d en el mercado de productos con alto grado de personalización.

La planificación de la producción en plantas con tecnologías de fabricación aditiva presenta varias diferencias frente a las tecnologías tradicionales. Esto implica un cambio en la forma de programar los lotes de fabricación y planificar su producción. Por ello, se ha realizado un estudio de la breve literatura que existe actualmente acerca de la planificación de la producción en plantas con tecnologías de fabricación aditiva. De esta forma hemos sido capaces de entender cuáles eran los aspectos fundamentales a tener en cuenta a la hora de elegir un lote de fabricación.

Gracias a todo el conocimiento adquirido en este primer bloque hemos sido capaces de redefinir el problema de planificación de la producción y ajustarlo a los requerimientos que presentarán los fabricantes de la plataforma Lonja3D.

De esta primera etapa de estudio acerca de impresión 3d y planificación de la producción hemos obtenido valiosas lecciones que se exponen a continuación:

- La mayor ventaja de la impresión 3d frente a tecnologías de fabricación tradicionales es la posibilidad de obtener piezas con alto grado de personalización y costes relativamente bajos.

- Mediante la fabricación de forma simultánea de piezas en una impresora 3d podemos reducir considerablemente los costes individuales de las piezas. Por ello, el objetivo principal del fabricante debe ser optimizar el área de fabricación disponible en cada una de las impresoras 3d.

- El margen de beneficio de un fabricante dependerá de la cantidad de material empleado en un lote de producción por encima del porcentaje de área ocupado.

De la etapa de formación en Python y las pruebas realizadas con el programa propuesto hemos obtenidos diferentes conclusiones. A continuación, se exponen las más relevantes:

- El lenguaje de programación utilizado por Python es sencillo y fluido. Esto ha permitido que el aprendizaje de este lenguaje desde cero sea rápido y cómodo.

- Python es un lenguaje que cuenta con una cantidad de librerías inmensa. Estas librerías han permitido que el programa desarrollado cuente con lectura de datos desde archivos Excel, salida de datos en Excel y la creación de imágenes que representen la distribución de las piezas en la plataforma de fabricación. De esta forma, no sólo obtenemos la solución que reporta mayor beneficio al fabricante, sino que además podemos guardar los datos de todos los posibles lotes y comparar.

Por último, gracias a las pruebas que enfrentaron a nuestro programa con diferentes casos reales de planificación de la producción en una impresora 3d podemos extraer las siguientes conclusiones:

- El programa se comporta mejor cuando parte de listas de piezas ordenadas de mayor a menor área que en el orden contrario. Esto permitirá comenzar con una solución de mejor calidad cuando se mejore el programa con búsqueda de óptimos locales en futuros trabajos.

- El hecho de ordenar las piezas de forma aleatoria permite generar diferentes lotes de forma rápida pero no permite explorar las proximidades de las soluciones con mayor ocupación. Hay un amplio margen de mejora en este aspecto y deberán desarrollarse algoritmos más óptimos en pasos futuros.

- Cuando los pedidos cuentan con un orden de entre 30 y 40 piezas, los resultados obtenidos son de calidad. Se encuentra un gran número de combinaciones con altos porcentaje de ocupación y el tiempo requerido es de 5-10 minutos. Lo cual no supone ningún problema para la planificación diaria e incluso puede aumentarse el número de simulaciones en busca de mejores soluciones.

- En pedidos que cuentan con aproximadamente 100 piezas, el espacio de soluciones crece estrepitosamente. En estas situaciones, el programa no obtiene soluciones tan buenas como las presentadas por otras técnicas. Además, obtener buenas soluciones implica realizar un alto número de simulaciones, dando lugar a tiempos de computación superiores a los 15 minutos y fallos en memoria por parte del programa si se guarda un número excesivo de lotes posibles.

- El factor porcentaje de relleno de las piezas es clave en el problema de elección de ganador. Además de ser una característica distintiva de las piezas fabricadas por impresión 3d, la fabricación de piezas con alto porcentaje de relleno puede dar lugar a situaciones con un menor porcentaje de ocupación de la cama de fabricación disponible pero mayor utilización de material.

- La altura de las piezas influye de la misma forma que lo hace el relleno. Dando lugar a situaciones con menor porcentaje de área ocupado pero mayor cantidad de material empleado. Siendo esta segunda opción la que reportará mayores beneficios al fabricante.

En vista del cumplimiento de los objetivos planteados y de las lecciones aprendidas que hemos recogido, podemos considerar este trabajo un éxito. Por lo tanto, una vez que se ha presentado una primera aproximación del programa que resolverá el problema de elección del lote que se fabricará en cada máquina de impresión 3d, es el momento de proponer futuros trabajos.

5.2. Trabajos futuros

El programa resuelve de forma eficaz el problema presentado. Hasta ahora la finalidad perseguida era ofrecer una primera aproximación del programa que gestionará la asignación de lotes de piezas a las máquinas de los fabricantes. Con esta premisa ya cumplida, podemos plantearnos su optimización.

A lo largo del proyecto se han detectado ciertas deficiencias en el programa que se deben tratar de solventar en futuros trabajos. Entre las más relevantes podemos destacar las que exponemos a continuación:

- El orden de la lista de piezas se crea de forma aleatoria en cada simulación. Cuando encuentra una solución mejor que la anterior el programa no realiza ningún tipo de búsqueda de óptimos locales. Esto perjudica la obtención de soluciones cercanas al óptimo. Sería interesante eliminar parte de la aleatoriedad en la generación de las listas con el fin de optimizar la búsqueda del lote óptimo.

- No se tienen en cuenta tiempos de set-up, procesado ni postprocesado de piezas. El tiempo de fabricación de la pieza es una variable que puede marcar las diferencias entre elegir un lote de piezas u otro y en este trabajo no se tiene en cuenta de forma directa. Será conveniente incorporar al modelo algoritmos que sean capaces de calcular el tiempo total necesario para fabricar el lote según se asignen unas piezas u otras.

- El modelo presentado resuelve el problema de empaquetado en dos dimensiones, dejando de lado la altura de las piezas. Fabricar piezas con gran altura y piezas muy bajas de forma simultánea impide que podamos extraer aquellas que son más bajas hasta la finalización de todo el lote. De nuevo se pueden realizar mejoras del modelo presentado añadiendo criterios de penalización para aquellos lotes en los que la diferencia de altura sea excesiva. De esta forma, trataremos de agrupar piezas con alturas similares con el fin de optimizar la capacidad productiva de las máquinas.

- Por último, el programa está preparado para resolver el problema de planificación de la producción en una sola máquina. Los fabricantes contarán con máquinas de diferentes dimensiones y velocidades de impresión que podrán ser capaces de fabricar piezas. Con la solución aportada en este trabajo permitimos al usuario realizar la asignación de lotes de forma individual. Una propuesta para trabajos futuros consiste en implementar la asignación de los lotes-máquinas de forma simultánea en todas las máquinas del fabricante.

6. Bibliografía

3D Works (2019). Available at: https://www.3dworks.cl/post/porcentaje-de-relleno-en-impresión-3d.

De Antón Heredero, J. (2018) 'Puesta en servicio del laboratorio de impresión 3D del Grupo de Investigación INSISOC'.

Berman, B. (2012) '3-D printing: The new industrial revolution', *Business Horizons*. 'Kelley School of Business, Indiana University', 55(2), pp. 155–162. doi: 10.1016/j.bushor.2011.11.003.

Canellidis, V. *et al.* (2006) 'Pre-processing methodology for optimizing stereolithography apparatus build performance', *Computers in Industry*, 57(5), pp. 424–436. doi: 10.1016/j.compind.2006.02.004.

Canellidis, V., Giannatsis, J. and Dedoussis, V. (2013) 'Efficient parts nesting schemes for improving stereolithography utilization', *CAD Computer Aided Design*. Elsevier Ltd, 45(5), pp. 875–886. doi: 10.1016/j.cad.2012.12.002.

Chergui, A., Hadj-Hamou, K. and Vignat, F. (2018) 'Production scheduling and nesting in additive manufacturing', *Computers and Industrial Engineering*. Elsevier, 126(September), pp. 292–301. doi: 10.1016/j.cie.2018.09.048.

Cui, Y. (2005) 'An exact algorithm for generating homogenous T-shape cutting patterns', *Computers & Operations Research*, 34(4), pp. 1107–1120. doi: 10.1016/j.cor.2005.05.025.

Cui, Y. P., Cui, Y. and Tang, T. (2015) 'Sequential heuristic for the two-dimensional bin-packing problem', *European Journal of Operational Research*. Elsevier B.V., 240(1), pp. 43–53. doi: 10.1016/j.ejor.2014.06.032.

Cui, Y., Yao, Y. and Cui, Y. P. (2016) 'Hybrid approach for the two-dimensional bin packing problem with two-staged patterns', *International Transactions in Operational Research*, 23(3), pp. 539–549. doi: 10.1111/itor.12188.

Fleszar, K. (2013) 'Three insertion heuristics and a justification improvement heuristic for two-dimensional bin packing with guillotine cuts', *Computers and Operations Research*. Elsevier, 40(1), pp. 463–474. doi: 10.1016/j.cor.2012.07.016.

Fleszar, K. and Charalambous, C. (2011) 'Average-weight-controlled bin-oriented heuristics for the one-dimensional bin-packing problem', *European Journal of Operational Research*. Elsevier B.V., 210(2), pp. 176–184. doi: 10.1016/j.ejor.2010.11.004.

Jakobs, S. (1996) 'On genetic algorithms for the packing of polygons', *European Journal of Operational Research*, 88(1), pp. 165–181. doi: 10.1016/0377-2217(94)00166-9.

Li, Q., Kucukkoc, I. and Zhang, D. Z. (2017) 'Production planning in additive manufacturing and 3D printing', *Computers and Operations Research*. Elsevier Ltd, 83, pp. 1339–1351. doi: 10.1016/j.cor.2017.01.013.

LifeHacks3D (2019). Available at: https://lifehacks3d.com/ciencia-y-tecnologia/infill-ultimaker-cura/.

Mirledy Toro, E., Garcés, A. and Ruiz, H. (2008) 'Solución al problema de empaquetamiento bidimensional usando un algoritmo híbrido constructivo de búsqueda en vecindad variable y

recocido simulado Two dimensional packing problem using a hybrid constructive algorithm of variable neighborhood search and s', *Facultad de ingeniería, Universidad de Antioquia*, 46, pp. 119–131.

Mirledy Toro, E. and Granada Echeverri, M. (2007) 'Problema de empaquetamiento rectangular bidimensional tipo guillotina resuelto por algoritmos genéticos', (35), pp. 321–326.

Piili, H. *et al.* (2015) 'Cost Estimation of Laser Additive Manufacturing of Stainless Steel', *Physics Procedia*. Elsevier B.V., 78(August), pp. 388–396. doi: 10.1016/j.phpro.2015.11.053.

Polyakovsky, S. and M'Hallah, R. (2009) 'An agent-based approach to the two-dimensional guillotine bin packing problem', *European Journal of Operational Research*, 192(3), pp. 767–781. doi: 10.1016/j.ejor.2007.10.020.

Wegener, K., Spierings, A. and Rickenbacher, L. (2013) 'An integrated cost-model for selective laser melting (SLM)', *Rapid Prototyping Journal*. Emerald, 19(3), pp. 208–214. doi: 10.1108/13552541311312201.

Zhang, Y. *et al.* (2016) 'Feature based building orientation optimization for additive manufacturing', *Rapid Prototyping Journal*. Emerald Group Publishing Ltd., 22(2), pp. 358–376. doi: 10.1108/RPJ-03-2014-0037.

Zhang, Y. and Bernard, A. (2013) 'Generic build time estimation model for parts produced by SLS', *High Value Manufacturing: Advanced Research in Virtual and Rapid Prototyping*, (February 2016), pp. 43–48. doi: 10.1201/b15961-10.

Zhang, Y., Gupta, R. K. and Bernard, A. (2016) 'Two-dimensional placement optimization for multi-parts production in additive manufacturing', *Robotics and Computer-Integrated Manufacturing*. Elsevier, 38, pp. 102–117. doi: 10.1016/j.rcim.2015.11.003.

Zhou, L. *et al.* (2018) 'Multi-task scheduling of distributed 3D printing services in cloud manufacturing', *International Journal of Advanced Manufacturing Technology*. The International Journal of Advanced Manufacturing Technology, 96(9–12), pp. 3003–3017. doi: 10.1007/s00170-017-1543-z.