Ingeniería Investigación y Tecnología, volumen XVII (número 2), abril-junio 2016: 179-190
ISSN 1405-7743 FI-UNAM
(artículo arbitrado)
doi:10.1016/j.riit.2016.06.003

Un algoritmo para el *Strip Packing Problem* obtenido mediante la extracción de habilidades de expertos usando minería de datos

An Algorithm for the Strip Packing Problem Obtained by Extracting Expert Skills Using Data Mining

Gatica Gustavo
Universidad Andrés Bello, Chile
Universidad de Santiago de Chile
Facultad de Ingeniería
Correo: ggatica@unab.cl

Reyes Pablo
Universidad Andrés Bello, Chile
Facultad de Ingeniería
Correo: pabl.reyes@uandresbello.edu

Contreras-Bolton Carlos
Universidad de Santiago de Chile
Departamento de Ingeniería Informática
Correo: carlos.contrerasb@usach.cl

Linfati Rodrigo
Universidad del Bío-Bío, Chile
Departamento de Ingeniería Industrial, Chile
Correo: rlinfati@ubiobio.cl

Escobar John Willmer
Universidad del Valle, Colombia
Departamento de Contabilidad y Finanzas
Correo: john.wilmer.escobar@correounivalle.edu.co

Información del artículo: recibido: mayo de 2014, reevaluado: enero de 2015, aceptado: septiembre de 2015

Resumen

La capacidad del ser humano para resolver problemas NP-Duro de forma manual no ha recibido la debida atención por la comunidad científica. Este artículo considera el problema del *Strip Packing*, que consiste en posicionar ortogonalmente un conjunto de piezas rectangulares dentro de un contenedor de ancho fijo y altura infinita, sin solaparlas, minimizando la altura alcanzada de las piezas dentro del contenedor. Se desarrolló un juego computacional que permite obtener soluciones manuales, propuestas por jugadores expertos, para distintas instancias del problema. La contribución del artículo consiste en presentar un algoritmo que se extrajo mediante patrones y minería de datos aplicada a soluciones encontradas por los jugadores expertos. El algoritmo generado se basa en elementos de árboles y heurísticas presentes en la literatura. Adicionalmente se presentan resultados computacionales, donde se logra encontrar la mejor solución conocida en 94.3% de un conjunto de instancias de la literatura y 79% para instancias generadas aleatoriamente.

Descriptores:

- *Strip Packing Problem*
- minería de algoritmos y patrones
- habilidades de expertos

Abstract

The ability of the humans to manually solve NP-hard problems had not received much attention of the scientific community. This paper considers the Strip Packing Problem (SPP), in which a set of rectangular pieces has to be placed orthogonally in a container with a given width and an infinite length. The pieces are not allowed to overlap (i.e. be stacked one over the other). The aim of the SPP is to minimize the overall length of the strip. In this paper, we have developed a computational game to allow manual solutions by expert gamers for different instances of the problem. The main contribution of the paper is the presentation of an algorithm based on patterns and data mining retrieved from the results achieved by expert gamers. The proposed algorithm is based on decision-trees and heuristics proposed in literature. Finally, the proposed approach is able to find the best-known solutions for the 94.3% of a set of instances proposed in the literature, and 79% for instances generated randomly.

Keywords:

- *Strip Packing Problem*
- *data mining of algorithms and patterns*
- *skills of experts*

Introducción

El problema del *Strip Packing* (SPP) se define a partir de una región rectangular de ancho W definido y alto infinito, en el cual se deben ubicar todas las piezas de un conjunto predefinido $R = \{r_1, r_2, \ldots, r_n\}$ que tienen dimensiones de ancho w_i y alto h_i, sin sobreponerlas, con el fin de minimizar la altura H utilizada en el contenedor (Gatica *et al*., 2015). En este artículo se analiza el SPP de dos dimensiones (2-SPP), con rotación de piezas en 90° y sin corte guillotina (Lodi *et al*., 2002). El SPP se deriva de la familia de problemas de corte y empaque, y pertenece a la clase NP-Duro; por ello la búsqueda de algoritmos eficientes es un área de constante evolución. Este tipo de problema es de gran interés en industrias tales como papeleras y madereras, debido a que está directamente asociado a la optimización del uso de materias primas, para así, reducir los costos de producción (Alvarez *et al*., 2008).

La solución de problemas NP-Duro mediante la utilización de la capacidad del cerebro humano recibe una baja atención, principalmente se analiza desde el punto de vista de la complejidad computacional. En la línea de la capacidad del cerebro humano, solo Acuña y Parada (2010) generan un algoritmo, que se obtuvo mediante la capacidad de 28 jugadores, quienes participaron en un torneo con un juego computacional del vendedor viajero. En tanto, desde el punto de la complejidad computacional, se han presentado muchos juegos del tipo rompecabezas combinatorios tan o más difíciles que cualquier problema del mundo real. Más aún, una disciplina como la teoría de los juegos combinatorios se encarga de analizar conocidos juegos que envuelven complejas decisiones, y de clasificar su dificultad (Fraenkel, 1997).

El fanatismo es un factor a considerar por el nivel de especialización que logran los jugadores, como muestran los estudios de Russell y Norvig (1995), así como De Jong y Schultz (1988). El esfuerzo usual de dedicación se transforma en una adquisición de conocimiento progresivo que desencadena nuevas y mejores técnicas para juegos NP-Completo (Sunnucks, 1970) tales como: buscaminas (Stewart, 2005), tetris (Demaine *et al*., 2002), criptoaritmética (Epstein, 1987). Sólo resta imaginar todas las horas, días y años que invierten miles de jugadores alrededor del mundo en estos juegos obteniendo un conocimiento adquirido en el cerebro, lo que los convierte en expertos. Si los juegos son reflejo de bien conocidos problemas NP-Completo: ¿qué tan buenas son las heurísticas utilizadas por los seres humanos para resolver tales juegos?

En este artículo se analiza la capacidad humana para descubrir una heurística mediante la solución de un juego computacional. Se analizan las jugadas obtenidas y se identifican los patrones típicos de juegos para descubrir ¿Cuál es la heurística que utilizan los distintos jugadores? Para dicha labor se emplean técnicas de aprendizaje automático. Los resultados obtenidos proporcionan un algoritmo de solución basado en árboles de decisión, los cuales presentan elementos de algoritmos y heurísticas existentes en la literatura. Se puede concluir que la heurística generada es capaz de resolver dos conjuntos de instancias de evaluación comparativa de la literatura, igualando los mejores resultados conocidos en 94.3% de los casos y en 79% para instancias generadas aleatoriamente.

En la siguiente sección se presentan los materiales y métodos, donde se explica cómo se diseña un experimento que permita recuperar patrones típicos de jugadores. Después se realiza un análisis de los resultados y finalmente algunas reflexiones.

Materiales y métodos

El desarrollo de este trabajo se puede dividir en dos etapas, la primera es desarrollar el juego computacional, y la segunda la aplicación de técnicas de descubrimiento de patrones a través de minera de datos.

Desarrollo del juego

La primera etapa consiste en desarrollar el *software*, denominado *Strip Packing Problem Game* (SPPG). El cual consiste en un rompecabezas que representa directamente una instancia del SPP y cumple con la condición de que todas las piezas del juego sean visibles desde el primer instante para poder analizar su complejidad y variabilidad, ya sea por la dimensión de las piezas o el número de ellas (Demaine, 2001). Como en un rompecabezas tradicional, las personas pueden jugar independientemente de los demás, no hay interacción entre ellos y las acciones de uno no afectan al otro. SPPG puede utilizar cualquier instancia del SPP; sin embargo se considerararon cinco instancias distintas en cinco niveles de dificultad, es decir, un total de 25 instancias. No se permite realizar retroceso de jugadas y se cuenta con un tiempo limitado de jugada. El *software* se desarrolló en C# usando .NET 3.5. La figura 1 muestra el diseño del juego, en donde en la parte izquierda se puede visualizar la grilla en donde se deben posicionar las piezas ubicadas en la parte derecha de la misma figura, se observa que todas son rectangulares y pueden rotarse por el jugador.

SPPG se implementa siguiendo la metodología RAD (*Rapid Application Development*), bajo el paradigma orientado a objetos (Jaaksi, 1998), que integra dos componentes: un sitio web-módulo de interacción con la aplicación SPPG y una base de datos diseñada para capturar la trazabilidad de una partida de juego usando MySql 4.1, el cual posteriormente se analiza mediante árboles de decisión para la extracción de conocimiento. La figura 2 muestra la estructura usada para la implementación del juego.

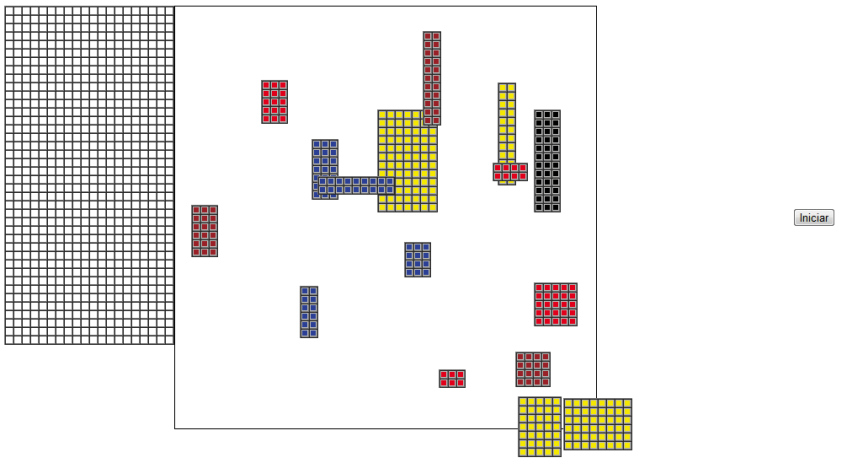

Figura 1. *Strip Packing Problem Game*

Fuente: Elaboración propia

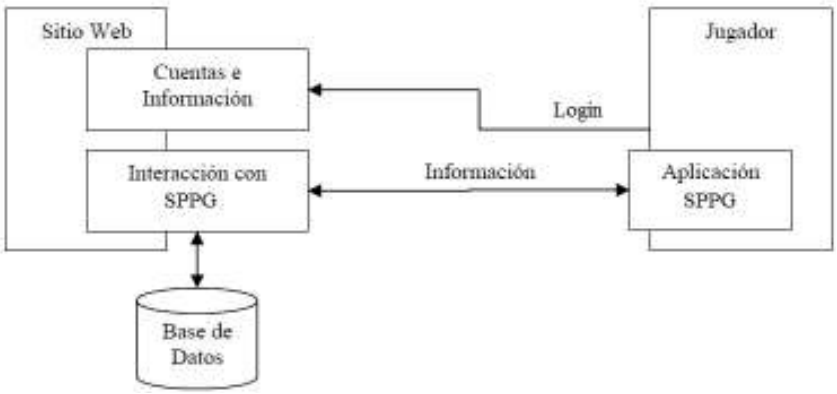

Figura 2. Implementación *Strip Packing Problem Game*

Fuente: Elaboración propia

Aplicación del minado de datos

En esta segunda etapa se aplica el proceso de minado de datos a las soluciones de los juegos para una parte del algoritmo descrito por Klemettien (1999). Esta aplicación típicamente sigue los pasos descritos en la figura 3, la cual se conforma por cuatro etapas: la limpieza y selección de datos, la correspondiente transformación de datos, el reconocimiento de patrones, la interpretación, evaluación y finalmente la generación del algoritmo para resolver el SPPG.

Para realizar un análisis de patrones es necesario contar con una buena representación del registro de las jugadas, el cual almacena las partidas realizadas por los jugadores. En la tabla 1 se observa el registro que contiene nueve columnas: *idJugadas* es el identificador correlativo del movimiento ejecutado, *FK_idPartidas* corresponde a la identificación de la partida realizada, *Rotación*, si la pieza está o no rotada; *Posicion_X*, la posición en el eje x donde la pieza se insertó respecto a la esquina inferior izquierda y *Posicion_Y*, la posición en el eje y dónde se insertó la pieza respecto a la esquina inferior izquierda, *FK_idNivel* corresponde al nivel de dificultad en el que se realiza la jugada, *Numero_pieza*, es el número de la pieza utilizada en la jugada; *Largo*, el largo de la pieza usada en la jugada; *Ancho*, el ancho de la pieza empleada en la jugada.

A continuación se detalla lo que se realizó en cada una de las etapas del minado de datos:

Selección y limpieza de datos: como primera etapa dentro de minado de datos se seleccionan los datos correspondientes a las jugadas completas. Se cuenta con un total de 4.121 jugadas distribuidas entre las 25 instancias del SPPG, utilizadas en esta investigación. Dado que los datos almacenados corresponden a jugadas completas, no fue necesario realizar limpieza de datos, ya que la totalidad fueron jugadas completas.

Transformación de datos: se realiza una codificación de las jugadas para su correcto análisis, cada jugada se anexa a la jugada anterior y a la siguiente. Para obtener las partidas completas, se realiza una codificación piramidal para analizar todas las jugadas posibles, almacenando cada una de las jugadas intermedias desde el inicio del juego.

De esta manera, se pueden realizar comparaciones de jugadas, así como dos jugadores: jugador A, jugador B, pudieron realizar las mismas inserciones en los primeros 5 pasos, pero luego tomaron distintas decisiones.

Para mantener una uniformidad de los factores a utilizar independiente de la configuración del nivel, los valores de las variables se calculan en relación con el nivel, para ello se utilizan deciles, es decir, cada tramo representa 10% del total del nivel, y para las piezas insertadas, se utilizan cuartiles, es decir, cada tramo representa 25% del total del nivel. Los atributos de tamaño se calculan de acuerdo con el espacio disponible para inserción; los atributos relativos a la pieza actualmente insertada se calcula con base en el total de piezas restantes; y los atributos relativos a la jugada actual dentro del problema se calculan en relación con el número total de jugadas posibles (número de piezas).

Descubrimiento y análisis de atributos

Para descubrir los atributos es necesario un amplio conocimiento del campo en estudio. Es necesario identificar la mayor cantidad de atributos que potencialmente describan las clases o atributos clase. Una clase es la inducción que realiza un sistema de aprendizaje automático sobre un conjunto de ejemplos (Witten y Frank, 2000). Además, y aunque parezca una tarea trivial, es necesario describir qué se desea caracterizar, es decir, qué significa la clase en cada ejemplo. En este trabajo los atributos caracterizan la decisión que toman los jugadores cuando seleccionan la pieza a insertar dentro del contenedor. Los atributos se describen a continuación:

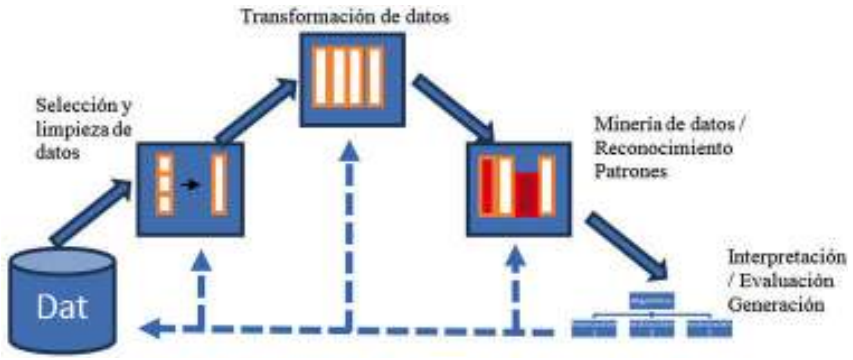

Figura 3. Descubrimiento de conocimiento en bases de datos

Fuente: Witten, 2000.

Tabla 1. Estructura log de datos

IdJugadas	FK_idPartidas	Rotación	Posición_X	Posición_Y	FK_idNivel	Número_Pieza	Largo	Ancho
399	138	0	0	0	1	2	40	16
400	138	0	0	16	1	4	24	24
401	138	1	39	0	1	8	4	20
402	138	0	24	16	1	1	7	6
403	138	0	31	16	1	10	7	6
404	138	1	24	35	1	3	20	5
405	139	0	24	22	2	7	7	8
406	139	0	24	30	2	5	7	4
407	139	1	31	22	2	6	4	4

Fuente: Elaboración propia

Atributo relacionado con la decisión: este atributo indica cuál heurística de inserción realiza el jugador. Las heurísticas consideradas son: inserción decreciente (comienza la inserción con la pieza de mayor tamaño y termina con la pieza de menor tamaño), inserción creciente (se inicia con la inserción de la pieza de menor tamaño y finaliza con la pieza de mayor tamaño), inserción inferior izquierda (comienza la inserción desde el lado inferior izquierdo), inserción mejor encaje (se realizan inserciones que solo completen líneas horizontales en la tira), inserción encaje inicial (realiza inserciones de la primera pieza que ocupe el espacio restante horizontal de la tira), inserción por mayor suma de áreas (realizar inserciones de manera decreciente en combinatoria de piezas, es decir, piezas que se encajan perfectamente se toman como una sola pieza, por ejemplo 2 piezas de 2x1, se toman como una sola pieza de 2x2). La figura 4 muestra los atributos relacionados con la decisión.

Atributos relativos a las piezas no insertadas

El primer conjunto de atributos que aquí se describe están relacionados con algunas características de las piezas aún no insertadas. En el trabajo de Campitelli y Gobet (2005) se muestran algunos indicios de las diferencias entre jugadores expertos y novatos cuando juegan ajedrez, en el sentido de la percepción que tienen del tablero de juego. Por ejemplo, los jugadores expertos logran captar características globales del tablero que les permite recordar cierta jugada que realizaron con anterioridad. Esta percepción global se refleja en la agrupación de piezas que realizan los jugadores, en el caso del SPPG.

En los métodos de agrupación de puntos, generalmente se utiliza el concepto de tamaño promedio, máximo y mínimo. Los tres atributos siguientes implementan estos índices.

1. **Tamaño Promedio** (*SPP_Tamaño_Promedio*): Calcula el área promedio entre todas las piezas que no se han insertado. Mientras mayor este atributo, mayor es el área que existe en las piezas no insertadas. Donde $d(i,j)$ es la diferencia de área entre la pieza i y la pieza j. Si W es el conjunto de piezas no insertadas, la ecuación (1) calcula este atributo

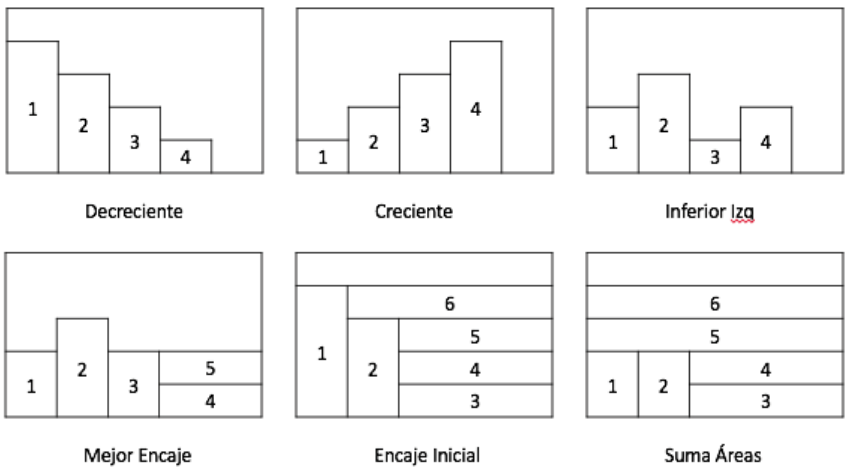

Figura 4. Descubrimiento de conocimiento en bases de datos

Fuente: Elaboración propia

$$SPP\_Tamaño\_Promeedio = \frac{\sum\sum_{\forall i,j \in W} d(i,j)}{|W|(|W|-1)} \quad (1)$$

2. **Tamaño Mínimo** (*SPP_Tamaño_Mínimo*): Calcula el área mínimo que existe entre dos piezas no insertadas. Este atributo refleja cuál es la diferencia mínima de área que tiene la configuración actual de piezas insertadas, definida por la ecuación (2)

$$SPP\_Tamaño\_Mínimo = min\ \{d\ (i,\ j)\ \forall\ i,\ j \in W\} \quad (2)$$

3. **Tamaño Máximo** (*SPP_Tamaño_Máximo*): Calcula el área máximo que existe entre piezas no insertadas. Refleja el máximo área que existe entre las piezas no insertadas. Mientras más cercano este atributo a *SPP_Tamaño_Mínimo*, más homogéneas son las piezas no insertadas, definida por la ecuación (3)

$$SPP\_\ Tamaño\_Máximo = max\{d(i,j)\ \forall\ i,j \in W\} \quad (3)$$

Atributos relativos a la geometría y costos de inserción

La lista de piezas insertadas juegan un papel fundamental en el algoritmo, pues determinan la próxima pieza a insertar. Es por ello que resulta interesante reflejar en un atributo cuál es el costo asociado a la selección de una u otra pieza. Los tres atributos que a continuación se detallan reflejan el costo mínimo, máximo y promedio de inserción de las piezas del problema. En figura 5 se aprecian los distintos enfoques de estos atributos. Suponiendo que las piezas pueden insertarse en A y B, existen tres tipos de costos de acuerdo con cuál se seleccione, se describen a continuación:

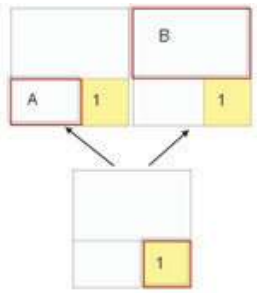

Figura 5. Ejemplo de posicionamiento durante la inserción
Fuente: Elaboración propia

1. **Costo mínimo de insertar pieza** (*SSP_pieza_Insertable_Barata*)**:** Calcula el costo mínimo de inserción dentro de las piezas de la lista de piezas marcadas. En este caso, *PM* corresponde a las piezas marcadas puestas en el contenedor, C_a corresponde a la pieza actual del problema y C_b representa la siguiente pieza. Este atributo, junto con el costo máximo y promedio de inserción, permite reflejar la dispersión que tienen las piezas a insertar en el sentido de su distancia. Mientras mayor es la diferencia entre el costo mínimo y máximo, es más complicado para el jugador seleccionar la próxima pieza a insertar, esto se define con la ecuación (4)

$$SPP\_pieza\_Insertable\_Barata = min\ \{d(i,\ C_a) + d(i,\ C_b) - d(C_a,\ C_b)\ \forall\ i \in PM\} \quad (4)$$

2. **Costo máximo de insertar pieza** (*SSP_pieza_Insertable_Cara*)**:** Calcula el costo máximo de inserción dentro de las piezas de la lista de piezas marcadas. Se asumen las mismas condiciones del atributo *SSP_pieza_Insertable_Barata*, se define en la ecuación (5).

$$SPP\_pieza\_Insertable\_Cara = max\ \{d(i,\ C_a) + d(i,\ C_b) - d(C_a,\ C_b)\ \forall\ i \in PM\} \quad (5)$$

3. **Costo promedio de insertar pieza marcada** (*SSP_pieza_Insertable_Proemedio*)**:** Calcula el costo promedio de insertar las piezas a la tira. Se asumen las mismas condiciones del atributo *SSP_pieza_Insertable_Barata*, se define en la ecuación (6)

$$SPP\_pieza\_Insertable\_Promedio = \frac{\sum_{\forall i \in PM} d(i,C_a) + d(i,C_b) - d(C_a,C_b)}{|PM|} \quad (6)$$

4. **Costo actual de insertar pieza marcada** (*SSP_pieza_Insertable_Actual*)**:** Calcula el costo actual de insertar las piezas a la tira, se obtiene un valor positivo si la pieza a insertar es mayor al espacio horizontal y un valor negativo si la pieza a insertar es menor al espacio horizontal. Se asumen las mismas condiciones del atributo *SSP_piezaInsertable_Barata*, que se define en la ecuación (7)

$$SPP\_pieza\_Insertable\_Actual = \frac{\sum_{\forall i \in PM} d(i,C_a) + d(C_a,\ C_b)}{|PM|} \quad (7)$$

Atributos relativos al avance de las jugadas

Se entiende por jugada la operación de insertar una pieza en la tira. En relación con las jugadas existen atributos interesantes que permiten reflejar si estas corresponden al principio, al medio o al final de una partida. Generalmente, los logs reflejan que el jugador comienza con una estrategia al principio de una partida, inten-

tando realizar buenas jugadas (realizar encajes de piezas grandes a lo largo completo de la tira), mientras tanto, en las jugadas finales se intenta solo "terminar el nivel". Los dos atributos anteriores son:

1. **Número de piezas totales no insertadas** (*SPP_numero_Piezas_Totales*): Calcula la cantidad de piezas que no se han insertado aún. Si este atributo es grande, indica que está en el comienzo de la partida y que los movimientos son los primeros. Además, puede reflejar cuál es la proporción de piezas *PP* versus las que están insertadas.

2. **Número de piezas insertables** (*SPP_numero_Piezas_Insertables*): Calcula el número de piezas que se pueden insertar. Este valor se establece después de comenzar el nivel y no se modifica durante las inserciones.

Aplicación de minería de datos

En esta transformación se considera el algoritmo BLDA (*Bottom Left Decreasing Area*) como modelo para el comportamiento que tiene el ser humano.

La idea básica es reproducir cada una de las 4.121 jugadas, movimiento a movimiento, generando un ejemplo para el entrenamiento del árbol de decisión. En cada movimiento se calculan todos los atributos y se compara la decisión del jugador con la decisión que toman las heurísticas de inserción. Si la decisión coincide, entonces se crea un nuevo ejemplo para la clase. Si no coincide con ninguna heurística, entonces se genera un ejemplo asumiendo que la heurística utilizada por el jugador es la inserción aleatoria. Luego se aplica un formato, que consiste en ingresar los datos a un archivo de texto plano, ordenados por partidas y serie de jugadas (figura 3), para su ingreso como parámetros en la herramienta de aprendizaje automático KNIME, cuya salida es el árbol completo de las decisiones realizadas por los jugadores, con un conteo de las repeticiones que se ejecutaron, ya sea por el mismo o más jugadores, divididas por nivel de dificultad (figura 6).

Selección de atributos

Se realiza una comprobación de la calidad de los atributos. Dentro de la herramienta KNIME, se ejecuta una

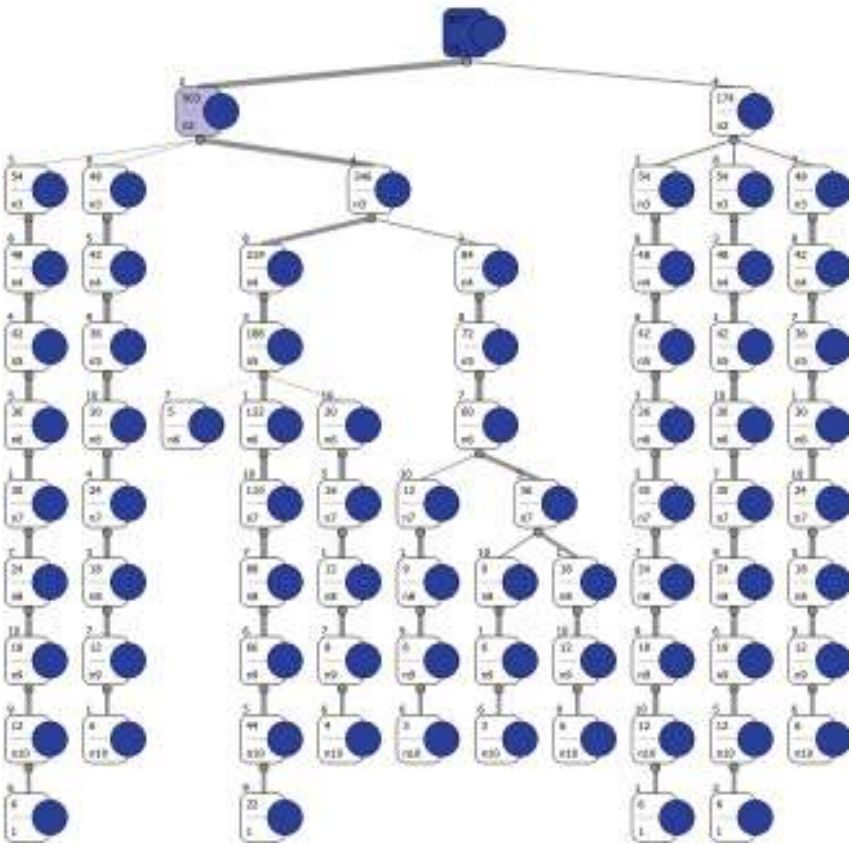

Figura 6. Árbol completo de las decisiones realizadas por los jugadores

Fuente: Elaboración propia

prueba de selección de atributos considerando los algoritmos ExhaustiveSearch (Nievergelt, 2000) y CfsSubsetEval (Pereira, 2007). Al final de este proceso, se obtiene la tabla 2 que refleja la relevancia de los atributos según los algoritmos mencionados. Los atributos que aparentemente guardan mayor relación con la clase son *SP_numero_Piezas_Totales* y *SPP_pieza_Insertable_Promedio*. Estos atributos no se pueden deducir a partir de la combinación de otros atributos. De forma intuitiva se puede observar que el atributo *SPP_numero_Piezas_Insertables* es totalmente imprescindible para caracterizar las heurísticas a seguir. Esto tiene sentido, dado que al analizar los registros, la estrategia del jugador cambia dependiendo si las jugadas corresponden al principio o al final de estas.

Otro atributo preponderante es *SPP_pieza_Insertable_Promedio*, el cual refleja el costo promedio de inserción de las piezas, es decir, las heurísticas a utilizar dependen de este costo. Finalmente, el tercer atributo imprescindible según este método es *SPP_pieza_Insertable_Actual*, el cual indica si las jugadas actuales son las primeras o las últimas en la lista de piezas. Esto también se corrobora empíricamente, ya que la estrategia cambia de acuerdo con qué sección de la lista de piezas se inserta actualmente. Sin embargo, existen atributos que al parecer son irrelevantes para caracterizar las decisiones del jugador. Este es el caso del atributo *SPP_numero_Piezas_Insertables*. Su eliminación indica que no es relevante el tamaño de la lista de piezas a la hora de seleccionar la estrategia a seguir.

Árbol de decisión

Se deben reproducir las 4121 jugadas, movimiento a movimiento, generando un ejemplo para el entrenamiento del árbol de decisión, en cada movimiento se calculan todos los atributos y se compara la decisión del jugador con la decisión de los atributos seleccionados. Si la decisión coincide, entonces se crea un nuevo ejemplo para la clase, en caso contrario, se crea un nuevo ejemplo asumiendo que la decisión es aleatoria. El árbol generado clasifica correctamente 87.03% de los casos las decisiones de los jugadores. A pesar que el árbol tenga más de 500 hojas, es necesario revisar en general los patrones obtenidos, donde se advierte que el nodo raíz hace referencia al atributo *SPP_Tamaño_Maximo*. El significado de este atributo se relaciona con la inserción de la pieza de mayor tamaño. La bifurcación de este atributo separa al árbol en las decisiones tomadas después de la segunda mitad de la lista de piezas. Esto puede parecer lógico dado que la lista de piezas, en un principio, se va insertando de mayor tamaño, para luego volver hacia las más pequeñas en las inserciones posteriores.

Resultados y discusión

El experimento se realizó con 20 jugadores en un rango etario de 22 y 34 años, que en sus actividades diarias deben realizar operaciones con características del *Strip Packing* Problem. En la tabla 4 se puede observar el número total de jugadas realizadas y el detalle de cada uno de los 5 niveles definidos para el experimento. El análisis de los resultados se realiza bajo la cantidad de jugadas y calidad de solución, dada por P, el valor de la mejor solución conocida, P^* el valor del área obtenida por el jugador y CR la calidad de respuesta obtenida por el jugador, luego $CR = 100\,[1 - (P^*/P - 1)]$, donde P^* es siempre mayor o igual a P.

Adicionalmente, en la tabla 3 se puede observar que ninguno de los jugadores logró obtener en el último nivel una calidad de jugada por sobre 90%, en cambio, en los niveles 1, 2, 3 y 4 se alcanzó 9.3, 9.16, 6.14 y 0.70% respectivamente, es decir, el descenso del porcentaje de efectividad tiene una directa relación con la complejidad y combinatoria del SPP. En el nivel 5, donde el óptimo tiene pérdida interna, los jugadores no lograron resolver el ni-

Tabla 2. Relevancia de atributos

Atributo	Porcentaje de relevancia
SSP_numero_Piezas_Totales	100%
SPP_pieza_Insertable_Promedio	100%
SPP_pieza_Insertable_Actual	100%
SPP_Tamaño_Promedio	30%
SPP_Tamaño_Maximo	30%
SPP_pieza_Insertable_Barata	30%
SPP_pieza_Insertable_cara	30%
SPP_Tamaño_Minimo	20%
Spp_numero_Piezas_Insertables	0%

Fuente: Elaboración propia

Tabla 3. Jugadas por Nivel

	Total jugadas	Jugadas sobre 90% de calidad	% de jugadas sobre 90% de calidad
Nivel 1	677	63	9.30 %
Nivel 2	1484	136	9.16 %
Nivel 3	976	60	6.14 %
Nivel 4	852	6	0.70 %
Nivel 5	132	0	0.00 %
Total	4121	265	6.43 %

Fuente: Elaboración propia

vel, principalmente porque el algoritmo utilizado buscó eliminar la pérdida interna, pero la mejor distribución conocida tiene una disposición diferente a los demás.

Patrones y procedimientos

El procedimiento utilizado por los jugadores se divide en dos etapas. La primera se compone por el proceso de construcción de la solución inicial, la cual se construye sin el conocimiento previo del problema, es decir, el jugador ha estructurado este proceso a través de todas las jugadas anteriores, como parte de su experiencia. La segunda, consiste en rescatar el posicionamiento de la jugada actual basado en dos criterios para seleccionar buenas configuraciones: a) este criterio se produce en el ordenamiento decreciente de área de las piezas, manteniéndose en la solución final; b) este criterio se produce en jugadas donde las piezas se ordenan según agrupamiento de piezas, formando figuras cuadradas y ordenadas en forma decreciente.

La solución más recurrente es formar áreas decrecientes insertadas de izquierda a derecha y de abajo hacia arriba, la cual es muy similar a la heurística *Bottom Left Decreacing Height* (Lodi, 2002). A nivel procedimental el algoritmo se describe en los 5 pasos siguientes:

1. Ordenar todas las piezas por tamaño de área en orden decreciente.
2. Insertar de izquierda a derecha y de abajo hacia arriba las piezas previamente ordenadas.
3. Cuando una pieza no es posible de insertar sin sobreponer, esta se gira y se prueba insertar, si no es posible se prueba con la pieza siguiente en el orden no creciente.
4. Si una pieza se omite porque no puede insertarse, y se colocó la siguiente, entonces se ordenan nuevamente en orden no creciente.
5. Al completar toda la fila, se comienza nuevamente de izquierda a derecha.

Algoritmo de solución obtenido

Mediante minería de datos y árboles de decisión se establece el algoritmo de jugadores para solucionar los problemas, este se denomina BLDA (*Bottom Left Decreasing Area*). La implementación del algoritmo se detalla en el Algoritmo 1. En el peor caso, en cada ingreso de una pieza se utiliza el espacio vertical. Por lo tanto, existen tantas iteraciones como piezas, lo cual conduce a una complejidad computacional de $O(n2)$.

Comparación de algoritmos

Para realizar un análisis comparativo con algoritmos existentes en la literatura, en la tabla 4 se presentan similitudes entre los distintos algoritmos, donde la primera fila ubica al autor, después la técnica utilizada para ordenar y la división de tira, respectivamente. Se destaca la similitud obtenida por BLDA con FFDH (*First Fit Decreasing Height*), al igual que la división de la tira es semejante a lo realizado por Wang (1983) para el corte guillotina de piezas regulares. Se observó semejanza en la forma de división de espacios con el algoritmo de Wang (1983), ya que realizan divisiones laterales y superiores, penalizando las divisiones superiores y priorizando las divisiones laterales. Una diferencia importante con Wang (1983), es no calcular el porcentaje de pérdida interna, por lo cual el algoritmo de jugadores no resuelve de manera adecuada diseños con espacio entre piezas.

En la tabla 5, se presentan pruebas del algoritmo con instancias de Burke (2003) e instancias generadas de manera aleatoria, en la primera y segunda columna, respectivamente. Se obtiene 94.3% promedio para las tres instancias de Burke (2003) y 79% para las generadas de manera aleatoria considerando pérdidas internas. Obteniendo 88.2% del óptimo para todas las instancias extras. Además, en la figura 7 se observa el diseño de la instancia B1 de Burke, donde en el sector izquierdo se detallan las cantidades y los tamaños de piezas dados por la forma [ancho] x [largo].

Algoritmo 1. Heurística BLDA generada

Algoritmo solución

```
1.  Entrada: W: Área de la tira, r: rectángulo, r_rotado = rectángulo rotado, r_saltado:
cantidad de rectángulos saltados, R: lista de n rectángulos, P: Árbol de solución
2.  Salida: H: altura
3.  Inicio
4.  Ordenar R por área no creciente
5.  bin = W;
6.  mientras cardinalidad de R != 0 hacer
7.      para cada rectángulos r en R hacer
8.          si r<bin, entonces
9.              insertar r en P
10.             insertar espacio_horizontal en P
11.             insertar espacio_vertical en P.
12.             bin = espacio_horizontal
13.             quitar r_i de R
14.             fin si
15.         si r>bin, entonces
16.             rotar e iniciar de nuevo
17.         fin si
18.         si r_rotado>bin, entonces
19.             r_saltado ++
20.             r++
21.         fin si
22.         si r_saltado == cantidad de r en R, entonces
23.             r_saltado = 0
24.             bin = espacio_vertical
25.         fin si
26.     fin para cada
27. fin mientras
28. Fin
```

Tabla 4. Comparación de algoritmos

Algoritmo	Ordenamiento	División de la tira
BLDA (este estudio)	Realiza un ordenamiento de áreas no crecientes	Genera una división en función de la pieza encajada, la subárea superior y la sub área lateral
FFDH (Lodi, 2002)	Realiza un ordenamiento de rotación de piezas para que todas tengan la altura máxima y luego realiza un ordenamiento de altura no creciente para insertar la primera pieza que encaje y luego reordena nuevamente	Genera una nueva división cada vez que una pieza no encaja en ningún nivel anterior
NFDH (Lodi, 2002)	Realiza un ordenamiento de rotación de piezas para que todas tengan la altura máxima y luego realiza un ordenamiento de altura no creciente; después de encajar la primera pieza, intenta con las restantes sin reordenar	Genera una nueva división cada vez que una pieza no encaja en el nivel actual
Wang (Wang, 1983)	No realiza ordenamiento Aunque es posible agregar la variación ya sea por área o por altura	Genera una división en función de la pieza encajada y su porcentaje de pérdida asociado, la sub área superior y la sub área lateral

Fuente: Elaboración propia

Tabla 5. Pruebas con otras instancias

Instancias de Burke (2003)		Generadas aleatorias	
Nombre	Calidad (%)	Nombre	Calidad (%)
B1	100	Aleatorio1	78
B2	100	Aleatorio2	80
B3	83		
Promedio	94.3	Promedio	79.0
Promedio final: 88.2 %			

Fuente: Elaboración propia

Figura 7. Solución de instancia B1

Fuente: Elaboración propia

Conclusiones

En este artículo se obtuvo un algoritmo a partir de las jugadas sobre un juego NP-Completo. Además, se verificó la validez del algoritmo y se probó con instancias nuevas, no presentes en la etapa de identificación de patrones de juego.

Los jugadores expertos entregan en cantidad y calidad jugadas suficientes para permitir identificar patrones. Encontrar estos patrones requiere de un análisis exhaustivo de las jugadas en un proceso de minado de datos. Una parte importante del algoritmo propuesto se extrae mediante árboles de decisión, una técnica utilizada en minado de datos con el objeto de representar el conocimiento o patrones.

El algoritmo obtenido valida la hipótesis de trabajo que afirma la posibilidad de identificar patrones de juego a partir de las jugadas. Aparte de verificar la hipótesis de trabajo se presentan pruebas que dan mayor soporte a esta afirmación: el algoritmo que representa los niveles de SPPG se ajusta 87.03% en promedio de los casos al comportamiento humano.

Como ejercicio adicional, se prueba el algoritmo sobre nuevas instancias, no resueltos por los jugadores, con la finalidad de demostrar que es un modelo general de resolución de *Strip Packing Problem*. La calidad promedio de la solución para problemas de 10 a 40 piezas alcanzó 88.2% de la mejor solución conocida.

Agradecimientos

Los autores agradecen el soporte de sus respectivas universidades. R.L. fue parcialmente financiado por el proyecto FONDECYT 11150370. G.G. fue parcialmente financiado por el grupo SEPRO de la Universidad Nacional de Colombia.

Referencias

Acuña D.E. y Parada V. People efficiently explore the solution space of the computationally intractable traveling salesman problem to find near-optimal tours. *PLoS ONE*, volumen 5 (número 7), 2010: 10.

Alvarez-Valdés R. y Parreño F. Reactive GRASP for the strip-packing problem. *Computer Operation Research*, volumen 4 (número 35), 2008: 1065-1083.

Burke E., Kendall G., Soubeiga E. A tabu-search hyperheuristic for timetabling and rostering. *Journal of Heuristics*, volumen 9 (número 6), 2003: 451-470.

Campitelli G. y Gobet F. Structure and stimulus familiarity: a study of memory in chess-players with functional magnetic resonance imaging. *The Spanish Journal of Psychology*, volumen 2 (número 8), 2005: 238-245.

De Jong K. y Schultz A.C. Using experience-based learning in game playing, en: Proceedings of the Fifth International Conference on Machine Learning, 1998, pp. 284-290.

Demaine E. Playing games with algorithms algorithmic combinatorial game theory, en: Proceedings of the 26th Symposium on Mathematical Foundations in Computer Science, Czech Republic: Lecture Notes in Computer Science, 2001, pp. 18-32.

Demaine E., Hohenberger S., Liben-Nowell D. Tetris is hard, even to approximate, en: Proceedings of the 9th annual international conference on Computing and combinatorics, Berlin: Springer-Verlag, 2002, pp. 351-363.

Epstein D. On the NP-completeness of cryptarithms. *ACM SIGACT News*, volumen 18 (número 3), 1987: 38-40.

Fraenkel A. Combinatorial game theory foundations applied to digraph kernels. *Electronic Journal of Combinatoric*, volumen 1, 1997: 13-42.

Garey M., Johnson D.S. *Computers and intractability*: *a guide to the theory of NP-completeness,* Nueva York, W.H. Freeman, 1979.

Gatica G., Villagran G., Contreras-Bolton C., Linfati R., Escobar J.W. A new genotype-phenotype genetic algorithm for the two-dimensional Strip Packing problem with rotation of 90° degrees. *Ingeniería y Universidad,* volumen 20 (número 1), 2016: 177-196.

Jaaksi A. A method for your object-oriented project. *Journal of Object-Oriented,* volumen 10, 1998: 17-25.

Klemettien M. A knowledge discovery methodology for telecommunication network. Proceedings of the twelfth international conference on data engineering, Finlandia, University of Helsinkin, 1999, pp. 115-122.

Lodi A., Martello S., Vigo D. Recent advances on two-dimensional bin packing problems. *Discrete Applied Mathematics*, volumen 1, 2, 3 (números 1-3), 2002: 379-396.

Nievergelt J. Exhaustive search, combinatorial optimization and enumeration: exploring the potential of raw computing power, en: Proceedings of the 27th Conference on current trends in theory and practice of informatics (SOFSEM ‹00), Springer-Verlag, Londres, 2000, pp. 18-35.

Pereira, J., Domínguez, M., Ocejo, J. Sáez. Modelos de Previsão do Fracasso Empresarial: Aspectos a considerar. *Revista de Estudos Politécnicos Polytechnical Studies Review,* volumen IV (número 7), 2007: 111-148.

Russell S. y Norvig P. *Artificial intelligence: A modern approach*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1995.

Stewart I. Minesweeper as an NP-complete problem. *ACM SIGCSE Bulletin*, (número 1), 2005: 39-40.

Sunnucks A. *Encyclopaedia of Chess, Nueva York,* Robert Hale Ltd, 1970.

Wang P.Y. Two algorithms for constrained two-dimensional cutting stock problem. *Computers & Industrial Engineering,* volumen 31, 1983: 109-115.

Witten I.H. y Frank E. *Data mining: practical machine learning tools and techniques*, Nueva York, Morgan Kaufmann, 2000.

Este artículo se cita:

Citación estilo Chicago

Gatica, Gustavo, Pablo Reyes, Carlos Contreras-Bolton, Rodrigo Linfati, John Willmer Escobar. Un algoritmo para el *Strip Packing Problem* obtenido mediante la extraccion de habilidades de expertos usando minería de datos. *Ingeniería Investigación y Tecnología*, XVII, 02 (2016): 179-190.

Citación estilo ISO 690

Gatica G., Reyes P., Contreras-Bolton C., Linfati R., Escobar J.W. Un algoritmo para el *Strip Packing Problem* obtenido mediante la extraccion de habilidades de expertos usando minería de datos. *Ingeniería Investigación y Tecnología*, volumen XVII (número 2), abril-junio 2016: 179-190.

Semblanzas de los autores

Gustavo Gatica. Doctor en ciencias de la ingeniería con mención automática por la Universidad de Santiago de Chile. Actualmente es profesor en las facultades de ingeniería de la Universidad de Andrés Bello y Santiago de Chile. Sus intereses de investigación consideran el diseño e implementación de algoritmos exactos y heurísticos para problemas de optimización combinatoria y desarrollo de aplicaciones computacionales para empresas chilenas.

Pablo Reyes. Ingeniero civil informático de la Universidad Andrés Bello. Actualmente se desempeña en la empresa privada en el área de las telecomunicaciones.

Carlos Contreras-Bolton. Magíster en ciencias de la ingeniería informática por la Universidad de Santiago de Chile. Actualmente es profesor en las facultades de ingeniería de la Universidad de Andrés Bello y Santiago de Chile. Sus intereses de investigación consideran el diseño e implementación de algoritmos exactos y heurísticos para problemas de optimización combinatoria.

Rodrigo Linfati. Doctor en automatización e investigación de operaciones por la Universidad de Bologna (Italia). Actualmente es profesor en el departamento de ingeniería industrial de la Universidad del Bío-Bío, Chile. Sus intereses de investigación incluyen el diseño e implementación de efectivos algoritmos exactos y heurísticos para problemas de optimización combinatoria y sus aplicaciones en problemas reales.

John Willmer Escobar. Doctor en investigación de operaciones por la Universidad de Bologna (Italia). Es docente de programas de pregrado y posgrado de la Pontificia Universidad Javeriana y Universidad del Valle. Sus intereses de investigación incluyen el diseño e implementación de efectivos algoritmos exactos y heurísticos para problemas de optimización combinatoria y sus aplicaciones en problemas reales.