



ELSEVIER

SCIENCE @ DIRECT®

European Journal of Operational Research xxx (2006) xxx–xxx

EUROPEAN  
JOURNAL  
OF OPERATIONAL  
RESEARCH[www.elsevier.com/locate/ejor](http://www.elsevier.com/locate/ejor)

## Using pattern matching for tiling and packing problems

Mahmood Amintoosi<sup>a,c,\*</sup>, Hadi Sadoghi Yazdi<sup>b</sup>,  
Mahmood Fathy<sup>c</sup>, Reza Monsefi<sup>d</sup>

<sup>a</sup> *Mathematics Department, Tarbiat Moallem University of Sabzevar, Sabzevar 397, Iran*

<sup>b</sup> *Engineering Department, Tarbiat Moallem University of Sabzevar, Sabzevar 397, Iran*

<sup>c</sup> *Faculty of Computer Engineering, Iran University of Science and Technology, Tehran, Iran*

<sup>d</sup> *Department of Computer, Faculty of Engineering, Ferdowsi University of Mashhad, Iran*

Received 15 October 2004; accepted 15 February 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 Published by Elsevier B.V.

*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](mailto:amintoosi@sttu.ac.ir) (M. Amintoosi), [Sadoghi@sttu.ac.ir](mailto:Sadoghi@sttu.ac.ir) (H.S. Yazdi), [mahfathy@iust.ac.ir](mailto:mahfathy@iust.ac.ir) (M. Fathy), [rmonsefi@ferdowsi.um.ac.ir](mailto:rmonsefi@ferdowsi.um.ac.ir) (R. Monsefi).

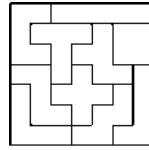


Fig. 1. The only solution for a  $7 \times 7$  problem without rotation or reflection.

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

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

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

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

## 52 2. Related works

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

59 Asai et al. [5] proposed a modified version of Takefuji and Lee approach. They add a new term (fitting vio-  
 60 lation function) to the energy function proposed by Takefuji and Lee and used an analog neural network array  
 61 [5]. Their energy function is given by Eq. (1).

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

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

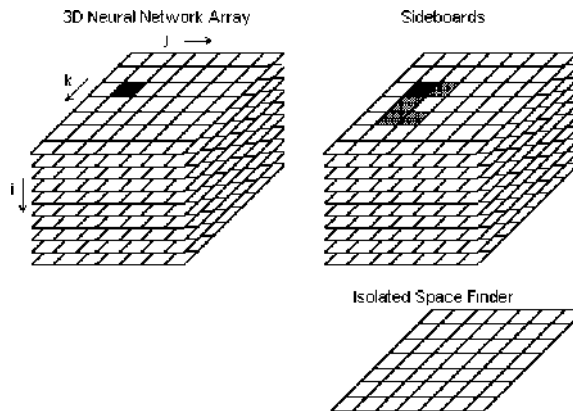
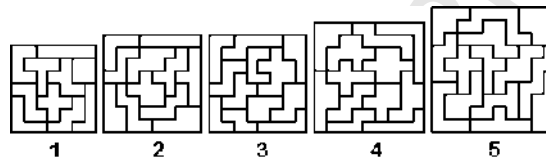
Fig. 2. Neural representation for the  $7 \times 7$  tiling problem.

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

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

73 Pargas and Jain proposed a stochastic approach for 2D-bin packing. Their technique is similar to those  
 74 used in GA<sup>1</sup> or in SA<sup>2</sup> algorithms. It is suitable for bin packing, but does not do well for tiling problems.  
 75 Due to usage of their terminology in the next section, some component of their work is briefly explained here.  
 76 A solution structure in their work has the following format:

$$78 \quad [(f_1, o_1), (f_2, o_2), \dots, (f_N, o_N), L],$$

79 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  
 80 steps of  $90^\circ$ , and  $L$  is the length of the solution. A bin has a predefined height  $H$ . As an optimization problem,  
 81  $L$  is objective function that must be minimized. Their main algorithm is as follows:

82 //Pargas and Jain Algorithm

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

86 repeat

87 Select a solution using a linearly biased random number generator;

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

90 Determining the value,  $L$ , of the new solution; Insert the solution into the popu-  
 91 lation maintaining sorted order;

93 Increment the iteration count;

<sup>1</sup> Genetic algorithm.

<sup>2</sup> Simulating annealing.

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

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

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

### 109 3. The proposed method

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

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

123

#### 124 3.1. The perimeter against the area

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

#### 130 3.2. The string representation of the board and the figures

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

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

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

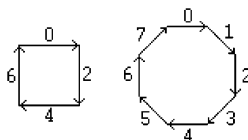
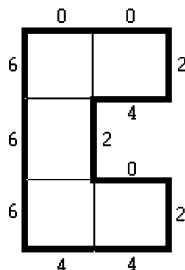


Fig. 4. Direction numbers for four- and eight-directional chain codes.



4-Directional Chain Code: 002420244666

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

141 000022222444466666. It is not hard to produce a chain code string for any shape. In this study the left upper  
 142 most cell of each figure is starting point for its direction code generation. Top edge of the cell is selected, and  
 143 with a clockwise movement; direction code would be in hand.

144 3.3. Rotation and reflection of the figures

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

148 3.4. The placement algorithm

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

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

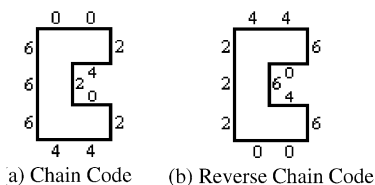


Fig. 6. Directional chain code and its reverse for a ‘C’ shaped figure. Its 4-Directional Chain Code is: 002420244666, and its Reverse Directional Chain Code is: 222006460644.

```

158 function Reverse(D:string)
159 begin
160   for i:=1 to length(D) do
162     R[length(D) - i + 1] := (D[i] + 4) mod 8;
164   Return R;
165 end;
166

```

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


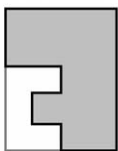
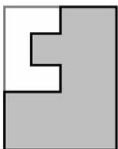
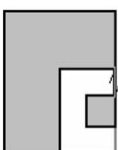

Inserting a 'C' shaped figure in a 5x4 board	
	
(a) – No Problem Insertion	
	Bin DirCode: 000022222444466666 Figure DirCode: 002420244666 Figure RevDirCode: <u>22200</u> 6460644 Bin DirCode after insertion: 0000222224444646064466
(b) – Cyclic Checking	
	Bin DirCode: <u>0000</u> 22222444466666 Figure DirCode: <u>0024</u> 20244666 Figure RevDirCode: <u>2220</u> 06460644 Bin DirCode after insertion: 00222224444660064606
(c) – Board Splitting	
	Bin DirCode: 000022222444466666 Figure DirCode: 002420244666 Figure RevDirCode: <b>2220</b> 06460644 Bin DirCode after insertion: 00002222 <b>460644</b> 2224466666 Bin DirCode after splitting: <b>2460-44</b> 2224466666000022
(d) – Outing of the board	
	Bin DirCode: 000022222444466666 Figure DirCode: 002420244666 Figure RevDirCode: <b>2220</b> 06460644 Bin DirCode after insertion: 00002222 <b>4422</b> 200646444466666

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

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

173 With these direction codes, placement of a figure can be done in a simple way. The following Place Proce-  
174 dure shows the placement algorithm.

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

```

180 procedure Place(F,B);
181 // B:= Bin Directional Chain Code String;
182 // F:= Figure Directional Chain Code String;
183 begin
184 R:=Reverse Direction Code String of the Figure;
185 SubString:= FindAMatch(B,F);
186 RevSubString:= Reverse(SubString);
187 index:= Delete(B,SubString);
188 //Delete: deletes common substring from B and
189 //returns the position of the deleted substring
190 Insert(B,R-RevSubString,index);
191 end;
192
193
194
```

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

- 197 • Circular sequence.
- 198 • Match finding.
- 199 • Out of the board checking and
- 200 • Board splitting.

201  
202 In the following subsections these components will be explained.

### 203 3.4.1. Circular sequence

204 In finding a common substring between two DirCode strings, one portion of the common substring may be  
205 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).  
206 We simply treat the chain code as a circular sequence of direction numbers.

### 207 3.4.2. Match finding

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

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

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

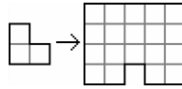


Fig. 8. Placing an L-shaped figure on a board; the Fig-DirCode (pattern) is 02024466 and the Bin-DirCode is 00000222244642446666.

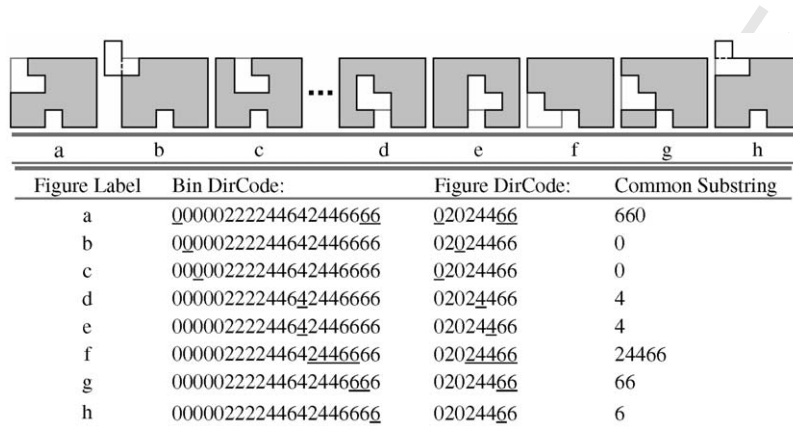


Fig. 9. Some of the common substrings of bin and figure chain codes and their corresponding possible locations on 2D surface. Common substrings, shown with underlined letters. The figure moves around the inside of the board. Some of cases such as (b) should be distinguished (out of the board). The first 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).

220

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

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

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

### 232 3.4.3. Out of the board checking

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

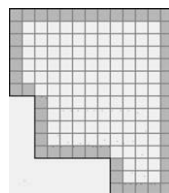


Fig. 10. The grayed cells are the boundary cells of a  $15 \times 13$  board, after placing some figures on the left bottom corner of the board.



### 237 3.4.4. Board splitting

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

### 244 3.5. Generation of a possible solution

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

```

250 procedure RandomSolutionGeneration;
251   B := Bin Directional Chain Code String;
252   F := array of Figures DirCode Strings with random order;
253 begin
254   for i := 1 to NumberOfFigures do
255     begin
256       while there is a match between B,F[i] do
257         begin
258           Place(F[i],B); // It places ith figure on the board
259           if not OutOfBoard() then break;
260         end;
261         if necessary SplitTheBoard();
262       end;
263     end;
264   end;
265 end;
266
267
268
  
```

269 For implementation of the proposed approach, we used the Pargas and Jain method in solving 2D bin  
 270 packing problems, with a bit modification. In the Pargas and Jain algorithm, the bin length is unlimited,  
 271 and the goal is to find a minimum length to pack all of figures, but in tiling problems, the length and height  
 272 of the board is fixed. Hence we modify their fitness function to satisfy our constraint. In tiling problems, on the  
 273 solution state the remaining area of the board is zero, hence we redefine fitness function as equation (2).  
 274

$$276 \text{ fitness\_value} = \sum_{i \in S} A_i + \text{NumberOfHoles} / \text{TotalArea}, \quad (2)$$

277 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  
 278 of the produced tiling pattern, (that not covered with any figure) and TotalArea is the total area of the board.  
 279 In addition we used the proposed representation for the board and the figures. The next section summarizes  
 280 some simulation results.

## 281 4. Simulation results

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

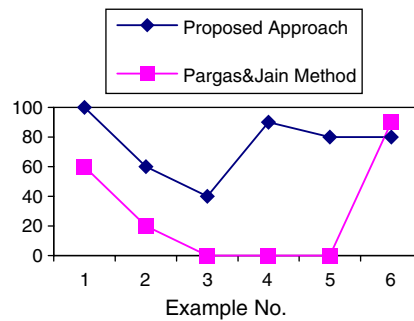


Fig. 11. Percentage to reach to the right solution in the proposed method and Pargas and Jain method.

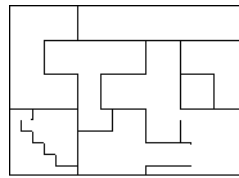


Fig. 12. Example No. 6: an obtained solution of placing 10 polyominoes on a  $15 \times 21$  board.

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

### 291 5. Concluding remarks and future works

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

### 303 Acknowledgements

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

### 306 References

- 307 [1] H. Dyckhoff, G. Wascher (Eds.), European Journal of Operational Research 44 (2) (1990), Special Issue on Cutting and Packing.  
 308 [2] M. Reid, Tiling rectangles and half strips with congruent polyominoes, Journal of Combinatorial Theory, Series A 80 (1) (1997) 106–  
 309 123.  
 310 [3] E.D. Goodman, A.Y. Tetelbaum, M. Kureichik, A genetic algorithm approach to compaction, bin packing, and nesting problems,  
 311 Technical Report of Case Center for Computer-Aided Engineering and Manufacturing, Michigan State University, No. 940702, 1994.  
 312 [4] Y. Takefuji, K.C. Lee, A parallel algorithm for tiling problems, IEEE Transactions on Neural Networks 1 (1) (1990) 143–145.

- 313 [5] H. Asai, T. Nakayama, H. Ninomya, Tiling algorithm with fitting violation function for analog neural array, Proceedings of the  
314 International Conference on Neural Network (1996) 565–570.
- 315 [6] R.P. Pargas, R. Jain, A parallel stochastic optimization algorithm for solving 2D bin packing problems, in: 9th Conference on AI for  
316 Applications, 1993, pp. 18–25.
- 317 [7] P. Poshyanonda, A. Bahrami, C.H. Dagli, Two-dimensional nesting problems: Artificial neural network and optimization approach,  
318 International Conference on Neural Network (1992) 572–577.
- 319 [8] P. Poshyanonda, C.H. Dagli, Genetic neuro-nester for irregular patterns Intelligent Engineering Systems through Artificial Neural  
320 Networks, vol. 3, ASME Press, 1993, pp. 825–830.
- 321 [9] R. Monsefi, M. Amintoosi, A genetic-neuro algorithm for tiling problems, Iranian Journal of Science and Technology, Shiraz  
322 University, 2002.
- 323 [10] S. Anand, C. McCord, R. Sharma, An integrated machine vision based system for solving the non-convex cutting stock problem using  
324 genetic algorithms, Journal of Manufacturing Systems 18 (6) (1999) 396–415.
- 325 [11] W. Hochstattler, M. Lobel, C. Moll, Generating convex polyminoes at random, Discrete Mathematics 153 (1–3) (1996) 165–176.
- 326 [12] J.J. Hopfield, D.W. Tank, Neural computation of decisions in optimization problems, Biological Cybernetics 52 (1985) 141–152.
- 327 [13] R.C. Gonzalez, R.E. Woods, Digital Image Processing, Second ed., Prentice Hall, 2002.
- 328 [14] C. Charras, T. Lecroq, Handbook of Exact String Matching Algorithms, King's College Publications, 2004. Available from: [http://](http://www-igm.univ-mlv.fr/~lecroq/string/string.ps)  
329 [www-igm.univ-mlv.fr/~lecroq/string/string.ps](http://www-igm.univ-mlv.fr/~lecroq/string/string.ps).
- 330 [15] T. Lecroq, Experimental results on string matching algorithms, Software—Practice and Experience 25 (7) (1995) 727–765.
- 331 [16] D. Gusfield, Algorithms on Strings, Trees, and Sequences, Cambridge University Press, New York, 1997.
- 332