Heuristics for the Score-Constrained Strip-Packing Problem

Asyl L. Hawa, Rhyd Lewis, and Jonathan M. Thompson

School of Mathematics, Cardiff University, Senghennydd Road, Cardiff, UK
`hawaa|lewisr9|thompsonjm1@cardiff.ac.uk`

Abstract. This paper investigates the Score-Constrained Strip-Packing Problem (SCSPP), a combinatorial optimisation problem that generalises the one-dimensional bin-packing problem. In the construction of cardboard boxes, rectangular items are packed onto strips to be scored by knives prior to being folded. The order and orientation of the items on the strips determine whether the knives are able to score the items correctly. Initially, we detail an exact polynomial-time algorithm for finding a feasible alignment of items on a single strip. We then integrate this algorithm with a packing heuristic to address the multi-strip problem and compare with two other greedy heuristics, discussing the circumstances in which each method is superior.

Keywords: Strip-Packing · Heuristics · Graphs and Networks

1 Introduction

The Constrained Ordering Problem (COP) is defined as follows:

Definition 1. *Let $\mathcal{M}$ be a multiset of unordered pairs of positive integers $\mathcal{M} = \{\{a_1, b_1\}, \{a_2, b_2\}, ..., \{a_n, b_n\}\}$, and let $\mathcal{T}$ be an ordering of the elements of $\mathcal{M}$ such that each element is a tuple. The* Constrained Ordering Problem (COP) *consists of finding a solution $\mathcal{T}$ such that, given a fixed value $\tau \in \mathbb{Z}^+$,*

$$\mathbf{rhs}(i) + \mathbf{lhs}(i+1) \geq \tau \qquad \forall\, i \in \{1, 2, ..., n-1\}, \tag{1}$$

where $\mathbf{lhs}(i)$ and $\mathbf{rhs}(i)$ denote the left- and right-hand values of the ith tuple. The inequality is referred to as the vicinal sum constraint*.*

For example, given $\mathcal{M} = \{\{1,2\},\{1,7\},\{2,4\},\{3,5\},\{3,6\},\{4,4\}\}$ and $\tau = 7$, one possible solution is $\mathcal{T} = \langle(1,2),(6,3),(5,3),(4,4),(4,2),(7,1)\rangle$.

A prominent application of the COP is in a strip-packing problem brought to light as an open-combinatorial problem by Goulimis [4]. Here, a set $\mathcal{I}$ of rectangular items (where $|\mathcal{I}| = n$) of equal height H made from cardboard are to be packed onto a strip of height H from left to right. Each item $i \in \mathcal{I}$ has width $w_i \in \mathbb{Z}^+$ and possesses two vertical score lines, marked in predetermined places. A pair of knives mounted onto a bar cuts along the score lines of two adjacent items simultaneously, which allows the items to be folded with ease (see

Figure 1). However, by design, the scoring knives cannot be placed too close to one another and, as such, have a "minimum scoring distance" (around 70mm in industry). The distances between each score line and the nearest edge on an item $i \in \mathcal{I}$ are the score widths, $a_i, b_i \in \mathbb{Z}^+$ where $a_i + b_i < w_i$, assigned such that $a_i \leq b_i$. Since these score widths are not necessarily equal, an item i can be positioned in one of two orientations: "regular", denoted (a_i, b_i), or "rotated", denoted (b_i, a_i), where the smaller of the two score widths a_i is on the left- and right- hand side respectively.

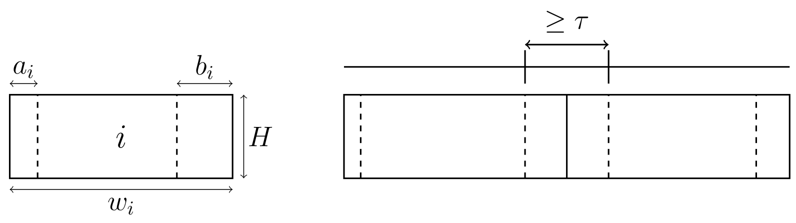

Fig. 1: Dimensions of an item $i \in \mathcal{I}$ in a regular orientation (a_i, b_i), and a feasible alignment of two items whose adjacent score lines can be scored simultaneously.

Clearly, for two items to be feasibly placed alongside one another on a strip, the distance between the two score lines must be equal to or exceed the minimum scoring distance, else the knives will not be able to score the items in the correct locations. Thus, the problem consists of finding a suitable ordering and orientation of the items such that the sum of every pair of adjacent score widths is greater than or equal to the minimum scoring distance.[1] It follows that there are $\frac{n!}{2}2^n$ distinct arrangements, making complete enumeration infeasible for non-trivial values of n.

It can be seen that, when using a single strip, this packing problem is equivalent to the COP, where each unordered pair in an instance $\mathcal{M}$ contains the score widths of an item and τ is the minimum scoring distance. It then follows that the vicinal sum constraint (1) corresponds to the requirement for the sum of adjacent score widths to exceed τ. Figure 2 shows the same instance $\mathcal{M}$ mentioned earlier depicted as a packing problem.

Observe that in this particular strip-packing problem the widths of the individual items are disregarded, since the aim is to arrange the items onto a single strip of seemingly infinite width. However, in industrial applications, strips of material will often be provided in fixed finite widths. Given a large problem instance, multiple strips may therefore be required to feasibly accommodate all of the items. For this reason, a more generalised problem can also be formulated as follows:

[1] Note that the left-hand score width of the first item and the right-hand score width of the last item on the strip are not adjacent to any other item, and can therefore be ignored.

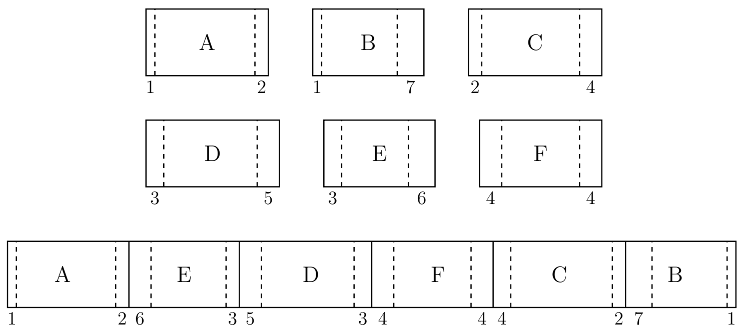

Fig. 2: Example of a single strip-packing problem and a corresponding solution with $\tau = 7$.

Definition 2. *Let $\mathcal{I}$ be a set of n rectangular items of height H with varying widths $w_i \in \mathbb{Z}^+$ and score widths $a_i, b_i \in \mathbb{Z}^+$ for each item $i \in \mathcal{I}$. Given a minimum scoring distance $\tau \in \mathbb{Z}^+$, the Score-Constrained Strip-Packing Problem (SCSPP) consists of finding the minimum number of strips of height H and width W required to pack all items in $\mathcal{I}$ such that the sum of every pair of adjacent score widths is greater than or equal to τ and no strip is overfilled.*

Note that in the special case of $\tau = 0$, the SCSPP is equivalent to the classical one-dimensional bin-packing problem (BPP). When $\tau > 0$, the problem also involves deciding the order in which the items are packed from left to right, and whether each item should be placed in a regular or rotated orientation. Thus, we define the following sub-problem associated with the SCSPP.

Definition 3. *Let $\mathcal{I}' \subseteq \mathcal{I}$ be a set of items whose total width is less than or equal to the capacity of a strip, (i.e. $\sum_{i\in\mathcal{I}'} w_i \leq W$). Given a minimum scoring distance τ, the Score-Constrained Packing Sub-Problem (SCPSP) involves finding an arrangement of the items in $\mathcal{I}'$ such that the sum of every pair of adjacent score widths is greater than or equal to τ.*

The remainder of this article is structured as follows: In Section 2, we will detail an exact polynomial-time algorithm for the COP and show how it is applicable to the SCPSP. Section 3 then introduces three heuristics that can be used to find feasible solutions to the SCSPP, one of which makes particular use of the exact algorithm from the previous section, and their associated advantages and disadvantages. An analysis of results gained from extensive experiments and a comparison of the heuristics is provided in Section 4, and finally Section 5 concludes the paper and proposes some potential directions for future work.

2 Solving the COP

We now present an exact polynomial-time algorithm for the COP. The underlying algorithm was originally proposed by Becker in [1], and is used to determine whether or not a solution exists for a given instance. Here, we extend this algorithm so that, if a solution does indeed exist, it is also able to formulate and present us with this final solution. This is especially useful for problems such as the strip-packing problem.

Let $\mathcal{M}$ be an instance of the COP of cardinality n. It is useful to model $\mathcal{M}$ as a graph G in which each vertex is associated with a single value in $\mathcal{M}$ in non-decreasing order. A pair of vertices, called "dominating vertices" are also added to the graph, both of which are assigned values equal to τ. These dominating vertices aid the solution process and are removed at the end. Thus, the graph G has $2n + 2$ vertices.

As seen in Definition 1, the values in $\mathcal{M}$ are arranged in pairs. This is represented in G by adding a set of "blue" edges, B, between vertices that are "partners", i.e. whose values make up a pair in $\mathcal{M}$. By introducing a bijective function $p : V \to V$ that associates each vertex with its partner, $p(v_i) = v_j$, we can denote this set of edges as $B = \{(v_i, p(v_i)) : v_i \in V\}$. Note that B is a perfect matching in G, with $|B| = n + 1$. Next, a set of "red" edges R is added to G that consists of edges between vertices whose sum equals or exceeds τ, provided they are not partners. It can be seen that the edges in R represent all possible orderings of values from different pairs in $\mathcal{M}$ that fulfil the vicinal sum constraint (1). Thus, we have a simple, undirected graph G with vertex set $V = \{v_1, ..., v_{2n+2}\}$ and two distinct edge sets B and R such that $B \cap R = \emptyset$. Figure 3 illustrates an example construction of G.

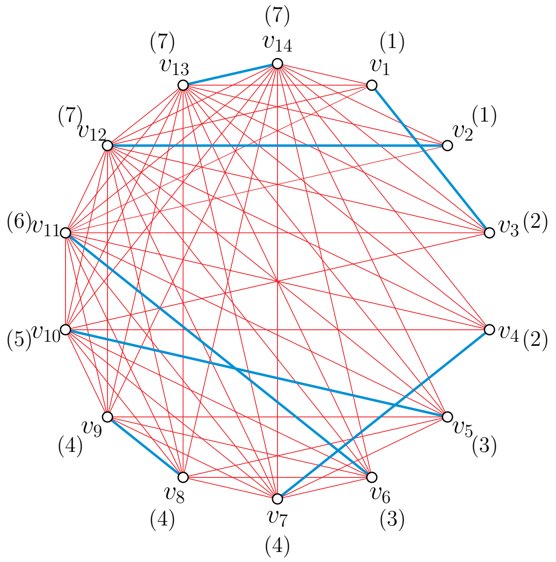

Fig. 3: $G = (V, B \cup R)$ using $\mathcal{M} = \{\{1, 2\}, \{1, 7\}, \{2, 4\}, \{3, 5\}, \{3, 6\}, \{4, 4\}\}$ and $\tau = 7$, where the thinner red edges are in R, and the thicker blue edges are in B. The corresponding values of each vertex are show in parentheses.

Recall that a Hamiltonian cycle in a graph G is a cycle that visits every vertex exactly once. Now, consider the following definition describing a variant of the Hamiltonian cycle problem involving two edge sets.

Definition 4. *Let $G = (V, B \cup R)$ be a simple, undirected graph where each edge is a member of one of two sets, B or R. G contains an alternating Hamiltonian cycle if there exists a Hamiltonian cycle such that successive edges alternate between sets B and R.*

It is clear that an alternating Hamiltonian cycle in G, if one exists, correpsonds to a feasible solution $\mathcal{T}$, as each "blue" edge from B represents a pair of values in $\mathcal{M}$, and each "red" edge from R indicates the values that meet the vicinal sum constraint, and thus can be placed alongside one another feasibly. Also, note that every edge in B must be present in the alternating Hamiltonian cycle. Consequently, the task can also be seen as finding a matching $R' \subseteq R$ of cardinality $n+1$ such that the edge sets B and R' form an alternating Hamiltonian cycle as described in Definition 4.

The problem of finding an alternating Hamiltonian cycle in general graphs is NP-hard as it generalises the classical Hamiltonian cycle problem [5]. However, the special structure of graphs derived from instances of the COP allows them to be solved in polynomial-time using the following method.

To find a suitable matching $R' \subseteq R$, a Maximum Cardinality Matching (MCM) algorithm is executed, which is based on [1] and also the earlier work of Mahadev and Peled [10]. First, take each vertex $v_1, v_2, ..., v_{2n+2}$ in turn and select the edge in R connecting v_i to the highest-indexed vertex v_j that is not already incident to an edge in R'. Now, add this edge (v_i, v_j) to R' and proceed to the next vertex until all the vertices have been assessed. The two vertices incident to each matching edge in R' are now referred to as being "matched". Similarly to partners, let $m : V \to V$ be a bijective function that assigns each vertex with its match, $m(v_i) = v_j$. Then, we can denote this matching set as $R' = \{(v_i, m(v_i)) : v_i \in V\}$.

During MCM, if a vertex v_i is not adjacent to any other unmatched vertex except its partner $p(v_i)$ via a blue edge, the preceeding vertex v_{i-1} can be "rematched", provided that (a) v_i is not the first vertex; (b) the previous vertex v_{i-1} has been matched; and (c) v_{i-1} and $p(v_i)$ are adjacent via a red edge in R. Then, v_i is matched with the vertex that is currently matched with v_{i-1}, and v_{i-1} is rematched with $p(v_i)$. Due to the initial order of the vertices, MCM is guaranteed to produce a maximum cardinality matching.

Clearly, if $|R'| < n+1$ after MCM has completed, there are an insufficient number of red edges to form an alternating Hamiltonian cycle, and therefore no feasible solution exists for the given instance $\mathcal{M}$. Otherwise, R' is a perfect matching of cardinality $n+1$, and the spanning subgraph $G' = (V, B \cup R')$ is a 2-regular graph where each vertex $v_i \in V$ is incident to one blue edge and one red edge, as can be seen in Figure 4. G' therefore consists of one or more cyclic components $C_1, C_2, ..., C_l$.

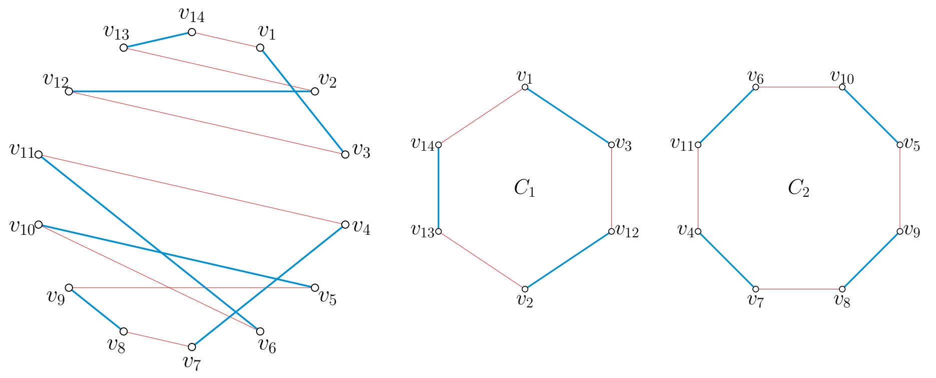

Fig. 4: Subgraph $G' = (V, B \cup R')$ created after MCM. When in planar form, it is clear that $l = 2$.

In the case where G' comprises one component, i.e. $l = 1$, then this alternating cycle is in fact Hamiltonian and therefore specifies a solution to the COP. However, if $l > 1$, the components of G' must be combined to form a single alternating Hamiltonian cycle. To do this, a Bridge Recognition (BR) procedure is executed that selects suitable edges from $R \backslash R'$ to replace edges in R' to connect the components of G'.

BR operates by first ordering the edges in R' such that the lower-indexed vertices of each edge are in increasing order and the higher-indexed vertices are in decreasing order. Any edges that cannot be placed in such an order are then removed from this list. For instance, in the example in Figure 4, the edges would be sorted as follows: $(v_1, v_{14}), (v_2, v_{13}), (v_3, v_{12}), (v_4, v_{11}), (v_6, v_{10}), (v_7, v_8)$. Note that, since v_5 was not matched with the highest-indexed vertex possible during MCM, the edge (v_5, v_9) does not adhere to the required structure of the list and is therefore omitted.

Starting from the first edge in the list, BR then searches through the list to find an edge that meets the following conditions: (a) the lower-indexed vertex of the current edge and the higher-indexed vertex of the next edge in the list are adjacent via an edge in R; and (b) the current edge and the next edge are members of different components of G'. If these conditions are met, BR adds the current edge to a set R_1, and continues to add all succeeding edges in the list to R_1 provided that, for each edge, both conditions hold and the succeeding edge is not a member of a component of G' that already has an edge in R_1. Once there are no more valid edges available to add to R_1, BR resumes its search through the remaining edges in the list to find another edge that meets the conditions, and can begin a new set R_2. The procedure terminates once the penultimate edge in the list has been assessed.

Now, one of the following three cases occurs:

1. In the event that BR has produced no sets, there are no suitable edges that can combine the components of G', and therefore an alternating Hamiltonian

cycle in G cannot be created. Consequently, no feasible solution exists for the corresponding COP.

2. If there exists a set R_i such that $|R_i| = l$, then all components of G' can be merged together to form a single alternating Hamiltonian cycle. This is achieved by adding the red edge from $R \backslash R'$ connecting the lower-indexed vertex of each edge in R_i to the higher-indexed vertex of the next edge to R' (for the final edge in R_i, add the red edge connecting its lower-indexed vertex to the higher-indexed vertex of the first edge). Edges that appear in both R_i and R' are then removed from R' so that R' remains a perfect matching and $R_i \cap R' = \emptyset$. G' then consists of a single alternating Hamiltonian cycle; hence a solution has been found.
3. It may be that multiple sets need to be used to connect the components of G'. For two edge sets R_i and R_j to "overlap", each set must have exactly one edge from the same component in G', with the other edges in each set being from different components. A collection of sets $\mathcal{R}^*$ therefore needs to be found such that each set in $\mathcal{R}^*$ overlaps with at least one other set, and each component of G' has at least one edge in one of the sets.

In the final case above, a Modified Bridge Recognition (MBR) algorithm is used to find suitable overlapping sets. Firstly, a copy of the set R_i with the highest cardinality[2] generated by BR is created, called R_1^*, and added to $\mathcal{R}^*$. Then, MBR takes the sorted list used in BR and removes edges from the list that are in $\mathcal{R}^*$. Similarly to BR, MBR proceeds through the list to find an edge that meets the following conditions: (a) the lower-indexed vertex of the current edge and the higher-indexed vertex of the next edge in the list are adjacent via an edge in R; and (b) the current edge and an edge in $\mathcal{R}^*$ are members of the same component of G' and the next edge is a member of a component that does not have an edge in $\mathcal{R}^*$, or vice versa. If both conditions hold, the current edge is added to a new set R_2^* which is then added to $\mathcal{R}^*$. MBR continues to add succeeding edges to R_2^*, provided (a) holds and the succeeding edge is a member of a component that does not have an edge in $\mathcal{R}^*$. Then, if every component of G' has an edge in one of the sets in $\mathcal{R}^*$, these sets are able to connect all the components of G' together, and so MBR terminates. Otherwise, the edges in R_2^* are removed from the sorted list, and MBR repeats the search for suitable edges to start a new set R_3^*. This procedure continues until either $\mathcal{R}^*$ contains overlapping sets that cover all components of G', or until there are no more suitable edges in the list to start a new set. If MBR has produced a feasible collection of sets then the components of G' can be merged to create an alternating Hamiltonian cycle by applying the connecting procedure above to every set in $\mathcal{R}^*$.

Using the instance illustrated in Figure 5 as an example, the edges (v_3, v_{12}) and (v_4, v_{11}) are in the set R_1 formed by BR, as (a) $(v_3, v_{11}) \in R$, i.e. the lower-indexed vertex of the first edge is adjacent to the higher-indexed vertex of the next edge; and (b) the edges are members of different components ($(v_3, v_{12}) \in C_1$ and $(v_4, v_{11}) \in C_2$). Note that since v_4 and v_{11} are adjacent, v_4 must also be adjacent to v_{12}, i.e. $(v_4, v_{12}) \in R$, since the value associated with v_{12} is greater

[2] In the event of a tie, MBR selects the set with the lowest index.

than or equal to the value associated with v_{11}. Then, because $|R_1| = l$, the edges (v_3, v_{12}) and (v_4, v_{11}) are removed from R' and replaced by the edges (v_3, v_{11}) and (v_4, v_{12}) from R_1. Removing the dominating vertices and any incident edges results in an alternating Hamiltonian path, which corresponds to a feasible solution $\mathcal{T}$.

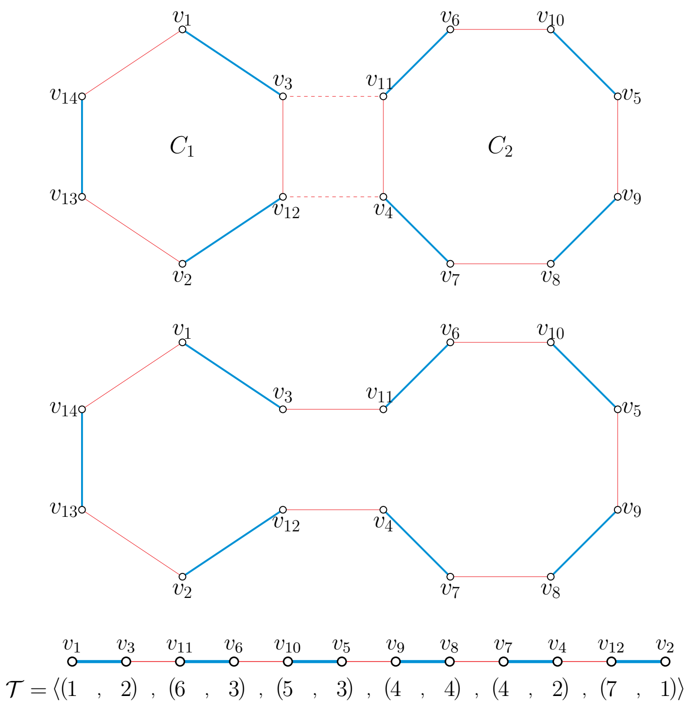
v_1
v_6
v_{10}
v_{14}
v_3
v_{11}
v_5
C_1
C_2
v_{13}
v_{12}
v_4
v_9
v_2
v_7
v_8
v_1 v_3 v_{11} v_6 v_{10} v_5 v_9 v_8 v_7 v_4 v_{12} v_2
$\mathcal{T} = \langle(1, 2), (6, 3), (5, 3), (4, 4), (4, 2), (7, 1)\rangle$

Fig. 5: Specific edges found using BR merge the components of G' together, forming an alternating Hamiltonian cycle which corresponds to a solution.

This entire algorithm that has been described will be referred to as the Alternating Hamiltonian Construction (AHC) algorithm. The time complexity of AHC is quadratic, as stated in the following theorem:

Theorem 1. *Let $G = (V, B \cup R)$ be a graph created from an instance $\mathcal{M}$ of cardinality n of the COP. Then, AHC terminates in at most $O(n^2)$ time.*

Proof. We assess each subprocedure of AHC in turn. Firstly, MCM produces a matching set $R' \subseteq R$ in at most $O(n \lg n)$ time due to the sorting of the vertices in lexicographical order. In BR, sorting the $n + 1$ edges of R' and removing unsuitable edges also requires $O(n \lg n)$ time. The sets R_i can be created in $O(n)$ time, as each edge in the list is considered once. As each set R_i contains at least two edges, BR can produce up to $\frac{n+1}{2}$ sets. Thus, examining each set to find one such that $|R_i| = l$ takes at most $O(n)$ time. The same method is also used to find the set with the highest cardinality in MBR. Since G' comprises a maximum of $\frac{n+1}{2}$ components, it follows that the number of edge sets in $\mathcal{R}^*$ needed to connect all the components is bounded by $\frac{n+1}{2} - 1$. The initial sorted list consists of at most $n-1$ edges, and therefore MBR is of quadratic complexity $O(n^2)$. Finally, the connecting procedure replaces up to $n+1$ edges, and so can be executed in $O(n)$ time. Consequently, AHC has an overall worst case complexity of $O(n^2)$. $\square$

3 Heuristics for the SCSPP

In this section, we now turn our attention to the multi-strip version of the problem. As mentioned in the introduction, the SCSPP is a generalisation of the one-dimensional BPP in that we also require the sum of every pair of adjacent score widths to be greater than or equal to a minimum scoring distance τ. It follows that the SCSPP is at least as hard as the BPP, which is known to be NP-hard [3], and so (under the assumption that $P \neq NP$) there is no known algorithm that is able to find an optimal solution for every instance of the SCSPP in polynomial time. Instead, heuristics can be used to find near-optimal solutions in a shorter amount of time.

For an instance of the SCSPP, a feasible solution is represented by the set $\mathcal{S} = \{S_1, S_2, ..., S_k\}$ such that

$$\bigcup\nolimits_{i=1}^{|\mathcal{S}|} S_i = \mathcal{I}, \tag{2a}$$

$$S_i \cap S_j = \emptyset \quad \forall\, i,j \in \{1,2,...,|\mathcal{S}|\},\ \ i \neq j, \tag{2b}$$

$$\sum\nolimits_{i=1}^{|S_j|} w_i \leq W \quad \forall\, S_j \in \mathcal{S}, \tag{2c}$$

$$\mathbf{rhs}(i) + \mathbf{lhs}(i+1) \geq \tau \quad \forall\, i \in \{1,2,...,|S_j|-1\},\ \ \forall\, S_j \in \mathcal{S}. \tag{2d}$$

That is, all items must be packed onto a strip (2a), each item can only be placed on one strip (2b), the strips cannot be overfilled (2c), and the items on each strip S_j in the solution must be arranged such that the vicinal sum constraint is fulfilled (2d). Note that constraints (2a)–(2c) are the necessary conditions for the BPP. An optimal solution $\mathcal{S}$ for the SCSPP is a solution that consists of the fewest number of strips k needed to feasibly contain the n items in the given problem instance. A simple lower bound for k is the theoretical minimum $t = \lceil \sum_{i=1}^{n} w_i/W \rceil$ which can be computed in $O(n)$ time [12].

Perhaps the simplest and most well-known heuristic for one-dimensional bin packing problems is First-Fit (FF), a greedy online algorithm that places each item, presented in some arbitrary order, onto the lowest-indexed strip such that the capacity of the strip is not exceeded. It is known that there always exists at least one ordering of the items such that FF produces an optimal solution [8]. A minor modification to FF yields the First-Fit Decreasing (FFD) heuristic, which sorts the items in non-increasing order of size prior to performing FF. In 1973, Johnson [7] showed that FFD is guaranteed to return a solution that uses no more than $\frac{11}{9}k + 4$ strips. More recently, Dosa [2] has proven that the worst case for FFD is in fact $\frac{11}{9}k + \frac{6}{9}$, and that this bound is tight. Due to the initial sorting of the items in non-increasing order of sizes, the time complexity of FFD is $O(n \log n)$.

As mentioned in the introduction, the SCSPP shares many similarities with the BPP, however the addition of (2d) brings complications. One obvious difference is the order in which the items appear on the strips. The order of the items in the BPP is unimportant; however in the SCSPP the items must be ordered in a way that meets the vicinal sum constraint. In addition, removing an item from a bin in the BPP retains the feasibility of the bin, whereas in the SCSPP this is not guaranteed, as it may leave a subset of items for which the vicinal sum constraint is not satisfied. Furthermore, the theoretical minimum t has the potential to be less accurate for the SCSPP, as the minimum scoring distance τ is not considered. For example, if the minimum scoring distance is greater than twice the largest score width, then it is clear that n strips will be required, regardless of the items' widths.

To gain an understanding of this problem, three heuristics for the SCSPP have been developed: a basic FFD heuristic with a simple modification; a heuristic that packs strips individually and prioritises score widths; and a more advanced version of FFD that incorporates the polynomial-time AHC algorithm.

The first heuristic is the Modified First-Fit Decreasing (MFFD) heuristic which acts in the same manner as the original FFD, attempting to place each item onto the end of the lowest-indexed strip. If an item is able to be packed onto a strip without exceeding the strip's capacity, MFFD then checks to see if the vicinal sum constraint is met between the right-most score on the strip and one of the score widths on the current item. If the constraint is met, MFFD places the item on the end of the strip in the appropriate orientation, otherwise the next strip is considered. The most prominent issue with this heuristic is due to the items being placed on the end of the strips. Although an item could potentially be packed in a different location on the strip other than the end, it may end up being placed on another strip, or perhaps even begin a new strip, thus increasing the number of strips in the final solution.

The next heuristic is the Pair-Smallest (PS) heuristic, which is an extension of an inexact procedure defined by Lewis et al. [9]. Unlike MFFD, which packs each item in turn, PS focusses on packing one strip at a time, only starting a new strip once the current strip is unable to accommodate any further items. Each strip is initialised by choosing the item from $\mathcal{I}$ with the smallest score width, and packing it in a regular orientation. PS then continues to fill the strip by selecting the item with the smallest score width that meets the vicinal sum constraint with the right-most score width on the strip, and whose width will not cause the strip to be overfilled. This heuristic aligns the smallest score widths with the largest ones, eliminating the possibility of placing larger score widths together unecessarily. Note that PS therefore prioritises the vicinal sum constraint over the item widths, choosing to fulfil this constraint first before considering whether the item can actually be accommodated. Consequently, there is no use for a procedure such as AHC to find a feasible arrangement of the items.

The last heuristic, Modified First-Fit Decreasing with AHC (MFFD$^+$), incorporates the AHC algorithm from Section 2. It operates in a similar fashion to the MFFD, placing items sorted in decreasing order onto the lowest-indexed strip. However, rather than attempting to place the item onto the right-most side of the strip, MFFD$^+$ executes AHC on all items on the strip. If AHC finds a feasible solution, the items are placed on the strip in the order of the solution, which includes the current item, else MFFD$^+$ attempts to pack the current item on the next strip. Using AHC means that if a feasible alignment of the items exists, there is a guarantee that it will be found. Unlike MFFD, where the current item can only be placed on the end of the strip, MFFD$^+$ allows the items to be entirely rearranged (see Figure 6). This reduces the possibility of having to start a new strip for an item, thus preventing increasing the number of strips in the final solution.

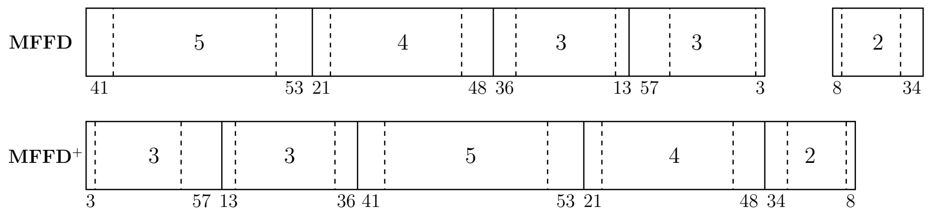

Fig. 6: Example instance of the sub-problem, with $W = 20$ and $\tau = 70$. In MFFD the constraint is not fulfilled in either orientation, however MFFD$^+$ is able find a feasible arrangement.

4 Experimental Results

Benchmark instances currently do not exist for the SCSPP, and so artificial problem instances were produced to compare the performance of the three heuristics. For our experiments, we generated 1000 problem instances for $|\mathcal{I}| \in \{500, 1000\}$. In each problem instance the items have varying widths $w_i \in [150, 1000]$ and score widths $a_i, b_i \in [1, 70]$ to ensure that each item has exactly two score lines, all selected randomly from a uniform distribution. Strips of widths $W = 5000$, 2500 and 1250 were used to influence the number of items per strip. As the width of the strips decrease, the average number of items per strip also decreases, making the problem more constrained. Both the items and the strips have equal height of $H = 1$. Similarly to experiments performed in [9], we also introduced a parameter δ to denote the proportion of pairs of score widths from different items that meet the vicinal sum constraint, i.e. whose sum is greater than or equal to τ. Values of δ from 0.0 to 1.0 were created by changing the value of τ. Clearly, when $\delta = 0.0$, there are no items that can be packed together feasibly, and so n strips will be required (one for each item), whereas if $\delta = 1.0$ all pairs of score widths meet the constraint, and the problem is equivalent to the BPP.

The heuristics were implemented in C++ and executed on a computer with an Intel Core i3-2120 3.30GHz processor. Our source code and all data is provided at [6]. Since optimal solutions are not available, in our case solution quality q is estimated by comparing each solution to the theoretical minimum, $q = |\mathcal{S}|/t$. For each heuristic, we calculated the average solution quality for every combination of n, δ, and W from 1000 instances. All individual trials were seen to complete in under 160ms.

Table 1: Average solution quality q for $n = 500$.

	$W = 5000$, $t = 58.039$			$W = 2500$, $t = 115.571$			$W = 1250$, $t = 230.648$		
δ	MFFD	PS	MFFD$^+$	MFFD	PS	MFFD$^+$	MFFD	PS	MFFD$^+$
0.0	8.618	8.618	8.618	4.328	4.328	4.327	2.169	2.169	2.169
0.1	5.214	**4.842**	5.161	2.659	**2.477**	2.657	1.515	**1.479**	1.515
0.2	4.031	**3.459**	3.976	2.121	**1.847**	2.118	1.326	**1.318**	1.326
0.3	3.111	**2.348**	3.038	1.730	**1.397**	1.708	1.195	1.229	**1.194**
0.4	2.436	**1.529**	2.297	1.460	**1.128**	1.410	1.110	1.181	**1.108**
0.5	1.911	**1.041**	1.691	1.263	**1.033**	1.196	1.058	1.154	**1.053**
0.6	1.491	**1.013**	1.246	1.124	**1.030**	1.069	1.029	1.135	**1.024**
0.7	1.196	**1.013**	1.045	1.049	1.028	**1.019**	1.017	1.114	**1.014**
0.8	1.050	1.012	**1.008**	1.016	1.027	**1.008**	1.013	1.091	**1.012**
0.9	1.009	1.012	**1.005**	1.007	1.026	**1.006**	1.012	1.073	**1.011**
1.0	**1.004**	1.012	**1.004**	**1.006**	1.026	**1.006**	**1.011**	1.065	**1.011**

Table 2: Average solution quality q for $n = 1000$.

	$W = 5000$, $t = 115.534$			$W = 2500$, $t = 230.563$			$W = 1250$, $t = 460.623$		
δ	MFFD	PS	MFFD$^+$	MFFD	PS	MFFD$^+$	MFFD	PS	MFFD$^+$
0.0	8.657	8.657	8.657	4.338	4.338	4.338	2.171	2.171	2.171
0.1	5.173	**4.842**	5.140	2.636	**2.481**	2.643	1.511	**1.467**	1.511
0.2	3.976	**3.462**	3.952	2.093	**1.858**	2.102	1.318	**1.311**	1.319
0.3	3.047	**2.350**	3.013	1.698	**1.409**	1.688	**1.183**	1.221	**1.183**
0.4	2.374	**1.520**	2.266	1.426	**1.131**	1.384	1.097	1.171	**1.096**
0.5	1.847	**1.026**	1.642	1.230	**1.030**	1.170	1.044	1.144	**1.040**
0.6	1.433	**1.012**	1.203	1.099	**1.027**	1.050	1.020	1.124	**1.015**
0.7	1.155	**1.012**	1.026	1.034	1.026	**1.012**	1.012	1.102	**1.009**
0.8	1.032	1.012	**1.006**	1.011	1.025	**1.007**	1.009	1.078	**1.008**
0.9	1.007	1.012	**1.004**	1.007	1.024	**1.006**	**1.008**	1.059	**1.008**
1.0	**1.004**	1.012	**1.004**	**1.006**	1.024	**1.006**	**1.008**	1.054	**1.008**

Tables 1 and 2 compare the results obtained from the three heuristics using the different values of δ and W, for $n = 500$ and 1000 respectively. Figures in bold indicate the best average solution quality for the given combination of parameters. We see that q tends towards 1 as δ increases since the proportion of score widths meeting the vicinal sum constraint increases, consequently permissing more items to be packed on each individual strip[3], thus reducing the number of strips required. Note that when $\delta = 1.0$ MFFD and MFFD$^+$ have identical solution qualities, as the instances are equivalent to the original BPP ($\tau = 0$); hence they operate in the same fashion as the original FFD heuristic.

Looking at results for both $n = 500$ and 1000, we see there is a clear pattern with respect to q across all widths and proportion levels. PS has the best solution quality for a wider range of δ when the strips are wider, and a much smaller range when the strip width decreases. Conversely, MFFD$^+$ obtains better solution qualities for a wider range of δ when $W = 1250$. Although PS does have the best solution quality of the three heuristics for $\delta = 0.1$ and 0.2 using the smallest strip width, we can see that it only marginally superior to MFFD$^+$. For example, take $\delta = 0.2$ in Table 1 for $W = 1250$. The difference between q for PS and MFFD$^+$ is 0.008, which translates to fewer than 2 strips difference between the average number of strips generated by each heuristic. In this particular instance, PS and MFFD$^+$ produced 303 and 305 strips on average, respectively.

Although using the average solution quality from 1000 instances provides a useful overview of the efficiency of a heuristic, there are other characteristics that we can consider. Take, for example, the results obtained with parameters $\delta = 0.7$ and $W = 5000$ in Table 2. Clearly PS obtains solutions with the fewest strips on average, however, we noted that out of the 1000 instances, PS did not produce a single solution containing t strips. On the other hand, there were 152 instances in which MFFD$^+$ was able to generate a solution S such that $|\mathcal{S}| = t$, thus implying that there are at least 152 instances that can be solved to optimality. Despite this, MFFD$^+$ has a higher average solution quality than PS, suggesting that the variance in the number of strips required is higher.

5 Conclusions and Further Work

This paper has investigated the Score-Constrained Strip-Packing Problem (SCSPP), a packing problem in which the order and orientation of the items is crucial to the feasibility of the solution. We begun by introducing the Constrained Ordering Problem (COP), and described the Alternating Hamiltonian Construction (AHC) algorithm, an exact polynomial-time algorithm that operates by modelling the problem graphically and using the concept of Hamiltonian cycles. We then showed how AHC can be used to find a solution for the Score-Constrained Packing Sub-Problem (SCPSP), the single-strip version of the SCSPP. Thus, the main problem was to tackle the multi-strip problem. Three heuristics, one

[3] The average number of items per strip when $n = 500$ for $W = 5000$, 2500 and 1250 are 8.475, 4.310, and 2.165 respectively, and the average number of items per strip when $n = 1000$ for $W = 5000$, 2500 and 1250 are 8.621, 4.329, and 2.169 respectively.

of which included the exact AHC algorithm, were described, and thorough experiments using a variety of parameters were executed.

A potential avenue for further research would be to produce an evolutionary algorithm (EA) for the SCSPP which incorporates AHC during local search. One possible addition would be to introduce an approach similar to one used in [11]. During each iteration of the evolutionary algorithm, high quality strips are chosen from each offspring solution and stored in separate set. On completion of the EA, a postoptimisation procedure based on the exact cover problem is executed. The procedure aims to find the fewest number of strips from the set of high quality strips that contains every item exactly once.

References

1. Becker, K.H.: Twin-Constrained Hamiltonian Paths on Threshold Graphs - An Approach to the Minimum Score Separation Problem. PhD Thesis, London School of Economics (2010)
2. Dósa, G.: The tight bound of first fit decreasing bin-packing algorithm is $FFD(I) \leq 11/9OPT(I) + 6/9$. Combinatorics, Algorithms, Probabilistic and Experimental Methodologies pp. 1–11 (2007)
3. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. WH Freeman Co., San Francisco pp. 90–91 (1979)
4. Goulimis, C.: Minimum Score Separation - an open combinatorial problem associated with the cutting stock problem. Journal of the Operational Research Society **55**(12), 1367–1368 (2004)
5. Häggkvist, R.: On F-Hamiltonian graphs. University of Umeå, Department of Mathematics (1977)
6. Hawa, A.L., Lewis, R., Thompson, J.: Source code for algorithms in the article "Heuristics for the Score-Constrained Strip-Packing Problem" (2018). https://doi.org/10.5281/zenodo.1311857
7. Johnson, D.S.: Near-optimal bin packing algorithms. Ph.D. thesis, Massachusetts Institute of Technology (1973)
8. Lewis, R.: A general-purpose hill-climbing method for order independent minimum grouping problems: A case study in graph colouring and bin packing. Computers & Operations Research **36**(7), 2295–2310 (2009)
9. Lewis, R., Song, X., Dowsland, K., Thompson, J.: An investigation into two bin packing problems with ordering and orientation implications. European Journal of Operational Research **213**(1), 52–65 (2011)
10. Mahadev, N.V., Peled, U.N.: Longest cycles in threshold graphs. Discrete Mathematics **135**(1-3), 169–176 (1994)
11. Malaguti, E., Monaci, M., Toth, P.: A metaheuristic approach for the vertex coloring problem. INFORMS Journal on Computing **20**(2), 302–316 (2008)
12. Martello, S., Toth, P.: Lower bounds and reduction procedures for the bin packing problem. Discrete applied mathematics **28**(1), 59–70 (1990)