Applying Ant Algorithms and the No Fit Polygon to the Nesting Problem

Edmund Burke and Graham Kendall

The School of Computer Science and Information Technology
The University of Nottingham
Nottingham NG7 2RD UK
{ekb,gxk}@cs.nott.ac.uk

Abstract. In previous work solutions for the nesting problem are produced using the no fit polygon (NFP), a new evaluation method and three evolutionary algorithms (simulated annealing (SA), tabu search (TS) and genetic algorithms (GA)). Tabu search has been shown to produce the best quality solutions for two problems. In this paper this work is developed. A relatively new type of search algorithm (ant algorithm) is developed and the results from this algorithm are compared against SA, TS and GA. We discuss the ideas behind ant algorithms and describe how they have been implemented with regards to the nesting problem. The evaluation method used is described, as is the NFP. Computational results are given.

Keywords. Genetic Algorithm, Search, Ant Algorithms, No Fit Polygon, Simulated Annealing

1 Introduction

In the nesting problem it is necessary to place a number of shapes onto a larger shape. In doing so the shapes must not overlap and they must stay within the confines of the larger shape. The usual objective is to minimise the waste of the larger shape. Only two dimensions, height and width, of the shapes are considered and the larger piece is sometimes considered to be of infinite height so that only the width of the placements need be checked. This is a realistic assumption for the real world as the larger shapes are sometimes rolls of material which can be considered as being of infinite length for the purposes of the placement procedure. In this paper a number of assumptions are made. The height of the bin (the larger piece) is considered infinite, although it remains the aim of the evaluation function to minimise this height. Only convex polygons are considered. Only one bin is used (that is, there is no concept of filling a bin and having to start another). Only guillotine cuts are allowed (that is, a cut must be made from one edge to the other). In future research, once we have shown the method to be effective, we will relax these constraints.

Ant Algorithms (described in section 4) are a relatively new search mechanism, having been introduced by Marco Dorigo in his PhD thesis [16] and in [9, 10]. Initially the algorithm was applied to the Travelling Salesman Problem (TSP) [19].

N. Foo (Ed.): AI'99, LNAI 1747, pp. 453-464, 1999.

In [25], when ant algorithms are compared against an iterated local search (ILS) algorithm, which is known to be among the best algorithms for the TSP, the ant algorithm finds a better average solution quality than ILS. In recent years, ant algorithms have also been applied to other problems. A recent paper [23] compares the results of an ant algorithm using test problems from QALIB. The ant algorithm was able to beat GRASP (greedy randomized adaptive search procedure) in all cases. The algorithm was also applied to a real world case and found a solution that was 22.5% better then the current one (although it is noted that this figure must be read with caution as other factors have been included in the real world planning). Ant Algorithms have also been applied to Vehicle Routing Problems (VRP) [4]. In [4] the algorithm could not improve on published results but it is seen as a viable alternative when tackling VRP's. Scheduling [11, 21], Graph Colouring [12], Partitioning Problems [22] and Telecommunication Networks [15] have also been addressed by ant algorithms. An overview of ant algorithms can be found in [17, 18].

We are not aware of any published work that applies ant algorithms to the nesting problem. The only work we have seen is a presentation at Optimization 98, University of Coimbra, Portugal (10-22 July 1998), which used an ant algorithm to place shapes. Unfortunately the conference did not publish proceedings.

In [3] the no fit polygon (NFP) is introduced (described in section 2). The NFP is also used in [1] to calculate the minimum enclosing rectangle for two irregular shapes. Albano [2] used a heuristic search method to place irregular pieces onto stock sheets. The search is based on the A* algorithm but some restrictions are introduced due to the size of the search space. For example the heuristic function is not admissible. Their evaluation function, f(n), is given by g(n) + h(n) where g(n) is the cost of the solution so far, which is a measure of the waste that cannot be used in later placements. h(n) is a heuristic measure that estimates the amount of waste in the optimal solution should the current piece be included in the placement. If it were possible to find suitable values for h(n) then an optimal solution could be found. However, this is not possible so only "good" solutions can be found. The Albano and Sapuppo approach makes use of the no fit polygon when deciding where pieces should be placed. More recently [24] also used the no fit polygon. Pieces are placed onto the stock sheet one at a time. The location of the next piece is calculated using the no fit polygon. Once the best placement has been found the piece is added to the partial solution and the next piece is placed. Oliveira experimented with three evaluation functions when considering the placement of the next piece.

This paper presents a method to produce solutions to the nesting problem. We employ an ant algorithm and compare our results with other evolutionary and meta-heuristic algorithms that have been applied to the same problem [5, 6].

2 No Fit Polygon

The no fit polygon (NFP) determines all arrangements that two arbitrary polygons may take such that the polygons touch but do not overlap. If we can find the NFP for two given polygons then we know we cannot move the polygons closer together in order to obtain a tighter packing. In order to find the placements we proceed as follows (see fig. 1). One of the polygons (P_1) remains stationary. The other polygon,

P_2, orbits around P_1, staying in contact with it but never intersecting it. Both polygons retain their original orientation. That is, they never rotate. As P_2 moves around P_1 one of its vertices (the reference point – shown as a filled circle) traces a line (this becomes the NFP).

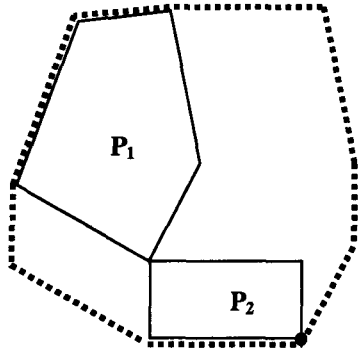

Fig. 1. No Fit Polygon

Fig. 1 shows the starting (and finishing) positions of P_1 and P_2. The NFP is shown as a dashed line. It is slightly enlarged so that it is visible. In fact, some of the edges will be identical to some of the edges of P_1 and P_2. Once we have calculated the NFP for a given pair of polygons we can place the reference point of P_2 anywhere on an edge or vertex of the NFP in the knowledge that P_2 will touch but not intersect P_1. In order to implement a NFP algorithm it is not necessary to emulate one polygon orbiting another. [13] and [14] present an algorithm that works on the assumption that (for convex polygons only) the NFP has its number of edges equal to the number of edges of P_1 and P_2. In addition, the edges of the NFP are simply copies of the edges of P_1 and P_2, suitably ordered. To build the NFP it is a matter of taking the edges of P_1 and P_2, sorting them and building the NFP using the ordered edges.

3 Evaluation

3.1 Placing the Polygons

In order to fill the bin we proceed as follows. The first polygon to be placed is chosen and becomes the stationary polygon (P_1 in the example above). The next polygon (P_2 from the above example) becomes the orbiting polygon. Using these two polygons the NFP is constructed. The reference point of P_2 is placed on each vertex of the NFP and for each position the convex hull for the two polygons is calculated. Once all placements have been considered the convex hull that has the minimum area is returned as the best packing of the two polygons. This larger polygon now becomes the stationary polygon and the next polygon is used as the orbiting polygon. This process is repeated until all polygons have been processed. As each large polygon (i.e. output from the convex hull operation) is created, its width is checked. If this exceeds

the width of the bin, then a new row within the bin is started. In this case the polygon which forced the width of the bin to be exceeded becomes the stationary polygon. That is, the large polygon built thus far forms one row and the next row is constructed using a single polygon as a starting point.

There are a number of points that need to be considered when using the above evaluation method. For reasons of space, a complete discussion of these points cannot be provided but they are detailed in [7], with the main points summarised below

A number of computational geometry algorithms have been implemented in order to effectively manipulate polygons. Executing computational geometry algorithms is computationally expensive. This makes the evaluation function the slowest part of the algorithm. In view of this a cache has been implemented which retains n (user defined) previous evaluations. If a solution (partial or complete) is in the cache then the results are retrieved from the cache rather than executing the evaluation function. To further improve the algorithm the concept of a polygon type has been introduced. This allows polygons which are identical to be classified together so that the cache can work at a higher level of abstraction when deciding if a partial or complete solution is in the cache. Once the NFP has been calculated there may be a number of optimal placements for the two polygons being evaluated (that is, the convex hulls have equal area). In this case, the convex hull to use is selected at random. However, all the optimal solutions are stored in the cache so that if the solution (partial or complete) is seen again, one of the other convex hulls could be selected.

If n polygons have been evaluated the solutions in the cache for the n^{th} polygon will be influenced by earlier evaluations. However, even though the convex hulls with the lowest area has been chosen earlier in the evaluation, this may not be the best choice once other polygons are added to the solution. Therefore, it may be beneficial to reevaluate a complete solution even if it is stored in the cache. In order to accommodate this a reevaluation parameter was introduced which, with some probability, forces a solution to be re-evaluated even if it is in the cache.

3.2. Evaluating the Placement

Our cost function is based on that used in [20]. It can be stated as follows

$$(\sum_{1}^{n}((1-(UsedRowArea/TotalRowArea))^2)*k)/n \qquad (1)$$

Where *UsedRowArea* is the total area of the polygons placed in that row
TotalRowArea is the total area of the bin occupied by that row
k if a factor simply to scale the result. We used 100 but 1 could be used
n is the number of rows in the bin

In words, we are trying to minimise the area used by each row.

This evaluation function is preferable to the more obvious method of simply measuring the bin height as, using the bin height, many solutions will map to the same evaluation function. This makes it much more difficult to effectively explore the search space. In previous work, we have compared these two different types of

evaluation and we find that the above function (1) produces better quality solutions; which is the same experience as reported in [20].

4 Ant Algorithms

4.1 Ant Algorithms and the TSP

Ant algorithms are based on the real world phenomena that ants, despite being almost blind, are able to find their way to a food source and back to their nest, using the shortest route. In [19] this phenomena is discussed by considering what happens when an ant comes across an obstacle and it has to decide the best route to take around the obstacle. Initially, there is equal probability as to which way the ant will turn in order to negotiate the obstacle. If we assume that one route around the obstacle is shorter than the alternative route then the ants taking the shorter route will arrive at a point on the other side of the obstacle before the ants which take the longer route. If we now consider other ants coming in the opposite direction, when they come across the same obstacle they are also faced with the same decision as to which way to turn. However, as ants walk they deposit a pheromone trail. The ants that have already taken the shorter route will have laid a trail on this route so ants arriving at the obstacle from the other direction are more likely to follow that route as it has a deposit of pheromone. Over a period of time, the shortest route will have high levels of pheromone so that all ants are more likely to follow this route. This form of behaviour is known autocatalytic behaviour. There is positive feedback which reinforces that behaviour so that the more ants that follow a particular route, the more desirable it becomes.

To convert this idea to a search mechanism for the Travelling Salesman Problem (TSP) there are a number of factors to consider. Below is a summary of [19].

At the start of the algorithm one ant is placed in each city. Time, t, is discrete. t(0) marks the start of the algorithm. At t+1 every ant will have moved to a new city and the parameters controlling the algorithm will have been updated. Assuming that the TSP is being represented as a fully connected graph, each edge has an *intensity of trail* on it. This represents the pheromone trail laid by the ants. Let $T_{i,j}(t)$ represent the intensity of trail edge (i,j) at time t. When an ant decides which town to move to next, it does so with a probability that is based on the distance to that city and the amount of trail intensity on the connecting edge. The distance to the next town, known as the *visibility*, n_{ij}, is defined as $1/d_{ij}$, where, d_{ij}, is the distance between cities i and j.

At each time unit *evaporation* takes place. This is to stop the intensity trails building up unbounded. The amount of evaporation, p, is a value between 0 and 1.

In order to stop ants visiting the same city in the same tour a data structure, Tabu, is maintained. This prevents ants visiting cities they have previously visited. $Tabu_k$ is defined as the list for the k^{th} ant which holds the cities that have already been visited.

After each ant tour the trail intensity on each edge is updated using the following formula

$$T_{ij}(t + n) = p \,.\, T_{ij}(t) + \Delta T_{ij} \qquad (2)$$

where

$$\Delta T_{ij}^{k} = \begin{cases} \frac{Q}{L_k} & \text{if the kth ant uses edge}(i, j) \text{ in its tour} \\ & (\text{between time } t \text{ and } t+n) \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

This represents the trail substance laid on edge (i, j) by the k^{th} ant between time t and t+n. Q is a constant and L_k is the tour length of the k^{th} ant. Finally, we define the transition probability that the k^{th} ant will move from city i to city j.

$$p_{ij}^{k}(t) = \begin{cases} \frac{[T_{ij}(t)]^{\alpha} \cdot [n_{ij}]^{\beta}}{\sum k \in allowed_k \; [T_{ik}(t)]^{\alpha} \cdot [n_{ik}]^{\beta}} & \text{if } j \in allowed_k \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

where α and β are control parameters that control the relative importance of trail versus visibility.

4.2 Ant Algorithms and the Nesting Problem

Using the TSP ant system as a model, an ant system has been developed for the nesting problem using the no fit polygon and the evaluation method described in section 3. Each polygon can be viewed as a city in the TSP and these are fully connected so that there is an edge between each polygon and every other. An ant is placed at each city (polygon) and using the trail and visibility values (formula 4) the ant decides which polygon should be visited (placed) next. Once an ant has placed all the polygons the edge trail values are updated. The edge trail values are calculated using the value returned from evaluating the nesting. This is equivalent to using the tour length in the TSP (L_k in formula 3).

Using the method described above we can use very similar formulae to those described in [19] (and shown above (2, 3, 4)), with some minor amendments. Visibility is now defined as how the polygon, just placed, fits with the piece about to be placed. For example, two rectangles of the same dimensions would fit together with no waste so the visibility would be high. Two irregular shapes, when placed together, may result in high wastage. This would result in a low visibility value. In order to calculate the visibility, the combined area of the two pieces is divided by the best placement of the two shapes (using the NFP). That is

$$n_{ij} = \text{TotalArea}_{ij} / \text{BestPlacement}_{ij} \quad (5)$$

Where i is the polygon just placed and j is the polygon about to be placed. This returns a value between 0 and 1. In order to improve the speed of the algorithm all the visibility values are calculated at the start of the algorithm and held in a cache. The

transition probability is defined as the probability of a polygon being placed next taking into account the polygons placed so far and the visibility of the next polygon. The same formula as 4, above, can be used to calculate the transition probability.

5 Testing and Results

5.1 Test Data

Two problems were used in our testing. The packing shown in fig. 2 is from [8]. The reason that this data was chosen is because it consists of convex polygons and the optimum is known. The only change made is to multiply the measurements in the original paper by a factor of two. This assists us when displaying results.

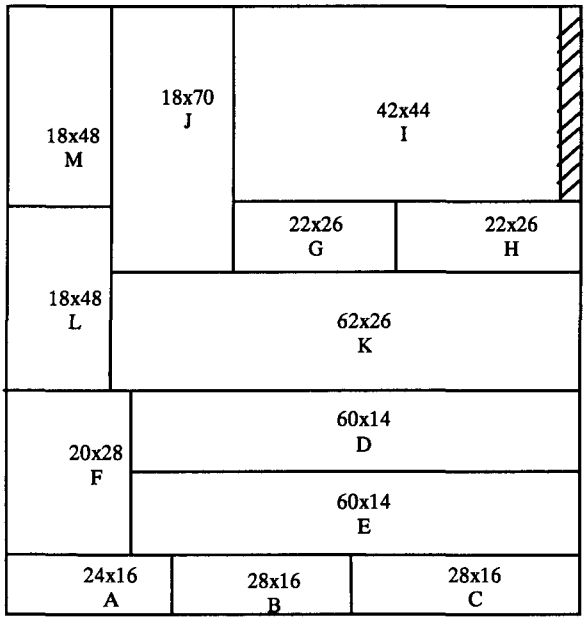

Fig. 2. Test Data 1

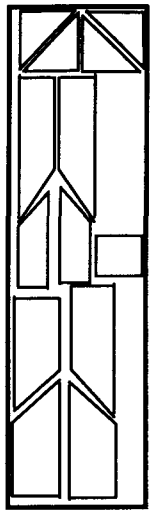

Fig. 3. Test Data 2

Our algorithms will not be able to find the optimum. The first two rows (A, B, C and D, E, F) can be constructed without problems. However, to find the optimum for the third row the polygons, would need to be presented in the order of G, H, I, J, K, L and M. The optimal solution could be built until the last polygon (M) came to be placed. At this time, due to the convex nature of the large polygon that had been built, the final polygon will not be placed in the position shown. In fact, the final polygon would be placed on a new row. Under these circumstances the total bin height would be 188 (as opposed to the optimal height of 140). Therefore, the permutation

ABCDEFGHIJKLM produces a bin height of 188. We accept that we will never achieve a value of 140 but would like to get as close as possible to this figure.

The second problem (fig. 3) is taken from the real world. A company has to cut polycarbonate shapes from larger stock sheets. The solution shown (fig. 3) is one stock sheet on a given day. Like the first problem, this is constrained by the convex properties of the algorithm. For example, the bottom four shapes cannot be packed in the way they have been as once three of the shapes have been packed, the fourth shape cannot be placed in the position shown. The same is true for the four shapes above (excluding the rectangle). The height of the stock sheet is 23940 units. Its width is 6240 units. Due to the convex properties of the algorithm it is not possible to achieve the solution shown but our main aim is to see if ant algorithms can match or beat the solutions we have already obtained.

5.2 Results

In this section we compare the ant algorithm with the results we have achieved using a genetic algorithm, tabu search and simulated annealing [5, 6]. All results are averaged over ten runs. The number of evaluations performed are equivalent in all cases. This leads to runtimes that are approximately equivalent (about 300 seconds on a Cyrix 166 processor) so that the results are compared fairly. The results show the evaluation value as well as the bin height. Although the bin height is not part of the evaluation, it is an important measure with regard to the quality of the solution and is therefore worth recording.

Initially we attempted to find suitable values for the parameters that control the ant algorithm. This aspect of advanced search remains an art, rather than a science and, in order to find suitable values we carried out several hundred runs simply setting the parameter values at random. We used these results, along with the best parameters found by Dorigo [19], to conduct more selective testing. Dorigo reported that the value of Q (the constant in formula 3) had little effect on the algorithm. We experimented with various values, {1, 10, 100, 1000}, and reached a similar conclusion. Therefore in the remainder of our tests $Q = 100$. In order to find a good value for the evaporation parameter, $p \in \{0.1, 0.5, 0.9\}$ was tested, using a trail importance, $\alpha = 1$ and a visibility importance, $\beta \in \{0, 1, 2, \ldots, 30\}$. These tests were carried out on test data 1. The results from these tests are shown in fig. 4.

All tests in fig. 4 show the highest evaluation when $\beta = 0$. This is expected as when $\beta = 0$ the search is effectively transformed into randomised greedy search with multiple starting points. All three runs also show, in the early stages, a downward trend as the visibility parameter increases. With $p = 0.1$ the evaluation values are generally higher than when p is 0.5 or 0.9. $p = 0.5$ performs better than when $p = 0.9$, at least in the early stages of the algorithm (until $\beta = 19$). In the latter stages, when the visibility is high, the graph has either flattened or is showing an upward trend for all values of p. Again, this would be expected as having too high a visibility starts returning the algorithm to a greedy search. This is due to the effect of the intensity trail becoming diminished. $p = 0.5$ as a good value agrees with the results in [19] in

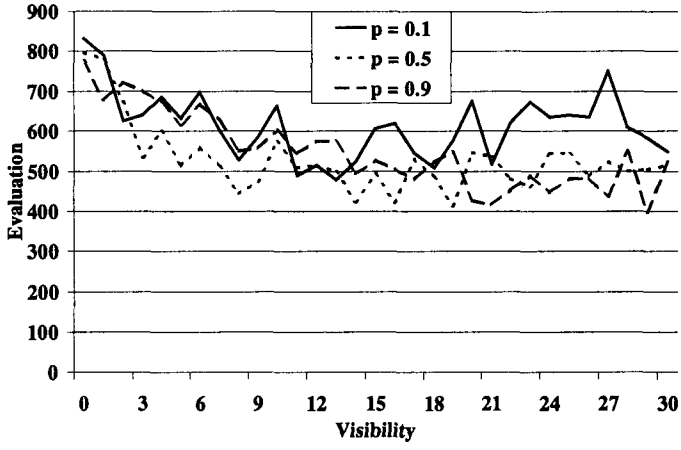

Fig. 4. Test Data 1, $\alpha = 1$, p = {0.1, 0.5, 0.9}, β = {0, 1, ..., 30}

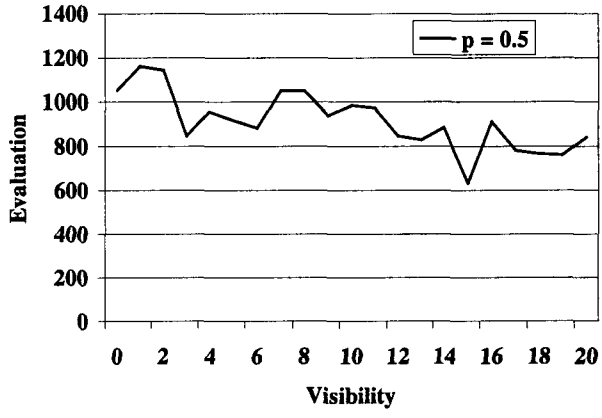

Fig. 5. Test Data 1, $\alpha = 5$, p = 0.5, β = {0, 1, ..., 20}

which it was reported that this was the best value found for p. In [19] the best value for α was found to be 1 (which is the value used above). In order to see if our algorithm agrees with this a test was carried that that set $p = 0.5$, $\alpha = 5$ and β = {0,1,..., 20}. Fig. 5 shows that this set of parameters produces worse results than when $\alpha = 1$. None of the tests produce an evaluation below 600, which was consistently done when $\alpha = 1$ (fig. 4)

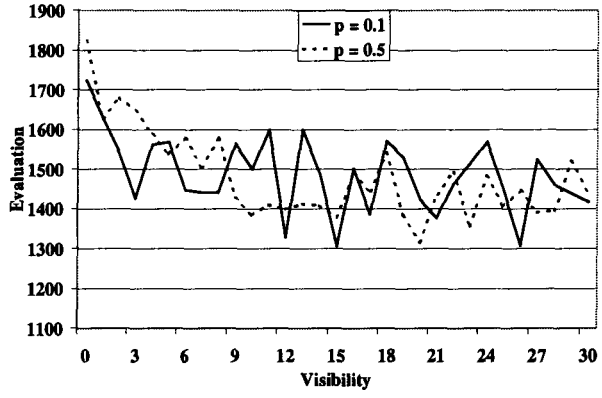

Fig. 6. Test Data 2, $\alpha = 1$, $p = \{0.1, 0.5\}$, $\beta = \{0, 1, \ldots, 30\}$

Having established a good parameter set we used test data 2 to confirm these values. Fig. 6 shows two runs that compare the effect of p (evaporation) on test data 2. In fact the two runs mimic each other closely but it is interesting to note that the lowest evaluation for $p = 0.5$ is when visibility is around 20. This matches the result from the first set of test data. This test appears to confirm that $p = 0.5$ is a good choice and we used this in the remaining tests.

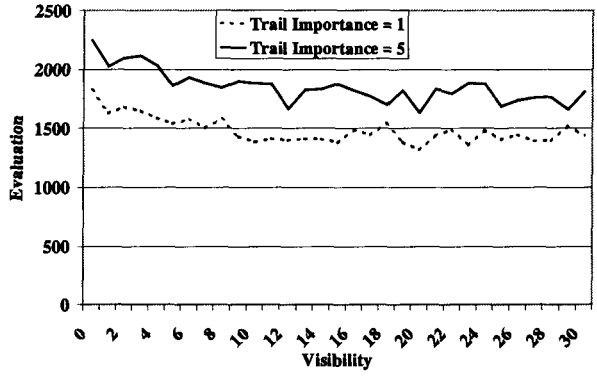

Fig. 7. Test Data 2, $\alpha = \{1, 5\}$, $p = 0.5$, $\beta = \{0, 1, \ldots, 30\}$

Fig. 7 shows the effect of trail importance, $\alpha = \{1, 5\}$ using test data 2. It shows that a higher value of α leads to inferior solutions. Again, this confirms the results from the first set of test data. In summary, the best results were achieved on both sets of test data when $\alpha = 1$, $p = 0.5$, $\beta = 20$. The best results we have achieved using other search algorithms [5, 6] and the modified Falkenauer function are shown in table 1. We also show the best results from the ant algorithm using the parameters just described.

Table 1. Results using the same test data with different search algorithms

	Test Data 1		Test Data 2	
Search Method	**Evaluation**	**Bin Height**	**Evaluation**	**Bin Height**
Genetic Algorithm	516.41	179.00	1377.99	30478.50
Tabu Search	323.77	165.80	1006.75	27566.40
Simulated Annealing	397.01	170.80	1661.77	33052.20
Ant Algorithm	412.97	173.20	1316.85	29316.60

Tabu search finds the best quality solutions of the algorithms implemented but the ant algorithm compares favourably with simulated annealing, performing significantly better with test data 2. The ant algorithm also out performs the other population based search method (genetic algorithm).

6 Summary

This is the first time that ant algorithms have been applied to the nesting problem and the results are encouraging. Ant algorithms out perform genetic algorithms and provide a viable alternative to simulated annealing, although more work is required for it to compete with tabu search. We will continue using ant algorithms in our future research when we will tackle more complex problems as well as relaxing some of the constraints we outlined above. The parameters to the ant algorithm are critical and we plan to carry out more work in this area. In addition we plan to look at hybridsation to combine ant algorithms with other search techniques in order to produce even better solutions.

References

1. Adamowicz, M., Albano, A. 1976. Nesting Two-Dimensional Shapes in Rectangular Modules. Computer Aided Design, 8, 27-33
2. Albano, A., Sappupo, G. 1980. Optimal Allocation of Two-Dimensional Irregular Shapes Using Heuristic Search Methods. IEEE Trans. Syst., Man and Cybernetics, SMC-10, pp 242-248
3. Art, R.C., 1966. An Approach to the Two-Dimensional Irregular Cutting Stock Problem. Technical Report 36.008, IBM Cambridge Centre.
4. Bullnheimer B., R.F. Hartl and C. Strauss (1999). An Improved Ant system Algorithm for the Vehicle Routing Problem. The Sixth Viennese workshop on Optimal Control, Dynamic Games, Nonlinear Dynamics and Adaptive Systems, Vienna (Austria), May 21-23, 1997, to appear in: Annals of Operations Research (Dawid, Feichtinger and Hartl (eds.): Nonlinear Economic Dynamics and Control, 1999
5. Burke, E., K., Kendall, G. 1999. Applying Evolutionary Algorithms the No Fit Polygon to the Nesting Problem. Accepted for The 1999 International Conference on Artificial Intelligence (IC-AI '99), Monte Carlo Resort, Las Vegas, Nevada, USA, 28 June - 1 July 1999

6. Burke, E., K., Kendall, G. 1999. Applying Simulated Annealing and the No Fit Polygon to the Nesting Problem. Accepted for WMC (World Manufacturing Congress), September 1999, Durham, UK
7. Burke, E., K., Kendall, G. 1999. Evaluation of Two Dimensional Bin Packing using the No Fit Polygon. Submitted to CIE26 (Computers and Industrial Engineering), December, 1999, Melbourne, Australia
8. Christofides, N., Whitlock, C. 1976. An Algorithm for Two_Dimensional Cutting Problems. Operations Research, 25, 30-44
9. Colorni A., M. Dorigo & V. Maniezzo (1992). An Investigation of Some Properties of an Ant Algorithm. Proceedings of the Parallel Problem Solving from Nature Conference (PPSN 92), Brussels, Belgium, R.Männer and B.Manderick (Eds.), Elsevier Publishing, 509-520
10. Colorni A., M. Dorigo & V. Maniezzo (1992). Distributed Optimization by Ant Colonies. Proceedings of the First European Conference on Artificial Life, Paris, France, F.Varela and P.Bourgine (Eds.), Elsevier Publishing, 134-142
11. Colorni A., M. Dorigo, V. Maniezzo and M. Trubian (1994). Ant system for Job-shop Scheduling. JORBEL - Belgian Journal of Operations Research, Statistics and Computer Science, 34(1):39-53. [116]
12. Costa D. and A. Hertz (1997). Ants Can Colour Graphs. Journal of the Operational Research Society, 48, 295-305
13. Cunninghame-Green, R. 1989. Geometry, Shoemaking and the Milk Tray Problem. New Scientist, 12, August 1989, 1677, pp 50-53
14. Cunninghame-Green, R., Davis, L.S. 1992. Cut Out Waste! O.R. Insight, Vol 5, iss 3, pp 4-7
15. Di Caro G. & Dorigo M. (1998). AntNet: Distributed Stigmergetic Control for Communications Networks. Journal of Artificial Intelligence Research (JAIR), 9:317-365
16. Dorigo M. (1992). Optimization, Learning and Natural Algorithms. Ph.D.Thesis (in italian), Politecnico di Milano, Italy, in Italian
17. Dorigo M. and G. Di Caro (1999). The Ant Colony Optimization Meta-Heuristic. In D. Corne, M. Dorigo and F. Glover, editors, New Ideas in Optimization, McGraw-Hill, in press
18. Dorigo M., G. Di Caro & L. M. Gambardella (1999). Ant Algorithms for Discrete Optimization. Artificial Life, 5(2), in press
19. Dorigo M., V. Maniezzo & A. Colorni (1996). The Ant System: Optimization by a Colony of Cooperating Agents. IEEE Transactions on Systems, Man, and Cybernetics-Part B, 26(1):29-41
20. Falkenauer, E. 1998. Genetic Algorithms and Grouping Problems. John Wiley and Sons
21. Forsyth P. and A. Wren (1997). An Ant System for Bus Driver Scheduling. Presented at the 7th International Workshop on Computer-Aided Scheduling of Public Transport, Boston, August 1997
22. Kuntz P., P. Layzell and D. Snyers (1997). A Colony of Ant-like Agents for Partitioning in VLSI Technology. Proceedings of the Fourth European Conference on Artificial Life, P. Husbands and I. Harvey, (Eds.), 417-424, MIT Press
23. Maniezzo V. and A. Colorni (1999). The Ant System Applied to the Quadratic Assignment Problem. IEEE Transactions on Knowledge and Data Engineering, to appear
24. Oliveira, J.F., Gomes, A.M., Ferreira, S. 1998. TOPOS A new constructive algorithm for nesting problems. Accepted for ORSpektrum
25. Stützle T. and M. Dorigo (1999). ACO Algorithms for the Traveling Salesman Problem. In K. Miettinen, M. Makela, P. Neittaanmaki, J. Periaux, editors, Evolutionary Algorithms in Engineering and Computer Science, Wiley, 1999