GPU-Based Computing for Nesting Problems: The Importance of Sequences in Static Selection Approaches

Pedro Rocha[1], Rui Rodrigues[2], A. Miguel Gomes[2], and Cláudio Alves[3]

[1] INESC TEC, Porto, Portugal
[2] INESC TEC and Faculdade de Engenharia, Universidade do Porto, , Porto, Portugal
[3] Escola de Engenharia, Universidade do Minho, Braga, Portugal

Abstract. In this paper, we address the irregular strip packing problem (or nesting problem) where irregular shapes have to be placed on strips representing a piece of material whose width is constant and length is virtually unlimited. We explore a constructive heuristic that relies on the use of graphical processing units to accelerate the computation of different geometrical operations. The heuristic relies on static selection processes, which assume that a sequence of pieces to be placed is defined *a priori*. Here, the emphasis is put on the analysis of the impact of these sequences on the global performance of the solution algorithm. Computational results on benchmark datasets are provided to support this analysis, and guide the selection of the most promising methods to generate these sequences.

1 Introduction

Given their practical and theoretical relevance, cutting and packing problems have deserved the attention of both operations research and computer science practitioners [8]. The general problem consists in finding the best way to place a set of items (pieces) on a larger object (board). The feasibility of a solution may be subject to different constraints. In the simplest form of the cutting and packing problem, both the pieces and the board are 1-dimensional objects and the only constraint that applies is related to the size of the board, which should not be exceeded. On higher dimensional problems, other constraints become more critical such as the non-overlapping of the pieces and the fact that pieces must be placed inside the board (although these constraints also arise in 1-dimensional settings, they are treated very easily in these cases). Furthermore, in real settings, many different specific operational constraints may be considered, such as the existence of conflicts between the pieces or the limitation of the rotations applied to the pieces. One interesting aspect is that they may also arise as subproblems of other integrated optimization problems, as for example in transportation problems where the capacity of the vehicles are strong constraints.

In this paper, we address the so-called *nesting problem* or irregular strip packing. The problem is a 2-dimensional packing problem involving pieces whose contour may be irregular. Here, irregularity contrasts with many other cases studied in the literature where only specific convex shapes are considered (squares, rectangles, circles). In our case, the shapes may have concavities, and they are not restricted to any particular family of polygons. The board is a rectangular strip representing a piece of material whose interior is homogeneous, has fixed width, and virtually unlimited length. Given these definitions, the optimization problem consists in finding the positions where the pieces should be placed such that they are completely inside the board, do not overlap, and the total length of the used strip is minimized. The problem is clearly NP-hard [4], and, in practice, it remains challenging even for small datasets due to its combinatorial and geometric nature. Different types of approaches have been described in the literature [2, 3, 5]. Here, we explore the potential of graphical processing units (GPU) to accelerate the computation of some geometrical operations, and we study the importance of good sequences of pieces in approaches based on greedy constructive heuristics. To the best of our knowledge, this is the first time that GPU are used to evaluate the quality of layouts from a representation of the solutions based on sequences of pieces. The objective of our study is mainly experimental. It aims to provide insights towards the development of efficient approaches from similar platforms.

The paper is organized as follows. The second section presents the basic definitions and concepts related to the nesting problem. Third section describes the solution framework adopted in our study, and discusses the most promising sequencing rules that should be used in pure greedy constructive heuristics. Further experiments and results are reported and discussed in the fourth section. In the last section, we draw some final conclusions.

2 The Nesting Problem and Underlying Concepts

The nesting problem addressed in this paper consists in finding the best layout for a set of 2-dimensional pieces on a board so that no pieces overlap and all of them are placed inside the board. The pieces have irregular contours that may potentially include concavities. The board is a strip with a fixed width and a length virtually unlimited. The quality of a layout is measured as the length of the used strip, or equivalently, as the usage of the area occupied up to the length it reaches on the strip. Due to technological constraints and physical properties of the raw materials, pieces are only allowed to be placed under a discrete set of orientations. Furthermore, the pieces may be placed in any unused position of the strip given that the internal part of both the pieces and the strip are homogeneous.

The geometrical representation of the pieces has a significant impact on the efficiency of the solution approaches. Pieces can be represented by defining their vertices as sequences of points. Although this is a simple representation, it requires complex trigonometric operations to compute the relative position of the shapes. An alternative is based on a raster representation (grid) where shapes are represented through a matrix of values (pixels). This representation can be used to check easily for overlaps, in particular when it is combined with the use of no-fit polygons (NFP). However, the accuracy of the representation depends on the unit size of the grid, and it increases, as the unit size gets smaller. Similarly, the computational burden increases significantly with smaller unit sizes, due to the large number of total grid units used. The raster representation is also more adequate for pieces with orthogonal edges, since non-orthogonal edges are only approximated.

The placement of the pieces is usually tackled with the assistance of NFP and inner-fit polygons (IFP) [1]. The NFP between two pieces can be described as a set of points that define the relative position between two polygons (whether they are overlapping or touching). Its main advantage is the simplification of the overlap verification process, but it is only efficient for discrete orientations since they can be computed offline in a pre-processing stage. The IFP is similar to the NFP, but it is used to ensure that a polygon is placed inside another.

Given the computational burden involved in the geometric operations inherent to the nesting problem, a promising approach has been to consider the use of dedicated hardware to support this computation, and in particular, the GPU [6]. The GPU is able to execute operations on multiple pixels simultaneously, which can be explored to improve the efficiency of the approaches when compared to the use of a normal CPU. The rasterization capabilities of the GPU enable producing feasible layouts considering the raster representation of both the pieces and the board. Using a higher representation quality also requires an increase in the total number of pixels, leading to a higher computational cost.

3 Solution Framework

In this paper, we explore and analyze a GPU-based greedy heuristic for the nesting problem. To take full advantage of the GPU, we used a raster representation to define the pieces and layout. The heuristic relies on a left-bottom placement rule that places the pieces iteratively in the layout according to a predefined sequence. Given the importance of these sequences on the quality of the layouts, we study the influence of these sequences on the performance of the heuristic.

Sofia *et al.* [7] present details about the GPU-based placement heuristic used in this work (GPU-Nest). Only a brief description of this heuristic is presented here. Given a predefined sequence of pieces, the heuristic places the pieces iteratively using a left-bottom placement rule. At each iteration, the placement rule is used to determine the most left-bottom feasible placement point of the next piece for all admissible orientations. Among all the alternatives, the heuristic selects the one that corresponds to the leftmost and bottommost position. In our implementation, this is achieved by keeping buffers with the admissible placement positions (pixels), one buffer for each piece type and orientation. Each time one piece is placed in the layout, the buffers are updated by drawing one NFP in each one (the NFP between the placed piece and the buffer piece type and orientation). This process has a drawback since it is not able to detect perfect fit situations due to the rasterization. To overcome this issue, the pieces are placed with a gap of 1 pixel between them. The heuristic was completely implemented in the GPU, which makes it computationally efficient. The NFPs and IFPs of all the pieces are generated in a pre-processing phase, and copied to the GPU memory.

The main control parameters are the grid resolution (which impacts on the approximation quality of the pieces), and the empty space left between pieces (in order to allow placement of pieces in small holes). An increase in the grid resolution leads to better approximations, which may reduce the layout length and the empty spaces between the pieces, while it also increases the computational cost.

Sofia *et al* [7] explored several greedy criteria to create static sequences of pieces. The static criteria implemented were the following: random (pieces are sorted randomly), height (pieces are ordered by height, taller first), width (pieces are ordered by width, wider first), irregularity (pieces are ordered by irregularity, most irregular first), rectangularity (pieces are ordered by rectangularity, less rectangular first), and size (pieces are ordered by area, larger first). The results clearly showed a significant influence of the criteria on the layout quality, with the best results being obtained by the size criteria. Several other authors obtained similar results. One of the main reasons for this conclusion is due to the fact that the layout length is mainly determined by the placement of the large pieces, since the smallest ones can frequently be accommodated among the largest ones.

The evaluation and assessment of the effectiveness of a piece sequence sorting criterion is done by comparing it to the solutions obtained by random sequences. For a given sorting criterion to be clearly effective, it should produce better layouts than the ones produced with random sequences. Additionally, the best results produced by random pieces sequences may allow identifying specific patterns that can enable the creation of new rules that consistently produce high quality layouts.

4 Computational Experiments

The computational experiments were executed on a platform with an Intel Xeon E5-5690@3.46Ghz processor, 48Gb RAM@1.33Ghz, Windows 7 x86-64, and a GPU Tesla C2070. The datasets used in the computational experiments were obtained from the ESICUP (EURO Special Interest Group on Cutting and Packing, www.fe.up.pt/esicup) website. These datasets, ordered by increasing geometric complexity, have been selected due to their diverse geometry (convex/irregular), total pieces, piece type, variation in size, and other factors (Table 1). This allows the evaluation of the approach under different circumstances. All pieces can be placed in 0 and 180 degrees rotations. Table 1 also presents the average computational times of the pre-processing phase (NFP) and the GPU-Nest heuristic (GPU-Nest). These computational times are independent of the pieces sequence, since they mainly depend on the pieces geometric characteristics. Namely, the pre-processing time depends on the number of vertices, while the nesting time depends on the total number of pieces, piece types and the number of pixels (given by the ratio between the board width and the grid resolution).

Table 1 Datasets characteristics and average computing times

	Characteristics						Average time	
Datasets	total pieces	piece types	average vertices	board width	unit grid size	total pixels (10^3)	NFP (s)	GPU-nest (s)
shapes	43	4	8.75	40	0.20	140.0	0,4	0.3
shirts	99	8	6.63	40	0.20	200.0	0.9	1.3
trousers	64	16	5.06	79	0.25	632.0	2.5	7.8
swim	48	10	21.90	5752	20.00	200.9	22.5	1.9

4.1 Randomly Generated Sequences

In order to assess the impact that a sequence may have on the final layout quality, a set of 200 randomly generated sequences were created and evaluated for the four datasets. Figure 1 shows the layout density histogram for dataset shapes. It can be seen that there is a large concentration of results within a certain range of solution quality, and that as the quality of the solutions increases or decreases, the total number of solutions produced diminishes significantly. This shows that there is room for improvement in the quality of the solutions by using better piece sequencing rules. Histograms for the other datasets (*shapes*, *swim* and *trousers*) have similar profiles. These results will serve as a baseline comparison against the results obtained with other pieces sequence sorting criteria.

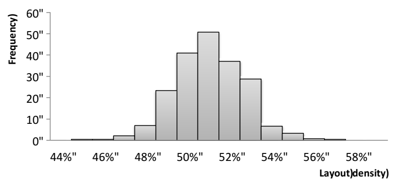

Fig. 1 Layout density histogram for dataset shapes

4.2 Divide Pieces by Size in Two Groups

An alternative to the random sequence approach was explored, based on the division of the pieces in two groups according to their area. A random sequence is generated within each group. A full sequence is created by adding the sequence from the group with the smaller pieces to the end of the sequence from the group with the bigger pieces. This strategy is based on the idea that the larger pieces define the main structure of a layout, and the smaller pieces may be placed in holes between the largest ones.

In order to test this approach based on subgroups organized by piece size, 200 sequences were produced for each subgroup in each dataset. The relative sizes of the pieces from each dataset were compared and a good allocation of pieces to each group selected. Figure 2 shows the layout density frequency polygons for datasets *shapes* and *shirts* when considering one and two groups. The two datasets clearly exhibit distinct behaviours when considering pieces divided in two groups: dataset *shapes* shows a small density improvement, while dataset *shirts* shows a bigger improvement. Dataset *swim* shows a similar behaviour to dataset *shapes*, while dataset *trousers* shows a similar behaviour to dataset *shirts*.

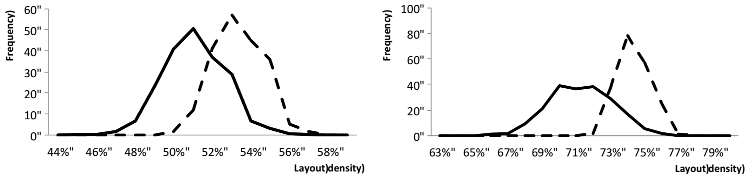

Fig. 2 Layout density frequency polygons for 1 group (solid line) and 2 groups (dashed line) for datasets *shapes* (left) and *shirts* (right)

The main reason for these different behaviours is due to the pieces' relative size in each dataset, as seen in figure 3. For dataset *shapes*, none of the smaller pieces could be placed among the bigger ones and only some of them for dataset *swim*. On the other hand, for datasets *trousers* and *shirts* the smaller pieces were all placed in holes between the larger ones, without increasing the layout length.

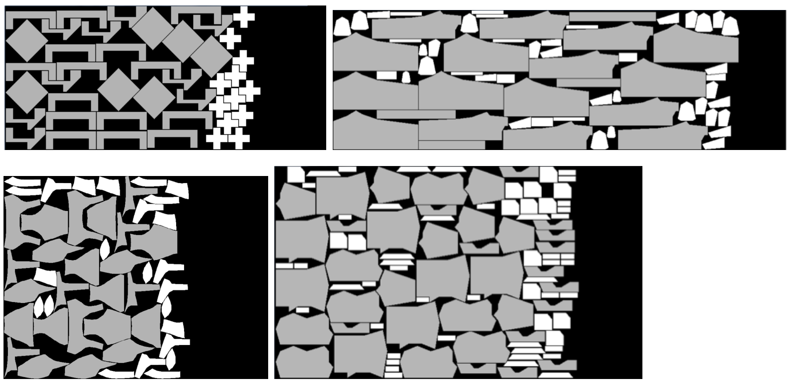

Fig. 3 Layouts with 2 groups of pieces (largest pieces in grey, smallest in white) for datasets *shapes* (top-left), *trousers* (top-right), *swim* (bottom-left) and *shirts* (bottom-right).

4.3 Best of 20 Random Sequences Divided in Two Groups

The solution quality obtained when using only one random sequence divided by groups has a large variability (figure 2). To overcome this issue and achieve a trade-off between the solution quality and the computational time, we tested an alternative approach where 20 random sequences, divided in two groups, are created and the best solution is selected. Table 2 compares the results between ordering the pieces by size (static criteria, i.e., fixed sequence), and random sequences (one and two groups). In the last column, we provide the results when considering the best layout among 20 random sequences generated with two groups. Selecting the best of 20 layouts allows obtaining a good solution, which has a very high probability of being better than the average solution of 200 random sequences. The downside of this approach is the increased computational cost.

Table 2 Impact of sequence generation procedure in the solution quality

	(1 seq,)	(200 random seq.)		(200 random seq.)		(20 random seq.)
Dataset	static	1 group		2 groups		2 groups
	—	avg.	(st.dev.)	avg.	(st.dev.)	best
shapes	51.7%	51.0%	(1.6%)	53.3%	(1.3%)	53.8%
shirts	74.8%	71.2%	(1.8%)	74.3%	(0.9%)	76.3%
trousers	76.5%	71.1%	(2.4%)	76.6%	(2.1%)	78.5%
swim	58.4%	56.2%	(1.8%)	59.3%	(1.6%)	58.9%

These algorithms have been implemented in the GPU, which improves the computation efficiency of this approach. It is specially true for datasets with complex geometries, as shown by the GPU-Nest heuristic average times on table 1. For instance, the best of 20 layouts for dataset *swim*, where pieces have in average more than 20 vertices, can be obtained in about 1 minute.

5 Conclusion

The results reported in this paper show that the solution quality of the layout is strongly dependent on the combination of the pieces sequence and placement rule. These results enabled the setting of a baseline for comparison for other experiments. The definition of the sequence based on the division into ordered sets of pieces allowed to determine the influence that such sequences may have on the quality of the layouts with a certain degree of variability. Noticeable improvements over the base random sequences were achieved with a simple division into two sets. The datasets with the largest variation in piece size showed to benefit significantly from this sequencing method, while more uniform datasets did not. Further research may lead to sequencing rules that enable generating consistently high quality solutions. Due to the implementation of these algorithms in a GPU, significant improvements in computational efficiency were achieved.

Acknowledgments. This work is financed by the FCT -- Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within project UID/EEA/50014/2013.

References

[1] Bennell, J., Oliveira, J.: The geometry of nesting problems: A tutorial. European Journal of Operational Research 184, 397–415 (2008)
[2] Bennell, J., Oliveira, J.: A tutorial in irregular shape packing problems. Journal of the Operational Research Society 60, 93–105 (2009)
[3] Burke, E., Hellier, R., Kendall, G., Whitwell, G.: A new bottom-left-fill heuristic algorithm for the two-dimensional irregular packing problem. Operations Research 54, 587–601 (2006)
[4] Fowler, R.J., Paterson, M.S., Tanimoto, S.L.: Optimal packing and covering in the plane are np-complete. Information Processing Letters 12, 133–137 (1981)
[5] Gomes, A.M., Oliveira, J.: A 2-exchange heuristic for nesting problems. European Journal of Operational Research 141, 359–370 (2002)
[6] Owens, J.D., Luebke, L., Govindaraju, N., Harris, M., Krüger, J., Lefohn, A.E., Purcell, T.J.: A survey of general-purpose computation on graphics hardware. Computer Graphics Forum 26, 1467–8659 (2007)
[7] Sampaio, S., Gomes, A.M., Rodrigues, R.: Exploring graphical processing in irregular strip packing problems. To be published in CIM Series in Mathematical Sciences, by Springer Verlag, for IO2013 - XVI Congresso da APDIO (June 2013)
[8] Wäscher, G., Hau*β*ner, H., Schumann, H.: An improved typology of cutting and packing problems. European Journal of Operational Research 183, 1109–1130 (2007)