UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

# DESARROLLO DE PRODUCTO DE OPTIMIZACIÓN PARA LA EMPRESA 1D SOLUTIONS EN BASE A ALGORITMOS DE INTELIGENCIA COMPUTACIONAL

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO

MARCELO ENRIQUE JIMÉNEZ DA FONSECA

PROFESOR GUÍA:
ANDRÉS CABA RUTTE

MIEMBROS DE LA COMISIÓN:
FRANCISCO RIVERA SERRANO
FRANCISCO CASADO CASTRO

Este trabajo ha sido parcialmente financiado por:
1D Solutions

SANTIAGO DE CHILE
2022

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: MARCELO ENRIQUE JIMÉNEZ DA FONSECA
FECHA: 14/03/2022
PROF. GUÍA: ANDRÉS CABA RUTTE

# DESARROLLO DE PRODUCTO DE OPTIMIZACIÓN PARA LA EMPRESA 1D SOLUTIONS EN BASE A ALGORITMOS DE INTELIGENCIA COMPUTACIONAL

El presente documento corresponde al informe del trabajo de título del alumno Marcelo Jiménez. En el informe se presenta la introducción, marco teórico, metodología, desarrollo, evaluación de los resultados y se comentan las principales conclusiones. El trabajo de título se destina a desarrollar un prototipo de alta resolución de un nuevo producto de optimización para la empresa 1D Solutions en base a algoritmos de inteligencia computacional.

La empresa 1D Solutions es una empresa digital de optimización que entrega a un público global una herramienta de optimización de corte lineal. En el presente informe se realiza una caracterización cuantitativa de la empresa y sus clientes, usando la información presente en los canales de ventas de la empresa. Se crea una página web, con tres prototipos de baja resolución de las alternativas de productos a desarrollar y se hace una campaña de *mail marketing* para obtener una validación temprana del prototipo final. En base a esta experimentación se define el producto a desarrollar: un optimizador web de cortes guillotinados en 2 dimensiones con un método de optimización inteligente.

Definido el producto, se construye el prototipo de alta resolución, integrando recursos de optimización externos basados en árboles de búsqueda, con el desarrollo de una interfaz de usuario, desplegada en una plataforma web. Se desarrolla el método de optimización inteligente, que interpreta los archivos de entrada y configura, automáticamente, los recursos de optimización para entregar uno o más resultados ajustados a la variante del problema de optimización que el usuario desea resolver.

Luego, se realiza una extensa comparativa entre el prototipo final y otros software comerciales, se evalúa el prototipo final en base a una rúbrica y se realizan 2 experiencias de usuario. Finalmente, se comentan las principales conclusiones del trabajo y se lista el trabajo futuro.

*Al hombre que me enseñó a pensar, que me dio preguntas,*
*que me hizo valorar el esfuerzo y que me mostró el camino,*

***a mi padre.***

# Agradecimientos

Le agradezco primero a mi madre, a quien le debo todo lo que soy, todos mis éxitos, toda mi felicidad, todo el amor que soy capaz de dar y recibir, se lo debo a ella, a su fuerza, a su corazón abnegado, a su vida que dedicó a nuestra familia. Le agradezco también a mi hermana, por ser mi amiga, por quererme, por ayudarme y por comprenderme cuando nadie más puede.

El mundo está en las personas, y mi mundo está en mis amigos, en aquellos que me acompañaron durante este recorrido universitario y que marcaron quien soy hoy. A Matías Villela, que me enseñó la gallardía, la astucia y que me acompañó codo a codo durante toda la carrera, gracias amigo por estar siempre ahí. A Ignacio del Río, mi amigo del colegio, mi socio en proyectos y mi compañero de viaje, las aventuras que vivimos las llevo bajo la piel, y a ti te llevo en el corazón. A mi grupo de amigas, Isadora, Francisca, Daniela, Sofía, Rocío y Javiera, gracias por ser las primeras que me acogieron en la facultad, gracias por todas las risas, por todas las fiestas, por todos los viajes a Papudo, por todos los buenos momentos que vivimos juntos y por los recuerdos que me traen alegría cuando más me hace falta. A Lucas Glasner, gracias amigo por todas las conversaciones interesantes, por ayudarme a escapar de lo superfluo, por los atardeceres en el mar y las aventuras en las olas. A Esteban Paredes, mi amigo de alma paralela y caminos divergentes, gracias por enseñarme a confiar en mi, por mostrarme la rebeldía, por hacerme parte de tus aventuras y ampliar mi mundo con el tuyo.

También quiero agradecer a los académicos y funcionarios que marcaron mi paso por esta institución. Le agradezco a mi profesor guía Andrés Caba, por ayudarme a encontrar mi pasión en los proyectos de innovación y emprendimiento, y guiar mi camino hasta el fin de mi carrera. A Francisco Casado, por confiarme como su sucesor en Beauchef Proyecta y por sembrar un prolífico camino futuro en la facultad. A Fernando Nowajewski, por su apoyo y mentoría en mis primeros proyectos de emprendimiento, logros que son el cimiento sobre el que me paro hoy a vislumbrar el futuro, y finalmente le agradezco a la dirección de relaciones internacionales por el intercambio que me cambió la vida.

Pasados ya 2 años desde el fin de la presencialidad, no puedo recordar sin gran nostalgia la facultad y la hermosa comunidad de la que fui parte, agradezco mi paso por la rama de natación y la rama de Taekwondo, donde aprendí el coraje y la resilencia. Al grupo de circo que me inició en una de mis grandes pasiones. Le agradezco a todos con quienes compartí risas los días viernes, con quienes salimos adelante en tareas y trabajos. Le agradezco a todos los que hicieron posible este hermoso camino universitario que hoy termina.

# Tabla de Contenido

# Índice de Tablas

# Índice de Ilustraciones

# Capítulo 1

# Introducción

## 1.1. Motivación

1D Solutions es una empresa digital de optimización de corte lineal fundada en 2001. La empresa ofrece un programa de escritorio que optimiza el corte lineal de materiales unidimensionales (perfiles metálicos, tubos, vigas, barras, etc), entregando un informe con los cortes a realizar y reduciendo la cantidad de material sobrante desechado. 1D Solutions tiene un alcance global, llegando con su producto 1D Nest a más de 2000 clientes en 78 países y en 4 idiomas distintos. Los principales clientes del producto 1D Nest son empresas relacionadas a la metalurgia y/o manufactura de estructuras metálicas.

A pesar del sostenido éxito comercial de la empresa 1D Solutions, gracias al posicionamiento, eficacia y facilidad de uso de su producto estrella 1D Nest, hoy la empresa se empieza a quedar atrás de su competencia. Canales de marketing modernos y nuevos productos de optimización le han otorgado a las empresas competidoras una mayor participación del mercado objetivo. Dado el potencial que tiene la empresa 1D Solutions, se proyecta este trabajo de título en desarrollar el prototipo de un nuevo producto de optimización que le permita crecer sus ventas y permanecer competitivo.

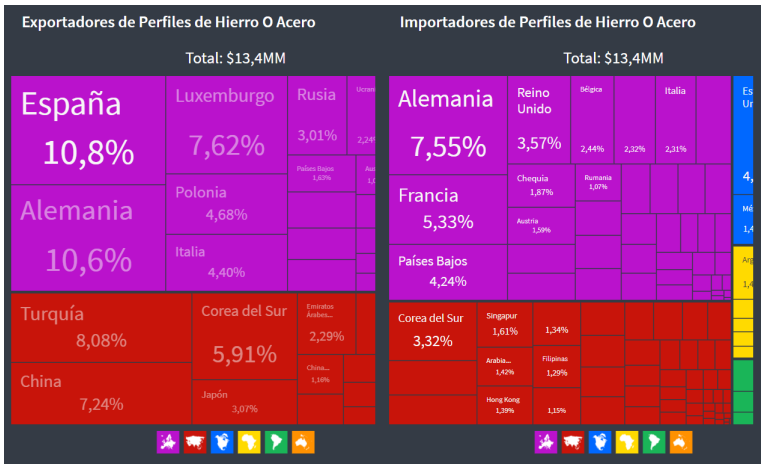
## 1.2. Antecedentes de la empresa y del mercado

El modelo de negocios de 1D Solutions en torno a su producto 1D Nest es un modelo *freemiun*, donde los clientes pueden descargar el programa con todas sus funcionalidades por 30 días, para luego tener que pagar por una licencia de por vida. Existen 3 tipos de licencias: individuales, empresa o adicionales. Las licencias individuales están sujetas a un computador; para instalar el programa en otro dispositivo se debe comprar una licencia adicional, que tiene un precio menor por ser ya cliente o comprar la licencia de empresa, que permite ser instalada en todos los computadores que se desee sin restricción. 1D Solution también vende la librería *CutLib*, que permite integrar el motor de optimización en otros sistemas o programas externos a la empresa.

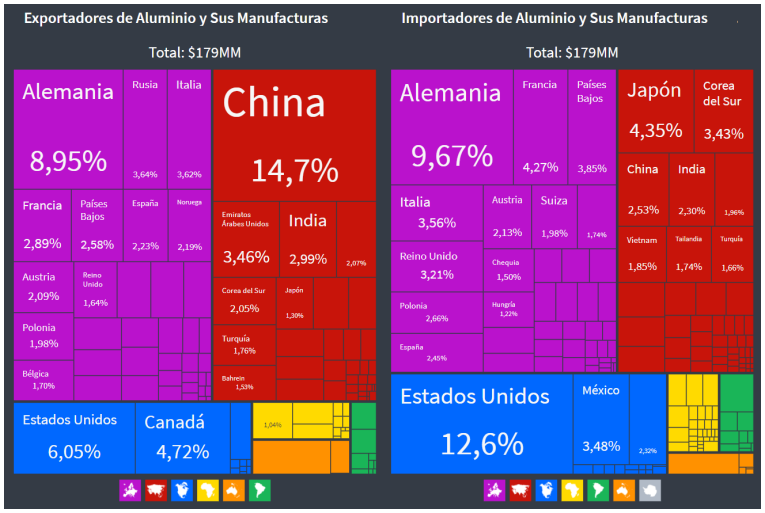
1D Solutions hoy en día tiene presencia en 78 países y genera aproximadamente 50.000 USD de ventas al año. Sus campañas de marketing están orientadas a América del norte

y Europa, donde destaca Estados Unidos como el país que lidera en ventas y en costos de adquisición de clientes, seguidos por Australia. La mayor cantidad de descargas se producen en Argentina y Brasil, sin embargo, sus tasas de conversión a pago son bajas, pues sólo usan los 30 días de prueba gratis.

El comercio mundial de perfiles de acero o hierro sin alear se estima en 13.4 miles de millones de dólares el 2019. Si se considera, además, el comercio mundial de tubos y tuberías de acero o hierro (24.9 MM), barras de acero aleado(6.85 MM), barras de acero sílico-manganoso (251 M) y barras de acero rápido (656 M), se puede estimar que la industria comerció en el 2019 cerca de 46.000 millones de dólares. Siguiendo la cadena de valor, se tiene que las industrias dedicadas a la manufactura de estructuras de acero y aluminio, a nivel mundial, comerciaron en 2019 64.900 millones de dólares. Con China y Alemania como los principales actores del mercado, que juntos acaparan el mayor porcentaje de participación de las exportaciones a nivel mundial, como se puede apreciar en la figuras 1.1 y 1.1.b [1].




(a) Exportaciones e importaciones mundiales de perfiles de hierro sin aleación durante el año 2019




(b) Exportaciones e importaciones mundiales de Aluminio y sus manufacturas durante el año 2019

Figura 1.1: Exportaciones e importaciones mundiales del mercado objetivo de 1D Solutions [1]

## 1.3. Descripción del problema

El problema a resolver en este trabajo de memoria, es la falta de diversificación de productos de la empresa 1D Solutions, que recibe el 98 % de sus ingresos gracias a su único producto 1D Nest. Esta es una de las causas por la cual la empresa ha visto un estancamiento de ventas anuales, estimadas en 49000 USD.

Como solución se propone el desarrollo de un nuevo producto de optimización, comenzando desde la concepción del producto hasta la entrega de un prototipo de alta resolución. La propuesta de valor es el desarrollo de un método de optimización inteligente, que permita a usuarios sin conocimientos de optimización ni programación, resolver distintas variantes del problema de corte y empaquetado de manera simple e intuitiva, sin tener que lidiar con configuraciones avanzadas.

## 1.4. Objetivos

El objetivo general de este trabajo de título es diseñar y construir un prototipo de alta resolución de un nuevo software de optimización para la empresa 1D Solutions, incorporando algoritmos de inteligencia computacional.

Los objetivos específicos son los siguientes:

1. Caracterizar cuantitativamente a la empresa y sus clientes.
2. Validar tempranamente el desarrollo mediante una experimentación con prototipos de baja resolución de las diferentes alternativas.
3. Obtener recursos de optimización funcionales en base a métodos de inteligencia computacional.
4. Desarrollar una interfaz de usuario y código de integración que permitan utilizar los recursos de optimización.
5. Implementar un método de optimización inteligente que permita configurar los recursos de optimización de manera automática, permitiendo abstraer al usuario final de la complejidad y configuraciones avanzadas del proceso de optimización.
6. Validar el prototipo de alta resolución desarrollado.

## 1.5. Resultados esperados

Finalizado el trabajo de título, se contará con:

1. Breve investigación de la empresa 1D Solutions y sus clientes, que entregue información útil para definir las características del prototipo de software de optimización a desarrollar.
2. Software de optimización con las siguientes características:

a) Resuelve una o más variantes del problema de corte y empaquetado, en 1, 2 o 3 dimensiones.

b) Posee uno o más algoritmos de optimización en base a un subconjunto de los siguientes métodos de inteligencia computacional:

- Algoritmos Genéticos
- Algoritmos de Árbol de búsqueda

c) Optimiza en un tiempo razonable (menor a 30 segundos) para el uso profesional de la herramienta.

d) Entrega resultados comparables con respecto a otros software disponibles en el mercado.

e) Cuenta con un método de optimización inteligente que lidia con las configuraciones avanzadas necesarias para las distintas variantes del problema de optimización, permitiendo su uso por parte de usuarios sin experiencia en optimización.

f) Cuenta con una interfaz de usuario simple que permite:

- Subir información de materiales y cortes.
- Ajustar parámetros de optimización.
- Ejecutar un optimización de cortes.
- Visualizar los planos resultantes de corte.

3. Tablas comparativas que muestren el desempeño del software frente a otros optimizadores de la competencia.

# Capítulo 2

# Marco teórico

## 2.1. Nomenclatura

Para comunicar con claridad, se procederá a fijar una nomenclatura básica de términos que se repetirán con mucha frecuencia a lo largo del trabajo de memoria:

- **Bin**: Se refiere a la materia prima usada en un problema de corte y empaquetado. En el caso de una dimensión, un bin se refiere a la viga, tubo u otro elemento unidimensional que será cortado para obtener ítems. En el caso de dos dimensiones, un bin se refiere a una plancha, lámina, tabla u hoja de material a ser cortada.
- **Ítem**: Pieza que se desea obtener a partir del corte de un bin. Usualmente, son varios de diferentes dimensiones y el trabajo del proceso de optimización es hacer un arreglo de estos ítems sobre el bin para reducir pérdidas o costos.
- **Recursos de optimización**: Apodo que denota de manera genérica tanto a algoritmos de optimización específicos, como a implementaciones de software que incluyan varios algoritmos de optimización y parámetros configurables.

## 2.2. Algoritmos de optimización

Un algoritmo de optimización se puede formular de la siguiente manera:

Tabla 2.1: Estructura de un algoritmo de optimización.

**Algoritmo 1** Estructura general de un algoritmo de optimización

**Requiere:** Función objetivo $f$
  $x^{(0)} \leftarrow$ punto aleatorio en el dominio de $f$
  **for** $i$ = 1,2, ... **do**
    $\Delta x \leftarrow \pi(f, x^{(0)}, ..., x^{(i-1)})$
    **if** se logra condición de detención **then**
      **return** $x^{(i-1)}$
    **end if**
    $x^{(i)} \leftarrow x^{(i-1)+\Delta x}$
  **end for**

En todo algoritmo de optimización, se parte desde una posición aleatoria en la función objetivo, el algoritmo iterativamente actualiza la posición computando cierto funcional $\pi$ que involucra la función objetivo, posición actual y la posición pasada. La diferencia fundamental entre los distintos tipos de algoritmos de optimización es la elección del funcional $\pi$ [9].

## 2.3. Problemas de corte y empaquetado

Los problemas de corte y empaquetado son un amplio grupo de problemas de optimización. Dada la variedad de disciplinas en las que aparecen, se les ha denominado de variadas formas, entre ellas:

- Problema de corte de stock y pérdidas de recorte
- Problemas de empaquetado (*bin packing*) , empaquetado de tiras (*strip packing*) y el problema de la mochila (*knapsack packing problem*)
- Problema de carga en vehículo, carga en pallets, carga en contenedores
- Problema del presupuesto de capital, asignación de memoria y problemas de planificación de multiprocesador [3, p.145]

Todos los problemas mencionados anteriormente se caracterizan por tener la misma estructura lógica:

- Hay 2 grupos de datos básicos cuyos elementos definen cuerpos geométricos de formas fijas en un espacio dimensional de uno o más números reales:
  - El stock de los llamados 'objetos grandes ' por un lado
  - La lista de los llamados ' ítem pequeños ' por el otro
- El proceso de corte o empaquetado realiza patrones, que son combinaciones geométricas de ítems pequeños asignados a los objetos grandes. Las piezas residuales son, usualmente, tratadas como 'pérdidas de recorte'.

Dado el rol dominante de los patrones geométricos en este tipo de problemas, se puede considerar a los problemas de corte y empaquetado como pertenecientes a la rama de combinatoria geométrica [3, pp. 147-148]. Si se consideran problemas donde se busca optimizar dimensiones espaciales, los problemas de corte y empaquetado se dividen en 2 ramas: corte de material o embalaje.

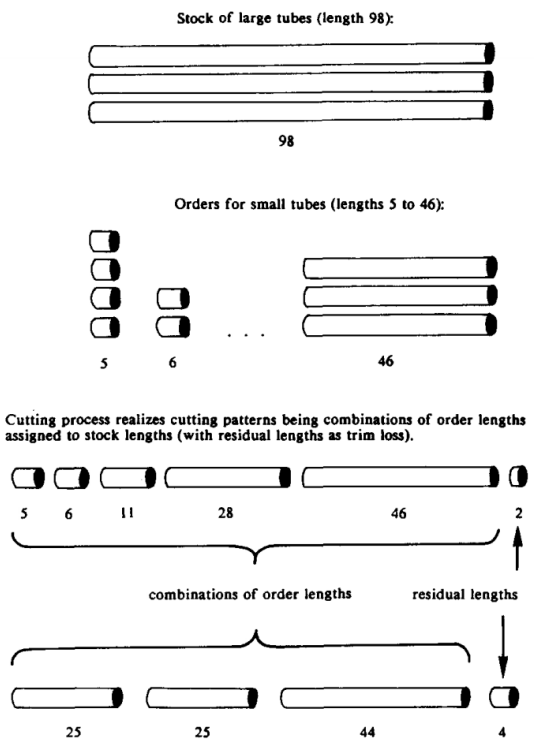
- Los problemas de corte son usualmente problemas de 1 o 2 dimensiones, donde se corta el material y se empacan las figuras en cierto espacio. Normalmente, se optimiza el corte para reducir lo más posible los residuos de recorte, o en otras palabras, el material sobrante de dimensiones menores a la pieza más pequeña. Ejemplos son el corte de vigas o planchas de material como acero, madera, textiles, etc. Dentro de estos problemas existen numerosas variantes: cortes con ángulos, cortes de guillotina, anidamiento de figuras irregulares, etc.

- Los problemas de embalaje son usualmente problemas que consideran las 3 dimensiones espaciales. En estos problemas se corta el espacio y se empaca el material, en contraposición al problema de corte mencionado anteriormente. Este tipo de problemas emerge en la carga de vehículos, pallets, contenedores, entre otros. En este caso se busca optimizar el uso de espacio, minimizando la cantidad de espacio no ocupado dentro del contenedor.

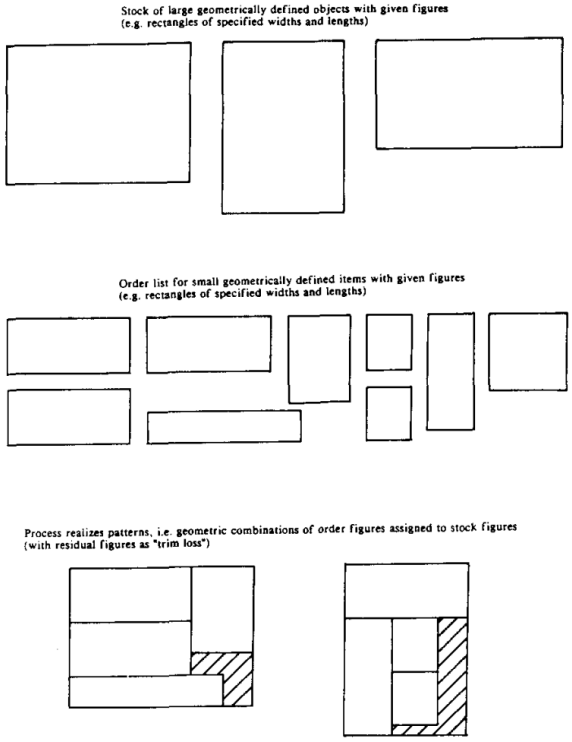
Si se consideran problemas que lidian con dimensiones no espaciales, encontramos una inesperada relación con problemas de áreas distantes a la manufactura de objetos o transporte. Dado que estos problemas escapan del contenido de este trabajo de memoria, no se detendrá en explicarlos detalladamente. Como ejemplos podemos nombrar la optimización del presupuesto de capital, donde se optimiza una dimensión financiera, la optimización de líneas de ensamblaje, donde se optimiza la dimensión temporal y el problema de asignación de memoria, donde se optimiza el espacio de memoria de un procesador.

El problema de la mochila, donde una serie de objetos con cierto peso o valor deben ser empacados en un contenedor, maximizando el valor total o minimizando el peso total, es la excepción ya que si bien se optimiza una dimensión no espacial (peso o valor) en la realidad se relaciona mucho con los problemas de corte y embalaje mencionados anteriormente [3].

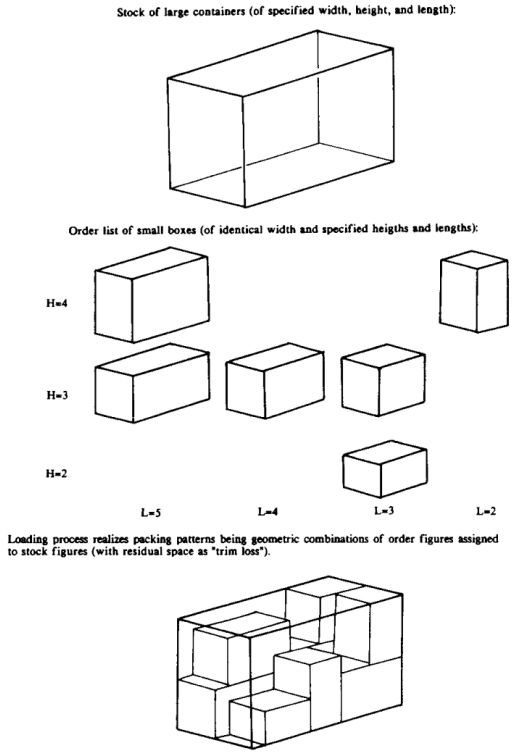
En la figura 2.1 podemos observar tres ejemplos de problemas de corte y empaquetado: en una dimensión, en dos dimensiones y en tres dimensiones.




(a) Ejemplo de problema de C. & P. en una dimensión




(b) Ejemplo de problema de C. & P. en dos dimensiones




(c) Ejemplo de problema C. & P. en tres dimensiones

Figura 2.1: Ejemplos de problemas de corte y empaquetado [3]

## 2.4. Técnicas de optimización

Los problemas de corte y empaquetado son en general, problemas de optimización combinatoria del tipo NP-completos, lo que implica que todos los algoritmos actualmente conocidos, que encuentran soluciones óptimas (exactas), requieren un número de pasos computacionales que crece exponencialmente con la escala del problema, en vez de crecer de acuerdo a una función polinomial [11]. En respuesta a la complejidad de los problemas se han desarrollado distintos algoritmos, que pueden ser clasificados como convencionales y no convencionales.

- Las técnicas convencionales buscan darle una solución exacta al problema, lo que en la mayoría de los casos, es una tarea difícil. En la programación lineal entera, rama matemática que abarca la resolución de este tipo de problemas, se encuentra que el problema combinatorio de la región factible de soluciones no es un conjunto convexo, por lo tanto se pueden tener muchos óptimos locales, lo que obliga a demostrar, mediante otro tipo de argumentos, cual de esas soluciones es el óptimo global. Hay un gran número de enfoques utilizados para resolver problemas de programación lineal entera; los más notables son: **enfoques enumerativos**, **técnicas de descomposición lagrangiana**, y **algoritmos basados en planos de corte** [6].

- Las técnicas no convencionales o heurísticas se basan en modelos extrínsecos o funciones objetivos desarrolladas, e intentan entregar una solución cercana al óptimo, sin asegurar su convergencia al óptimo global.[5, p.18]. Las técnicas heurísticas tienen mayor flexibilidad para tomar en consideración restricciones específicas a los problemas y entregan un buen balance entre la calidad de la solución y el esfuerzo computacional requerido.[4, p. 1]. Dentro de esta clasificación encontramos a los **algoritmos genéticos (GA)**, la **búsqueda tabú (TS)**, y el **recocido simulados (SA, simulated annealing)**[ 5, p.18].

### 2.4.1. Técnicas numerativas

La forma más intuitiva de resolver un problema de este tipo sería enumerar todas las soluciones factibles y elegir la mejor. Sin embargo, dada la naturaleza NP-Completa del problema, aumentar el tamaño o dimensión del problema puede hacer explotar la cantidad de soluciones factibles, lo que impide resolver de manera enumerativa. Una opción consiste en eliminar muchas soluciones argumentando aspectos de factibilidad y dominación. La técnica de Ramificación y Acotación (*Branch and bound*) es la más utilizada de los enfoques de enumeración. Consiste en generar un árbol de soluciones en la que cada una de las ramas que salen de un nodo conducen a una solución posterior a la del nodo. El método es capaz de detectar que ramas no son óptimas y las elimina del árbol. [6]

### 2.4.2. Descomposición y relajación Lagrangiana

La forma clásica de abordar los problemas de programación lineal entera es “relajar” las restricciones de integridad de las variable. El enfoque lagrangiano parte con un conjunto de las

restricciones “complicadas” del problema, que se añadirán a la función objetivo con una serie de multiplicadores de Lagrange, que se irán modificando de manera iterativa. Al eliminar las restricciones más complicadas del problema, se obtiene un subproblema que es mucho más fácil de resolver. Estos subproblemas se resuelven iterativamente hasta que se alcancen los valores óptimos para los multiplicadores. A este enfoque se le conoce como relajación de Lagrange. Por otro lado se tiene la técnica llamada descomposición de Lagrange, que consiste en aislar subconjuntos de restricciones de forma que se obtengan problemas fáciles de resolver para cada uno de estos subconjuntos. La mayor debilidad de estas técnicas de solución es que dependen fuertemente del problema y por lo tanto, no pueden ser fácilmente generalizables [6].

### 2.4.3. Algoritmos basados en planos de corte

Este tipo de algoritmo consiste en modificar el conjunto convexo del espacio de soluciones de la relajación lineal del problema, de forma que todos sus puntos extremos lleguen a ser enteros. Pese a los cambios en el espacio de soluciones, este debe permanecer convexo. Primero se resuelve la relajación lineal del problema y luego se encuentra un plano (desigualdad lineal), que corta la parte fraccional de la solución lineal, a la vez que asegura que todos los puntos enteros factibles cumplen con la desigualdad. Un plano de corte será una restricción válida deducida del conjunto de restricciones del problema. El algoritmo termina cuando se ha encontrado una solución entera al problema, o bien, la relajación lineal del problema no es factible y, por lo tanto, el problema entero tampoco, o bien, las últimas generaciones de planos de corte no mejoran la función objetivo, o bien, no es posible definir ningún plano de corte. El algoritmo de Ramificación y corte (*Branch and cut*) es un híbrido que incorpora esta noción de plano de corte. [6]

### 2.4.4. Búsqueda tabú (TS)

El algoritmo basado en búsqueda local llamado Búsqueda Tabú, desarrollado por Fred Glover en 1990, deriva su atractivo de la gran flexibilidad que posee y la facilidad de implementación en problemas de optimización combinatorios. El algoritmo TS inicia con una solución factible y se mueve paso a paso hacia un punto de solución mejorado. Se genera una muestra de vectores de decisión en la vecindad de la solución actual y se determina al mejor basado en una heurística. Se hace un movimiento desde el actual vector de decisión hacia un mejor vector de decisión que no esté en la lista tabú, lo que entrega, un valor mejorado de la función objetivo, vía simples modificaciones del vector de decisión actual. Una lista tabú contiene cierto número de los últimos vectores de decisión visitados. El mejor vector de decisión reemplaza al vector de decisión más viejo de la lista tabú, y al resto de los vectores sobrevivientes de la lista se les da un estado activo, que reduce el riesgo de repetir e bucle los mismos vectores de decisión [5, p. 24].

### 2.4.5. Recocido simulado

La técnica de recocido simulado, también llamada cristalización simulada o enfriamiento simulado, está basada en el concepto del modelamiento y simulación de un sistema termodinámico. La técnica parte con la selección de un vector de decisión de proceso aleatorio y se

mueve hacia una nueva vecindad de vectores de decisión, que mejoran el valor de la función objetivo. La técnica puede aceptar vectores de decisión inferiores basado en cierta medida probabilística para evitar caer en óptimos locales de una función de respuesta multimodal. La probabilidad de hacer un movimiento hacia un vector de decisión inferior (vector que genera un valor peor de la función objetivo) decrece mientras el valor del 'parámetro de temperatura', definido en el algoritmo, decrece. Esto último es análogo al enfriamiento lento en un proceso de recocido para alcanzar un estado cristalino perfecto [5, p.24].

### 2.4.6. Algoritmos genéticos

Los algoritmos genéticos son un método heurístico que permite optimizar una función objetivo sin conocer su forma analítica. Se inspiran en el proceso evolutivo de los organismos vivos presentes en la naturaleza, tomando, principalmente, los mecanismos de selección natural, reproducción y mutación.

Primero, se crea una población de individuos aleatorios, luego, se evalúa el desempeño de cada individuo, se selecciona a los mejores y se procede a mezclarlos y/o inducir mutaciones para obtener la próxima generación, finalmente, se procede a repetir el proceso. [11]

La estructura básica de un algoritmo genético es la siguiente:

- Cromosoma
- Individuo
- Función de Fitness
- Población
- Inicialización
- Selección
- Reproducción

Un **cromosoma** describe a una solución en particular, representa el plan de cómo se creará al individuo, o sus propiedades. Se puede codificar una solución dentro de un cromosoma, usando una representación binaria, o un set de números naturales, o reales. Normalmente, es mejor dividir un cromosoma en cromosomas más pequeños y que una solución esté representada por un conjunto de cromosomas.

Se esperan las siguientes habilidades de un cromosoma:

- Iniciarse aleatoriamente ( setear sus elementos aleatoriamente)
- Clonarse (crear una copia de sí mismo)
- Cruzarse con otro individuo
- Mutar (modificar sus elementos)

Un **individuo** es un grupo de cromosomas, que conforman el elemento principal de una población. El individuo es testeado y seleccionado en base a su rendimiento y/o calidad, dependiendo de la función de *fitness* que se le aplique.

La **función de *fitness*** simula al ambiente en el proceso evolutivo real: un tiburón tiene un muy buen rendimiento en el océano, sin embargo, como animal terrestre o aéreo, es un fracaso. La función de *fitness* define la calidad de los individuos. Debe ser simple y rápida, pues es necesario computar la calidad de cada individuo en cada generación, por lo tanto, su tiempo de ejecución se relaciona directamente con el tiempo que demora el algoritmo en encontrar un óptimo.

La **población** es un conjunto de individuos que se reproducen, mutan y compiten entre ellos por ser parte de la próxima generación. Determina el número máximo de individuos, cuántos nuevos individuos se generan en cada iteración y almacena al mejor de los individuos, incluso cuando ya fue reproducido y eliminado.

La **selección** es una de las funciones clave de los algoritmos genéticos: remueve individuos viejos, remueve individuos de mala calidad, selecciona a la élite y remueve individuos duplicados. Remover individuos viejos permite escapar de óptimos locales, donde los individuos seleccionados para reproducirse siempre son los mismos y la evolución se estanca. La remoción de malos individuos permite el mejoramiento progresivo de la población. Es posible que se generen 2 o más individuos con los mismos cromosomas, los que apuntan a la misma solución, para aumentar la calidad y la diversidad "genética" de la población; estos son removidos.

La **reproducción** es el premio que ganan los mejores individuos de cada generación. Hay distintas formas de reproducir a un individuo: se puede transferir directamente a la nueva generación sin cambios, se le pueden aplicar mutaciones, que son pequeñas alteraciones en sus cromosomas, o se pueden recombinar los cromosomas de 2 individuos para generar un tercero totalmente nuevo.

### 2.4.7. Algoritmo A* de memoria acotada (MBA*)

En inglés: *Memory bounded A** y en adelante referido como MBA* , es un algoritmo desarrollado por Libralesso y Fontan [8], para el concurso de corte de vidrio 2018 ROADEF/EURO. MBA* es un algoritmo búsqueda de árbol de cualquier momento (*anytime tree search algorith*m), lo que significa que representa el espacio de soluciones con una estructura de árbol y en cada iteración produce una solución factible mejor a la anterior, por lo tanto, el algoritmo puede ser detenido en cualquier momento. El algoritmo MBA* explora el árbol de soluciones poniendo prioridad en las ramas más interesantes, ya que explorarlo por completo no es factible dada la naturaleza NP-completa del problema.

El algoritmo parte con una cola que contiene únicamente el nodo raíz. En cada iteración, el algoritmo MBA* extrae el "mejor" nodo de la cola y sus hijos son añadidos a la cola. Si el tamaño de la cola supera cierto umbral, los "peores" nodos son descartados. Se parte con un umbral de 2 y, cada vez que la cola se vacía, se parte de nuevo con el umbral multiplicado por un factor de crecimiento $f$. Se parte con un umbral de 2 y los autores definieron al factor

de crecimiento $f = 1.5$. La función que determina los "mejores" o "peores" nodos es llamada función guía y se cuenta con varias definiciones para las distintas variables del problema de optimización. Las variantes de empaquetado se enfocan en el porcentaje de pérdida mientras que la variante de la mochila incluye el peso de cada solución.

## 2.5. Otros software de optimización

A continuación se nombran y se describen las principales empresas que proveen software comerciales que fueron identificados y revisados durante el periodo de elaboración de este informe.

### 2.5.1. Magic Logic

Software de optimización 3D que consiste en 3 principales aplicaciones: Cube-IQ, Cube-IQ-Web y BlackBox. Cada una diseñada para un uso especifico.

**Cube-IQ**: Planeador de carga de escritorio. Recorta costos de envíos por barco.

**Cube-IQ-Web**: Planeador de carga web, diseñado para uso en celulares y tablets.

**BlackBox**: Plug-in para cartonización y paletización compleja mixta; se añade a los sistemas ERP/WMS (Software de administración).

### 2.5.2. Alpacku

Servicios de embalaje de contenedor 3D para optimizar la eficiencia de envíos. Minimiza costo y área superficial, volumen, peso del contenedor y riesgo de daños accidentales. Algoritmo en base a aprendizaje de máquinas y algoritmos genéticos. Apunta a industrias de envíos de *e-commerce*, contenedores de carga y carga en camiones.

### 2.5.3. TMachines

Software de corte en 1 y 2 dimensiones.

**CutLogic1D**: Optimizador de corte lineal, cuenta con optimización por pérdidas y por costo. Gráficos de planos de corte. Inventario de material usado. Reportes predefinidos y con capacidad de edición. Importación desde MS Excel, CSV, TXT, MS Access, portapapeles y conexión a bases de datos (MS SQL Server, MySQL, Oracle, Firebird, etc.) Exportación a MS Excel, PDF, CSV, TXT, RTF, HTML, RazorGage y puertos seriales.

**CutLogic2D**: Optimizador de corte de paneles 2D. Optimización por pérdidas y por costo. Corte de guillotina de 2 y 3 etapas y anidamiento de paneles. Permite sección de agarre para corte de vidrio. Optimización por grupos, optimización de parámetros para piezas remanentes. Inventario de stock de materiales y piezas remanentes. Importación desde MS

Excel, CSV, TXT, MS Access, portapapeles y conexión a bases de datos (MS SQL Server, MySQL, Oracle, Firebird, etc.) Exportación a MS Excel, PDF, CSV, TXT, RTF, HTML, AutoCAD DXF, G-Code para máquinas CNC.

### 2.5.4. Real Cut 1D

Software de corte lineal con opción de corte de ángulos distintos a 90 grados en los extremos.

### 2.5.5. OptiCutter

Software de corte lineal online, aplicación web que no necesita instalación y procesa la información en la nube.

### 2.5.6. CutMaster 2D

Software de optimización 2D, anidamiento de piezas rectangulares, optimización por pérdida de material y costo, cálculo de banda del borde, control de la dirección de grano. Importación de Excel o archivos XML. Exportación a AutoCAD DXF y archivos XML. Reportes detallados, con piezas etiquetadas.

### 2.5.7. NestandCut

Optimización 2D especializada en anidamiento de piezas irregulares para corte en CNC. Aplicación web online. Limpieza automática de geometrías DXF o DWG. Lectura y mejora del nesting DXF o DWG realizado con otro sistema. Generación automática del orden de recorte de piezas.

# Capítulo 3

# Metodología

A continuación, en la figura 3.1, se presenta la metodología a seguir durante este trabajo de memoria:


Figura 3.1: Metodología del trabajo de título

## 3.1. Comprender

La metodología comienza por comprender el desempeño de la empresa, su relación con clientes y las características principales de sus clientes. Se investigarán los canales de marketing de la empresa y la información histórica de ventas. Se busca comprender características demográficas y etarias del público objetivo de 1D Solutions. También, se desea cuantificar los volúmenes de ventas mensuales, anuales y por región geográfica, para descubrir características que puedan ayudar a definir el nuevo producto a desarrollar.

## 3.2. Experimentar

Luego se procederá a experimentar con usuarios. En base a la etapa anterior y a la opinión de la empresa, se generan distintas alternativas de los prototipos a desarrollar. Se construirán prototipos de baja resolución y de muy bajo costo, que serán presentados a clientes o a personas que cumplan con el perfil del público objetivo de la empresa. De esta interacción, se busca obtener evidencia empírica del interés por cada prototipo y descubrir las características más interesantes del producto a desarrollar para los futuros usuarios.

En paralelo a estas dos etapas, se hará una búsqueda de implementaciones de algoritmos de optimización, que por sí solos, o combinados, permitan resolver distintas variables del problema de corte y empaquetado en 1, 2 o 3 dimensiones.

## 3.3. Definir

En base a los aprendizajes obtenidos en las etapas anteriores, y tomando en consideración también los recursos de optimización disponibles, se definirá en detalle las características del prototipo final y se dará paso a la construcción de este.

## 3.4. Construir

La construcción del prototipo final de alta resolución, consta de 2 grandes etapas: el desarrollo de la interfaz gráfica y la implementación de recursos de optimización externos.

La **interfaz gráfica** permite a los usuarios interactuar con el programa usando botones y formularios, sin necesidad de ejecutar ni hacer configuraciones desde la consola de comandos del sistema operativo. Dado que las características del prototipo final son definidas durante el desarrollo del trabajo de título, se barajan distintas opciones para el desarrollo de la interfaz. En el caso de desarrollar un programa de escritorio, la primera opción considerada es el *framework* QT, que permite, desde un único código base, extender el uso a todas las plataformas existentes (Windows, Android e IOS). Los principales lenguajes que se utilizan en esta plataforma son C++ y QML. En el caso de una aplicación web, la primera opción considerada para el desarrollo es el *framework* Django, que entrega una serie de funcionalidades útiles para hacer un desarrollo rápido y escalable. Los principales lenguajes que se utilizan en esta plataforma son Python, JavaScript, CSS y HTML.

La implementación de **recursos de optimización** externos dependerá mucho de cuál o cuáles se elijan integrar, decisión que se llevará a cabo en la etapa de definición del prototipo. Al inicio de este trabajo de título, la empresa puso a disposición un software de optimización basado en algoritmos genéticos. En paralelo a las primeras dos etapas de la metodología, se hace una búsqueda de otros recursos de optimización que permitan construir el software sin necesidad de implementar un algoritmo matemático de optimización directamente, aunque, dependiendo de la cantidad y calidad de los recursos de optimización encontrados, no se descarta la posibilidad de hacer este desarrollo.

Una vez integrada la interfaz de usuario con los recursos de optimización, se tendrá terminada la primera parte del prototipo final, dando paso al desarrollo que aporta mayor valor en este trabajo de título: el método de optimización inteligente.

## 3.5. Desarrollar

El **método de optimización inteligente** es el aporte central del trabajo de título, y consta de todo el desarrollo necesario para configurar de manera óptima, automática y específica los recursos de optimización frente a los distintos problemas de optimización que el usuario desee resolver. De esta manera, este método de optimización deberá tomar decisiones inteligentes a partir de la información que provea el usuario en los archivos de entrada. Entre las decisiones a tomar se considera identificar el tipo de variante del problema de optimización a realizar y seleccionar la mejor configuración de los recursos de optimización. Todo lo anterior con el fin de abstraer al usuario final de las configuraciones avanzadas que necesita cada tipo de optimización y permitirle resolver distintos problemas de optimización de manera simple e intuitiva.

## 3.6. Evaluar

Finalmente se comparan y evalúan las características claves del prototipo final. Se hará una extensa comparación del prototipo contra otros software de optimización comerciales. Se busca comprender las ventajas del prototipo contra su competencia y también las carencias, para guiar el trabajo futuro. Se realizarán experiencias de usuario con clientes o personas que cumplan el perfil del público objetivo de 1D Solutions. De estas interacciones se busca validar tanto al prototipo como al método de optimización inteligente, y también, encontrar espacios de mejora para el trabajo futuro.

# Capítulo 4

# Comprender cuantitativamente a 1D Solutions y sus clientes

De acuerdo a la metodología, se parte el trabajo de memoria analizando la información histórica de ventas y descargas de 1D Solutions, para entender a los clientes de la empresa y, además, comprender también el desempeño de la empresa en búsqueda de pistas que permitan diseñar el nuevo producto de acuerdo a la relación ya existente entre la empresa y sus clientes.

Para obtener esta información se consultó a la empresa MyCommerce, que gestiona la venta del producto. Se pudo obtener la información histórica de ventas de la última década. También se investigó en la plataforma Google Ads, que es el canal principal de marketing de 1D solutions. En esta plataforma se pudo obtener información detallada del público objetivo que mayor interacción y conversión tiene con la empresa.

A continuación se presentan las figuras 4.1, 4.2, 4.3 y 4.4, que detallan los volúmenes de ventas y descargas de la empresa por área geográfica y por producto.

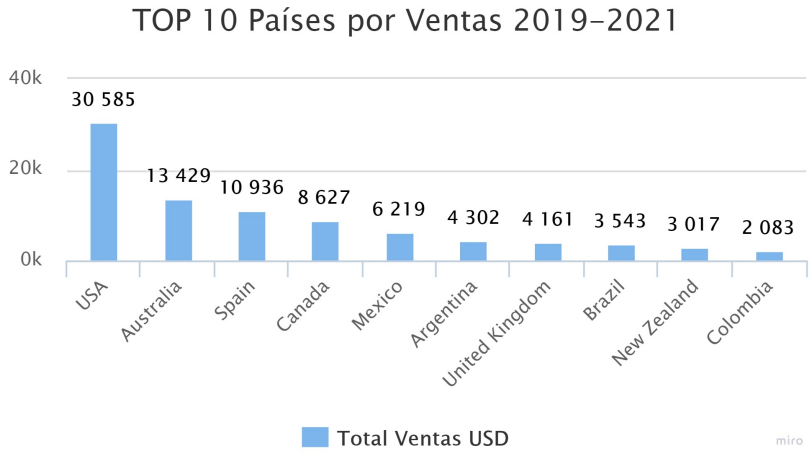



Figura 4.1: Volumen de ventas en USD de los 10 países con mayores ventas.

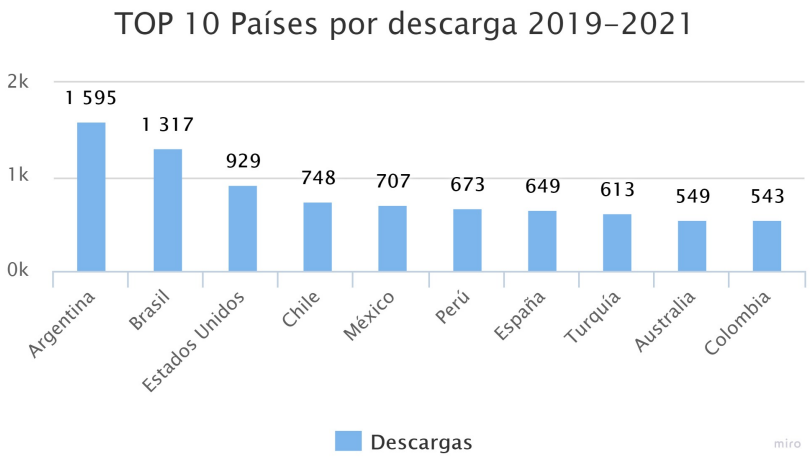



Figura 4.2: Cantidad de descargas de los 10 países con mayor cantidad.

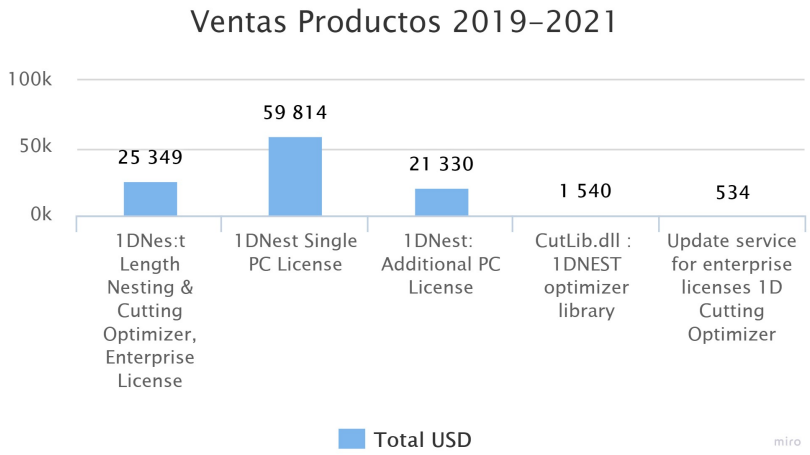



Figura 4.3: Volumen de ventas en USD de los productos y servicios ofrecidos.

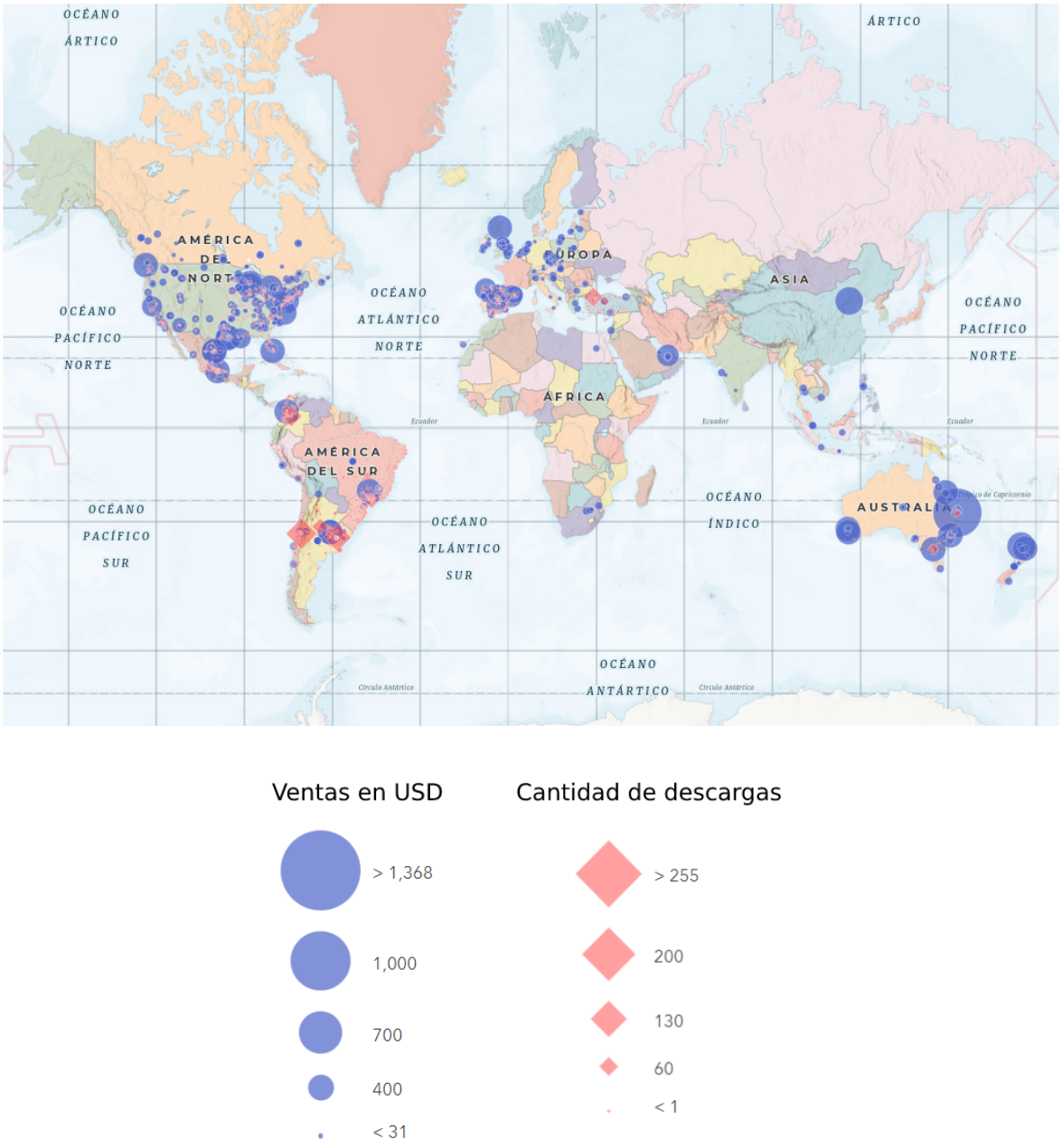
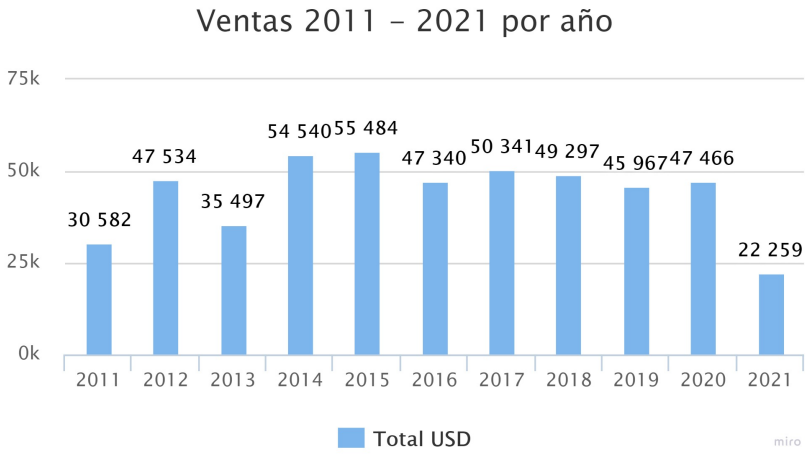



Figura 4.4: Mapa de ventas y descargas combinado

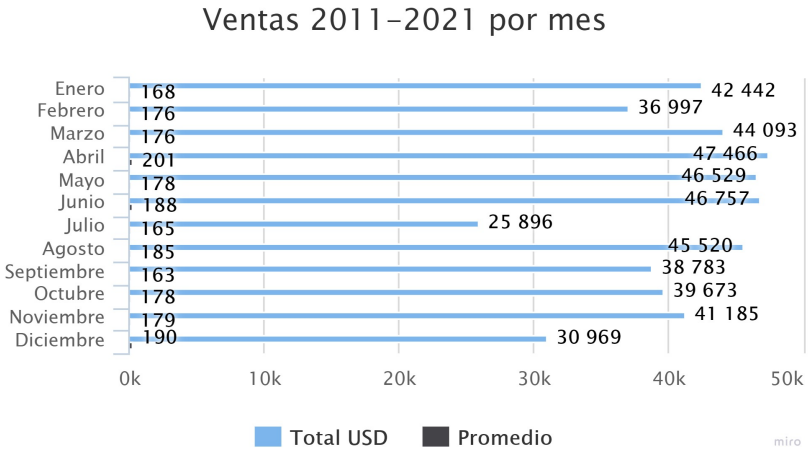
En las figuras 4.1 y 4.2 se puede observar que existe una diferencia significativa entre la descarga del producto y la venta. Se observa que dos países latinoamericanos, Argentina y Brasil lideran las descargas, sin embargo, es Estados Unidos quien domina las compras del producto. Esto muestra que existe una menor disposición a pago dentro de los países latinoamericanos, a pesar que la necesidad de optimización está presente. Analizando estos datos se puede concluir, primero, que el público más atractivo y con mejor disposición a pago es el público estadounidense, lo que ya determina al inglés como primer lenguaje a implementar en la interfaz de usuario (podría haber sido español también); decisión que también se apoya en el amplio abanico de regiones geográficas en donde se compra el producto (ver figura 4.4).

Proyectando los resultados se podría inferir que el modelo de precio que 1D Solutions está usando actualmente, no logra convencer a Argentina y Brasil y que, tal vez, para el nuevo producto a desarrollar, se pueda implementar un modelo de subscripción en vez de un sólo pago por una licencia de por vida, asumiendo como hipótesis que el precio de entrada es la barrera que abstiene a los potenciales clientes latinoamericanos.

Observando la figura 4.3, se confirma la problemática que da inicio a este trabajo de título que, prácticamente, el 100 % de los ingresos de la empresa corresponden a licencias del producto 1D Solutions, dejando a otras propuestas, como la librería CutLib y los servicios post-venta, al margen.



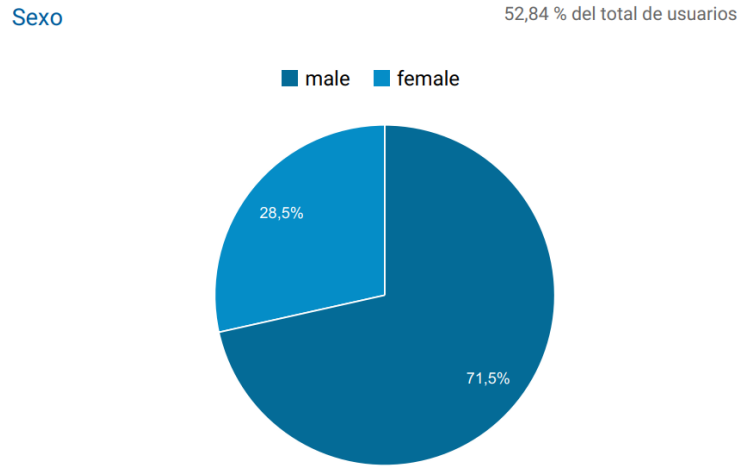

(a) Volumen de ventas en USD ordenado por año.



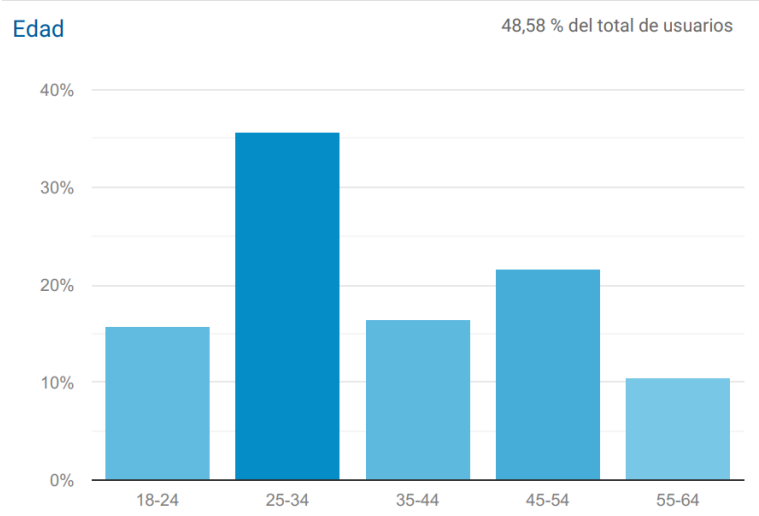

(b) Volumen de ventas en USD ordenado por mes.

Figura 4.5: Ventas de 1D Solutions en la última década.

Las ventas de la última década de 1D Solutions graficadas en la figura 4.5.a muestran un ingreso estable, lo que si bien no es malo, tampoco es ideal considerando que 8 años de trabajo no han logrado elevar las ventas. Es interesante notar en la figura 4.5.b que el mes de Julio constantemente registra la menor cantidad de ventas, probablemente porque coincide con el inicio del verano en muchos países del hemisferio norte. Lo mismo para diciembre con el hemisferio sur, donde también vemos una caída en las ventas.




(a) Distribución de género de público objetivo anglo-hablante.




(b) Distribución etaria de público objetivo anglo-hablante.

Figura 4.6: Características demográficas del público objetivo de 1D Solutions.

La distribución de género del público anglo-hablante, que es el público más numeroso y relevante, mostrada en la figura 4.6.a, denota una clara tendencia hacia usuarios hombres. Si bien no existen importantes características de la empresa o del prototipo a construir que tengan que ver con género, es relevante saberlo. La distribución etaria de la figura 4.6.b muestra que el rango más predominante es el de adultos jóvenes (25-34 años), lo que se contradice con la presunción de la empresa que su producto era mayoritariamente usado por adultos (45 - 54 años). Este dato es relevante en función de la estética y usabilidad de la interfaz de usuario que mejor aceptación tendrá dentro del público objetivo. Se apunta a una estética moderna, fluida, responsiva y minimalista, propia de las interfaces de usuario modernas.

# Capítulo 5

# Experimentación con usuarios y definición del prototipo

## 5.1. Alternativas del prototipo

A partir de conversaciones con 1D Solutions, la lluvia de ideas de los productos que podrían ser desarrollados durante la memoria se redujo a 3 opciones, ordenadas según el interés y entusiasmo mostrado por la empresa:

- Optimizador de corte de guillotina en 2 dimensiones para industrias de corte de vidrio.
- Plataforma web de optimización en 1 dimensión que integre el software 1D Nest ya existente.
- Optimizador en 3 dimensiones para industria de almacenamiento y logística.

La opción que más apoyo tiene es el optimizador en 2 dimensiones, puesto que muchos de sus clientes perteneces a la industria de manufactura de ventanas, donde usan el optimizador 1D Nest para optimizar el corte de los perfiles que sostienen las ventanas. Sin embargo, usan optimizadores de otras compañías para aprovechar el corte de los vidrios. Además, el corte en guillotina usado en esa industria (corte de extremo a extremo del material), es una de las variantes 'simples' del problema de optimización en 2 dimensiones, comparado, por ejemplo, con la compleja labor de anidamiento y corte de piezas irregulares en máquinas CNC, lo que ayuda en la factibilidad técnica de llevar este trabajo a cabo.

La segunda opción es crear una plataforma web para el software ya existente de la empresa 1D Nest. Esta alternativa nace naturalmente de la idea de modernizar y ampliar la gama de productos de la empresa. Es la opción con menos riesgo, puesto que el software está más que validado y, además, presenta la oportunidad de renovar el modelo de negocios del producto, permitiendo cobrar subscripciones periódicas en vez de vender licencias de por vida. Sin embargo, tiene la desventaja de no apuntar hacia un nuevo mercado, si no que entra a competir con el software que ya existe de la empresa.

La tercera opción, es un optimizador en 3 dimensiones para la industria de almacenamiento y logística, donde un caso de uso podría ser optimizar el orden de paquetes dentro de

contenedores para sus viajes transoceánicos. Esta opción destaca por su innovación y proyección, puesto que existen relativamente pocas empresas en el mercado que ofrecen este tipo de solución y, además, se prevé un aumento mundial del comercio en línea o *e-commerce* y en consecuencia un aumento en el transporte y almacenamiento de bienes. No obstante, la complejidad técnica de construir dicho optimizador es muy alta, y el proyecto se considera más apropiado para un equipo de desarrollo con altas competencias en ingeniería matemática e informática, puesto que involucra directamente desarrollar un algoritmo de optimización en 3 dimensiones.

## 5.2. Experimentación con prototipos de baja resolución

Para obtener mas información sobre estas 3 opciones, se crearon **3 prototipos** de baja resolución, que podrían ser llamados incluso pruebas de concepto. Se desarrolló **una nueva página web** para desplegar estos prototipos, en la cual se incorporó un **formulario de subscripción** a un 50 % de descuento y se hizo una **campaña de *mail marketing*** para hacer los llegar esta promoción a los potenciales clientes. Las principales métricas a evaluar son el rendimiento de la campaña de mail y la cantidad de subscripciones a los formularios. A continuación, en las figuras 5.1,5.2, 5.3 y 5.4 se muestra la página web:

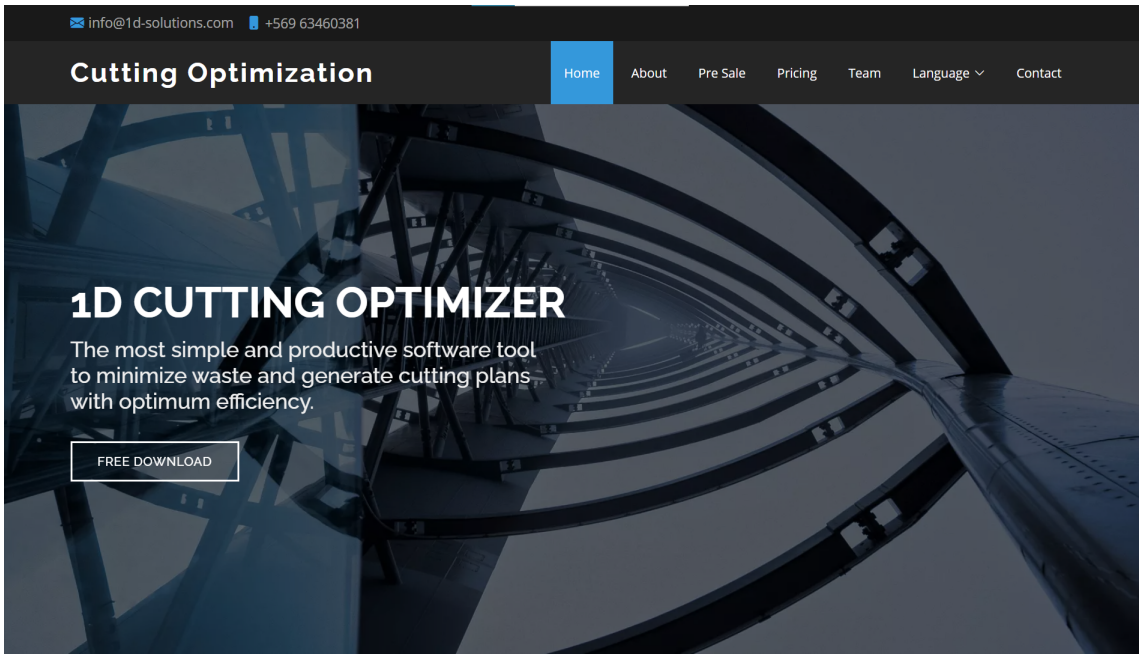



Figura 5.1: Inicio de la nueva página web para 1D Solutions www.1d-solver.com

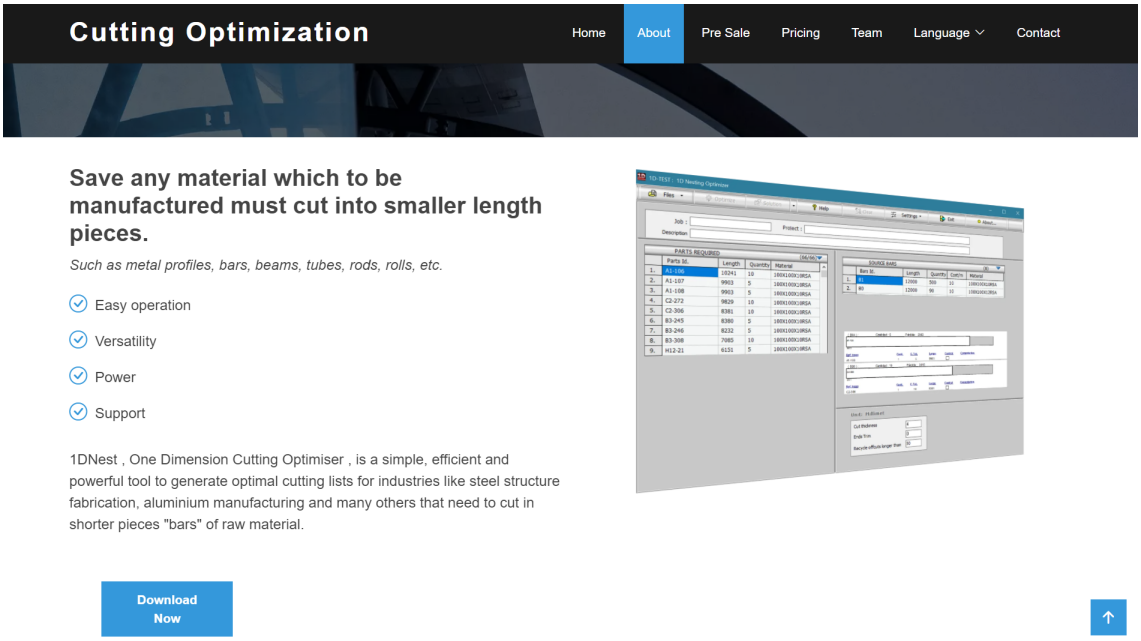



Figura 5.2: Acerca del producto 1D Nest en la nueva página web para 1D Solutions www.1d-solver.com

La nueva página (www.1d-solver.com) tiene un aspecto más moderno que la actual web (www.1d-solutions.com) y fue creada para poder experimentar y modificar libremente sin poner en riesgo el posicionamiento en Google de la página web principal de la empresa. En la página se permite descargar y comprar licencias del producto 1D Nest y, además, se incluye una sección de pre-venta (figura 5.3), donde se puede acceder a los 3 prototipos (figuras 5.4.a, 5.4.b, 5.4.c ) mencionados anteriormente:

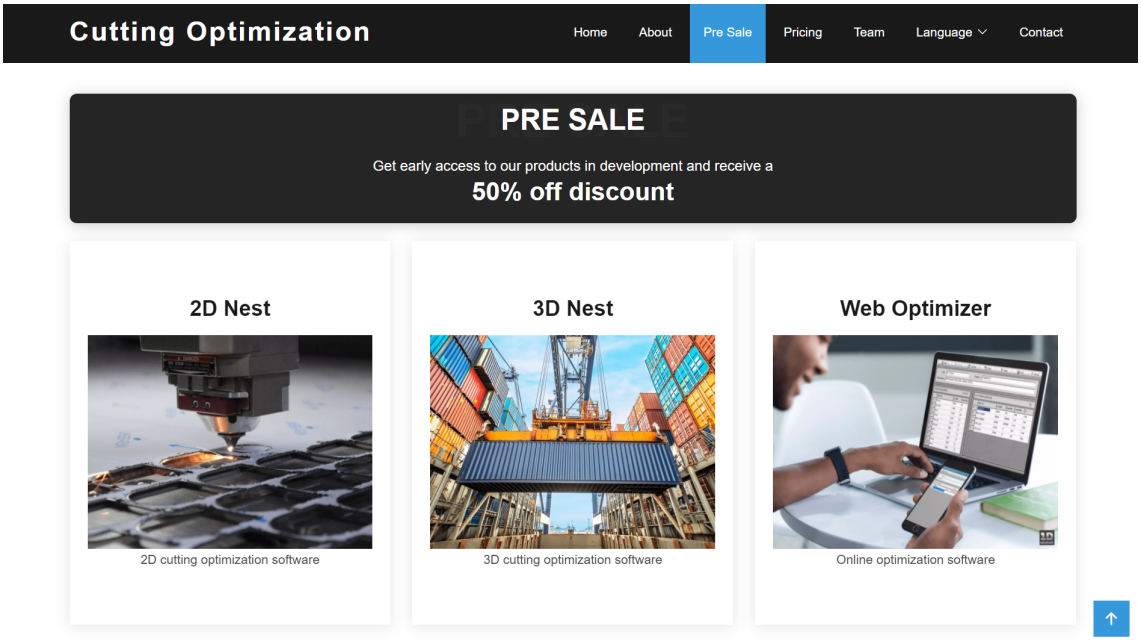
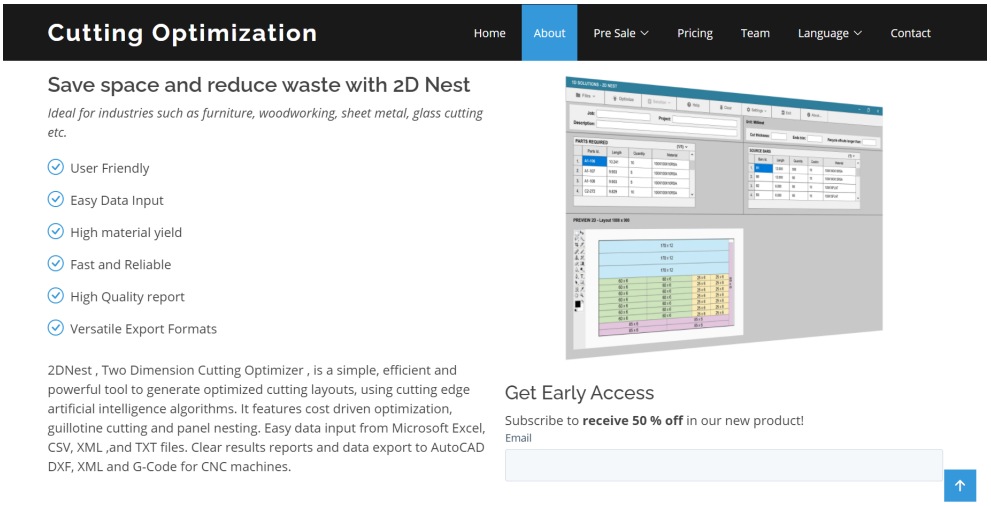
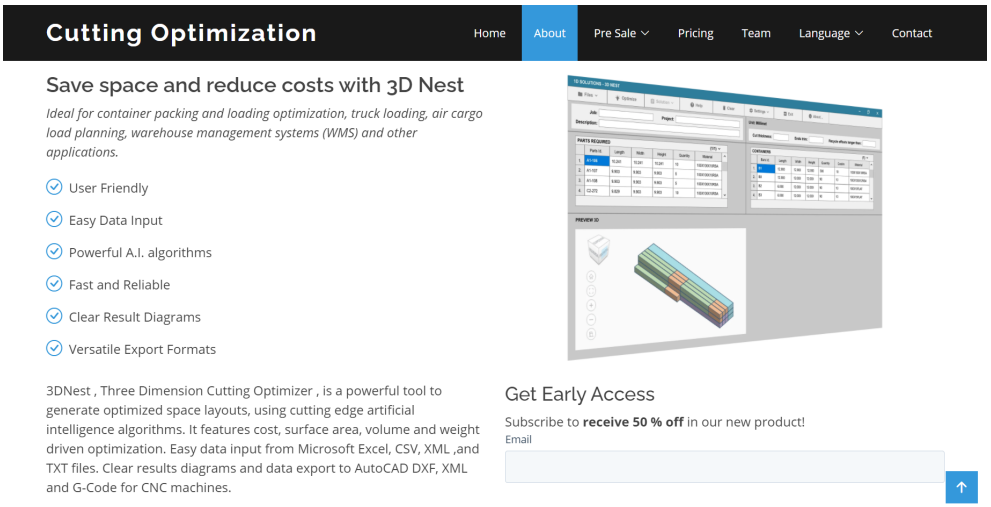


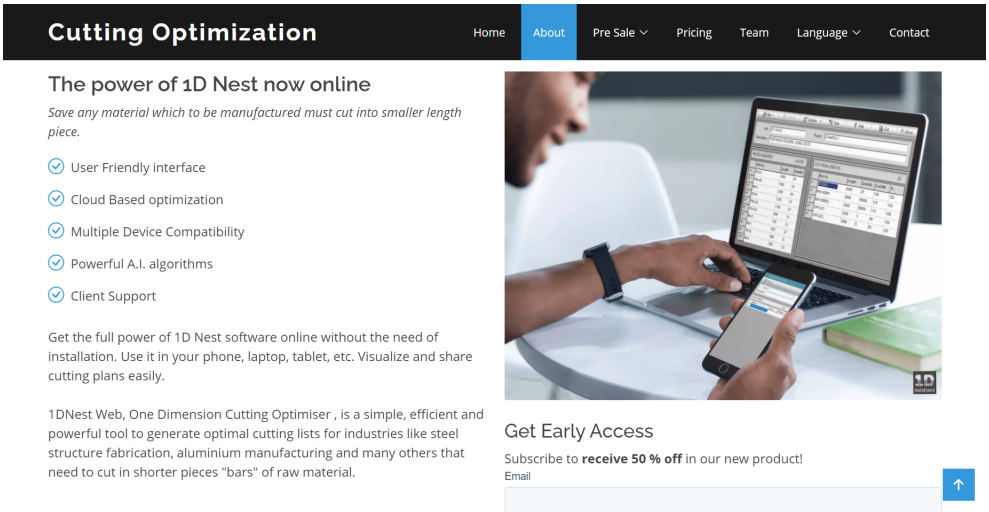

Figura 5.3: Sección de pre-venta de los prototipos en la nueva página web




(a) Prototipo de optimizador 2D http://1d-solver.com/en/2DOpti.html




(b) Prototipo de optimizador 3D http://1d-solver.com/en/3DOpti.html




(c) Prototipo de optimizador web 1D http://1d-solver.com/en/WebOpti.html

Figura 5.4: Prototipos de baja resolución para experimentación.

Para crear los prototipos del optimizador en 2 y 3 dimensiones se contrató a una diseñadora gráfica, que realizó las visualizaciones de la interfaz de usuario que se observan a la derecha de las figuras 5.4.a y 5.4.b. La visualización del optimizador web (figura 5.4.c) fue hecha por el autor. Las características que se describen a la izquierda de las imágenes son una recopilación de las funcionalidades y cualidades más interesantes que se observan en los software de la competencia. Esto con la idea de conseguir la mayor cantidad de subscripciones al acceso temprano del producto, que corresponde al formulario que se observa a la derecha abajo de las imágenes.

Usando la plataforma Hubspot se monitorean las nuevas subscripciones a los formularios, almacenando en una lista los correos de los clientes interesados y se creó la campaña de mailing mencionada anteriormente para darle a conocer a clientes antiguos de 1D Solutions la nueva página y los productos en desarrollo. Se eligió enviar un correo a los clientes que han tenido interacción con 1D Solutions hace máximo 2 años.

A continuación, en las figuras 5.5, 5.6 y 5.7 se muestra el mail enviado y los resultados de la campaña:

| NOMBRE | VISUALIZACIONES | TASA DE ENVÍO |
|---|---|---|
| 2D Suscripción Hub<br>Publicados \| Formulario clásico | 24 | 20,83% |
| Web Suscripción Hub<br>Publicados \| Formulario clásico | 21 | 9,52% |
| 3D Suscripción Hub<br>Publicados \| Formulario clásico | 14 | 7,14% |

Figura 5.5: Resultado de inscripción a formularios.

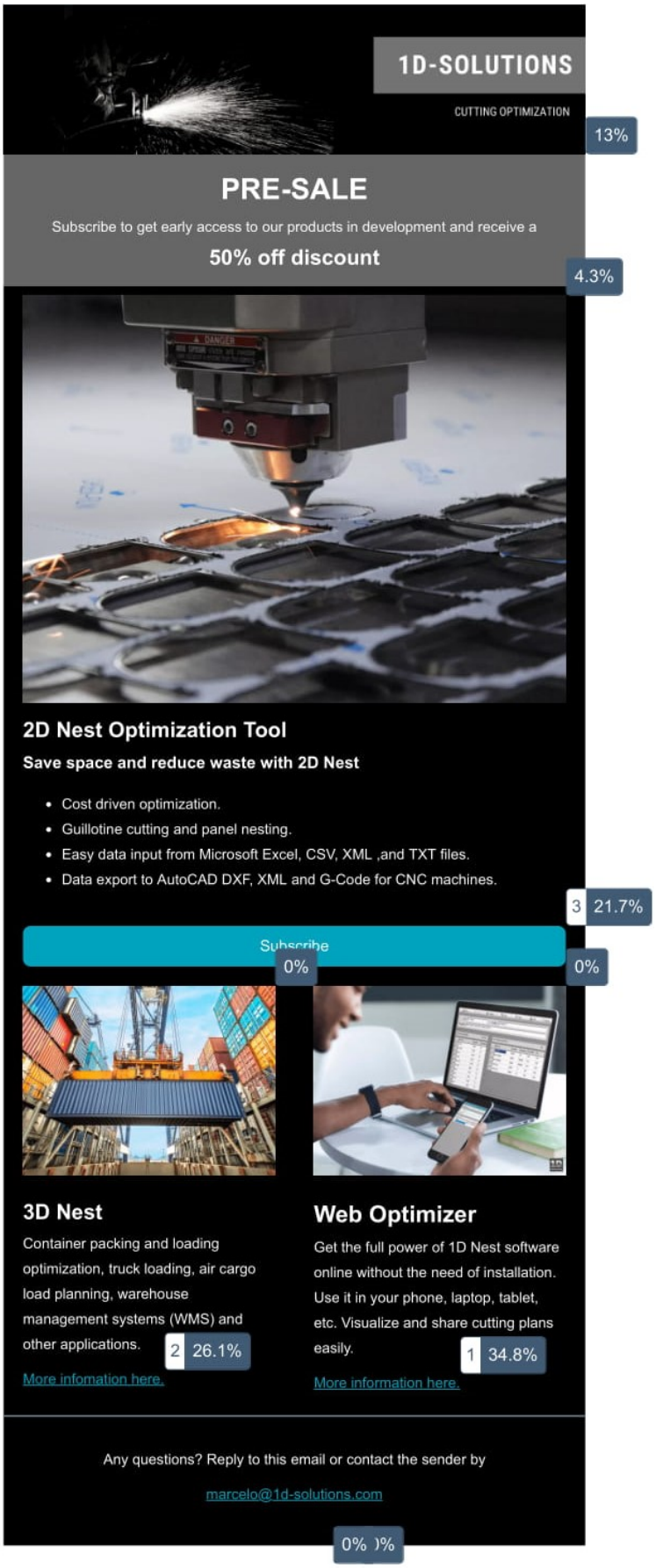



Figura 5.6: Mail enviado y porcentaje de clicks en distintos enlaces.


(a) Entrega


(b) Apertura


(c) Clicks

Figura 5.7: Resultados de campaña de mailing

Además de las cifras presentadas en las figuras 5.7.a, 5.7.b, 5.7.c, a la fecha se han recibido 4 solicitudes de clientes pidiendo mayor información sobre el nuevo producto de optimización en 2 dimensiones, con los cuales se coordinó la entrega de una versión beta del software para su uso gratuito a cambio de retroalimentación.

Según las estadísticas disponibles en MailChimp, una de las plataformas más grandes de email marketing del mundo, la tasa de apertura media en el sector de software y aplicaciones webs es de 21,29 % y la tasas de clicks es de 2,45 % [10]. Comparando con los resultados de la figura 5.7.b y 5.7.c se puede comprobar que el rendimiento de esta estrategia estuvo dentro de los parámetros previstos, incluso mostrando una mejor tasa de clicks que lo esperado, a pesar de que la campaña fue hecha en el mes de Julio, que según la figura 4.5.b, es el mes de menor actividad comercial para la empresa.

### 5.2.1. Discusión de resultados de experimentación

De los resultados se concluye que existe un claro interés por un producto de optimización en 2 dimensiones, puesto que el formulario de subscripción con mayor cantidad de entradas es el correspondiente a esta opción. Por otro lado, los clientes de 1D Solutions valoran un producto web, conclusión tomada a partir del alto porcentaje de clicks que recibió Web Optimizer (ver figura 5.6).

## 5.3. Definición del prototipo

En base al proceso de experimentación temprana con clientes, a la retroalimentación de la empresa y en vista de los recursos de optimización obtenidos (presentados en el capítulo siguiente), **se decidió que el software a construir será una aplicación web de optimización de corte guillotinado en 2 dimensiones**. Esta aplicación será accesible a través de un navegador web, estará albergada en un servidor por lo que los usuarios podrán acceder a ella sin necesidad de descargar ni instalar. Los archivos de los usuarios serán guardados en una base de datos y los recursos de optimización serán ejecutados en la nube. Esta decisión le da al programa una mayor compatibilidad con distintos sistemas operativos que una aplicación de escritorio, permite hacer actualizaciones a los clientes de manera instantánea, permite generar un modelo de negocios en base a subscripción, entrega la posibilidad de diseñar una interfaz responsiva para tener compatibilidad entre distintos dispositivos, y también, reduce los requisitos técnicos de los dispositivos del cliente, ya que la optimización no se ejecutará en su dispositivo local.

# Capítulo 6

# Construcción del prototipo

## 6.1. Recursos de optimización para el prototipo

Durante el desarrollo del trabajo de título, se buscaron recursos que pudieran servir de base para la obtención del primer prototipo funcional; un software capaz de resolver alguna variante del problema de corte y empaquetado. Finalizado este proceso, se cuenta con 2 alternativas:

### 6.1.1. Packing solver

Packing solver es un programa de corte guillotinado en 2 dimensiones diseñado por Libralesso y Fontan [8] para el concurso de corte de vidrios ROADEF / EURO 2018, en el cual la implementación de su algoritmo de corte ganó el primer lugar entre 64 participantes. El algoritmo de búsqueda que desarrollaron fue bautizado ‘ A* de memoria acotada’ (MBA* por sus siglas en inglés). Su funcionamiento está descrito en la sección de marco teórico de este informe. El repositorio del programa cuenta con una rigurosa comparación con otros algoritmos similares presentes en la literatura y demuestra entregar consistentemente resultados competitivos en términos de optimización de material y tiempo de ejecución.

Packing solver no solo incluye la implementación del algoritmo MBA*, si no que también ofrece una serie de mejoras y capacidades ampliadas que fueron desarrolladas después de la competencia ROADEF, entre las que se destacan la optimización por costo (*Knapsack problem*), optimización de tiras (*Strip Packing*), y una amplia gama de configuraciones y opciones. El programa cuenta con una licencia MIT libre que permite ser usado gratuitamente tanto para fines comerciales como particulares sin ningún tipo de restricción ni garantía. El programa no cuenta con una interfaz gráfica y, para ser ejecutado, se debe compilar usando BAZEL y ejecutar en el intérprete de comandos de Windows CMD, adjuntando las rutas de los archivos a cargar, especificando las rutas de los archivos de salida, y añadiendo, explícitamente, las configuraciones correspondientes al tipo de corte y optimización que se desea hacer. **El programa no cuenta con una guía de usuario**, solo cuenta con una somera documentación en su repositorio de GitHub, por lo que para usarlo se debe contar con conocimientos avanzados de programación y familiaridad con problemas de optimización matemática. El programa si cuenta con un visualizador en línea de los resultados de corte. https://github.com/fontanf/packingsolver

### 6.1.2. Genetic solver

Genetic solver (bautizado así por el autor de este informe) es un programa de corte de guillotina en 2 dimensiones desarrollado por Petr Horný en 2008 como trabajo de tesis para el master en Ingeniería eléctrica e informática de la universidad técnica checa de Praga [11]. Este paper fue traducido del checo a inglés por 1D Solutions y la empresa lo tiene hace años en carpeta para su eventual uso en un nuevo producto. Este software también cuenta con una licencia libre para fines comerciales.

El software funciona en base a un algoritmo genético de corte de guillotina de 4 etapas. Cuenta con una interfaz de usuario muy rudimentaria que permite ingresar cortes, ajustar parámetros y visualizar los resultados de la optimización, sin embargo, esta interfaz debe ser ejecutada a través de un archivo .bat donde se debe especificar la ruta al compilador de java y la ruta a las clases .JAVA correspondientes. Además, no compila bien con todas las versiones del Java SDK, por lo que se tuvo que investigar la versión original con la que fue compilada, se tuvo que descargar aquella versión, añadirla a las variables de entorno del sistema y luego modificar el archivo .bat para que pudiese ser ejecutado. Luego se encontraron problemas para cargar archivos .csv a la interfaz, lo que implica que un usuario común se vería obligado ingresar al programa una por una las piezas y planchas a cortar. Se modificaron los archivos fuente del programa para corregir los errores de carga y se volvió a compilar para, finalmente, obtener una interfaz funcional. Todo lo anterior se explica para dar cuenta que si bien es un algoritmo con interfaz, está lejos de ser un producto terminado y listo para entregárselo a un usuario real.

En el paper de Genetic solver [11] se detalla la construcción de 3 algoritmos genéticos que solucionan el problema de corte; cada uno es una versión mejorada del anterior. Se detallan también la comparación entre el algoritmo genético con otros programas de optimización de corte comerciales, en los cuales se observa un rendimiento bastante alto del optimizador.

### 6.1.3. Comparación de los recursos de optimización

El programa Packing solver cuenta con 5 algoritmos distintos de optimización (A*, DFS, IMBA*, DPA*, IBS) y el programa Genetic solver cuenta con 3 (G,B,C). Para entender sus rendimientos se estructuró una comparación.

Dado que el programa Packing solver consta de una variedad de funciones objetivo y configuraciones, se tuvo que ajustar poder ser comparado con la optimización de corte guillotinado de 4 etapas del programa Genetic solver. En particular, se configuró el programa Packing solver con la función objetivo *bin packing*, que busca reducir al máximo el desperdicio de material. Se configuró, además, un corte de guillotina de 3 etapas no exacto (se puede hacer un cuarto corte independiente a los ítems), lo que es comparable con el corte de guillotina de 4 etapas de Genetic solver. Se dio una orientación inicial aleatoria y se permitió rotaciones de ítems.

Se añadió al experimento de comparación un par de optimizadores comerciales, OptiCutter y Cut Optimal, con el objetivo de obtener una primera validación del rendimiento de los recursos de optimización. Dado que las versiones gratuitas de los programas presentan varias restricciones, no fue posible evaluar cada ejemplo del experimento.

A continuación se presentan las tablas 6.1, 6.2, 6.3, 6.4, 6.5 que comparan a todos los algoritmos respecto al porcentaje de pérdida, la cantidad de planchas de material usadas y el tiempo de convergencia, con distintas cantidades de piezas a cortar (ítems):

Tabla 6.1: Comparación de algoritmos con 12 ítems.

| Ej 1 12 ítems | Pérdida | Planchas | Tiempo [s] |
|---|---|---|---|
| A* | 21.08 % | 1 | 0.040981 |
| DFS | 29.76 % | 1 | 0.0037641 |
| IMBA* | 16.71 % | 1 | 0.0033293 |
| DPA* | 21.08 % | 1 | 0.023652 |
| IBS | 16.71 % | 1 | 0.0025557 |
| G | 3.60 % | 1 | 5 |
| B | 3.60 % | 1 | 5 |
| C | 3.60 % | 1 | 5 |
| Cut Optimal Pro | 3.29 % | 1 | 0,407s |
| OptiCutter | 2.98 % | 1 | |

Tabla 6.2: Comparación de algoritmos con 25 ítems.

| Ej 2 25 ítems | Pérdida | Planchas | Tiempo [s] |
|---|---|---|---|
| A* | - | - | - |
| DFS | 40.61 % | 4 | 10.0003 |
| IMBA* | 18.02 % | 3 | 0.0155745 |
| DPA* | - % | - | - |
| IBS | 18.22 % | 3 | 0.017999 |
| G | 14.38 % | 3 | 5 |
| B | 11.24 % | 3 | 5 |
| C | 4.55 % | 3 | 5 |
| Cut Optimal Pro | 13.04 % | 2 | 2,359 |

Tabla 6.3: Comparación de algoritmos con 50 ítems.

| Ej3 50 ítems | Pérdida | Planchas | Tiempo [s] |
|---|---|---|---|
| A* | - | - | - |
| DFS | 41.84 % | 4 | 0.004499 |
| IMBA* | 19.06 % | 3 | 0.0111296 |
| DPA* | - % | - | - |
| IBS | 19.06 % | 3 | 0.0145621 |
| G | 17.94 % | 3 | 10 |
| B | 15.94 % | 3 | 3 |
| C | 11.11 % | 3 | 10 |
| Cut Optimal Pro | 22.75 % | 3 | 1,593 |

Tabla 6.4: Comparación de algoritmos con 100 ítems.

| Ej4 100 ítems | Pérdida | Planchas | Tiempo [s] |
|---|---|---|---|
| A* | - | - | - |
| DFS | 30.13 % | 8 | 30.0029 |
| IMBA* | 6.75 % | 6 | 0.0516234 |
| DPA* | - % | - | - |
| IBS | 6.23 % | 6 | 0.0667187 |
| G | 27.86 % | 8 | 15 |
| B | 24.30 % | 8 | 17 |
| C | 12.65 % | 7 | 10 |
| Cut Optimal Pro | 6.86 % | 6 | 4,812s |

Tabla 6.5: Comparación de algoritmos con 500 ítems.

| EJ5 500 ítems | Pérdida | Planchas | Tiempo [s] |
|---|---|---|---|
| A* | - | - | - |
| DFS | 32.02 % | 41 | 30.0363 |
| IMBA* | 3.42 % | 29 | 30.0033 |
| DPA* | - % | - | - |
| IBS | 3.81 % | 29 | 30.0239 |
| G | 42.95 % | 49 | 28 |
| B | 39.42 % | 46 | 23 |
| C | 22.04 % | 36 | 40 |
| Cut Optimal Pro | 3.94 % | 29 | 35,172 |

#### 6.1.3.1. Discusión de resultados

Se le recuerda al lector que los algoritmos buscan reducir la pérdida y la utilización de material, por lo que en las tablas un porcentaje menor en la columna "Pérdida" y un número menor en la columna "Planchas" denotan un mejor resultado. El tiempo de convergencia fue difícil de comparar puesto que el programa Genetic solver no se detiene automáticamente, si no que continúa iterando infinitamente hasta que el usuario detiene la ejecución. Se optó por dar un tiempo de 5 segundos como límite para ambos optimizadores.

En este experimento, no se considera como desperdicio el material sobrante del último corte guillotinado, por lo que un patrón de corte que concentra a los ítems en un lado de la plancha, y permite conservar un trozo grande de material, tendrá menor porcentaje de pérdida que un patrón de corte, que distribuye los ítems por la plancha y entrega muchos trozos pequeños de material sobrante. Esto explica que en la tabla 6.1 existen distintos porcentajes de pérdida, a pesar que, todos los algoritmos cortan la misma cantidad de ítems desde una única plancha.

Respecto al tiempo de convergencia, es claro que los algoritmos del optimizador Packing solver convergen mucho más rápido que los algoritmos de Genetic solver, que necesitan entre 1 a 3 segundos en converger a una solución estable. También se aprecia que la optimización de material que logran los algoritmos de Genetic solver, en los ejemplos de pocos ítems, son superiores a los algoritmos de Packing solver (ver tablas 6.1, 6.2 y 6.3), sin embargo, para ejemplos con mayor cantidad de ítems, el algoritmo Packing solver entrega resultados con porcentajes de pérdida y utilización de planchas mucho menor (ver tablas 6.4 y 6.5).

En base a los resultados de la comparación, se puede afirmar que ambos optimizadores son recursos valiosos y se propone combinarlos en el prototipo para obtener los mejores resultados.

## 6.2. Evaluación de herramientas de desarrollo

Se procede a investigar las herramientas más aptas para el desarrollo del software. Se consideró brevemente Express, un *framework* de desarrollo web para Node.js, sin embargo, se tomó la decisión de usar Django, un *framework* de desarrollo web basado en Python que se especializa en rapidez del desarrollo, seguridad y escalabilidad. Django unifica el *front-end* (vistas y funcionalidades por el lado del cliente) con el *back-end* (servicios y funciones en servidor), permitiendo una comunicación fácil y rápida entre ambos mundos. También, permite gestionar la base de datos a través de una API que emula clases , objetos y métodos típicos de la programación orientada a objetos y entrega módulos pre-hechos para una gran cantidad de necesidades comunes en toda aplicación web como, por ejemplo, gestionar usuarios y, además, entrega un servidor local de desarrollo junto con un panel de administración listo para usar.

## 6.3. Inicialización del proyecto en Django y creación de pantalla principal

Para iniciar el desarrollo, se creó un ambiente virtual en la carpeta base del proyecto, usando el módulo Venv de Python [12]. Luego se instaló la versión más reciente de Django [13] y se inició el proyecto llamado 2DNest. Se descargó una plantilla de un sitio de administración genérico [14] basada en Bootstrap [15], una librería de desarrollo web enfocada en el *front-end*. Se configuró el proyecto para desplegar la plantilla al ejecutar el servidor local y así se obtuvo rápidamente una primera versión de la pantalla de inicio de la aplicación web. La pantalla principal o *dashboard* fue actualizada continuamente a medida que se añadieron funcionalidades y características a la aplicación. A continuación se presenta en la figura 6.1 la versión final de la pantalla principal de la aplicación web 2D Nest.

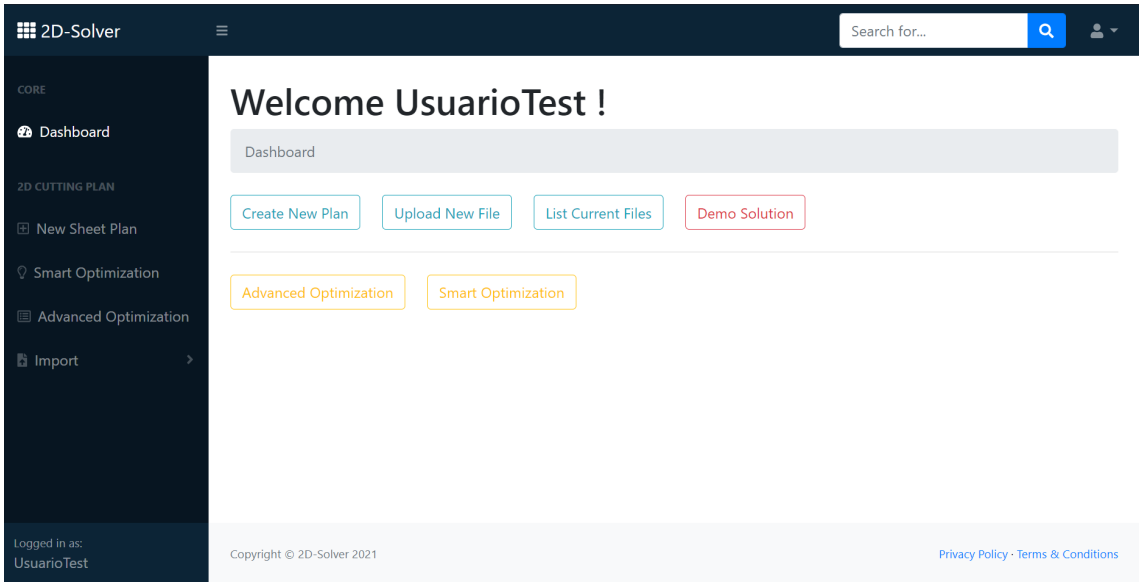



Figura 6.1: Pantalla de bienvenida de la aplicación web 2D Solver

## 6.4. Ejecutar optimizadores desde Django

El primer desafío que se propuso cumplir fue invocar a los programas de optimización desde el proyecto 2D-Nest en Django. Para esto se tuvo 3 entrevistas con desarrolladores web, a quienes se les consultó sobre el proyecto en general y en específico sobre como ejecutar procesos desde Python. La solución fue ocupar el módulo *subprocess* de Python. Este módulo permite ejecutar desde un archivo Python algún programa como si fuese ejecutado directamente desde la pantalla de comandos. Se creó una aplicación dentro del proyecto 2DNest llamada *algorithms*. Este módulo se encarga de manejar todas las acciones relacionadas con los programas Packing solver y Genetic solver.

Se inició con el programa Packing solver, creando una clase en el archivo *models.py* de la aplicación *algorithms*, que recibe como argumentos los parámetros necesarios para ejecutar el programa (función objetivo, nombre de archivo, tipo de corte, etc). Dentro de esta clase se creó un método llamado *execute*, que toma los argumentos y llama al módulo *subprocess* para ejecutar el programa, retornando un objeto que contiene el resultado del proceso. Adicionalmente, se crea un archivo .csv con los resultados de la optimización de corte.

El programa Genetic solver se ejecuta a través de una interfaz de usuario gráfica, con botones y formularios, por lo tanto, fue necesario crear, dentro de los archivos fuente del programa, un nuevo archivo llamado *CmdInterface.java*, el cual recibe parámetros a través de la línea de comando y ejecuta las funciones necesarias para realizar la optimización de corte. Una vez terminada y depurada la nueva interfaz por comandos, se hizo un proceso análogo al anterior, creando una clase en el archivo *models.py* que permite ejecutar el programa usando el módulo *subprocess*.

Siguiendo buenas técnicas de diseño y programación, se creó un archivo de testeo, donde las clases mencionadas anteriormente son ejecutadas y testeadas automáticamente, capturando errores, fallas del diseño y evitando conflicto entre las partes nuevas del código y las antiguas.

Se logró ejecutar exitosamente los programas Packing solver y Genetic solver. Por ahora, los parámetros y la ruta al archivo son configurados directamente en el código fuente de la aplicación web.

## 6.5. Subir archivos CSV

El siguiente paso para desarrollar una aplicación de optimización de corte funcional es implementar las funciones de subida y lectura de archivos CSV, que contienen las piezas (ítems) y planchas a cortar (bins). Para ello se crearon 2 nuevas aplicaciones dentro del proyecto Django, la primera llamada *csvs* y la segunda llamada *panels*. Se creó una clase llamada CSV en el archivo *models.py* de la aplicación csvs , que recibe como argumentos un archivo y un *string* de identificación del tipo (ítem o bin). La fecha de subida y el usuario, son parámetros autocompletados de esta clase y, también, un campo booleano llamado *activate* que indica si el archivo fue procesado o no. En la aplicación *panels* se crearon 2 clases : ítems y bins, que representan a cada pieza y plancha.

Para interactuar con la clase CSV se implementó un formulario, usando la *API Model Form* de Django, que permite crear rápidamente formularios directamente conectados al modelo descrito anteriormente y, en consecuencia, directamente conectados con la base de datos. El formulario implementado permite crear instancias de la clase CSV, que luego es capturada en la función de vista, la cual procesa el archivo CSV creando instancias de los objetos ítem y bins, que son almacenados en la base de datos.

A continuación se muestra en la figura 6.2 la versión final del formulario de subida de archivos CSV:

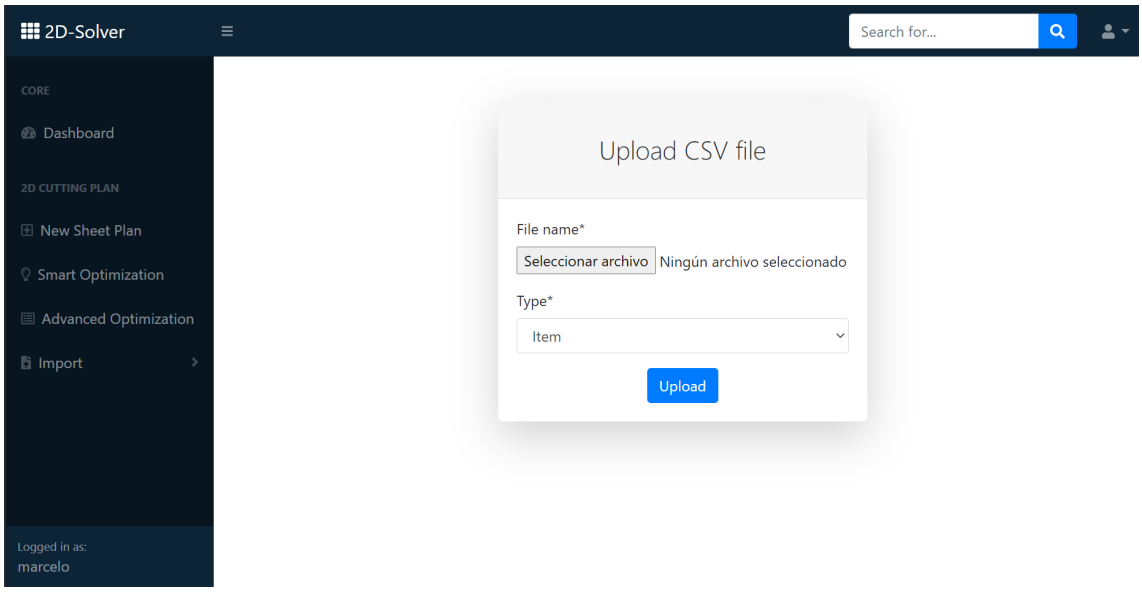



Figura 6.2: Formulario de subida de archivos CSV.

## 6.6. Homologar lectura de CSV

El programa Genetic solver y el programa Packing solver leen archivos CSV de manera distinta y requieren que estos estén configurados de 2 formas distintas, lo que complica la implementación del prototipo. Se decidió modificar la manera en que el programa Genetic solver lee archivos CSV para que fuera homóloga a la lectura de Packing solver. Para esto se tuvo que modificar el código fuente del programa, recompilarlo y volver a implementarlo en el archivo Django, actualizando los test y las llamadas al programa.

## 6.7. Ejecutar optimización desde *frontend*

Se creó un formulario simple para crear una instancia de las clases Packing solver y Genetic solver, y así, poder ejecutar la optimización de un plan de corte, esta vez desde el lado del cliente, usando la interfaz visual del formulario. El formulario es completado por un usuario y envía un *POST request* con las respuestas, que es manejado por la función *execute_algorithm_view()* en *views.py*. Aquí se recibe el *POST request*, se crea una instancia del formulario, se valida y se guarda. Luego se crea en la base de datos un objeto *packing*

*solver* con el campo booleano *executed* marcado como falso, así se identifica para llamar a la función *execute()*, que ejecuta el algoritmo y entrega un archivo CSV de resultados.

Si bien se hizo la implementación tanto para Packing solver como Genetic solver, solo se dejó en línea el formulario Packing solver, ya que se buscaba implementar de forma automática en el método de optimización inteligente al recurso Genetic solver. Se le puso al formulario el nombre de *Advanced Optimization*, ya que desde el formulario se pueden acceder a casi todas las características configurables del algoritmo.

A continuación se muestra en la figura 6.3 la mitad (dado que es muy extenso) de la versión final de este formulario:

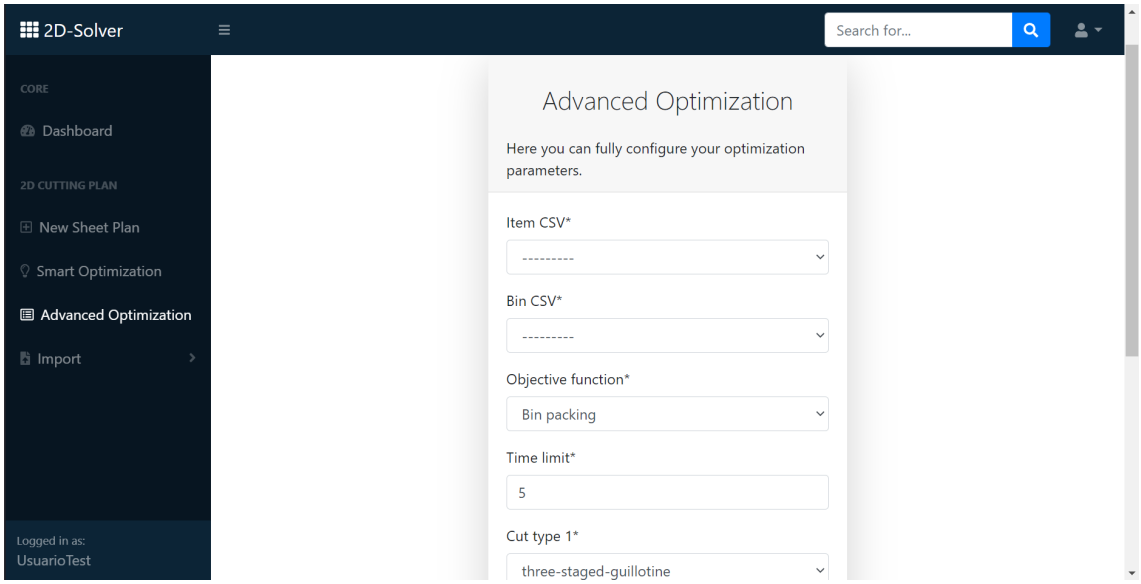



Figura 6.3: Formulario de optimización avanzada para ejecutar desde *front end*

## 6.8. Gestión de usuarios

Se desarrolló un nuevo módulo llamado *users* en el proyecto Django que gestiona los usuarios de la aplicación. Se implementó un formulario de registro para usuarios nuevos, uno de inicio de sesión y una pantalla de término de sesión. Se puso como requisito en las secciones de optimización y de subida de archivos tener la sesión iniciada, en caso contrario, se envía al usuario al formulario de inicio de sesión. En caso de si tener iniciada la sesión, el programa asocia los archivos a cada usuario y filtra en los formularios de optimización las opciones de archivos, para darle acceso a cada usuario sus archivos propios y no tener acceso a toda la base de datos.

A continuación se presentan en las figuras 6.4, 6.5.a y 6.5.b los formularios y pantallas usados para la gestión de usuarios:

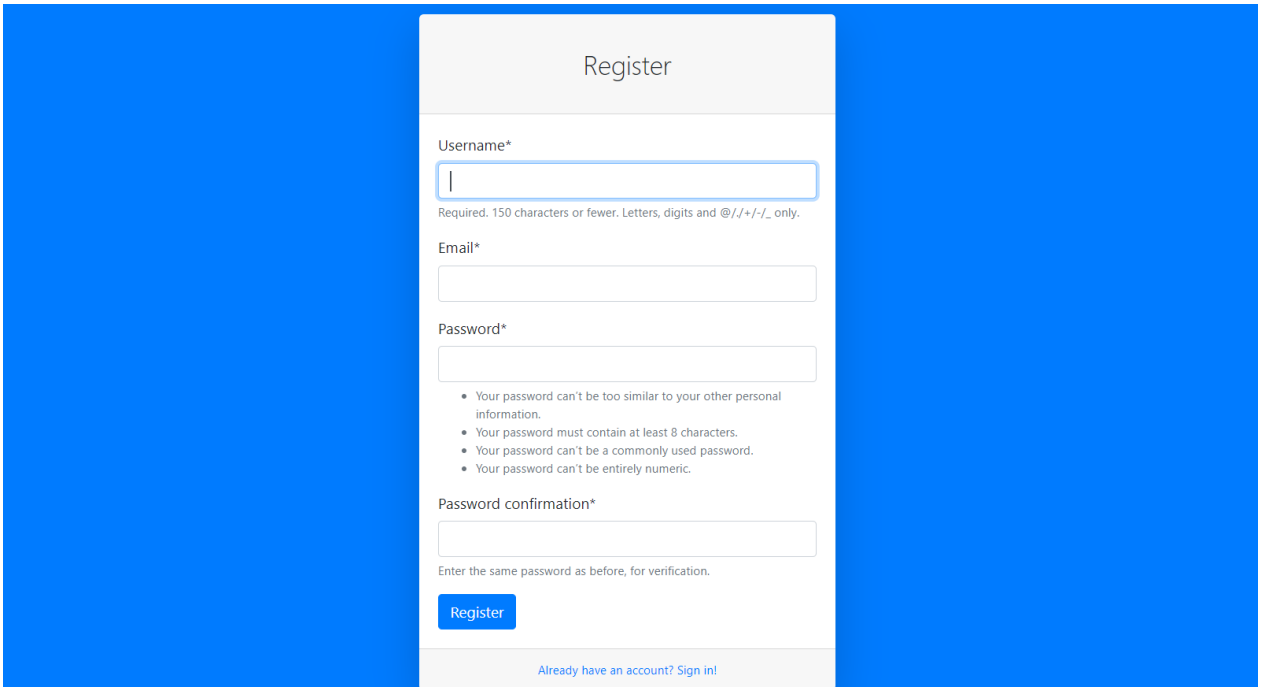
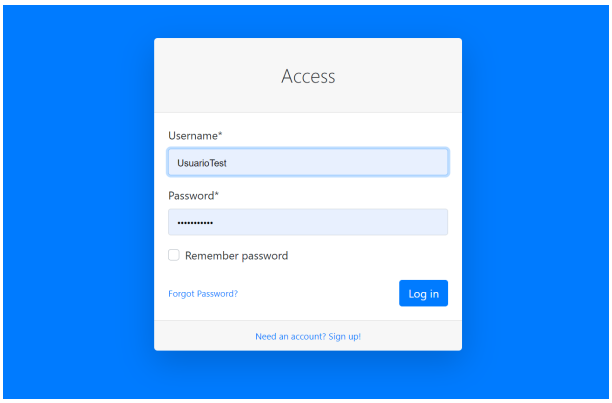
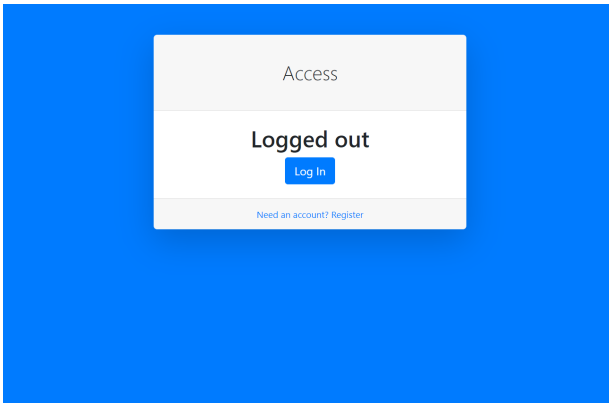



Figura 6.4: Formulario de registro para nuevos usuarios




(a) Formulario de inicio de sesión




(b) Pantalla de sesión cerrada

Figura 6.5: Visualización de resultados actualizada

## 6.9. Ingreso manual de datos

Se implementó una maqueta visual de la subida manual de datos, vista que le permitirá al usuario generar archivos de corte (ítems) y de planchas (bins) sin necesidad de escribirlo directamente en formato CSV y subirlo como archivo al sistema.

Dado que no se considera una función crítica del programa, y que a juicio del desarrollador supone un esfuerzo no menor implementarlo realmente, se decidió dejar la maqueta visual como *mock up*, para ser finalizada más adelante.

A continuación, en la figura 6.6 se presenta el *mock up* del ingreso manual de datos:

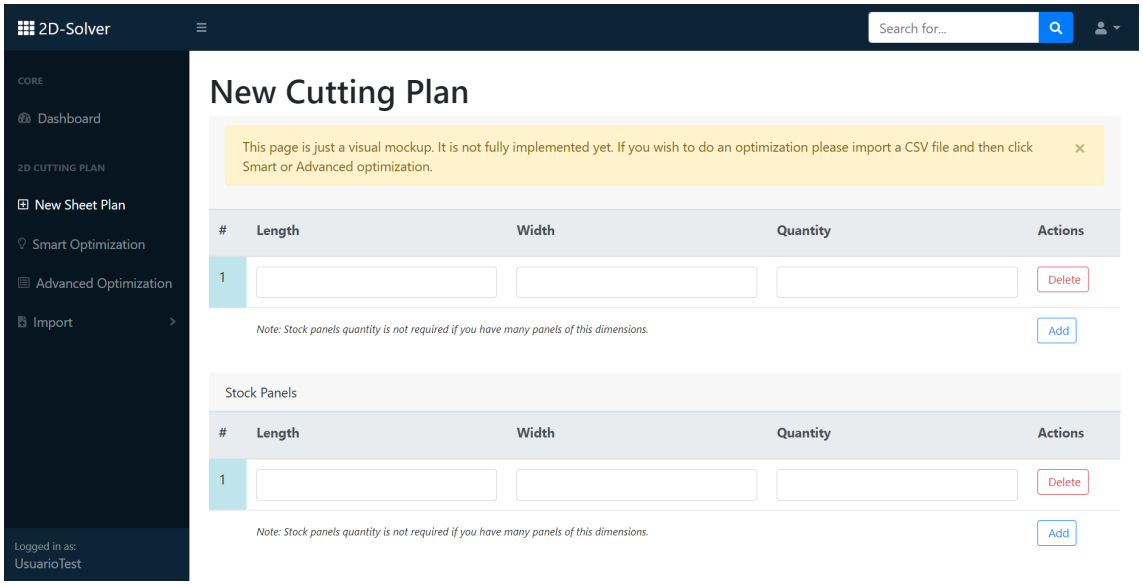



Figura 6.6: Maqueta del ingreso manual de datos

## 6.10. Listar archivos de usuario

Se implementó una funcionalidad que muestra en una tabla los nombres de los archivos subidos por el usuario, junto con el tipo (bin o ítem) y la fecha de subida. A continuación se presenta en la figura 6.7 la funcionalidad de listado de archivos del usuario:

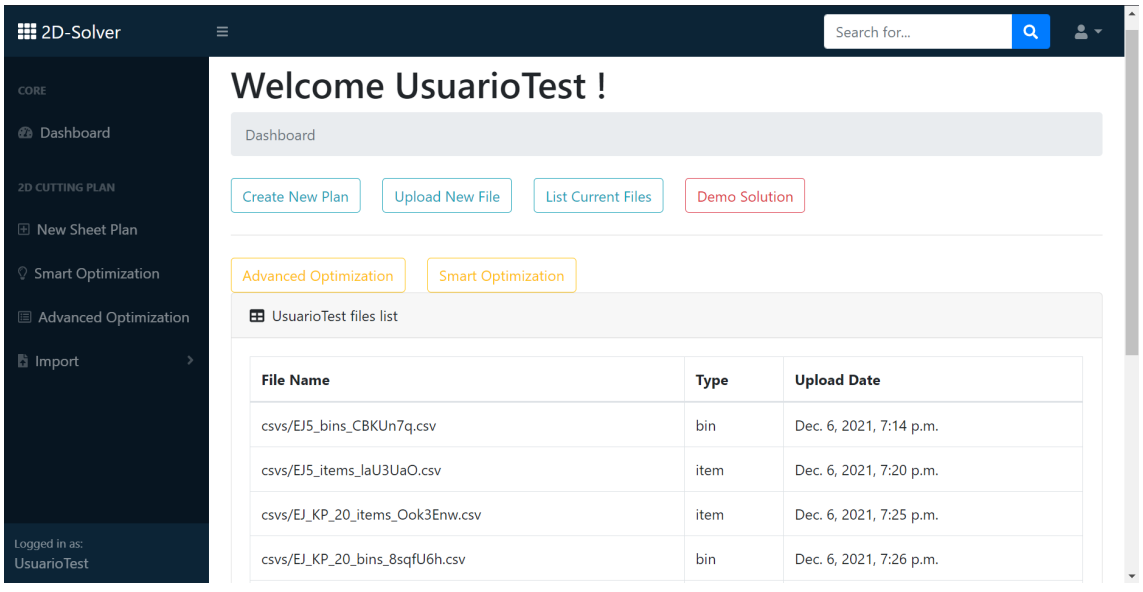



Figura 6.7: Listar archivos de usuario

## 6.11. Visualización de resultados

Para visualizar los resultados de la optimización se usó la librería de JavaScript D3 [16], que permite hacer visualización de datos en usando tecnología web compatible con todos los navegadores. Se evaluó el uso de GoJs [17] , una librería muy poderosa para hacer diagramas interactivos, sin embargo es pagada, por lo que se prefirió ir con la alternativa D3js que es igual de potente y de código abierto.

Los diagramas usados son del tipo *TreeMap*, una representación visual de una estructura de datos jerárquica. Los archivos CSV que entrega el programa Packing solver tienen una estructura de árbol, donde la plancha sin cortes es el nodo raíz y sus subsecuentes divisiones son sus ramas y hojas. A continuación en la figura 6.8 se muestra una primera visualización de un plan de corte real:

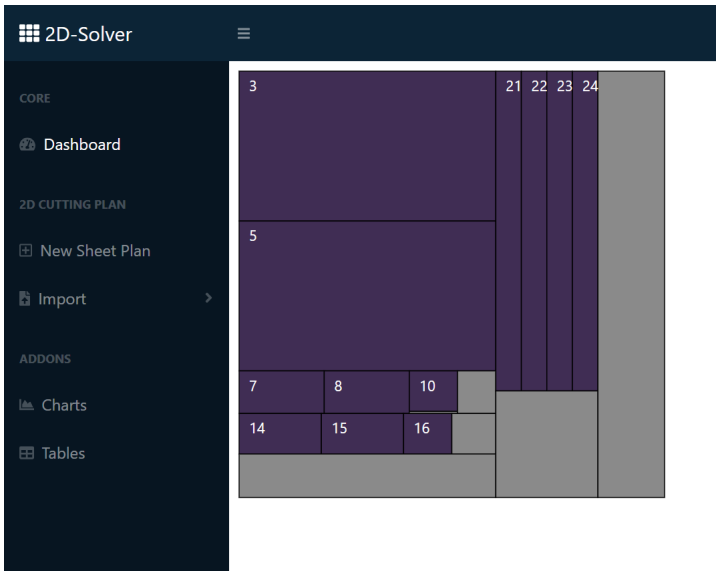
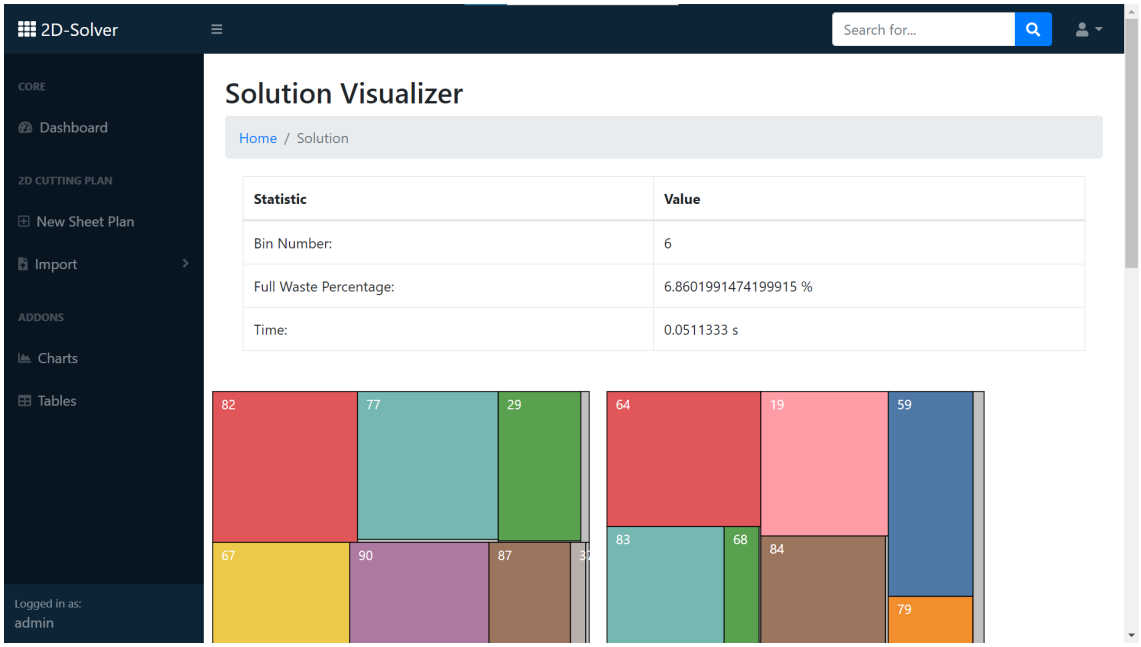



Figura 6.8: Visualización del plan de corte

Para mejorar la experiencia del usuario se añadió una tabla con información relevante respecto a la optimización. Se asignó una paleta de colores a cada corte, se implementó un *tooltip* que aparece cuando el cursor se posa sobre algún rectángulo de corte para mostrar las dimensiones de cada corte y se habilitó el despliegue de varias planchas de corte a la vez.

A continuación en las figuras 6.9.a y 6.9.b se muestra la página de visualización mejorada:




(a) Página de visualización con estadísticas del resultado.


(b) Diagramas de corte con *tooltip* de información.

Figura 6.9: Visualización de resultados final

## 6.12. Solución demo

Se implementó una demostración de la visualización de soluciones en la plataforma, de esta manera un usuario primerizo puede observar los resultados de una optimización sin necesidad de subir archivos ni ejecutar un proceso de optimización. El archivo de solución que se despliega en la solución demo es actualmente insertado manualmente por el desarrollador en la base de datos y se asocia a cada nuevo usuario.

Con esta parte terminada se tienen todos los bloques funcionales para ejecutar y visualizar los resultados de la optimización, por lo que el primer prototipo se encuentra terminado.

## 6.13. Redefinición de alcances del prototipo

El trabajo descrito en las secciones anteriores resume la construcción del prototipo final, sin embargo, la implementación de la subida de archivos, ejecución y especialmente el visualizador se hizo específicamente para el programa Packing solver. Si bien se trabajó bastante en el código fuente del programa Genetic solver para compatibilizar con Packing solver, en particular, se desarrolló una interfaz para ejecutar desde la consola de comandos y se modificó el método de lectura de archivos CSV para aceptar archivos con el formato de Packing solver, se estima que el trabajo necesario para compatibilizar el resto de funciones necesarias con Genetic solver es muy grande para el contexto de este trabajo de memoria, y no aporta suficiente valor al prototipo final. **Dado el poder y versatilidad del recurso Packing solver se decide enfocar el resto del desarrollo únicamente en este programa y dejar la integración del programa Genetic solver a futuro, fuera del marco de trabajo de este trabajo de título.**

# Capítulo 7

# Desarrollo del método de optimización inteligente

El método de optimización inteligente son funciones o clases dentro de la aplicación web que se encargan de hacer una elección y configuración inteligente del los recursos de optimización a usar. Se desea que a partir de la información extraída del archivo de datos que el usuario sube a la plataforma, la aplicación web sea capaz de decidir con qué parámetros configurar la optimización para obtener los mejores resultados.

Para lograr esto se hace un estudio en profundidad de todos los parámetros configurables del recurso de optimización Packing solver. Primero, se explica un **ejemplo de ejecución** del programa, para entender cómo y dónde se insertan los parámetros configurables. En seguida, se estudian las **funciones objetivo** del programa y se decide, en cada una de ellas, bajo que condiciones el método de optimización inteligente debe implementarlas. A continuación se listan las 6 **opciones** que permiten modificar propiedades de los archivos de entrada y se comenta su relación con las distintas funciones objetivos. Luego se describen las configuraciones del **esquema de ramificación** y se deciden los valores por defecto a integrar en el método de optimización inteligente. Finalmente se estudia el funcionamiento de cada **algoritmo** presente en el programa Packing solver y se deciden las mejores opciones para implementar en el método de optimización inteligente.

A continuación se muestra un ejemplo de ejecución del programa Packing solver en la consola de comandos:

```
C:/path/to/main --verbose --problem-type rectangleguillotine  --objective knapsack --items
    ↪ path/file_items.csv  --bins path/file_bins.csv  --certificate file_solution.csv  --output
    ↪ file_output.json  --time-limit 1  -q "RG -p 3NHO -c 4" -a "IBS"  -q "RG -p 3NHO -c
    ↪ 5" -a "IBS"
```

- –**verbose** : este parámetro determina la cantidad de texto explicativo que se imprime con la ejecución del programa.
- –**problem-type** : el tipo de problema define el formato del certificado. Cada tipo de problema tiene funciones objetivos y esquemas de ramificación disponibles. La única opción que tiene el programa actualmente es *rectangle guillotine*, por lo que no es un

parámetro controlable.

- –**objective** : la función objetivo. Fórmula matemática que traduce los planos de corte resultantes a una métrica de desempeño que se intenta maximizar o minimizar. Se estudiarán las 6 opciones en profundidad más adelante.
- –**ítems** : ruta y nombre del archivo csv que contiene las medidas de los ítems a producir.
- –**bins** : ruta y nombre del archivo csv que contiene las medidas de las planchas a cortar.
- –**certificate** : ruta y nombre del archivo que se crea luego de la ejecución. Contiene la información de cada corte a realizar.
- –**output**: ruta y nombre del archivo de salida que se crea luego de la ejecución. Contiene métricas del desempeño de la ejecución y del plan de corte generado.
- –**time-limit** : segundos máximos de ejecución del programa.
- **-q**: esquema de ramificación. En el se describen hasta 9 opciones que configuran la forma en que el algoritmo crea y destruye ramas durante su ejecución.
- **-a** : algoritmo. Cada esquema de ramificación puede ejecutar distintos algoritmos de búsqueda. Se discutirán en profundidad más adelante. Cada esquema de ramificación y algoritmo se ejecutan en diferentes *threads*, por lo que pueden ejecutarse más de uno en paralelo.

## 7.1. Objective function

La función objetivo es la formula matemática que permite evaluar la calidad del resultado de la optimización. Siempre se busca maximizar o minimizar su valor. Está intrínsecamente ligada al tipo de problema de optimización que se desea resolver. El programa Packing solver entrega 6 opciones que se discutirán enseguida.

**Bin packing problem (BPP)** : esta función objetivo se caracteriza por buscar minimizar el área total desperdiciada o no usada. Se puede formular matemáticamente como :

$$min \quad nHW - \sum_{i=I} w_i h_i$$

donde n es el número de bins o planchas usadas, H es la altura y W es el ancho, $I$ es el set de los ítems producidos y $h_i$,$w_i$, son la altura y ancho de cada ítem [8].

Esta función es estándar para problemas de corte y no cuenta con ninguna característica diferenciadora o específica, por lo que para efectos de este estudio, se considera que cumple un rol de función objetivo por defecto frente a diversos tipos de problemas de optimización.

**Knapsack problem (KP)** : esta función objetivo se caracteriza por buscar maximizar la suma de los valores asociados al conjunto de ítems producidos. Esta función proviene, como sugiere el nombre, del problema de la mochila, donde se busca llenar una mochila de volumen

finito, con objetos de distintos tamaños y valor, maximizando el valor total almacenado en la mochila. Se puede expresar matemáticamente de la siguiente forma:

$$max \quad \sum_{i=1}^{n} v_i x_i$$
$$tq \quad \sum_{i=1}^{n} a_i x_i < A$$

donde $v_i$ indica el valor, $x_i$ representa las copias y $a_i$ el área del ítem $i$. $A$ es el área total disponible.

Esta función objetivo debe ejecutarse cuando cada ítem tenga un valor asociado, por lo que el método de optimización inteligente deberá ser capaz de identificar e interpretar una columna correspondiente a este dato en el archivo de entrada al programa.

**Strip packing width (SPPW)**: esta función objetivo es utilizada en problemas donde se tiene un material de alto fijo y de ancho semi-infinito, por ejemplo un rollo industrial de papel de 1 metro de alto y 100 m de ancho. En este tipo de optimización se busca posicionar los ítems a producir de manera que el ancho total usado sea mínimo.[18]

**Strip packing height (SPPH)**: esta función es análoga a la anterior, sólo que la dimensión de ancho es fija y el alto es semi-infinito.

Para configurar correctamente la optimización con estas dos últimas funciones se deberá identificar bins que tengan una razón entre ancho y alto fuera de lo común. Se tomará como parámetro de decisión que si una longitud es 5 veces más que la otra, se aplicará la función objetivo, este valor queda abierto a modificación en el futuro cuando se tenga mayor información sobre este tipo de optimización. También en la configuración manual de los bins en la interfaz de usuario se deberá insertar una opción que permita setear una de las dimensiones del bin como infinita, implicando el uso de una de estas funciones.

**Bin packing with leftovers (BPPL)** : esta función objetivo es muy similar a la función BPP, busca minimizar el área desperdiciada total, sin embargo, no se considera desperdicio el remanente del último corte hecho en el último bin. Se formula matemáticamente como:

$$min \quad nHW - Hw - \sum_{i=I} w_i h_i$$

donde n es el número de bins o planchas usadas, H es la altura y W es el ancho, $w$ es la posición del último corte, $I$ es el set de los ítems producidos y $h_i$,$w_i$, son la altura y ancho de cada ítem [8].

Esta función es específica para el problema de optimización que fue propuesto en la competencia que dio origen al programa Packign solver. Si bien se espera implementar una opción manual para que el usuario tenga acceso a esta función, no se considera que deba ser implementada automáticamente por el método de optimización inteligente, ya que no hay

información en los archivos de entrada que permita asumir alguna preferencia entre BPP y BPPL.

**Variable sized bin packing (VBPP)**: esta función objetivo proviene de una variante del BPP, en donde los bins disponibles no son todos del mismo tamaño y tienen costos distintos. El objetivo es posicionar los ítems requeridos en los bins de manera que la suma de los tamaños de los bins usados sea mínima o en el caso de que tengan costos distintos; se busca minimizar el costo del material. Es valioso notar que si todos los tamaños y costos de los bins disponibles son iguales, volvemos al problema clásico BPP [19].

Para implementar esta función se deberá interpretar los bins disponibles en el archivo de entrada. Si se presentan varias opciones y/o si contienen una columna de costo, se procederá a usar esta función.

## 7.2. Opciones

Existen 6 opciones que permiten modificar directamente ciertas propiedades de los datos de entrada, que permiten ejecutar distintas funciones objetivos sin tener que cambiar manualmente los archivos que contienen la información de bins e ítems.

- –**bin-infinite-copies**: sobrescribe las copias de los bins como infintas. Útil para funciones BPP y VBPP.
- –**item-infinite-copies**: sobrescribe las copias de cada ítem a producir como infinitas. Puede ser útil para la función KP.
- –**bin-infinite-width**: sobrescribe el ancho de cada bin como infinito. Útil para la función SPPW.
- –**bin-infinite-height**:sobrescribe el alto de cada bin como infinito. Útil para la función SPPH.
- –**bin-unweighted**: elimina el costo o valor asociado a cada bin. Útil para optimizar un conjunto de datos destinado a la función KP con otra función objetivo, por ejemplo BPP.
- –**unweighted**: elimina la ganancia o valor asociado a cada ítem, lo hace asignando el área de cada ítem como valor. Útil para optimizar un conjunto de datos destinado a la función KP con otra función objetivo, por ejemplo BPP.

## 7.3. Esquema de ramificación

Entregadas todas las configuraciones descritas anteriormente, se puede configurar varios esquemas de ramificación, que permiten al programa ejecutar, en threads paralelos, distintos algoritmos con distintas configuraciones. A continuación, se presentan las distintas opciones que existen:

- –**cut-type-1**: three-staged-guillotine, two-staged-guillotine
- –**cut-type-2**: roadef2018, non-exact, exact, homogenous
- –**first-stage-orientation**: vertical, horizontal, any
- –**min1cut, –max1cut, –min2cut**: positive integer
- –**min-waste**: positive integer
- –**one2cut**
- –**no-item-rotation**
- –**cut-through-defects**

**Cut-type-1** configura la cantidad de etapas de corte que seguirá el algoritmo. Una etapa de corte corresponde a una serie de cortes guillotinados paralelos en un mismo bin o sub-bin, obtenido de una etapa de corte anterior. Esta es una restricción que se impone en la manufactura real de piezas, ya que hacer demasiadas etapas de corte puede ser ineficiente, a pesar de que una mayor cantidad de etapas de corte le da al optimizador mayor libertad para realizar planos de cortes que aprovechen más material.

En este caso la restricción de etapas de corte a 2 o 3 se fundamenta en aspectos técnicos y materiales de la producción real de piezas, por lo tanto, se considerarán 3 etapas de corte por defecto para el método de optimización inteligente, que es la opción por defecto que consideraron los desarrolladores de Packing solver. Si el usuario desea cambiarlo será de manera manual y no una función automática del método de optimización inteligente.

**Cut-type-2** describen el tipo de corte que seguirá el algoritmo. La configuración **exact** implica que finalizadas las etapas de corte guillotinado, los ítems debes estar perfectamente producidos. Al contrario con las configuración **non-exact**, se permite una etapa de corte extra, pero que sólo puede ser utilizada para quitar material sobrante de un ítem, no para generar ítems nuevos. La opción **roadef2018** es particular para la convocatoria que dio origen al programa, y permite que luego de 3 etapas de corte, se haga una única cuarta etapa de corte guillotinado, pudiendo dividir una sub-pieza y generando 2 ítems nuevos.

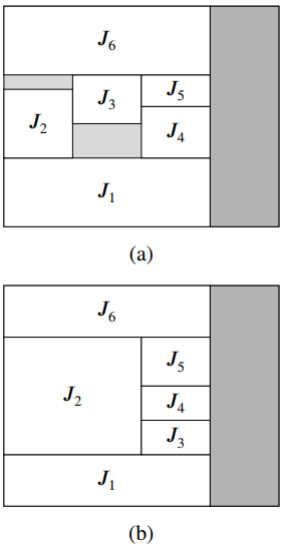



Figura 7.1: Ejemplo tipo de corte Roadef2018 [8]

Como se aprecia en la figura 7.1 , solo un corte de cuarta etapa guillotinada que genere ítems nuevos está permitido. Por lo tanto, el plan de corte (a) está permitido, puesto que hay solo un corte que genera a $J_4$ y a $J_5$, pero el (b) no lo está, ya que se hacen 2 cortes de cuarta etapa que generan a $J_5$, $J_4$ y $J_3$.

La opción que entrega mayor libertad para obtener planos de corte óptimos es **non-exact**, por lo que se considerará como opción por defecto del método de optimización inteligente, asumiendo como hipótesis que para la mayor cantidad de usuarios, la métrica de desempeño más importante es minimizar el desperdicio.

**First-stage-orientation** define la orientación de la primera etapa de corte. La opción **any** le da al optimizador mayor libertad para generar patrones de corte óptimos, por lo que se considerará como opción por defecto. La configuración de esta opción quedará manual, puesto que depende de restricciones técnicas específicas a cada usuario y no de información extraíble del problema de corte ingresado por el usuario.

**Min1cut, max1cut, min2cut, max2cut y one2cut** son restricciones manuales a la cantidad de cortes que se realizan en cada etapa. La opción **max2cut** no aparece en la documentación del programa pero si es una opción válida programada para el esquema de ramificación. Las primeras 4 opciones reciben números enteros y la última es un campo booleano. Nuevamente, se encuentra frente a restricciones que pueden ser útiles para usuarios específicos pero que, en general, ponen mayor dificultad al algoritmo para encontrar patrones de corte óptimo, por lo que configurar manualmente estas variables de optimización entregarán , en la mayoría de los casos, resultados de menor calidad. Se procede en consecuencia a considerar como valores por defecto, para el método de optimización inteligente, estas opciones: **0, -1, 0 , -1 y false**, considerando el valor -1 equivalente a infinito.

**Min-waste** es una opción que permite definir una cantidad mínima de desperdicio que debe tener la solución. Dado que se busca, justamente, minimizar este aspecto de la solución,

se considerará el valor por defecto **1** y configuración avanzada manual.

**No-item-rotation** es una opción que permite impedir que el optimizador rote los ítems a producir durante el proceso de ramificación de soluciones del algoritmo. Se considera el valor por defecto **false** y configuración avanzada manual.

**Cut-through-defects** es una opción del esquema de ramificación que permite cortar a través de regiones defectuosas del bin. El optimizador por defecto evita estas zonas defectuosas y las incluye en el área de desecho, esta opción le indica hacer lo contrario. Se considera esta dejar esta opción por defecto como **false**.

## 7.4. Algoritmos

El programa Packing solver incluye 5 algoritmos distintos con los que se puede recorrer el esquema de ramificación. Entre los 5 destaca IMBA*, algoritmo creado por los desarrolladores del Packing solver para el concurso ROADEF EURO 2018, con el cual se ganó el primer lugar de la convocatoria.

A continuación se procederá a estudiar cada algoritmo con el objetivo de entender cuándo y cómo conviene ser usado.

### 7.4.1. A*

A* (pronunciado « A estrella » ) es un algoritmo computacional muy usado en problemas de búsqueda de ruta y recorrer grafos. El algoritmo entrega un camino factible entre múltiples nodos del grafo.

A* es una extensión del algoritmo Dijkstra, ambos funcionan haciendo el camino de menor costo desde el nodo de inicio hasta el nodo de meta. Mientras que el algoritmo Dijkstra evalúa nodo a nodo los costos asociados a todos los caminos posibles desde el nodo actual, el algoritmo A* usa una función heurística que le permite estimar el costo total de usar uno u otro camino. De esta manera el algoritmo A* es capaz de discriminar con anticipación cual de los posibles caminos a tomar tiene más sentido, ya que la función heurística le da un valor respecto a la calidad global de ese camino.

Como ejemplo, se puede considerar encontrar la ruta óptima desde un punto a otro en el congestionado centro de Santiago. En este ejemplo cada intersección de calles es un nodo del grafo, cada calle es un camino y la congestión vehicular de cada calle es su costo. El algoritmo Dijkstra parte en la intersección de inicio, mide la congestión en las 3 calles a las que le es posible avanzar y las ordena desde la más despejada a la más congestionada, a continuación avanza por la calle más despejada y repite el proceso, sumando los costos de cada cuadra para llevar una lista ordenada de rutas posibles, siempre se explora primero el camino con menor costo, y en cada nodo se actualizan los valores, por lo que el algoritmo también retrocede para explorar otras rutas y así asegurar un camino óptimo.

El algoritmo A* incluye en su función de costo $f(n)$ una función heurística $h(n)$, que en este caso podría ser la distancia euclidiana entre el nodo siguiente y el nodo de destino.

$$f(n) = g(n) + h(n)$$

donde

$f(n)$ = costo total estimado del camino a través del nodo $n$
$g(n)$ = costo acumulado para llegar al nodo $n$
$h(n)$ = costo estimado desde $n$ hasta la meta.

Por lo tanto, en el ejemplo el algoritmo A* suma el valor de la congestión del camino siguiente con la distancia entre el nodo siguiente y el nodo de destino. Así frente a 2 caminos con la misma congestión, el algoritmo A* prefiere el camino que se dirija hacia el nodo objetivo. De esta manera converge mucho más rápido a la ruta óptima, puesto que tiene cierta perspectiva global de la dirección que lleva en cada iteración, con la desventaja que al incorporar una función heurística, el algoritmo ya no asegura resultados óptimos globales [21].

En la figura 7.2, encontramos una representación simple del ejemplo. En amarillo y rojo rutas posibles del algoritmo Dijkstra. En azul la ruta de A estrella guiada por la distancia euclidiana de sus nodos a la meta representada por la línea verde.

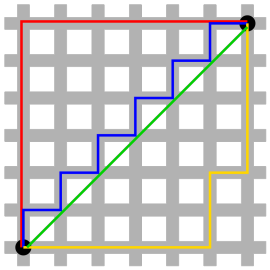


Figura 7.2: Ejemplos de rutas de A*(azul), Dijkstra (rojo y amarillo) y la distancia euclidiana del origen a la meta (verde).

La implementación de este algoritmo en el programa Packing solver es un poco distinta al ejemplo mostrado anteriormente. El grafo que recorre el algoritmo es, en este caso, el esquema de ramificación; un árbol de planos de corte que se ramifica con cada corte guillotinado. Este árbol presenta un costo nulo por explorar cada camino , es decir , $g(n) = 0$ . Esto tiene sentido, ya que el camino para llegar a un nodo u otro en este grafo son equivalentes, no existe un peso o costo intrínseco al camino que une un nodo con otro. La diferencia que sí existe, es el valor de la función heurística $h(n)$ asociada a cada nodo, que en este caso es el porcentaje de desperdicio, o el valor total de la solución, dependiendo si nos encontramos frente a una función objetivo BPP o KP respectivamente.

### 7.4.2. DFS

*Depth first search*, en español, búsqueda de profundidad primero, es un algoritmo de búsqueda muy simple, que recorre nodo a nodo el grafo, evitando visitar 2 veces el mismo nodo. Tiene un funcionamiento muy básico. En el nodo actual, visita al azar uno de los nodos conectados al nodo actual y repite este proceso hasta que se encuentra con un nodo ya visitado; en ese momento retrocede en el grafo hasta encontrar el primer nodo con caminos no visitados, elige al azar uno y continúa el proceso hasta cumplir con su condición de detención, que puede ser, llegar a un nodo objetivo, visitar todos los nodos, etc.

La implementación en el programa Packing solver es recursiva, se llama a si misma al final del código para continuar con la expansión del árbol de soluciones. Al parecer no hace un recorrido ciego, si no que usa una heurística para decidir que nodo visitar y expandir a continuación. Sin embargo, al ser una estructura de árbol, donde no existen loops, no implementa una lista de nodos visitados o una marca booleana, por lo que sólo recorre en una única dirección el árbol de soluciones hasta el fondo. A diferencia del algoritmo A*, que implementa una lista ordenada de nodos visitados que le permite ir actualizando en cada iteración el mejor candidato a expandir. La condición de detención es encontrar una solución factible, que difícilmente es óptima dada la forma en que recorre el árbol.

### 7.4.3. DPA*

*Dynamic Programing A Star*, en español, programación dinámica A estrella, es un algoritmo de búsqueda más complejo que los últimos 2 algoritmos descritos. Al igual que los casos anteriores, recorre un grafo en búsqueda de un nodo objetivo, y como resultado entrega un camino óptimo. A diferencia de los algoritmos anteriores, DPA* no solo entrega un plan óptimo desde el punto de partida elegido, sino que computa rutas óptimas desde cada nodo del grafo. Esto, en aplicaciones reales para encontrar rutas óptimas en ciudades por ejemplo, es muy útil, dado que la posición actual puede variar de manera estocástica, y en ese caso es ventajoso tener un algoritmo resistente a esa variable aleatoria.

### 7.4.4. IBS

*Iterative beam search*, en español, búsqueda iterativa de haz, es un poderoso algoritmo de búsqueda, que mezcla la noción de los algoritmos *Breadth first search* y *A star*.

*Breadth first search(BFS)*, en español búsqueda de amplitud primero, es un algoritmo muy similar a *Depth first search*, con la importante diferencia que BFS no elige sólo un nodo para continuar su exploración, si no que explora cada nivel del grafo, es decir, desde cierto nodo, BFS explora todos los posibles nodos a los que es posible acceder desde su posición actual, y luego hace lo mismo con el siguiente nivel de nodos, mapeando todas las rutas posibles, hasta encontrar el nodo objetivo, con un tiempo de convergencia muy alto.

IBS toma este concepto pero limita la cantidad de nodos a explorar a cierta amplitud de haz (*beam width*), por lo que en cada nivel, sólo explora un número limitado de nodos y elige estos nodos usando una función heurística, estrategia prestada del algoritmo A*.

El algoritmo IBS mezcla las mejores características de los algoritmos simples de búsqueda, lo que le entrega una gran robustez y permite converger a resultados de buena calidad en un tiempo óptimo, de ahí su gran desempeño mostrado en la comparación de algoritmos de la sección 6.4.

### 7.4.5. IMBA*

*Iterative memory bounded A star* es un algoritmo desarrollado por los creadores de Packing solver. Su descripción se encuentra en la sección 2.4.7

### 7.4.6. Conclusión

Estudiados los algoritmos y considerando su desempeño en la comparativa realizada en capítulo 6, se concluye que los 2 algoritmos que entregan resultados valiosos son IMBA* y IBS, ambos algoritmos pertenecientes al recurso de optimización Packing solver, por lo que serán configurados para correr en threads paralelos y así obtener el mejor resultado entre ambos cada vez que se ejecute la optimización.

## 7.5. Compilación para VBPP

Para habilitar la generación de heurística de columnas necesarias para usar la función objetivo VBPP, se tuvo que recompilar el programa Packing solver usando Bazel, sin embargo, se encontró un error en la compilación puesto que se hace uso de un componente llamado *IBM ILOG CPLEX Optimization Studio*, el cual no se encontraba instalado. Se descargó desde la página oficial de IBM con la licencia gratis para estudiante, que pone un riesgo legislativo para el programa puesto que al ser gratis no debería permitir su utilización con fines comerciales. Se leyeron los términos y condiciones del programa al momento de instalación pero no se pudo determinar con total certeza esta restricción, se deberá pedir ayuda legal al respecto antes de lanzar el producto terminado a producción.

Instalado el componente se vuelve a recibir los mismos errores de compilación. El problema es que no existen los componentes de CPLEX en el repositorio del autor de Packing solver.

Se hizo contacto con Florian Fontan, autor del programa Packing solver y se siguieron sus instrucciones para usar el componente *CPLEX Optimization Studio* localmente. Se pudo avanzar en el proceso de compilación, sin embargo se tuvo que enfrentar a nuevos problemas, dado que el software Packing solver y CPLEX estaban configurados para ser compilado juntos en Linux. Se volvió a contactar a Florian quien amablemente entrego ayuda y documentación para sortear el problema. Luego de un trabajo de investigación no menor y de varios intentos se logró compilar nuevamente el programa Packing solver y habilitar la función objetivo VBPP.

En las conversaciones con Florian, se comentó la opción de usar *COIN-OR Linear Programming*, un software análogo a *CPLEX Optimization Studio*, pero de código abierto. Esta opción resuelve el incipiente problema legislativo que se podría tener con el uso de la licencia de estudiante de CPLEX. Como trabajo futuro queda recompilar el programa usando *COIN-OR Linear Programming*, ya sea por parte del autor de este informe o directamente contratando a Florian Fontan para que realice esta tarea específica.

## 7.6. Identificación de características en archivos CSV

En el modelo de archivos CSV , se crearon 5 métodos que sirven para evalúar características de los archivos csv que contienen la información sobre ítems y bins:

- **read_csv()** : abre el archivo csv y retorna un diccionario con la información de cada columna.
- **get_profit()** : usa el método anterior para abrir el archivo y buscar en las cabeceras de las columnas la palabra clave *PROFIT* , retorna un booleano.
- **get_cost()** : lee el archivo csv y buscar en las cabeceras de las columnas la palabra clave *COST* , retorna un booleano.
- **width_height_ratio()**: lee el archivo csv y retorna una lista con los cocientes entre el ancho y el alto.

- **multiple_bins()**: lee el archivo csv y retorna un valor booleano si el archivo csv contiene más de un bin (una línea de información).

Con estos métodos, en la función *smart_execute_view()* del script *view.py* del módulo *algorithms*, se evalúan los archivos csv seleccionados para la optimización.

## 7.7. Ejecución del método de optimización inteligente

En la figura 7.4, se exhibe un diagrama de flujo de la ejecución del recurso Packing solver mediante el método de optimización inteligente. A continuación se explicará paso a paso el flujo del programa.

Se parte con la recepción del formulario de optimización inteligente (ver figura 7.3) que contiene los id de los archivos de ítems y bins. Se recuperan ambos objetos de la base de datos y se almacenan en las variables *item_csv* y *bin_csv*. Luego se crea una lista vacía llamada *objective_function*, que almacenará el nombre de las funciones objetivos seleccionadas. Se crean, además, 2 variables booleanas llamadas ps1 y ps2, usadas para decidir, más adelante, si se ejecutaran y visualizarán dos instancias del optimizador Packing solver o solo una.

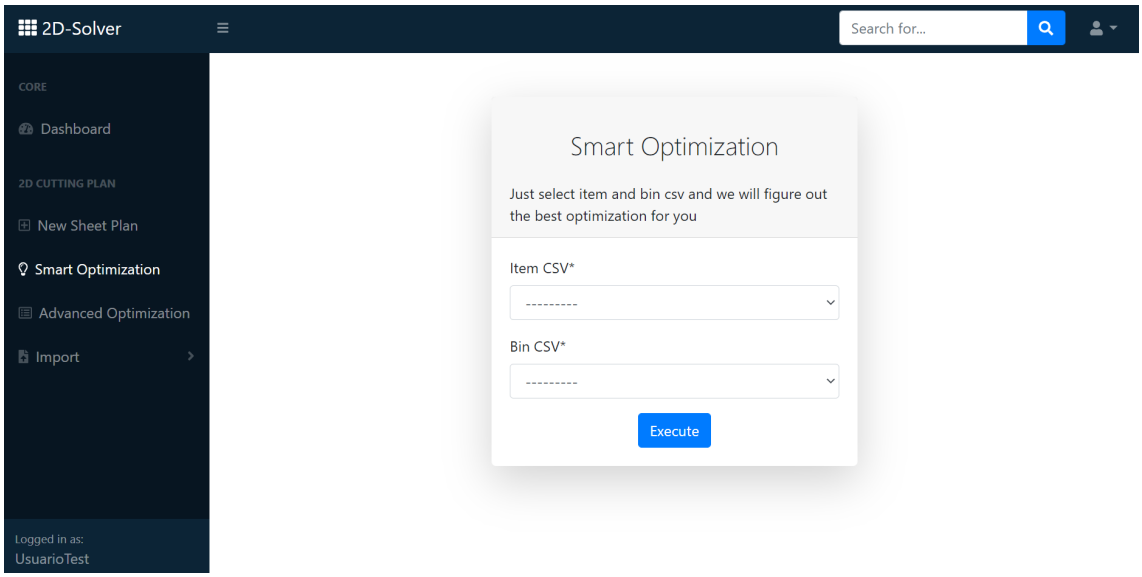



Figura 7.3: Formulario de optimización inteligente

Se procede a evaluar las variables *item_csv* y *bin_csv* usando las funciones descritas en la sección anterior.

Primero, se usan las funciones *multiple_bins()* y *get_cost()*, que se aplican sobre el archivo csv de bins, para evaluar si el usuario entrega bins de distintas dimensiones y con costos distintos. Si una de las funciones retorna un valor booleano positivo, se añade a la lista *objective_function* el *string* "VBPP", que indica que la función objetivo *Variable sized bin packing* fue seleccionada.

A continuación, se usa la función *get_profit()*, que se aplica sobre el archivo de ítems para evaluar la selección de la función objetivo *knapsack*. Al igual que en el caso anterior, si la función retorna verdadero, se añade el *string* "KP" a la lista *objective_function*, indicando que la función *knapsack* fue seleccionada.

Luego, la función *width_height_ratio()* se evalúa sobre el archivo csv de bins para encontrar valores inusuales de largo y ancho en los bins, Si el cociente es mayor a 5, se selecciona la función objetivo *Strip packing width*. Si el cociente es menor a 0.2, se selecciona la función objetivo *Strip packing height*

Finalizada la detección de las características clave en los archivos de entrada, se procede a crear 2 instancias del objeto packing solver y se evalúa el largo de la lista *objective_function*.

Si el largo de la lista es 0, osea se seleccionaron 0 funciones objetivos, se setea la variable ps2 como falsa y se ejecuta solo una instancia de packing solver con sus configuraciones predeterminadas (función objetivo *bin packing problem*) y se visualiza solo un resultado.

Si el largo de la lista es 1, o sea se seleccionó una sola función objetivo, se ejecutarán 2 instancias de packing solver, una con la función objetivo seleccionada y otra con la configuración predeterminada. Para cambiar la función objetivo se usa el método *packingSolver.set(objective_function)*. En este caso se visualizarán 2 resultados.

Si el largo de la lista es de 2 o más, se ejecutarán 2 instancias del optimizador, con las 2 primeras funciones objetivos seleccionadas.

A continuación, se procede a ejecutar los algoritmos de optimización, con el método *execute()*, que configura un tiempo de ejecución máximo de 5 segundos y ejecuta en 2 *threads* paralelos los algoritmos IBS y MBA*. En cualquiera de los casos anteriores, si una función objetivo es VBPP, la instancia del optimizador Packing solver se ejecutará con otro método específico llamado *execute_vbpp()*, que integra ciertas configuraciones específicas como, por ejemplo, un tiempo de ejecución máximo más alto (15 segundos) y la ejecución en 2 *threads* paralelos del algoritmo IBS, que en este caso tiene una mejor respuesta que el algoritmo MBA*.

Si la ejecución de alguna de las instancias de los objetos packing solver resulta en error, se setea su variable ps1 o ps2 como falsa. De esta manera se puede decidir, en última instancia, cuántos resultados mostrar, y así, no confundir al usuario con resultados vacíos o erróneos. Si ambas variables son seteadas como falsas, se envía al usuario de vuelta al formulario de optimización inteligente, con un mensaje de error explicativo.

El orden en que las características claves son identificadas y el flujo en general del método de optimización inteligente se muestra a continuación en la figura 7.4

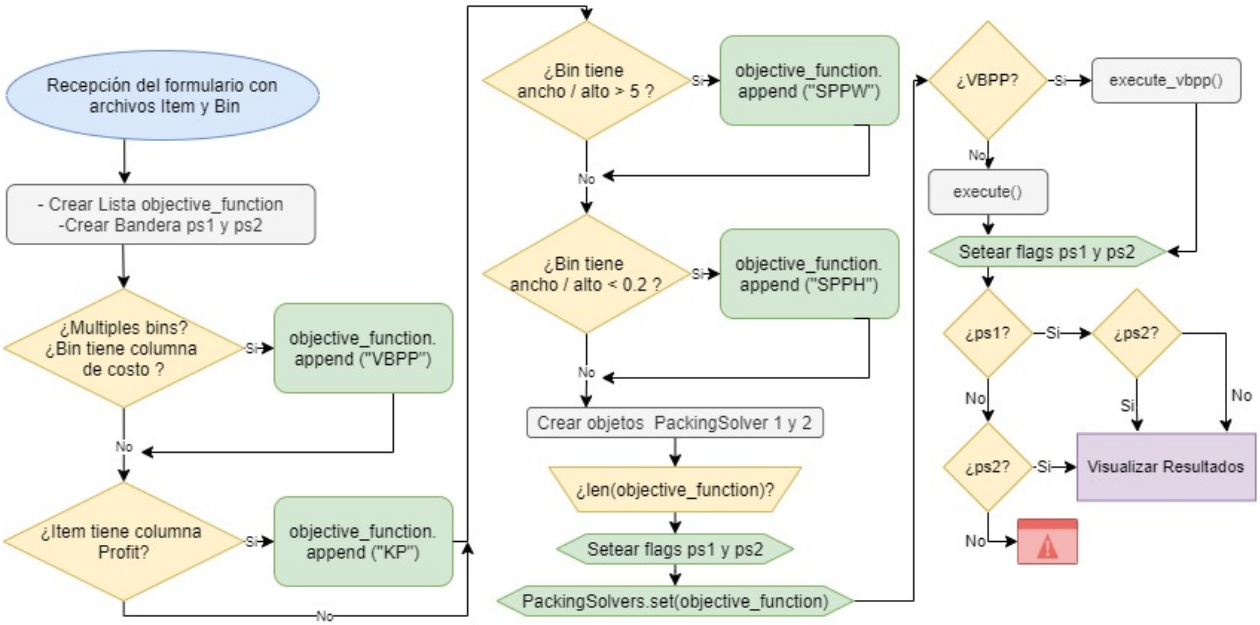



Figura 7.4: Diagrama del método de optimización inteligente

# Capítulo 8

# Evaluación del prototipo

## 8.1. Cuadros comparativos de resultados de optimización

### 8.1.1. Comparación *bin packing problem*

Tabla 8.1: Comparación de la variante BPP con 12 ítems.

| Ej 1 12 Ítems | Pérdida | Planchas | Cantidad de cortes | Tiempo [s] |
|---|---|---|---|---|
| Cut Logic 2D | 29.76 % | 1 | 22 | 21 |
| Cutting Optimization Pro | 29.76 % | 1 | 16 | 0.36 |
| CutList Optimizer | 29.76 % | 1 | 15 | 30 |
| 2D Solver | 29.76 % | 1 | 17 | 0.008 |

Tabla 8.2: Comparación de la variante BPP con 25 ítems.

| Ej 2 25 Ítems | Pérdida | Planchas | Cantidad de cortes | Tiempo [s] |
|---|---|---|---|---|
| Cut Logic 2D | 21.57 % | 3 | 41 | 52 |
| Cutting Optimization Pro | 13.04 % | 2 | 47 | 2.328 |
| CutList Optimizer | 13 % | 2 | 45 | 65 |
| 2D Solver | 13.04 % | 2 | 48 | 0.017 |

Tabla 8.3: Comparación de la variante BPP con 51 ítems3

| Ej 3 51 Ítems | Pérdida | Planchas | Cantidad de cortes | Tiempo [s] |
|---|---|---|---|---|
| Cut Logic 2D | 22.76 % | 3 | 99 | 163 |
| Cutting Optimization Pro | 22.76 % | 3 | 104 | 2.328 |
| CutList Optimizer | 23 % | 3 | 96 | 64 |
| 2D Solver | 22.75 % | 3 | 102 | 0.018 |

Tabla 8.4: Comparación de la variante BPP con 101 ítems.

| Ej 4 101 Ítems | Pérdida | Planchas | Cantidad de cortes | Tiempo [s] |
|---|---|---|---|---|
| Cut Logic 2D | 6.86 % | 6 | 190 | 240 |
| Cutting Optimization Pro | 6.86 % | 6 | 198 | 5.235 |
| CutList Optimizer | 7 % | 6 | 196 | 64 |
| 2D Solver | 6.86 % | 6 | 199 | 0.082 |

Tabla 8.5: Comparación de la variante BPP con 500 ítems.

| Ej 5 500 Ítems | Pérdida | Planchas | Cantidad de cortes | Tiempo [s] |
|---|---|---|---|---|
| Cut Logic 2D | 7.14 % | 30 | 946 | +1800 |
| Cutting Optimization Pro | 3.94 % | 29 | 796 | 28.39 |
| CutList Optimizer | 7 % | 30 | 748 | 67 |
| 2D Solver | 3.94 % | 29 | 833 | 5.004 |

Tabla 8.6: Comparación de la variante BPP con 1000 ítems.

| Ej 6 1000 ítems | Pérdida | Planchas | Cantidad de cortes | Tiempo [s] |
|---|---|---|---|---|
| Cut Logic 2D | 5.57 % | 59 | 1883 | +1800 |
| Cutting Optimization Pro | 2.25 % | 57 | 1444 | 79.625 |
| CutList Optimizer | 6 % | 59 | 1424 | 122 |
| 2D Solver | 3.94 % | 58 | 1627 | 5 |

#### 8.1.1.1. Discusión de resultados

*Bin packing problem* es la variante por defecto del problema de corte y empaquetado. En esta variante se optimiza el corte de la plancha buscando reducir el porcentaje de pérdida y la cantidad de planchas usadas. Una métrica interesante que se incluyó en la comparativa es la cantidad de cortes necesarios, que sirve para discriminar entre soluciones que posean la misma cantidad de pérdida y planchas utilizadas.

Se puede observar que 2D Solver entrega, en todos los cuadros, resultados muy competitivos y a la par con sus contentores comerciales. En especial se destacan las 2 últimas tablas 8.5 y 8.6 correspondientes a los ejemplos 5 y 6, donde 2D Solver obtiene mejores resultados que Cut Logic 2D y Cut List Optimizer a pesar de que a ambos se les entregó un tiempo de ejecución muchísimo mayor. Si bien 2D Solver llega a su límite de tiempo de ejecución (5 segundos) y entrega un resultado parcial de optimización, los otros 2 optimizadores ya señalados no logran entregar un resultado competitivo en sus largos tiempos de convergencia (más de 30 minutos para Cut Logic 2D).

Se puede inferir que con un límite de tiempo mayor, 2D Solver hubiera podido llegar a un porcentaje de optimización aún menor, igualando o superando al obtenido por Cutting Optimization Pro en la tabla 8.6. Queda como trabajo futuro evaluar el impacto del límite de tiempo en el rendimiento del algoritmo y evantualmente implementar una modificación dinámica de este límite.

Los resultados anteriores demuestra una característica muy valiosa y diferenciadora de 2D Solver, su rápido tiempo de convergencia. Este descubrimiento será explotado en fases ulteriores de desarrollo para apuntar hacia segmentos de clientes con necesidades de optimización de cantidades industriales de material.

### 8.1.2. Comparación *variable sized bin packing problem*

Tabla 8.7: Comparación de la variante VBPP con 12 ítems.

| Ej 1 10 Ítems | Pérdida | Planchas | Costo | Tiempo [s] |
|---|---|---|---|---|
| Cut Logic 2D | 19.90 % | 5 | 376 | 56 |
| Cutting Optimization Pro | 19.89 % | 5 | 297 | 3.438 |
| 2D Solver | 19.89 % | 5 | 376 | 5.003 |

Tabla 8.8: Comparación de la variante VBPP con 20 ítems.

| Ej 2 20 Ítems | Pérdida | Planchas | Costo | Tiempo [s] |
|---|---|---|---|---|
| Cut Logic 2D | 8.99 % | 11 | 633 | 13 |
| Cutting Optimization Pro | 6.08 % | 9 | 606 | 7.109 |
| 2D Solver | 8.98 % | 11 | 633 | 5.003 |

Tabla 8.9: Comparación de la variante VBPP con 40 ítems.

| Ej 3 40 Ítems | Pérdida | Planchas | Costo | Tiempo [s] |
|---|---|---|---|---|
| Cut Logic 2D | 2.93 % | 15 | 800 | 213 |
| Cutting Optimization Pro | 3.55 % | 15 | 807 | 7.781 |
| 2D Solver | 3.76 % | 16 | 788 | 5.013 |

Tabla 8.10: Comparación de la variante VBPP con 100 ítems.

| Ej 4 100 Ítems | Pérdida | Planchas | Costo | Tiempo [s] |
|---|---|---|---|---|
| Cut Logic 2D | 4.21 % | 43 | 3416 | 229.000 |
| Cutting Optimization Pro | 12.87 % | 48 | 3739 | 18.953 |
| 2D Solver | 6.89 % | 47 | 3460 | 5.004 |

Tabla 8.11: Comparación de la variante VBPP con 500 ítems.

| Ej 5 500 Ítems | Pérdida | Planchas | Costo | Tiempo [s] |
|---|---|---|---|---|
| Cut Logic 2D | 5.08 % | 225 | 17080 | 426.000 |
| Cutting Optimization Pro | 12.01 % | 244 | 18392 | 15.063 |
| 2D Solver | 4.80 % | 226 | 16994 | 15.013 |

#### 8.1.2.1. Discusión de resultados

*Variable sized bin packing problem*, o en español, problema de planchas de dimensiones variadas, es una variante del problema de corte y empaquetado donde se poseen varios bins, o planchas de materia prima de distintas dimensiones y costos, y se quiere optimizar los cortes en el material para reducir el costo total del plan de corte. También, en segundo plano, se busca reducir el porcentaje de pérdida y la cantidad de planchas usadas. En esta comparativa no se pudo incluir al software CutList Optimizer, dado que no entrega esta funcionalidad.

En esta comparativa, es necesario enfocarse en la columna de costo, ya que es la principal variable a optimizar. Nuevamente vemos un desempeño de 2D Solver sólido en los resultados de la comparativa, a la altura de los otros software comerciales.

Cut Logic 2D muestra un buen rendimiento en esta variante de optimización, sin embargo, 2D Solver ratifica su dominio en los ejemplos de mayor cantidad de ítems, como se evidencia en la tabla 8.11, donde 2D Solver supera ampliamente a sus contrincantes.

### 8.1.3. Comparación *strip packing problem*

Tabla 8.12: Comparación de la variante SPP con 12 ítems.

| Ej 1 12 Ítems | Pérdida | Largo (max 10000) | Tiempo [s] |
|---|---|---|---|
| Cut Logic 2D | 85.9 % | 1520 | 14 |
| Cutting Optimization Pro | 85.9 % | 1635 | 0.313 |
| CutList Optimizer | 86 % | 1565 | 62 |
| 2D Solver | 85.9 % | 1520 | 5 |

Tabla 8.13: Comparación de la variante SPP con 25 ítems.

| Ej 2 25 Ítems | Pérdida | Largo (max 10000) | Tiempo [s] |
|---|---|---|---|
| Cut Logic 2D | 76.47 % | 3643 | 64 |
| Cutting Optimization Pro | 76.47 % | 2632 | 0.891 |
| CutList Optimizer | 76 % | 2770 | 62 |
| 2D Solver | 76.47 % | 1784 | 5 |

Tabla 8.14: Comparación de la variante SPP con 50 ítems.

| Ej 3 50 Ítems | Pérdida | Largo (max 10000) | Tiempo [s] |
|---|---|---|---|
| Cut Logic 2D | 51.41 % | 10000 | 76 |
| Cutting Optimization Pro | 51.41 % | 5346 | 0.922 |
| CutList Optimizer | 51 % | 9511 | 9 |
| 2D Solver | 51.41 % | 5062 | 5 |

Tabla 8.15: Comparación de la variante SPP con 100 ítems.

| Ej 4 100 Ítems | Pérdida | Largo (max 20000) | Tiempo [s] |
|---|---|---|---|
| Cut Logic 2D | 27.35 % | 20000 | 136 |
| Cutting Optimization Pro | 27.35 % | 15377 | 2.89 |
| CutList Optimizer | 27 % | 19362 | 63 |
| 2D Solver | 27.35 % | 15997 | 0.015 |

#### 8.1.3.1. Discusión de resultados

*Strip packing problem*, o en español, el problema de empaquetado de tiras, es una variable del problema de corte y empaquetado donde tenemos una plancha o bin que tiene una dimensión semi infinita, es decir, se cuenta con un rollo de material que es muchísimo más largo que ancho, o más ancho que largo. En esta variable se busca optimizar el largo total del plan de corte, o en otras palabras, se buscan planes de corte que agrupen todas las piezas en la parte más próxima del rollo de material.

En esta comparativa, hay que centrarse en la columna 'Largo'. En la cabecera se indica entre paréntesis el largo máximo del bin o rollo a cortar, para hacer contraste con los resultados obtenidos. Mientras menor sea el largo del resultado entregado por el software, mejor es el plan de corte.

Se observa en este caso que no siempre los optimizadores comerciales fueron capaces de comprender o de optimizar de acuerdo a la variante del problema que se les entregó, a pesar de haber hecho todas las configuraciones posibles para esto. Por ejemplo, en las tablas 8.14 y 8.15, se puede observar que CutLogic 2D y CutList Optimizer entregan planes de corte que utilizan casi en su totalidad el largo del rollo de material, resultados que para una empresa textil, por ejemplo, que utiliza mucho este tipo de optimización, simplemente no le sirve.

En esta comparativa, 2D Solver es el ganador indiscutido, dado que en casi todos los ejemplos fue capaz de encontrar la solución más óptima. Cutting Optimization Pro fue el segundo optimizador con mejor desempeño, igualando a 2D Solver en el la tabla 8.12 y derrotándolo en la tabla 8.15

### 8.1.4. Comparación *knapsack problem*

*Knapsack problem*, o el problema de la mochila, corresponde a una variante del problema de optimización donde cada pieza a cortar posee un valor y se busca obtener el plano de corte que contenga, prioritariamente, a las piezas de mayor valor.

No se pudo realizar cuadros comparativos de esta funcionalidad dado que ningún optimizador comercial de los abordados en este trabajo cuenta con aquella funcionalidad. Esto le da a 2D Solver otra característica diferenciadora, que podría ser útil si se encuentra un segmento de clientes que necesiten este tipo de optimización.

## 8.2. Cuadros comparativos de características y funcionalidades

En las tablas 8.16, 8.17, 8.18, 8.19 y 8.20 se hace una comparación exhaustiva de las funcionalidades más relevantes que ofrecen los software de la competencia y el prototipo desarrollado 2D Solver en este trabajo de título.

A continuación se listan los software a comparar:

- Cut Logic 2D : **CL2D** [24]
- Cutting Optimiation Pro: **COP** [23]
- CutList Optimizer: **CLO** [28]
- Opti Cutter: **OC** [27]
- 2D Load Packer: **2DLP** [26]
- Corte 7: **C7** [25]
- 2D Solver: **2DS**

### 8.2.1. Comparación de importación

Tabla 8.16: Comparación de la funcionalidad de importación entre distintos optimizadores comerciales y el desarrollado.

| Importación | CL2D | COP | CLO | OC | 2DLP | C7 | 2DS |
|---|---|---|---|---|---|---|---|
| Portapapeles | x | x | | | x | | |
| CSV | x | | x | x | x | | x |
| MS Excel | x | x | | x | | | |
| MS Access | x | | | | | | |
| Base de datos | x | | | | x | | |

### 8.2.2. Comparación de exportación

Tabla 8.17: Comparación de la funcionalidad de exportación entre distintos optimizadores comerciales y el desarrollado.

| Exportación | CL2D | COP | CLO | OC | 2DLP | C7 | 2DS |
|---|---|---|---|---|---|---|---|
| PDF | x | x | x | x | x | x | x |
| CSV | x | | | x | | x | x |
| MS Excel | x | x | | x | | x | |
| AutoCAD DXF | x | | | | | x | |
| XML,HTML | x | | | | | x | |
| RTF, ODT, ODS | x | | | | | | |
| JPG,PNG | x | | x | | | | |

### 8.2.3. Comparación de configuraciones y parámetros

Tabla 8.18: Comparación de las configuraciones de optimización y ajuste de parámetros presentes en distintos optimizadores comerciales y el desarrollado.

| Configuraciones y parámetros | CL2D | COP | CLO | OC | 2DLP | C7 | 2DS |
|---|---|---|---|---|---|---|---|
| Ancho de sierra de corte | x | x | x | x | | x | |
| Corte de margenes en bin | x | x | | | | x | |
| Dimensión mínima de remanentes | x | x | x | | | | x |
| Guardado de remanentes | x | | | | | x | |
| Dimensiones de la mesa de corte | x | | | | | x | |
| Limitar cortes vert. / hor. | x | x | | | | x | |
| Limitar cortes por etapa | | | | | | | |
| Orientación de primer corte | | x | | | | | |
| Rotación de ítems | x | | | x | | | x |
| Gestión de zonas defectuosas | | | | | | | x |
| Copias infinitas bin | x | | | | | x | x |
| Largo / ancho infinito bin | | | | | | | x |

### 8.2.4. Comparación de propiedades de optimización

Tabla 8.19: Comparación de las propiedades de la optimización presentes en distintos optimizadores comerciales y el desarrollado.

| Optimización | CL2D | COP | CLO | OC | 2DLP | C7 | 2DS |
|---|---|---|---|---|---|---|---|
| Por altura/ancho | x | | | | | | x |
| Por costo | x | x | x | | | x | x |
| Por valor de corte | | | | | | | x |
| Múltiples Bins | | | x | x | | x | x |
| Considerando Restos | x | | | | | | x |
| Nesting (No guillotinado) | x | x | | | | x | |
| Guillotinado | x | x | x | x | x | x | x |
| 2 Etapas | x | x | | | | | x |
| 3 Etapas | x | x | | | | | x |
| Corte Exacto | x | x | | x | x | | x |
| Corte No Exacto | x | | | | | | x |
| Corte Homogéneo | | | | | | | x |

### 8.2.5. Comparación de otras características

Tabla 8.20: Comparación de características misceláneas entre diferentes optimizadores comerciales y el desarrollado.

| Otras características | CL2D | COP | CLO | OC | 2DLP | C7 | 2DS |
|---|---|---|---|---|---|---|---|
| Reportes predefinidos | x | x | x | x | x | x | x |
| Reportes editables | x | | | | | x | |
| Edición manual de solución | x | x | | | | x | |
| Máquinas CNC y Código G | x | | | | | x | |
| Múltiples soluciones | | | | | | x | x |
| Dirección del grano | | x | x | | | x | |

### 8.2.6. Discusión de resultados

Se puede observar que existen optimizadores más completos y con más características y funcionalidades que 2D Solver. Estos son CutLogic 2D y Corte 7, ambos software de escritorio con ya una larga trayectoria en el rubro de la optimización. En particular, ambos ofrecen cortes no guillotinados, que son una variante del problema de optimización más compleja y que no se planea abordar con el software 2D Nest, dada su complejidad matemática de resolución.

CutList Optimizer y Opticutter son ambos aplicaciones web, más livianas y con funcionalidades más restringidas que los casos anteriores. Estos 2 programas son los competidores más directos de 2D Solver, dado que ambos son aplicaciones web y ofrecen características similares. En el estado actual de desarrollo, estos programas comerciales superan al prototipo 2D Solver en características de usabilidad, sin embargo, 2D Solver cuenta con una versatilidad y potencia de optimización mayor, lo cual le entrega una ventaja clave frente a estos competidores, dado que para alcanzarlos sólo se debe trabajar en funcionalidades relacionadas a la experiencia del usuario.

Finalmente, tenemos al software 2D Packing Loader, que a pesar de ser un software totalmente desarrollado y en fase de producción, es simplemente un producto muy inferior a lo que ya ofrece 2D Solver, tanto en su versatilidad de optimización, como en su usabilidad, que simplemente es tan mala que no se pudo utilizar para las comparativas anteriores, dado el enorme tiempo que supone aprender a usar el programa.

Las comparativas anteriores permiten vislumbrar las características valiosas del prototipo y sus carencias. Se puede apreciar que en términos de propiedades de optimización, 2D Nest cumple con la mayoría, lo que le da una ventaja importante en comparación al resto, ya que son competencias medulares del algoritmo de optimización, que necesitan bastantes horas de desarrollo para igualar. Respecto a las carencias, 2D Nest está atrás respecto a características que mejoran la experiencia de usuario, que si bien son relevantes, requieren un esfuerzo menor para imitar e incluso mejorar, respecto a los otros software. En base a lo anterior se puede asegurar que el prototipo posee un gran potencial para entrar a competir y ganar clientes en el mercado objetivo de 1D Solutions.

## 8.3. Evaluación de características críticas

A continuación se presenta una evaluación de las funcionalidades más relevantes de un programa de optimización. Para evaluar se elaboró una rúbrica con la que se contrastarán distintos programas de optimización comerciales y el desarrollado en este trabajo de título. La rúbrica evalúa 5 características críticas, asignando un puntaje de 1 a 3 dependiendo del nivel de completitud de la característica. Para tener una mirada externa en la evaluación, se le pidió a un usuario de prueba que realizaran la evaluación del optimizador 2D Solver. El resto de los programas fueron evaluados por el autor de este trabajo de título. A continuación se presenta la rúbrica de evaluación para cada característica crítica:

### 8.3.1. Archivos de entrada

1. El software permite importar un único tipo de archivo con un formato particular.
2. El software permite importar archivos con un set limitado de formatos y acepta menos de 3 extensiones distintas.
3. El software permite importar archivos independiente del formato de los datos y acepta más de 3 extensiones distintas.

### 8.3.2. Proceso de importación

1. El proceso de importación es tedioso y complicado. El software no comprende la información presente en el archivo. No permite corregir la información.

2. El proceso de subida es suficientemente rápido y sencillo. El software comprende la información presente en el archivo pero no permite corregir o ajustar la información entrante.

3. El proceso de importación rápido y sencillo. Comprende la información presente en el archivo y permite corregir y ajustar de manera simple la información entrante.

### 8.3.3. Variantes del problema de optimización

1. El software no acepta distintas variantes del problema de corte y empaquetado en 2D. Sólo optimiza de acuerdo al porcentaje de pérdida de material.

2. El software acepta distintas variantes del problema de corte y empaquetado en 2D. El usuario debe configurar manualmente la optimización para obtener resultados de acuerdo a la variante del problema.

3. El software acepta distintas variantes del problema de corte y empaquetado en 2D. Comprende a partir de los datos de entrada la variante del problema y se configura automáticamente para entregar el resultado de optimización más acertado.

### 8.3.4. Velocidad de optimización

1. El software entrega resultados en más de 5 segundos, sin la posibilidad de agilizar el proceso.

2. El software entrega resultados en más de 5 segundos, permitiendo interrumpir el proceso para agilizar la obtención de resultados.

3. El software entrega resultados de manera instantánea, o en menos de 5 segundos.

### 8.3.5. Visualización de resultados

1. El software entrega una visualización de resultados confusa y poco comprensible. Entrega información útil para el corte, sin herramientas interactivas. No entrega estadísticas del proceso de optimización.

2. El software entrega una visualización de resultados estética, comprensible y útil. Entrega información útil para el corte, sin herramientas interactivas. Entrega estadísticas del proceso de optimización

3. El software entrega una visualización de resultados clara y comprensible. Entrega herramientas interactivas que permiten al usuario visualizar el orden de los cortes, la posición de los cortes, las dimensiones de los Ítems y estadísticas del proceso de optimización.

### 8.3.6. Cuadro de evaluación

En la siguiente tabla 8.21 se resumen los resultados de la evaluación:

Tabla 8.21: Cuadro de evaluación entre los optimizadores comerciales comparados anteriormente y el optimizador desarrollado.

| Software | Archivos | Importación | Variantes | Velocidad | Visual | Total |
|---|---|---|---|---|---|---|
| Cut Logic 2D | 3 | 3 | 2 | 2 | 3 | 13 |
| Cutting Opt. Pro | 2 | 2 | 1 | 3 | 1 | 9 |
| CutList Optimizer | 1 | 2 | 1 | 2 | 3 | 9 |
| Opti Cutter | 2 | 1 | 1 | 3 | 2 | 9 |
| 2D Load Packer | 1 | 1 | 1 | 1 | 1 | 5 |
| Corte 7 | 2 | 3 | 2 | 2 | 3 | 12 |
| **2D Solver** | **2** | **2** | **3** | **3** | **2** | **12** |

### 8.3.7. Discusión de resultados

Se observa en la tabla 8.21 un muy buen desempeño del software 2D Nest en comparación a la competencia. Si bien es superado por Cut Logic 2D y Corte 7 en algunos aspectos, como la facilidad de importación, versatilidad de archivos y visualización de resultados, 2D Nest destaca por su capacidad de abordar distintas variantes del problema de optimización, característica desarrollada ampliamente en este trabajo de título bajo la consigna del método de optimización inteligente. Se espera que con el trabajo a futuro desarrollado el software pueda igualar y superar a los demás optimizadores en todos los aspectos abordados en esta evaluación.

## 8.4. Experiencia de usuario

Los usuarios objetivo de este desarrollo son personas relacionadas al mundo de la manufactura, tanto de muebles o estructuras de madera como de ventanas. Según el análisis de los clientes de 1D Solutions, que probablemente comparten varias características con los futuros clientes de este desarrollo, la mayoría de ellos son hombres de edad entre 25 a 34 años, que trabajan en el área de ventas, es decir, usan la herramienta para estimar costos del producto o directamente trabajan en el área de manufactura, por ejemplo, operarios de las máquinas de corte guillotinado.

Si bien lo ideal sería hacer pruebas de usuario con personas que calcen perfectamente el perfil anteriormente descrito, se considera que el prototipo actual aun carece de mucho desarrollo respecto a la experiencia de usuario, por lo que se prefirió elegir a usuarios con un mayor nivel de manejo de software y una perspectiva amplia respecto al desarrollo de productos. Por lo anterior, se eligió como usuarios de prueba a dos miembros de FabLab,

laboratorio de fabricación de la facultad de ciencias físicas y matemáticas de la Universidad de Chile.

El objetivo de esta experiencia es validar que usuarios, experimentados en el uso de herramientas de software, logren ejecutar una optimización, con la menor cantidad de intervención externa.

Ambas experiencias fueron grabadas y sintetizadas a continuación:

### 8.4.1. Gonzalo Olave Wolff

Licenciado en Ingeniería eléctrica, miembro del staff de Fablab y co-fundador de la Comunidad de robótica de la U. de Chile. A continuación se resume su experiencia:

Gonzalo entendió intuitivamente que era necesario subir un archivo a la plataforma, primera acción que pudo realizar sin complicaciones, luego utilizó el botón *List Files* , donde intentó sin éxito revisar el contenido del archivo que acababa de subir, a continuación ingresó a la plantilla visual *New Cutting Plan*, donde comentó la falta de unidades de medida. Luego subió el otro archivo faltante a la plataforma y entró a la pantalla de optimización avanzada, donde comentó la necesidad de documentación respecto a los parámetros allí presentes. Luego entró a la pantalla *Smart Optimization*, donde pudo ejecutar la optimización sin problemas. En la página de visualización pudo identificar rápidamente los ítems de los restos y entender los cortes a realizar.

A continuación Gonzalo exploró los archivos disponibles para su prueba de usuario, donde hizo varias preguntas para comprender las columnas del archivo csv, el rol del archivo bin y del archivo ítem. Después de revisarlos comentó la necesidad de agregar información sobre ambos archivos en la pantalla principal para guiar mejor a los usuarios del programa, expresó su preferencia por un plan de corte, es decir, una sección que agrupe los archivos de bins e ítems que forman parte de la misma optimización. Notó la falta de una rueda de carga al apretar botón *Execute* que indique al usuario que debe esperar mientras se ejecuta la optimización. Al realizar una nueva importación de archivo, comentó que el mensaje de éxito debiese mostrar más información sobre el archivo que se acaba de cargar.

Lograda la segunda visualización, Gonzalo hizo varios comentarios respecto a la visualización de resultados, indicando que hace falta más información para el usuario que ocupará el plan de corte para realmente hacer los cortes, en particular el orden de los cortes, las dimensiones del corte y la necesidad de exportar la visualización. También indicó la importancia de incluir el grosor de la sierra en la optimización, para lograr resultados precisos.

A continuación se recogen una serie de citas literales de su experiencia de usuario.

- “ Si yo tengo un archivo, un CAD, que quiero ocupar, lo exporto en milímetros o en pulgadas?.”
- “Sería genial, por ejemplo, saber que significa esto , que me diga qué es lo que hace esta función o me lleve a un link que me diga qué significa esa función objetivo.”

- “ De partida, si tú no me dices y yo no te pregunto no sé que significa esto, igual es importante que sea auto-explicativo. (refiriéndose a la pantalla principal del programa) ... yo no siempre voy a estar seguro de como se hace,... si alguien quiere hacer algo más avanzado, ... que esto te apañe como a entender.”
- “Aquí a lo mejor sería bueno que saliera el nombre del archivo (Refiriéndose al mensaje de éxito de la pantalla de importación). Y que fue si un bin o un ítem, porque me dice que se cargó pero yo no me acuerdo si subí el bin o el ítem primero... Podría tener un historial de los últimos archivos subidos.”
- “¿En qué orden debo seguir los cortes?... Sería genial que me dijera : plancha 1 dimensión vertical corte uno tanto.”
- “¿Aquí el corte es adimensional? porque claro si tengo una sierra de 2 milimetros o 5 milimetos más el ítem no quedara de la dimensión que tu quieres.”

De esta experiencia de usuario se extraen las siguientes conclusiones:

1. Es necesaria una explicación detallada de los archivos ítem y bin necesarios para la optimización, sea como archivo de ayuda o como un Wizard.
2. Se debiese iniciar la cuenta de un usuario con un par de archivos de prueba, que le muestren formatos aceptables de archivo y comentarios explicativos.
3. La funcionalidad *List Files* debiese permitir revisar el contenido de los archivos.
4. La optimización avanzada debiese tener una documentación que le permita a usuarios más experimentados aprender a configurar la optimización a su gusto.
5. Es beneficioso implementar una rueda o barra de carga que indique al usuario el tiempo estimado a esperar para el resultado de la optimización.
6. Es necesario agrupar los archivos de ítems y bins en un plan de corte, que almacene también la solución de la optimización y los metadatos de la operación.
7. Se debe implementar una selección de unidades de medida.
8. Se debe implementar la optimización considerando el grosor de la hoja de corte.
9. Se debe entregar más herramientas para el operador de corte en la pantalla de visualización, en particular el orden de los cortes, la dimensión de cada ítem y un formato para exportar e imprimir.

### 8.4.2. Victor Contreras Rojas

Diseñador industrial y miembro del staff de Fablab. A continuación se resume su experiencia:

A Victor se le dió las mismas explicaciones que a Gonzalo, y una breve explicación de los archivos bins e ítems. Victor inició su experiencia subiendo un archivo de bins. Luego de

explorar varias secciones del programa, descubrió que necesitaba otro archivo para realizar la optimización. Subió el archivo restante y utilizó el formulario de *Smart Optimization* para llegar a la pantalla de visualización de resultados. En la pantalla de visualización comentó que preferiría tener la opción de definir en el programa la plancha utilizada.

A continuación Victor subió un nuevo archivo de ítems de ejemplo, intentó hacer una nueva optimización mezclando este nuevo archivo con el archivo de bins anterior, lo cual resultó en un error del programa dado que el archivo de bins no contenía suficientes copias para hacer la optimización, frente a lo cual Victor comentó que un comportamiento deseable del optimizador sería añadir las planchas necesarias para realizar la optimización y alertar al usuario del problema.

A continuación se recogen una serie de citas literales de su experiencia de usuario.

- " Lo primero que haría yo es que cambiaría el que tengas que subir una plancha, y dejaría un slot para definir yo el tamaño de plancha que tengo. "
- " En los que he usado yo, yo meto cortes, ... , la cuestión lo optimiza y te tira los cortes mejor distribuidos en la plancha, pero te va sumando planchas, no aparece como un error"
- " Si quieres trabajar con 2 archivos, haría algo visual donde la persona entienda que aquí va el archivo de las planchas y aquí el archivo de los cortes "

De esta experiencia de usuario se extraen las siguientes conclusiones:

1. Se debe evitar que la optimización inteligente entregue error por falta de copias de bins. Es mejor que se haga la optimización y se alerte al usuario que las copias especificadas eran insuficientes.
2. Si bien el ingreso manual de datos es una forma ineficiente para crear los archivos, tiene valor para usuarios primerizos que quieren entender y usar rápidamente la herramienta.
3. Se debe dar mayor flexibilidad al usuario para definir su plan de corte, una funcionalidad deseable es poder mezclar un archivo de ítems con una definición manual de las dimensiones del bin.

# Capítulo 9

# Conclusiones

Finalizado el trabajo de memoria, se puede concluir que se cumplieron satisfactoriamente los objetivo específicos planteado al inicio de esta labor en el plazo contemplado y con la calidad deseada.

En base a los canales de marketing y software de gestión de las ventas de 1D Solutions, se obtuvo una caracterización cuantitativa de la empresa y sus clientes. De esta caracterización se descubre que el perfil de cliente de 1D Solutions son hombres adultos, con un rango etario predominante entre los 25 a 34 años. Se concluye que el público más atractivo y con mejor disposición a pago es el público estadounidense y que el inglés es el lenguaje más relevante a implementar en el prototipo. También se observa que el modelo de precios de 1D Solutions no logra convencer a los países latinoamericanos, principalmente a Argentina y Brasil, que lideran en descargas pero no en ventas. Finalmente se concluye que el nuevo producto a desarrollar debiese implementar un modelo de subscripción en vez de un sólo pago por una licencia de por vida.

Se crea una página web donde se implementan prototipos de baja resolución para validar tempranamente el desarrollo y se hace una campaña de mail marketing dirigida a los clientes de los últimos 2 años. De esta experiencia, se concluye que el prototipo debe ser una aplicación web de optimización en 2 dimensiones, por su compatibilidad entre dispositivos y sistemas operativos.

Se obtienen 2 recursos de optimización: el software Packing solver, que integra 5 algoritmos de inteligencia computacional basados en árboles de búsqueda y Genetic solver, que implementa 3 algoritmos genéticos para resolver el problema de optimización. De su estudio a profundidad se concluye usar únicamente Packing solver por su versatilidad, potencia y problemas de compatibilidad con Genetic solver. También se eligen los 2 algoritmos más robustos de Packing solver: IBS y MBA*, para ser ejecutados en paralelo en los núcleos del dispositivo y obtener el mejor resultado entre ambos.

Se desarrolla un prototipo de alta resolución de una aplicación web de optimización de corte guillotinado en 2 dimensiones, basada en los dos algoritmos de árbol de búsqueda mencionados anteriormente. El prototipo, llamado 2D Nest, integra un **método de optimización inteligente** que es capaz de interpretar los archivos entregados por el usuario,

para ajustar de manera automática los parámetros de optimización y ejecutar procesos de optimización diferentes en paralelo, aprovechando los distintos núcleos del hardware que lo soporta y entregando distintos resultados de optimización en base a la variante del problema de corte y empaquetado que el usuario desea resolver, abstrayendo al usuario de todas las configuraciones avanzadas necesarias para el proceso.

Finalmente, se compara el optimizador desarrollado con otros optimizadores comerciales del mismo tipo, se evalúan las características críticas del prototipo en base a una rúbrica y se hacen 2 experiencias de usuario. De lo anterior se concluye que el prototipo muestra un excelente desempeño en comparación a la competencia, si bien es a veces superado en algunos aspectos, como la facilidad de importación, versatilidad de archivos y visualización de resultados, el prototipo 2D Nest supera a todos por su capacidad de abordar distintas variantes del problema de optimización y por su velocidad de optimización, características que lo colocan indiscutiblemente en el podio de esta competencia, más aun, considerando que se trata de un prototipo y no de un producto totalmente desarrollado.

En base a todo lo recién expuesto, se puede concluir, con seguridad, que el objetivo general del trabajo de título fue cumplido satisfactoriamente. El prototipo desarrollado es competitivo en todas las características claves definidas e incluso supera ampliamente a otros productos del mercado. La empresa 1D Solutions se ha manifestado conforme con los resultados del proyecto y desean, en un futuro, seguir con la segunda fase del desarrollo, invirtiendo más recursos para formar un equipo de desarrollo liderado por el autor de este trabajo, con el fin de evolucionar el prototipo desarrollado hacia un producto listo para el mercado.

Este resultado no se podría haber logrado sin aplicar las diversas herramientas que la formación de ingeniero eléctrico entrega a sus alumnos. La comprensión acabada y profunda de algoritmos de inteligencia computacional fue clave para la implementación exitosa del software de optimización Packing solver, desarrollada por terceros y que contaban con una documentación prácticamente nula. En este mismo punto, las herramientas de programación adquiridas en los distintos ramos y proyectos universitarios fueron una base muy necesaria para apuntar a un desarrollo informático tan ambicioso como es este proyecto. Además de todas las habilidades “duras”, las competencias en innovación y emprendimiento, que también forman parte de la malla curricular de ingeniería eléctrica tomaron un rol protagónico al inicio del proyecto, puesto que definir una problemática, diseñar un producto en torno a usuarios finales y gestionar el proceso de desarrollo es tan o más complejo como el desarrollo del software en si mismo.

## 9.1. Trabajo a futuro

En base a las experiencias de usuario, a la comparativa de funcionalidades de otros software comerciales y a la propia visión del desarrollador, se procede a discutir el trabajo a futuro que resta al proyecto.

En primer lugar, se debe suplir al prototipo de varias funcionalidades que, a pesar de no ser críticas para el funcionamiento del programa, son útiles para mejorar la experiencia de los futuros usuarios del optimizador:

- Extender la importación de archivos a Excel, portapapeles e implementar el ingreso manual de datos.
- Mejorar la gestión de usuarios, en particular, crear perfil de usuario con preferencias y permitir guardar y gestionar planes de corte y sus resultados.
- Permitir exportar los resultados de optimización a imágenes o diagramas con distintas extensiones de archivo.
- Mejorar visualización de resultados con herramientas interactivas que permitan seguir en orden los cortes y conocer su posición relativa en el bin.
- Desplegar tabla con la posición absoluta de los cortes a realizar en visualización e incorporar también en la exportación.
- Generar y habilitar a los usuarios documentación del programa.
- Implementar archivos de prueba por defecto, para explicar a usuarios primerizos las funcionalidades de la plataforma.
- Permitir visualizar las listas de ítems o bins en la plataforma.
- Implementar *widget* de espera durante la optimización e inhabilitar botón *execute* para no generar procesos de optimización paralelos indeseados.
- Implementar unidades de medida en visualización y en el ingreso manual de datos.
- Implementar optimización considerando el ancho de la hoja de corte.
- Mejorar el proceso de optimización inteligente para evitar el error por falta de bins.

También, se desea rehacer las comparativas con los otros optimizadores comerciales, variando el tiempo de ejecución máximo para reducir aún más los porcentajes de pérdida.

### 9.1.1. Paso a producción

Pasar a producción es el paso final del desarrollo de cualquier aplicación web. Se refiere al proceso de subir la aplicación a un servidor, conectarla con un servicio de base de datos y con otros servicios para hacerla accesible por cualquier usuario de internet.

Como primer paso se encontró Coockie Cutter, un *framework* que permite pre-configurar un proyecto Django para que esté listo para la fase de producción. El *framework* presenta una extensa serie de alternativas para conectar el proyecto Django con distintos software de apoyo, tanto para la base de datos, como el *hosting*, como el *email server*, gestión de archivos estáticos, etc.

Gracias a este *framework* se descubrió Heroku, un servicio que se hace cargo de la gran parte del trabajo de pasar a producción una plataforma web y entregar un servicio *cloud*. También se descubrió WhiteNoise, otro *framework* que permite simplificar la entrega de

archivos estáticos en software basado en Python y MailHog; es un servicio para *mails* en aplicaciones basadas en Python.

Otro factor a considerar es que el actual Packing solver está compilado para Windows, de pasar a producción el programa se ejecutará en un servidor Linux, por lo que se tendrá que crear una máquina virtual Linux local y volver a compilar el programa para este sistema operativo. Afortunadamente el desarrollador de Packing solver lo hizo en Linux, por lo que hay certeza de que este programa puede ser compilado y ejecutado en ese sistema operativo. También se debe recompilar el programa usando *COIN-OR Linear Programming*, un programa de optimización matemática de código abierto que reemplaza a CPLEX y evita ciertos conflictos legales.

Considerando el paso a producción, se listan una serie de tareas que deberán ser abordadas a futuro:

- Implementar distinción de clientes gratuitos y no gratuitos y habilitar las funcionalidades dependiendo de su categoría.
- Definir los planes de usuarios premiun y el modelo de precios.
- Integrar sistema de terceros para el pago de licencias premiun. MyCommerce es la opción prevista.
- Levantar y monitorear métricas de tráfico con Google Analytics.
- Levantar y monitorear campañas de marketing en Google Ads.
- Implementar un CRM para gestionar las interacciones con clientes en la plataforma. Hubspot es la opción prevista.

# Bibliografía


[1] Datawheel. Información del comercio mundial de perfiles de acero, tuberías de acero, Aluminio y articulos de aluminio. *Observador de complejidad económica mundial* [En línea] Disponible: https://oec.world/es/profile/hs92/iron-blocks https://oec.world/es/profile/hs92/iron-pipes https://oec.world/es/profile/hs92/barrod-of-high-speed-steel-not-in-coil https://oec.world/es/profile/hs92/barrod-of-silico-manganese-steel-not-in-coils https://oec.world/es/profile/hs92/barrod-alloy-steel-nesnfw-hot-rolleddrawnextrude https://oec.world/es/profile/hs92/raw-aluminium https://oec.world/es/profile/hs92/aluminium-and-articles-thereof

[2] Dinngo. Design Thinking en Español. [En línea] Disponible: https://www.designthinking.es/inicio/index.php

[3] H. Dyckhoff *"A typology of cutting and packing problems"* en European Journal of Operational Research 44 pp. 145–159, 1990.

[4] R. Macedo et al *"2D Cutting Stock Optimization Software Survey"* proyect SCOOP. 2008.

[5] M. Angamuthu *"A review of optimization techniques in metal cutting processes"* en Journal Computers and Industrial Engineering 50 pp. 15–34, 2006.

[6] C.P. García *"Métodos y Algoritmos para resolver problemas de Corte unidimensional en entornos realistas. Aplicación a una empresa del Sector Siderúrgico."* Tesis Doctoral, Universidad Politécnica de Valencia.

[7] Aryanezhad et al.: *"A simple approach to the twodimensional guillotine cutting stock problem."* Journal of Industrial Engineering International 2012 8:21.

[8] L. Libralesso, F. Fontan, *An anytime tree search algorithm for the 2018 ROADEF/EURO challenge glass cutting problem,* arXiv:2004.00963 [cs] http://arxiv.org/abs/2004.00963 arXiv: 2004.00963

[9] K. Li, J. Malik, *Learning to Optimize* arXiv:1606.01885, 2016 and International Conference on Learning Representations (ICLR), 2017

[10] MailChimp. Análisis comparativos y estadísticas de email marketing por sector. [En línea]. Dispible: https://mailchimp.com/es/resources/email-marketing-benchmarks/

[11] L. Horný, *Optimization of Board Cutting* Diploma Paper, Study program: Electrical Engineering and Informatics, Master's. May 2008

[12] Python Software Foundation. Virtual enviroment documentation. [En línea]. Disponible: https://docs.python.org/3/library/venv.html

[13] Django Software Foundation. Django Framework documentation. [En línea]. Disponible: https://docs.djangoproject.com/en/4.0/

[14] Start Bootstrap LLC. (May, 2021). Bootstrap template. https://startbootstrap.com/template/sb-admin

[15] Bootstrap team. Bootstrap documentation.[En línea]. Disponible: https://getbootstrap.com/docs/5.1/getting-started/introduction/

[16] Mike Bostock. (July, 2021). D3 Data-Driven Documents wiki. [En línea]. Disponible: https://github.com/d3/d3/wiki

[17] Northwoods Software. Get Started with GoJS. [En línea]. Disponible: https://gojs.net/latest/learn/

[18] Defu Zhang , Leyuan Shi , Stephen C.H. Leung , Tao Wu *A priority heuristic for the guillotine rectangular packing problem* January 2016 Information Processing Letters 116(1):15-21 DOI:10.1016/j.ipl.2015.08.008

[19] Guochuan Zhang. *On Variable-Sized Bin Packing* August 2002, Kiel Universitat, Germany.

[20] Roberto Baldacci, Marco A. Boschetti, Maurizio Ganovelli, Vittorio Maniezzo, *Algorithms for nesting with defects,* Discrete Applied Mathematics, Volume 163, Part 1, 2014,Pages 17-33, ISSN 0166-218X, https://doi.org/10.1016/j.dam.2012.03.026. https://www.sciencedirect.com/science/article/pii/S0166218X12001308

[21] Brilliant.org. A* Search. [En línea]. Disponible: https://brilliant.org/wiki/a-star-search/

[22] Patrick Winston. *6.034 Artificial Intelligence. Fall 2010.* Massachusetts Institute of Technology: MIT OpenCourseWare, https://ocw.mit.edu.com License: Creative Commons BY-NC-SA.

[23] Optimal Programs. Cutting Optimization Pro user manual.[En línea].Disponible: https://www.optimalprograms.com/help/cutting_optimization_pro_en/index.htm

[24] Tmachines(2002). Cut Logic 2D Online Help. [En línea]. Disponible: https://www.tmachines.com/cl2d/help/

[25] Corte Óptimo. Corte 7 soporte en línea. [En línea]. Disponible: https://corteoptimo.com/corte7-ayudaenlinea

[26] Astrokettle. 2D Load Packer. [En línea]. Disponible: https://www.astrokettle.com/pr2dlp.html

[27] OptiCutter. OptiCutter online features. [En línea]. Disponible: https://www.opticutter.com

[28] CutList Optimizer. Online optimization Tool CutList Optimizer. [En Línea]. Disponible: https://www.cutlistoptimizer.com

[29] S. Zamora, E. Sologuren, A. Galdames, F. Lillo, I. Lobos *Guías para escribir el informe final de proyecto de título de ingeniería civil informática*