

Using pattern matching for tiling and packing problems

Mahmood Amintoosi ^{a,c,*}, Hadi Sadoghi Yazdi ^b,
Mahmood Fathy ^c, Reza Monsefi ^d

^a *Mathematics Department, Tarbiat Moallem University of Sabzevar, Sabzevar 397, Iran*

^b *Engineering Department, Tarbiat Moallem University of Sabzevar, Sabzevar 397, Iran*

^c *Faculty of Computer Engineering, Iran University of Science and Technology, Tehran, Iran*

^d *Department of Computer, Faculty of Engineering, Ferdowsi University of Mashhad, Iran*

Received 15 October 2004; accepted 15 February 2006

Available online 18 October 2006

Abstract

This paper describes a new placement method based on pattern matching for 2D tiling problems. Tiling problem can be considered as a special case of bin packing. In the proposed method, the representation of the figures and the board is based on directional chain codes. Contrary to other works that the area has been used for the board and the figures, the proposed method is based on usage of their boundaries instead. With this representation, consideration of the area has been replaced with that of the exact string matching. With the proposed knowledge representation, rotation and reflection of the figures can be considered easily. The results of a hybrid approach of genetic algorithm and simulated annealing have been shown. This new method, introduces a novel approach for handling and solving a variety of 2D-packing problems.

© 2006 Elsevier B.V. All rights reserved.

Keywords: Tiling; Direction code; Pattern matching; String matching; 2D cutting and packing; Genetic algorithm

1. Introduction

The tiling problem is to pack a checkerboard (bin) with small pieces. In doing so, the figures must not overlap and they must stay within the confines of the board. In this study, we consider a special case of these sorts of problems i.e., Tiling with polyminoes. Each polymino is a rectilinear polygon that, the length of each edge is a multiple of some predefined unit length.

Most of the previous works make some restrictions on the problem. Many of them use rectangular figures [1], whereas some others use specified or congruent polygons [2]. Tiling problem can be considered as a bin packing or cutting stock problem that the global minima are required. Bin packing and cutting stock problem

* Corresponding author. Address: Mathematics Department, Tarbiat Moallem University of Sabzevar, Sabzevar 397, Iran.

E-mail addresses: amintoosi@sttu.ac.ir (M. Amintoosi), Sadoghi@sttu.ac.ir (H.S. Yazdi), mahfathy@iust.ac.ir (M. Fathy), rmonsefi@ferdowsi.um.ac.ir (R. Monsefi).

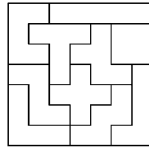


Fig. 1. The only solution for a 7×7 problem without rotation or reflection.

are the famous problems in combinatorial optimization problems and have received considerable amount of attention in the literatures [1,3].

In the next section, we explain two parallel algorithms among the reported algorithms on tiling and packing based on the modified McCulloch–Pitts neuron model [4,5] and a genetic algorithm based algorithm [6]. The first two algorithms are applicable for solving a 7×7 tiling problem with 10 polyominoes; which is the problem to find a single solution among 1.3×10^{14} possible candidates where there exists one and only one solution in the problem without rotation or reflection. Fig. 1 shows this solution of the problem. In the last paper, Pargas and Jain [6] proposed a stochastic approach, based on genetic algorithm and simulated annealing, for 2D-bin packing.

A common point in most of the algorithms on 2D-bin packing with irregular pieces, is the use of surface of the pieces for figures representation in their placement algorithm [4–9]. In this work instead of the surface, boundary of the figures and the bin has been used for their representation. The concept of direction code in image processing field has been used for boundary representation in the literatures. Anand et al. used the boundary of figures, but not for placement [10]. Hochstättler et al. used the boundary information for creating convex polygons [11]. To the best of our knowledge it is the first time that the circumference of the figures and pattern matching are used directly for placement. With this representation, working in the area (2D space) is transformed to working with the strings (1D space); in other words two-dimensional problem reduces to one-dimensional ones.

The rest of this article is organized as follows. In Section 2, we have a review on the related previous works. Section 3 contains a detailed explanation of how we applied pattern matching to the tiling problems. Section 4 gives an overview of the experimental results. The last section concludes with a summary of the paper and a quick look at our work in progress on this subject.

2. Related works

Takefuji and Lee [4] used a Hopfield neural network with optimization approach to solve this problem without rotation or reflection of figures. They defined an energy function that its value equals to zero, in solution state. As an optimization problem, energy function should be minimized. According to Hopfield and Tank [12], they update input of neurons to reach global minima. They reach to global minima in 100% of runs for the 7×7 problem shown in Fig. 1. This problem, can be solved by the three-dimensional $10 \times 7 \times 7$ neural network array, illustrated in Fig. 2.

Asai et al. [5] proposed a modified version of Takefuji and Lee approach. They add a new term (fitting violation function) to the energy function proposed by Takefuji and Lee and used an analog neural network array [5]. Their energy function is given by Eq. (1).

$$E = \frac{A}{2} \sum_{i=1}^l \left(\sum_{q=1}^m \sum_{r=1}^n V_{iqr} - 1 \right)^2 + \frac{B}{2} \sum_{i=1}^l \sum_{j=1}^m \sum_{k=1}^n f(i) V_{ijk}^2 + \frac{C}{2} \sum_{i=1}^l \sum_{j=1}^m \sum_{k=1}^n g(i) V_{ijk}^2 + \frac{D}{2} \sum_{i=1}^l \sum_{j=1}^m \sum_{k=1}^n V_{ijk} (1 - V_{ijk}), \quad (1)$$

where m , n are width and length of the checkerboard, l is the number of figures, $f(i)$ is overlap violation function and $g(i)$ is fitting violation function for i th polymino. A , B , C and D are adjustable parameters, that obtained with experience and trial and error methods. Their method found the global minima for five examples illustrated in Fig. 3.

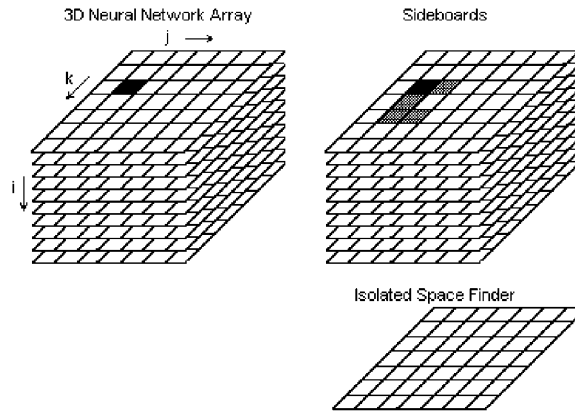
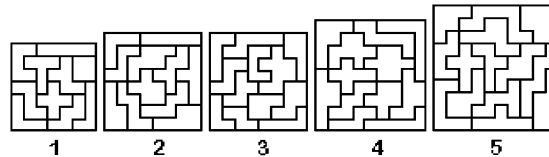
Fig. 2. Neural representation for the 7×7 tiling problem.

Fig. 3. Five examples illustrated in [5].

One of the disadvantages of above methods is that, they cannot be applied to the problems that have more than one figure with the same shape and orientation. Our simulation results showed that the aforementioned methods do not reach to the global minima for these sorts of problems. Another drawback of these methods is their parameters dependencies.

Pargas and Jain proposed a stochastic approach for 2D-bin packing. Their technique is similar to those used in GA¹ or in SA² algorithms. It is suitable for bin packing, but does not do well for tiling problems. Due to usage of their terminology in the next section, some component of their work is briefly explained here. A solution structure in their work has the following format:

$$[(f_1, o_1), (f_2, o_2), \dots, (f_N, o_N), L],$$

where f_i , $0 \leq f_i \leq N - 1$ represents the i th of N figures, and o_j , $0 \leq o_j \leq 3$ is the figure current orientation in steps of 90° , and L is the length of the solution. A bin has a predefined height H . As an optimization problem, L is objective function that must be minimized. Their main algorithm is as follows:

//Pargas and Jain Algorithm

Initialize the population by **generating a random solution**, determining its length L , and inserting it into the population according to L ;

repeat

Select a solution using a linearly biased random number generator;

Generate a new solution either by perturbing the solution selected (probability p) or by **generating a random solution** (probability $(1 - p)$);

Determining the value, L , of the new solution; Insert the solution into the population maintaining sorted order;

Increment the iteration count;

¹ Genetic algorithm.

² Simulating annealing.

```
until [(population has converged) or (No improvement of best solution) or
      (NumIterations > MaxIterations)]
```

Evaluation of a solution S involves assigning a location in the bin (board) to each figure such that no two figures overlap. The evaluation algorithm employs a two-dimensional bit array, of height H and arbitrary length. In the evaluation algorithm, attempt to place the figure with the prescribed orientation, and with its leftmost uppermost square unit on the first empty cell. If unsuccessful, try figure by rotating 90° if again unsuccessful, try figure by rotating 180° if once more unsuccessful, try figure by rotating 270° , otherwise, try next empty cell until figure is successfully placed. Scanning the cells of bin in search of the next empty cell, is done by a simple linear scan in column major order. In termination, convergence occurs when all solutions in the population have the same length.

With their algorithm but without rotation of the figures our simulation results showed that their algorithm could not find any solution for examples 3, 4 and 5, illustrated in Fig. 3. Percentage rate of reaching to the solution for examples 1, 2 were 60 and 20, respectively.

3. The proposed method

For our core framework we use the hybrid method of [6], which was mentioned in the previous section. The main difference is in the solution generation procedure, which is indicated with bold letters in the Pargas and Jain Algorithm. In the solution generation procedure, the figures are placed on the board, in sequence one after the other, with a random order of figures. In the proposed method, for the placement of the figures, the perimeter of the figures and the board instead of their areas has been used. Hence, working on two-dimensional area has been replaced with exact string matching. The following subsections explain in detail how the perimeter has been used for placement of the figures. Several basic components for proposed algorithm need to be explained. These include:

1. The perimeter against the area.
2. The string representation of the board and the figures.
3. Rotation and reflection of the figures.
4. The placement algorithm.
5. Generation of a possible solution.

3.1. The perimeter against the area

When a figure enlarges, its area increases in quadratic manner with respect to the size, whereas the increase of the perimeter is linear. If we want to use mathematic notation, for a circular figure with radius r the perimeter ($2\pi r$) has a linear proportion to radius but its area (πr^2) has a quadratic proportion to its radius. Hence, one can expect that, the time complexity of the placement algorithm based on the perimeter, might be better than that of the area. In the next section it is explained how a figure may be represented by its boundary.

3.2. The string representation of the board and the figures

Boundary detection is one of the well-known problems in image processing field. Chain codes are used to represent a boundary by a connected sequence of straight-line segments of specified length and direction. Typically, this representation is based on 4- or 8-connectivity of the segments [13]. The direction of each segment is coded by using a numbering scheme as the ones shown in Fig. 4. The representation of a right-angled polygon needs only four codes. For future extensions, 0, 2, 4 and 6 are chosen for four cardinal directions and 1, 3, 5 and 7 are reserved.

Fig. 5 shows one polymino and its 4-Directional Chain Code string. We will use *DirCode* instead of 4-Directional Chain Code notation.

As mentioned earlier, the board can be polymino and it does not restrict to the rectangular forms. Any shaped board can be represented in this way. Chain code string for a 5×4 rectangular bin is

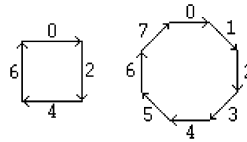
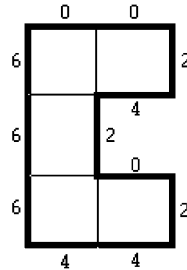


Fig. 4. Direction numbers for four- and eight-directional chain codes.



4-Directional Chain Code: 002420244666

Fig. 5. A polymino and its corresponding chain code.

000022222444466666. It is not hard to produce a chain code string for any shape. In this study the left upper most cell of each figure is starting point for its direction code generation. Top edge of the cell is selected, and with a clockwise movement; direction code would be in hand.

3.3. Rotation and reflection of the figures

With this representation, rotation and reflection of the figures can be done easily. For rotating 90° , and horizontal or vertical reflecting it is sufficient to exchange every character in chain code string with appropriate letter.

3.4. The placement algorithm

With this representation, placing figures on the board in two-dimensional space reduces to finding a match between two strings in one-dimensional space. Suppose B is chain code string of bin and F denotes the figure chain code. It is sufficient to find a common substring of B and F , and replace the common portion of B with *appropriate code*, based on F . For the generation of this proper code, a new term: *Reverse Direction Code* (Rev-DirCode) is defined. As pointed earlier, chain code of each figure, constructed based on clockwise movement. Reverse direction code of each figure, is achieved by movement under reverse clockwise direction. Fig. 6 illustrates chain code and its reverse for a ‘C’ shaped figure.

Reverse direction code can be constructed easily, if the direction code is available. If string D , indicates a chain code, the following procedure, produces its reverse in R .

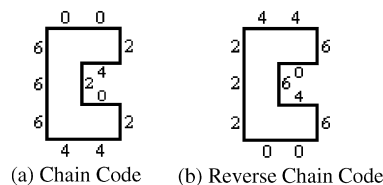


Fig. 6. Directional chain code and its reverse for a ‘C’ shaped figure. Its 4-Directional Chain Code is: 002420244666, and its Reverse Directional Chain Code is: 222006460644.

```

function Reverse(D:string)
begin
  for i:=1 to length(D) do
    R[length(D) - i + 1] := (D[i] + 4) mod 8;
  Return R;
end;

```

Fig. 7 shows some of the possible placement locations of a ‘C’ shaped figure on a 5×4 checkerboard, based on their common substrings. It also demonstrates various problems rose in placement of the figures by the proposed method. Common substrings of board and figures chain codes and corresponding substring of reverse direction codes shown with underlined letters. Underlined letters from the board (bin) chain code

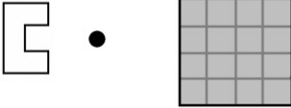
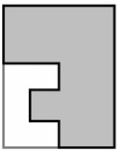
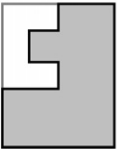
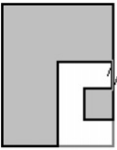
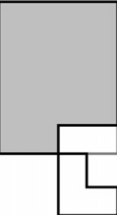
Inserting a ‘C’ shaped figure in a 5x4 board 	
(a) – No Problem Insertion	
	Bin DirCode: 000022222444466666 Figure DirCode: 002420244666 Figure RevDirCode: <u>22200</u> 6460644 Bin DirCode after insertion: 00002222244 6460644 66
(b) – Cyclic Checking	
	Bin DirCode: <u>0000</u> 22222444466666 Figure DirCode: <u>0024</u> 20244666 Figure RevDirCode: <u>22200</u> 6460644 Bin DirCode after insertion: 0022222444466 0064606
(c) – Board Splitting	
	Bin DirCode: 000022222444466666 Figure DirCode: 002420244666 Figure RevDirCode: 22200 <u>6460644</u> Bin DirCode after insertion: 00002222 46064422 24466666 Bin DirCode after splitting: 2460-4422 24466666000022
(d) – Outing of the board	
	Bin DirCode: 000022222444466666 Figure DirCode: 00 <u>24</u> 20244666 Figure RevDirCode: 22200646 <u>0644</u> Bin DirCode after insertion: 00002222 4422200646 44466666

Fig. 7. Different insertion forms and various problems rose in placement of a figure with the proposed method. Common substrings of the board (bin) and the figure chain codes and corresponding substring of reverse direction code shown with underlined letters. Underlined letters from the board chain code string is deleted and replaced with the remaining portion of the reverse direction code, illustrated with bold letters.

string has been deleted and replaced with the remaining portion of the reverse direction code, illustrated with bold letters.

With these direction codes, placement of a figure can be done in a simple way. The following Place Procedure shows the placement algorithm.

In the following Place Procedure, after finding a match between two strings—that are highlighted with underlined letters in Fig. 7—the corresponding strings are deleted from the board DirCode and the figures RevDirCode. The remaining substring of figure RevDirCode—that is highlighted with bold letters in Fig. 7, is inserted into board DirCode, at the position of the deleted string. An advantage of this method is that, checking for the overlapping figures is no longer needed.

```

procedure Place(F,B);
  // B:= Bin Directional Chain Code String;
  // F:= Figure Directional Chain Code String;
begin
  R:= Reverse Direction Code String of the Figure;
  SubString := FindAMatch(B,F);
  RevSubString := Reverse(SubString);
  index := Delete(B,SubString);
  //Delete: deletes common substring from B and
  //returns the position of the deleted substring
  Insert(B,R-RevSubString,index);
end;
```

With the proposed method, several problems, frequently arise, that must be explained with more details. These are included:

- Circular sequence.
- Match finding.
- Out of the board checking and
- Board splitting.

In the following subsections these components will be explained.

3.4.1. Circular sequence

In finding a common substring between two DirCode strings, one portion of the common substring may be at the end, and another portion lies at the starting point of one or both strings, as can be seen in the Fig. 7(b). We simply treat the chain code as a circular sequence of direction numbers.

3.4.2. Match finding

Among the string matching methods, there are effective algorithms for exact pattern matching, which can be used in FindAMatch procedure [14,16]. The size of our alphabet is small (equal with 4) and the length of the patterns (perimeter of the figure) is long. Experimental results showed that the Reverse Factor Algorithm is the most efficient algorithm for match finding in this situation [15]. But in this study we need to find all of the common substrings of two strings. Every partial exact matching corresponds with a possible location of the figure on the board. Figs. 8 and 9 demonstrate some of the common substrings of bin and figure direction codes and their corresponding possible locations, when a figure is placed on a board.

If the placement of a figure was not successful we have to test *the next possible location* for it. *The next possible location* may have several meanings:

- The next common substring as shown in Fig. 9.
- The next empty cell, in column major order, as Pargas and Jain method.
- The next best fit location that is related with the next longest common substring.

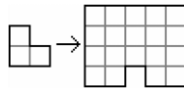


Fig. 8. Placing an L-shaped figure on a board; the Fig-DirCode (pattern) is 02024466 and the Bin-DirCode is 00000222244642446666.

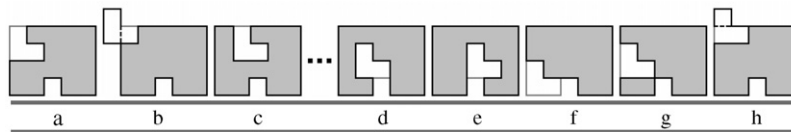


Figure Label	Bin DirCode:	Figure DirCode:	Common Substring
a	000002222446424466 <u>66</u>	020244 <u>66</u>	660
b	00000222244642446666	02024466	0
c	00000222244642446666	02024466	0
d	00000222244642446666	02024466	4
e	00000222244642446666	02024466	4
f	00000222244642446666	02024466	24466
g	00000222244642446666	02024466	66
h	00000222244642446666	02024466	6

Fig. 9. Some of the common substrings of bin and figure chain codes and their corresponding possible locations on 2D surface. Common substrings, shown with underlined letters. The figure moves around the inside of the board. Some of cases such as (b) should be distinguished (out of the board). The first common substring is 660 (the end and the beginning of the Fig-DirCode) (a). The second common substring is 0 (the third letter of the Fig-DirCode string) (b). The best match corresponds with the best-fit location (f).

We named the two first: *First Match* and the latest *Best Match*. These are different from First Fit and Best Fit notations in the bin packing literatures.

In this study the *Best Match* has been used for simulation results. The best common substring is the Longest Common Substring. It is possible to have some identical best positions for a figure. It is assumed that in these situations, all identical positions have equal chance for selection. The various tiling patterns that were examined by us, give this intuitive result that, every pattern can be produced with this way. Finding a formal correctness proof of the mentioned note is one of the future works.

We need a quick algorithm to find common substrings of two strings in non-increasing order, based on their lengths. At this stage of our research the brute force algorithm has been used for pattern matching. However its implementation could be largely improved using sophisticated string algorithms and data structures such as suffix trees [15].

3.4.3. Out of the board checking

As you can see in Fig. 7(d) a figure may lie out of the board. For exploring these situations and for the previous and next subsections reasons, we should use the coordinates of the boundary cells of the board. Fig. 10 shows a board and its boundary cells. With this cells, out of board checking can be done with an algorithm with time complexity $O(\text{length}(\text{inserted substring}))$.

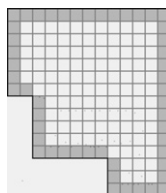


Fig. 10. The grayed cells are the boundary cells of a 15×13 board, after placing some figures on the left bottom corner of the board.

3.4.4. Board splitting

Sometimes the board may be split in placement figure. If the board chain code is leaved unchanged, the best match procedure would have some errors in the order of the best match positions. Hence for use of best match it is necessary to split the board. For board splitting, letters of the board chain code that are tangent with each other, must be distinguished (Fig. 7(c)). It is not hard to identify the mentioned letters and split the board, if we use boundary cells. It is done with a recursive function. For first match we do not need splitting.

3.5. Generation of a possible solution

Solution generation is obtained easily, with the stated components. The following procedure demonstrates the random solution generation. It is assumed that every call of FindAMatch procedure in the Place procedure, finds the next common substring of the board and figure. The input of the procedure is the bin and the figures chain codes, which mentioned in the previous sections. The output of the proposed method is the founded solution by the best member of the genetic algorithm.

```

procedure RandomSolutionGeneration;
  B := Bin Directional Chain Code String;
  F := array of Figures DirCode Strings with random order;
begin
  for i := 1 to NumberOfFigures do
    begin
      while there is a match between B,F[i] do
        begin
          Place(F[i],B); // It places ith figure on the board
          if not OutOfBoard() then break;
        end;
        if necessary SplitTheBoard();
      end;
    end;
  end;
end;

```

For implementation of the proposed approach, we used the Pargas and Jain method in solving 2D bin packing problems, with a bit modification. In the Pargas and Jain algorithm, the bin length is unlimited, and the goal is to find a minimum length to pack all of figures, but in tiling problems, the length and height of the board is fixed. Hence we modify their fitness function to satisfy our constraint. In tiling problems, on the solution state the remaining area of the board is zero, hence we redefine fitness function as equation (2).

$$\text{fitness_value} = \sum_{i \in S} A_i + \text{NumberOfHoles} / \text{TotalArea}, \quad (2)$$

where S is the set of non-placed figures, A_i is the area of the i th figure, NumberOfHoles is the number of holes of the produced tiling pattern, (that not covered with any figure) and TotalArea is the total area of the board. In addition we used the proposed representation for the board and the figures. The next section summarizes some simulation results.

4. Simulation results

The Pargas and Jain method [6] is implemented with a small modification: rotation of figures was not considered. In the proposed approach, the Best Match form is used as FindAMatch subroutine in Place Procedure. Fig. 11 shows the percentage rate to reach to global minima for six examples. We ran our proposed method and Pargas and Jain method on each example 10 times. Genetic population size is 20 and maximum generation number is set to 500. Examples no. 1–5 was illustrated in Fig. 3 and example no. 6 is illustrated in Fig. 12.

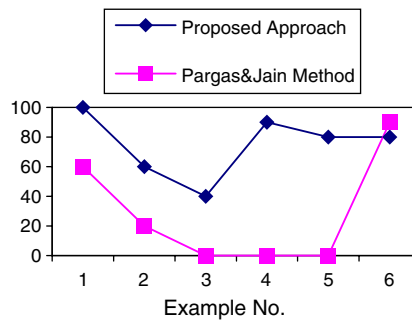


Fig. 11. Percentage to reach to the right solution in the proposed method and Pargas and Jain method.

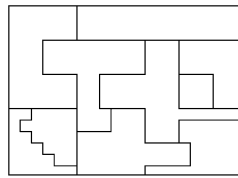


Fig. 12. Example no. 6: an obtained solution of placing 10 polyminoes on a 15×21 board.

As can be seen in Fig. 11 the average case of the proposed approach is better than Pargas and Jain method. As shown in Fig. 11, Pargas and Jain method cannot find a global solution over examples 3–5; but the proposed algorithm gives 40%, 90% and 80% to reach to the right solutions, respectively, for examples 3–5.

5. Concluding remarks and future works

In this study a new knowledge representation method has been discussed based on the perimeter of the figures and the board for 2D tiling and packing problems. The representation of the figures and the board is based on four-directional chain code in image processing. Hence finding a place for a figure on the board consists then in identifying a common substrings in the strings that represent them. With this representation, the two-dimensional problem reduces to one-dimensional ones. For a specified figure, the area increases in quadratic manner with respect to the size, but the increase of the perimeter is linear. Thus, one can expect that, the time complexity of perimeter-based placement algorithms might be better than that of the area-based algorithms. The simulation results yielded by a hybrid method, showed that the proposed knowledge representation is applicable for solving tiling problems. However its implementation could be largely improved using sophisticated string algorithms and data structures such as suffix trees. It is expected that using chain codes and pattern matching, initiate new methods for handling and dealing with 2D-packing problems.

Acknowledgements

The authors would like to thank the Prof. Pham Dinh Tao and an anonymous reviewer for detailed and constructive comments that greatly improved the manuscript.

References

- [1] H. Dyckhoff, G. Wascher (Eds.), European Journal of Operational Research 44 (2) (1990), Special Issue on Cutting and Packing.
- [2] M. Reid, Tiling rectangles and half strips with congruent polyominoes, Journal of Combinatorial Theory, Series A 80 (1) (1997) 106–123.
- [3] E.D. Goodman, A.Y. Tetelbaum, M. Kureichik, A genetic algorithm approach to compaction, bin packing, and nesting problems, Technical Report of Case Center for Computer-Aided Engineering and Manufacturing, Michigan State University, No. 940702, 1994.
- [4] Y. Takefuji, K.C. Lee, A parallel algorithm for tiling problems, IEEE Transactions on Neural Networks 1 (1) (1990) 143–145.

- [5] H. Asai, T. Nakayama, H. Ninomya, Tiling algorithm with fitting violation function for analog neural array, Proceedings of the International Conference on Neural Network (1996) 565–570.
- [6] R.P. Pargas, R. Jain, A parallel stochastic optimization algorithm for solving 2D bin packing problems, in: 9th Conference on AI for Applications, 1993, pp. 18–25.
- [7] P. Poshyanonda, A. Bahrami, C.H. Dagli, Two-dimensional nesting problems: Artificial neural network and optimization approach, International Conference on Neural Network (1992) 572–577.
- [8] P. Poshyanonda, C.H. Dagli, Genetic neuro-nester for irregular patterns Intelligent Engineering Systems through Artificial Neural Networks, vol. 3, ASME Press, 1993, pp. 825–830.
- [9] R. Monsefi, M. Amintoosi, A genetic-neuro algorithm for tiling problems, Iranian Journal of Science and Technology, Shiraz University, 2002.
- [10] S. Anand, C. McCord, R. Sharma, An integrated machine vision based system for solving the non-convex cutting stock problem using genetic algorithms, Journal of Manufacturing Systems 18 (6) (1999) 396–415.
- [11] W. Hochstattler, M. Lobel, C. Moll, Generating convex polyminoes at random, Discrete Mathematics 153 (1–3) (1996) 165–176.
- [12] J.J. Hopfield, D.W. Tank, Neural computation of decisions in optimization problems, Biological Cybernetics 52 (1985) 141–152.
- [13] R.C. Gonzalez, R.E. Woods, Digital Image Processing, Second ed., Prentice Hall, 2002.
- [14] C. Charras, T. Lecroq, Handbook of Exact String Matching Algorithms, King's College Publications, 2004. Available from: <http://www-igm.univ-mlv.fr/~lecroq/string/string.ps>.
- [15] T. Lecroq, Experimental results on string matching algorithms, Software—Practice and Experience 25 (7) (1995) 727–765.
- [16] D. Gusfield, Algorithms on Strings, Trees, and Sequences, Cambridge University Press, New York, 1997.