Optimization and Simulation: Sequential Packing of Flexible Objects Using Evolutionary Algorithms

H. Behnke, M. Kolonko, U. Mertins, S. Schnitter

Institut für Mathematik, Technical University Clausthal, D-38678
Clausthal-Zellerfeld, Germany

Abstract. We want to fill a given two-dimensional closed contour as accurately as possible with a fixed number of identical, flexible objects. These objects have to be packed sequentially. They adapt themselves to the surface they are packed on, but their deformation can only be simulated. This type of problem is the two-dimensional cross-section of manufacturing processes where soft material is wound onto a mandrel. We formulate this as a problem of dynamic programming with simulated law of motion. It allows an evolutionary algorithm approach that successfully produces approximate solutions in a non-sequential fashion.

KEYWORDS : evolutionary algorithms, optimization and simulation, packing problems, dynamic programming

1 Introduction

We consider the problem of filling a given two-dimensional container with a fixed number N of objects of identical type. The objects are smooth and adapt themselves to the surface their are packed on. The packing has to start at the bottom of the container and the objects have to be placed sequentially, see Fig. 1. The aim is to fill the container, which is given as a closed two dimensional contour, as accurately as possible. The particular additional constraint here is that the exact behaviour of the objects, their deformation function, is not available in a closed analytical form. Instead, it has to be simulated based on assumptions about properties of the material.

Hence, we have to solve an optimization problem with a simulated target function. We use a dynamic programming framework to give a precise definition of our problem. The dynamics of the model depend on the simulated deformation of the objects, which rules out most solution methods of dynamic programming like backward induction. Instead, we suggest an approximate procedure that consists of a genetic algorithm framework optimizing the relative position of the objects using the results of the simulation as fitness.

The main point here is to find a coding of the feasible solutions and of the cost function (i.e. the deviation of the final layout from the target contour) such

that genetic algorithm techniques can be applied efficiently. With our approach standard genetic operators yield feasible solutions without any problem specific adaptations. Also, these operations behave 'locally' enough to produce offspring that inherits properties of their parents leading to astonishingly good solutions in very short time for a number of real world problems.

The problem described is the two-dimensional cross-section of certain three-dimensional packing problems that arise in the manufacturing of rotational symmetrical bodies like tubes, rings or tanks from fibre-reinforced plastic. Here, a continuous fibre bundle impregnated with a polymere is wound around a rotating mandrel, see e.g. [6] for technical details. The shape of the mandrel determines the inner contour of the workpiece, the outer contour is formed by the layers of the composite fibre bundles after hardening or cooling. So the winding robot has to put more layers at places where the side of the workpiece is to be stronger and less where there is to be a groove. Generally the aim is to control the robot in such a way, that a predefined outer contour is filled as accurately as possible with a given number of windings. As the spooled material is soft and sticky, it will adapt to the surface or is even pressed on to it to prevent air inclusion.

Due to the rotational symmetry, we can restrict the problem to (one half of) the 2-dimensional cross-sections of the workpiece and of the rope to be spooled, see Fig. 1. Here, (a) shows the cross-section of a ring. The cross-section of the rope when wound onto an even surface will look like Fig. 1 (b), this will be referred to as an *object*. Fig. 1(c) shows one half of the ring (the *target shape*) with four objects placed, i.e. with four rounds of rope already applied. The target shape consists of the lower *starting* and the upper *target contour* . We assume that the axis of the mandrel is parallel to the $x-$axis as indicated in the Figue. 1. We assume further that the number of objects is chosen appropriately (i.e. the area of the target shape is N times the area of the objects). Note that we neglect possible problems in the original 3-dimensional problem that may occur when 'changing the lane' during winding.

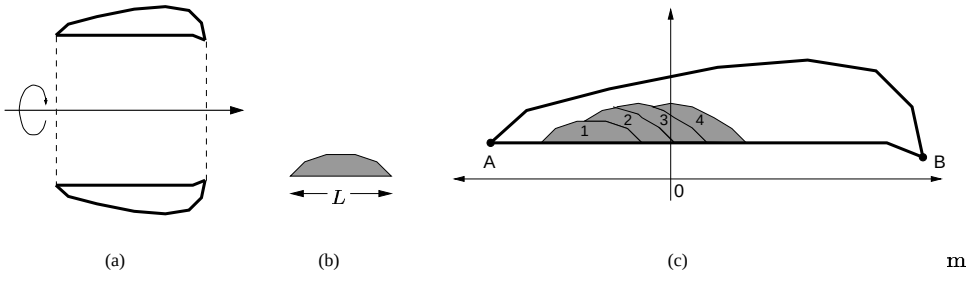

Fig. 1. Target shape and objects.

Classical problems of packing and cutting, see e.g. [2] mostly consider objects of fixed shape, an exception is [4]. Methods for solving include dynamic programming, integer programming, problem specific algorithms and several heuristics, see [2]. In [3], simulated annealing is applied to a simple packing problem.

The paper is organized as follows. In Section 2, we give a formulation of the problem in a dynamic programming set-up and define the solution. In Section 3 we sketch how to simulate the deformation of a single object and how to evaluate the cost of a complete solution approximately. In Section 4 we then present a genetic algorithm that produces good solutions in a non-sequential way. In the final Section 5 we report on some practical experiences with an implementation of the algorithm.

2 The Mathematical Model

Recall that a dynamic programming model (see [1]) consists of states s , actions a , a transition function $T(s,a)$ that maps state-action-pairs into new states, a cost structure with local costs $c(s,a)$ connected to each transition and a terminal cost function $V_0(s)$ for the final state.

Our packing problem fits into this framework if we regard the decision where to place the next object as an action. Then $N :=$ no. of objects to be placed equals the number of optimization stages. The state of the packing has to contain all information relevant for the further placement, in particular, the present upper contour formed by the objects already placed. This will be made precise in the following sub-sections.

2.1 Feasible Contours

We describe the contour by two functions for its upper and lower half. The set of possible contour functions is

$$\mathfrak{C} \;:=\; \{C : [\alpha,\beta] \to \mathbb{R}_+ \mid C \text{ bounded, continuous and piecewise differentiable}\}$$

with $\alpha < \beta, \alpha, \beta \in \mathbb{R}$. We assume that the given *starting contour* C_0 and the *target contour* $\overline{C}$ belong to $\mathfrak{C}$ with $C_0(x) < \overline{C}(x)$ for $x \in (\alpha,\beta)$ and $y_\alpha := C_0(\alpha) = \overline{C}(\alpha)$, $y_\beta := C_0(\beta) = \overline{C}(\beta)$, see Fig. 2. During the placement, the starting contour C_0 is changed to contours $C_1 \le C_2 \le \ldots \le C_N \in \mathfrak{C}$ by adding on the objects.

We characterize each point on a contour $C \in \mathfrak{C}$ by its distance from the left endpoint of C measured in arc length, '$C-$distance' for short. To do so, let

$$\Lambda_C(x) \;:=\; \int_\alpha^x \sqrt{1 + C'(z)^2}\, dz, \qquad x \in [\alpha,\beta],\ C \in \mathfrak{C}$$

denote the $C-$distance of the point $(x, C(x))$ from the left starting point $(\alpha, C(\alpha))$ measured in arc length along C.

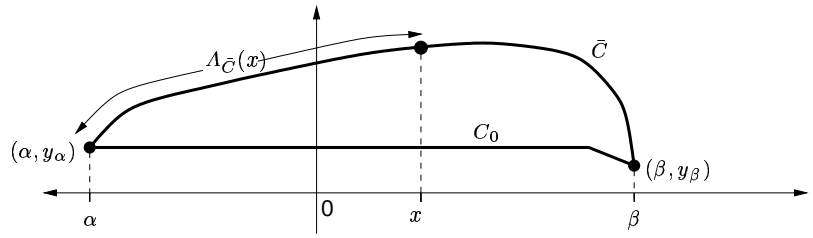

Fig. 2. A target shape with starting contour C_0 and target contour $\bar{C}$, point $(x, \bar{C}(x))$ has distance $\Lambda_{\bar{C}}(x)$ from the left endpoint.

2.2 The State Space

Next we define the state for the dynamic programming model described above.

We assume that the winding robot moves from one side of the target shape to the other and back again, i.e. it produces complete layers of material and may not change its direction freely. This restricts the space of possible solutions but simplifies matters considerably.

For a given starting contour $C_0 \in \mathfrak{C}$ with $y_\alpha = C_0(\alpha), y_\beta = C_0(\beta)$ we define the *state space*

$$S := \{(C,t,d) \mid C \in \mathfrak{C},\ C(\alpha) = y_\alpha, C(\beta) = y_\beta,\ C(x) \geq C_0(x), \alpha < x < \beta, \\ \text{and } 0 < t < \Lambda_C(\beta),\ d \in \{-1,+1\}\ \}.$$

Here, the state $s = (C,t,d)$ indicates that the objects placed so far form an upper contour C and that the location of the last object placed is represented on C by t (in $C-$distance). $d = +1$ indicates that the present layer runs from left to right, $d = -1$ indicates the opposite direction. Note that the last object was *not* placed *on* C, point t only represents that location, see details below. In the starting state (C_0,t,d), t indicates an arbitrary reference point for the first placement.

2.3 Action and Placement

As described above, the placement of an object has to be formulated as action in the dynamic programming model.

We assume that the objects all have identical shape before placement (i.e. the rope of material has a constant cross-section) and that the length L of their lower contour (*base line*) and the volume of the cross-section when placed on a surface are constant, see Fig. 1 and 3. As reference point for the placement (*basepoint*) we take the middle of this base line, it will be marked by a black triangle in the figures below.

Let $\lambda > 0$ be the maximal $C-$distance between two successive objects that can be managed by the winding machine. Then, *action* $a \in [0,1]$ applied to state

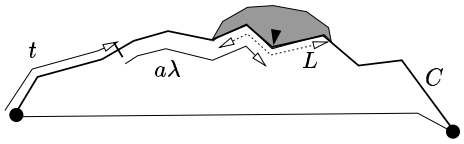

Fig. 3. An object is placed on C with its basepoint at $C-$distance $t + a\lambda$.

(C, t, d) means to place the next object with its basepoint at a $C-$distance of $da\lambda$ from t, i.e. at $C-$distance $t + da\lambda$ from the left endpoint, see Fig. 3. Near the left and right border of the target shape this may lead to infeasible positions $\notin [0, \Lambda_C(\beta)]$. In this case the direction d is changed to $-d$ and the object is placed at the next feasible position, hence the exact position of the new object is at a $C-$distance $\tau(s, a)$ where

$$\tau(s,a) = \tau((C,t,d),a) \;:=\; \begin{cases} t + da\lambda & \text{if } L/2 \le t + da\lambda \le \Lambda_C(\beta) - L/2 \\ L/2 & \text{if } t + da\lambda < L/2 \\ \Lambda_C(\beta) - L/2 & \text{if } t + da\lambda > \Lambda_C(\beta) - L/2 \end{cases} \tag{1}$$

and the direction after placement is given by

$$\delta(s,a) = \delta((C,t,d),a) \;:=\; \begin{cases} d & \text{if} \quad L/2 \le t + da\lambda \le \Lambda_C(\beta) - L/2 \\ -d & \text{else} \end{cases} .$$

Note that this notion of feasible placement does not prevent objects to be placed completely outside of the target shape, see Fig. 6 below for an example.

2.4 The State Transition

The state transition function has to describe the contour after a placement of a new object. Let the local deformation function for an object

$$l \mapsto \gamma(C, r, l)$$

be a bounded, continuous and piecewise differentiable function that gives the height of the object l units (in $C-$distance) from its left endpoint if the object is placed on C at a $C-$distance r, see Fig. 4. In the next Section, we shall give an idea how to simulate γ.

The new contour after placing an object in state $s = (C, t, d) \in S$ with a relative offset of $a \in [0, 1]$ is now given by

$$C_{s,a}(x) \;:=\; \begin{cases} C(x) & \text{for } \Lambda_C(x) \le \tau(s,a) - L/2 \\ C(x) + \gamma(C, \tau(s,a), l) & \text{for } \Lambda_C(x) = \tau(s,a) - L/2 + l,\ 0 \le l \le L \\ C(x) & \text{for } \Lambda_C(x) \ge \tau(s,a) + L/2 \end{cases} \tag{2}$$

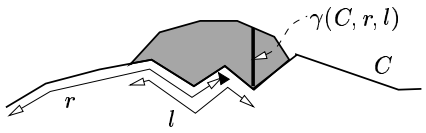

Fig. 4. The object is placed at $C-$distance r. Its height l units from its left endpoint is given by $\gamma(C,r,l)$.

for $\alpha \leq x \leq \beta$. Note that the region outside $\tau(s,a) \pm L/2$ refers to that part of the contour that is left untouched by the newly placed object. Obviously, $C_{s,a}$ belongs to $\mathfrak{C}$ for all s, a.

We finally have to determine how the location $\tau(s,a)$ at which the last object was placed on C is represented on the new contour $C_{s,a}$ as reference point for the next placement. We use the projection of the basepoint of the last object onto the new contour where the projection is perpendicular to the line connecting left and right endpoint of the object, see Fig. 5. From simple geometric considerations we

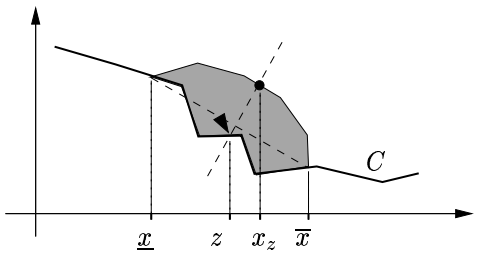

Fig. 5. The reference point for the next offset is marked by a black dot. It is a projection of the basepoint (black triangle) of the last placement onto the new surface.

obtain that the $x-$coordinate $x_z = x_z(s,a)$ of this point is the smallest solution y of

$$(y-z)(\overline{x}-\underline{x})/(C(\underline{x})-C(\overline{x}))+C(z) \;=\; C_{s,a}(y). \tag{3}$$

where $\underline{x}, \overline{x}$ and z are as in Fig. 5. Now the complete *transition function* for the dynamic programming model with local deformation functions γ can be defined from (2) and (3) as

$$T(s,a) \;=\; T((C,t,d),a) \;:=\; (C_{s,a}, \Lambda_{C_{s,a}}(x_z(s,a)), \delta(s,a)). \tag{4}$$

2.5 The Cost Structure

We have no local costs in our model but a *terminal cost function*

$$V_0(s) \;=\; V_0((C,t,d)) \;:=\; \int_\alpha^\beta \mid C(x)-\overline{C}(x) \mid \, dx. \tag{5}$$

which gives the deviation from the target contour in the final state.

Now the dynamic programming problem is completely determined. A *solution* to this problem is a sequence of offsets

$$a_0, \ldots, a_{N-1} \in [0,1]^N$$

for the consecutive placement of the N objects on the starting contour C_0. Note that *any* sequence from $[0,1]^N$ constitutes a feasible solution. Its costs are given by

$$\begin{aligned} V_N(s_0, a_0, \ldots, a_{N-1}) &= V_0(T(\ldots(T(T(s_0, a_0), a_1), \ldots, a_{N-1}) \\ &= \int_\alpha^\beta \mid C_N(x) - \overline{C}(x) \mid \, dx \end{aligned} \tag{6}$$

where C_N is the final upper contour after applying $a_0, \ldots, a_{N-1}$ to s_0. An *optimal solution* for starting state $s_0 = (C_0, t, d)$ is a sequence $a_0^*, \ldots, a_{N-1}^* \in [0,1]^N$ that minimizes (6) over $[0,1]^N$.

3 Simulating the local deformation

We only sketch the simulation very briefly as it is not in the focus of the present paper.

In our experiments it turned out to be sufficiently accurate to use piecewise linear functions for the contours as well as for the objects, and hence for the deformation function γ. This greatly simplifies the calculation described above and also the representation of a contour on the computer.

Let $\gamma_0 : [0, L] \to \mathbb{R}_+$ denote the piecewise linear upper contour of the object when placed on an even surface. We add γ_0 to the present contour at the location given by $\tau(s, a)$. Let Γ denote that part of the new contour $C_{s,a}$ that is formed by γ_0.

We use two different levels of simulation to improve Γ. The first level assumes that the surface of an object will always tend to form a section of a circle. From simple geometric considerations we can determine a circle that runs through the two endpoints of the object placed at $\tau(s, a)$ ($\underline{x}, \overline{x}$ in Fig. 5) and encloses the exact volume of the object. We then approximate the relevant section of the circle by a polygone which is slighly changed afterwards to enclose the correct volume. This yields an improved deformation function γ_1.

This is a very fast procedure that works well as long as the surface C is not too ragged. Otherwise the circle may intersect with C. This has to be detected and then the second more detailed level of simulation has to be called. Here, the polygone describing Γ is optimized to yield smallest length and minimal distortion with the correct enclosed volume. This is done using a standard library for non-linear optimization. The result is a simulated contour $C_{s,a}$ which takes into account more physical properties of the material.

For a given starting state s_0 and a sequence of off-sets $a_0, \ldots, a_{N-1} \in [0,1]$ we evaluate the cost function $V_N(s_0, a_0, \ldots, a_{N-1})$ as given in (6) successively by N calls of the simulation.

4 An Evolutionary Algorithm

We shall now describe an evolutionary heuristic algorithm that yielded excellent solutions in our experiments. Let us briefly collect the main ingredients of genetic algorithms (see e.g. [5]). A starting population of solutions ('individuals') is created at random. It is subject to random genetic operations that produce off-spring solutions. Standard operators are crossover of two randomly selected solutions and random mutation of a solution. Repeated application of these operators enlarges the population which is then reduced by a selection mechanism to its former size. This cycle (' generation') is repeated for a certain number of times. The selection prefers solutions with low costs. Thus the members of the successive populations tend to become good solutions of the optimization problem. There are many different ways to select solutions for the operations and to take into account their 'fitness', i.e. the cost of the solution.

In the preceding Sections we have formulated our problem such that the structure of a solution $a_0, \ldots, a_{N-1}$ is extremely simple : it lies in the unit rectangle $[0,1]^N$ and each vector in this space represents a feasible placement of N objects. Therefore we may use simple standard genetic operators without any problem specific mending operation. The fitness of a solution $a_0, \ldots, a_{N-1}$ is measured by the cost function $V_N(s_0, a_0, \ldots, a_{N-1})$ as defined in (6). Note that we minimize fitness.

For *crossover* we use a one-point or uniform operator. The one-point crossover takes two randomly selected solutions $(a_0, \ldots, a_{N-1})$ and $(a'_0, \ldots, a'_{N-1})$ as parents, selects a random position $l \in \{0, \ldots, N-1\}$ and returns the solution $(a_0, \ldots, a_l, a'_{l+1}, \ldots, a'_{N-1})$. In terms of controlling the winding machine this means to use off-sets from the first solution for the first l rounds and then to follow the second solution. Uniform crossover selects each position in the resulting solution independently from one of the parents.

Note that we restricted changes of the direction d to the lateral borders where it is calculated by the cost function. Including changes of direction into the solution would cause problems with crossover operators, as the impact of a change in direction depends heavily on its *absolute* location within a placement which in general will be changed during a crossover. Then the offspring individuum will be quite different from its parents which would make the algorithm search rather arbitrarily in the solution space.

We apply *mutation* to each new offspring solution created by crossover. Mutation operators randomly change single offsets of a solution. Again, even a small change of a single offset may lead to a completely different solution, as all subsequent objects are shifted. We therefore use a 'local' mutation that also changes the next offset to neutralize the widerange effect of the mutated offset. The procedure is as follows: select an index $0 \leq i \leq N-2$ randomly with a bias towards higher values. In the solution to be mutated, let $\sigma = a_i + a_{i+1}$. Select a new value a'_i uniformly distributed over

$$\Big[\max\{1,\sigma\} - 1,\ \min\{1,\sigma\}\Big]$$

and put $a'_{i+1} = \sigma - a'_i$. Then $a'_i, a'_{i+1} \in [0,1]$ and $a'_i + a'_{i+1} = \sigma$, i.e. the position of objects $i+2, i+3, \ldots$ on the their contour is unchanged. Note however, that the contours themselves will have changed due to the moving of objects i and $i+1$. This may be illustrated by looking at the $a_0, \ldots, a_{N-1}$ as distances of beads on a string. Mutation then moves exactly one of the beads and leaves the others unchanged.

Let M be the population size. Crossover and mutation are repeated for a fixed number M' of times enlarging the population to $M + M'$ elements . The selection mechanism then takes M solutions from the enlarged population to form the next population. These may be the M solutions with lowest (simulated) costs V_N , or they may be drawn with a probability density proportional to the V_N-values or proportional to their rank with respect to these cost values. Again these are standard selection procedures.

We provided the system with a graphical user interface , that allows to display the evolution of the population with their cost values. The evolution may be stopped and the individuals may be inspected graphically as shown in Fig. 6 and 7 below.

5 Experimental Results

We applied our model to several contours supplied by our industrial partner. Typically there were about 60 objects to be placed. Target contour and the cross-section of the object were given as polygones, similar to Fig. 1.

A starting population was created randomly, good results were obtained with individuals that had constant offsets $a_0, \ldots, a_{N-1}$ with different random values for different individuals. As starting state $(C_0, t, -1)$ we used the lower contour C_0 with its middle point as reference point for the first object.

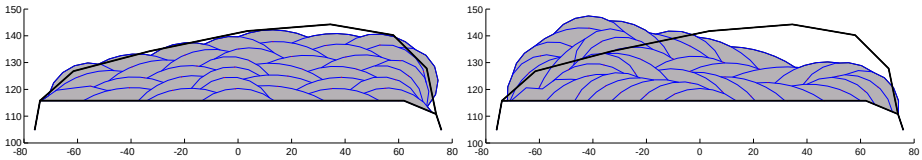

Fig. 6. Two individuals from a starting population.

Fig. 6 shows two individuals from a starting population. The one on the left was produced with econstant offsets, the other with random offsets. Note that these are obtained by placing a random vector of offsets into the contour using the simulation of the cost function as described in Section 3. The much improved results in Fig. 7 show the two best individuals obtained after about 300 generations of the genetic algorithm which took about 30 sec of cpu time on a 650 MHz Pentium.

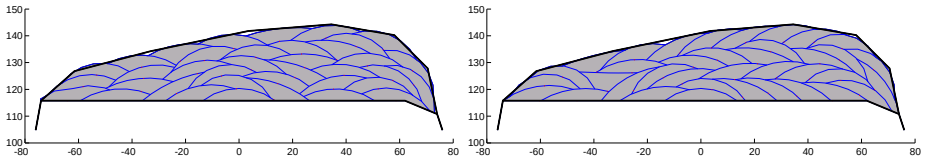

Fig. 7. Two individuals from a later population.

Results from this system were fed into a real winding machine. The contours obtained in reality were in very good accordance with our simulated contours. This shows that our simple model of deformation reflects reality with sufficient accuracy.

6 Conclusions

We modelled a complex technical process of placing flexible objects into a given contour as a deterministic dynamic programming problem. The classical sequential solution methods are not applicable as the main ingredient of the model, the state transition function is not known. It can only be approximated by a separate local optimization that simulates the physical properties of the flexible material.

We therefore took a non-sequential approach, that optimizes all placement decisions at once with a genetic algorithm. The formulation of the model allowed to produce feasible solutions with standard genetic operators without the need of problem specific mending operations.

In real world applications with about 60 objects this algorithm produced astonishingly good results in just a few seconds.

References

1. Bertsekas, Dimitri P. *Dynamic programming : deterministic and stochastic models* Englewood Cliffs, N. J.: Prentice-Hall, 1987.
2. Dowsland, Kathryn A. and William B. Dowsland *Packing Problems* Europ. J. Operational Research, 56 (1992), 2-14.
3. Dowsland, Kathryn A. *Some Experiments with Simulated Annealing Techniques for Packing Problems* Europ. J. Operational Research, 68 (1993), 389-399.
4. Scheithauer, G.*The Solution of Packing Problems with Pieces of Variable Length and Additional Allocation Constraints* Optimization, 34 (1995), 81-86.
5. Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolutionary Programs.* Springer Verlag, 1992.
6. T. J. Reinhart. *Composites.* Engineered materials handbook, ASM International / Handbook Committee, Metals Park, Ohio : ASM International, 1988.