Optimization for Part Nesting and Layout Using a Distributed SPMD Architecture

Joseph P. Wetstein, PE, MSEE[1], *Allon Guez, Ph.D.*[2]

Electrical and Computer Engineering Department,
Drexel University, Philadelphia, PA

ABSTRACT

There is a need to perform highly complex real-time optimization during manufacturing to solve the problem of maximizing the yield of production while minimizing the cost of materials. Although traditional linear and non-linear programming approaches have dealt with these problems, some problems are too complex and result in a combinatorial explosion for those methods. We are interested in developing a solution to the leather nesting task, an NP-hard problem. We feel the best solution to a large and complex problem such as this is to attempt to model the behavior that is already evident in nature. There are many examples in nature providing an optimal packing or nesting of items. Our algorithm will offer a better solution to problems of this class by presenting a controlled trade-off between computation time and regional optimality of the solution. Our goal is to develop an algorithmic scheme that will yield the best possible solution to this combinatorial optimization problem using a hierarchical approach to object determination and placing, an intelligent controller, and parallel executed simple local decision rules based on nearest neighbor parameters which achieve results that are competitive with or better than the results of a human nester, and to aid in manufacturing. In our SPMD model, each object to be placed will be a separate process in a parallel computing architecture, and will consider the parameters (location, orientation) of its nearest neighbors (other parts in the vicinity) to make translocation and re-orientation decisions. We intend to show that the fusion of existing algorithms and the establishment of uniform and 'simple' local decision rules will more quickly yield to a better optimum for the entire system. The algorithms developed in this study are applicable to a large class of optimization problems.

---

[1] Joseph P. Wetstein received his Master's Degree in Electrical Engineering Systems in 1994, and his B.S. in Electrical and Computer Engineering in 1991, from Drexel University in Philadelphia, Pennsylvania. He is a registered Professional Engineer (PE) in the State of Pennsylvania, and is currently a Candidate for the Ph.D. degree in Electrical Engineering at Drexel. He is a member of the NSPE and IEEE.

[2] Allon Guez is a Professor and Director of the Robotics and Automation Laboratory, Department of Electrical Engineering, Drexel University. Dr Guez received a B.Sc. in E.E. at the Technion, Israel, 1978, an M.S.E.E. in 1980, a Ph.D. in 1983 from the Univ. of Florida, and an MBA in 1996 from Drexel University.

* *The authors can be reached at: jpw@coe.drexel.edu*

I. INTRODUCTION

Optimization in Manufacturing

There is a need to perform highly complex real-time optimization during manufacturing, and there has been an on-going effort to use computers, robotics, and automation techniques and bring these devices down to the factory floor. Factory owners have long been interested in automation techniques where repetitive action can lead to human error, or in locations where the use of human labor is not advisable due to the hazards of the particular production environment, such as in radiation, heat, or other dangerous conditions.

Industry owners are starting to see advanced computer technology as a money saving device that can increase their profits by reducing the machine overhead in production and reduce the amount of use and waste of the raw materials required to initiate the manufacturing process. Factories specializing in metal cutting, fabric cutting, and leather cutting all face similar manufacturing problems in optimization, even when there are differences in the constraints presented. There is a direct budgetary implication to the ability, or lack of ability, to maximize the output (items produced) of a manufacturing operation while minimizing the cost of raw materials and the time of complex machine processing and human intervention. Traditional linear and non-linear programming approaches have dealt with problems such as these, and attempt to solve the problem of maximizing of the yield while minimizing the cost of production. A variety of algorithms ranging from neural network learning, traveling salesman, bin packing, operations research methods, and heuristics have all been applied to this problem. However, even with the advent of very fast computers, parallel architecture, and advances in algorithms, robotics, and vision, there still is a void to be filled in this area. Even when solutions or partial solutions exist in one industry, they seldom find their way into another application areas.

Motivation for Optimization

Packing and Nesting, or Stock Cutting, problems appear in a variety of manufacturing industries. The particular class of nesting problems that we have chosen fall into the description of "Stock Cutting," specifically, the cutting of patterns, or stencils, from the body of what is essentially a 2-dimensional surface, in order to manufacture some commodity. What we are *nesting* or *packing* are the stencils on the face of the material to be subdivided. In a nesting problem, we are given a piece of fabric or an animal hide, the so-called surface, and a set of geometric patterns (stencils) that define parts of a piece of clothing or, in the case of leather, possibly a piece of furniture, as a prelude to material cutting. We are focusing our research on leather, and shall describe the difficulties and challenges of leather with respect to other

materials. The task is to place the stencils on the surface, with resulting the placement called a marker, such as to minimize the amount of wasted material. Although the amount of automation in the field currently is minimal, we hope to use our research developments and algorithms to build an interactive system to aid the manufacturer in producing real-time solutions to the cutting task. These problems are similar to bin packing problems studied in transshipping, and we can benefit from the research done in that area.

Although rectangle and sphere packing have been well studied and industry-available software for performing packing and nesting exists (Reda, 1994), and (Haims, 1970), there is no uniform algorithm for performing the same task when the objects in question are not geometrically-friendly. (Raoot, 1991) reviews past approaches to layout planning, a closely-related discipline, that includes various schematic techniques, both traditional and graphical; optimization and heuristics algorithms, including branch-and-bound as well as computerized heuristics; interactive and graphics, to allow user intervention in obtaining a solution; multi-goal, formulating the problem as a quadratic assignment task; and knowledge-based approaches. In fact, the automatic generation of nesting routines for non-convex objects to be packed or cut is an active area of research, as current algorithms have yet to adequately match human performance on these problems. Some solutions allow the translation, although not the rotation, of 2D objects, and many rectangular estimation (Adamowicz, 1976) or various heuristics have been applied. Even the translation problem alone is considered NP-complete (Fowler, 1981).

The layout of parts in the cutting operation is a very important step in the manufacture of apparel. Because of the expense of the raw material used, a small savings on each marker can reach gains in efficiency of production that amount to millions of dollars per year. Although totally automated layout has not been widely utilized, any amount of automation will result in reduced costs, decreased manufacturing time, and quality improvements. Additionally, a computer will be able to provide a layout much faster than a human could, decreasing the time for cutting. Currently, human experts spend time trying to solve this problem heuristically in order to find the tightest non-overlapping packing. A computer will also be able to more consistently follow the rules established for the cutout, and therefore will make the optimization more uniform for each application; new products and new uses for this algorithm will be quickly learned by an automation system. The solution of cutting and packing problems in the textile and leather industries will play an important role in industrial practice. The result in the decreased cost of production will of course increase the competitiveness of the industry and will enable a quick turnaround for customer demands; the increase of profitability is clearly the result.

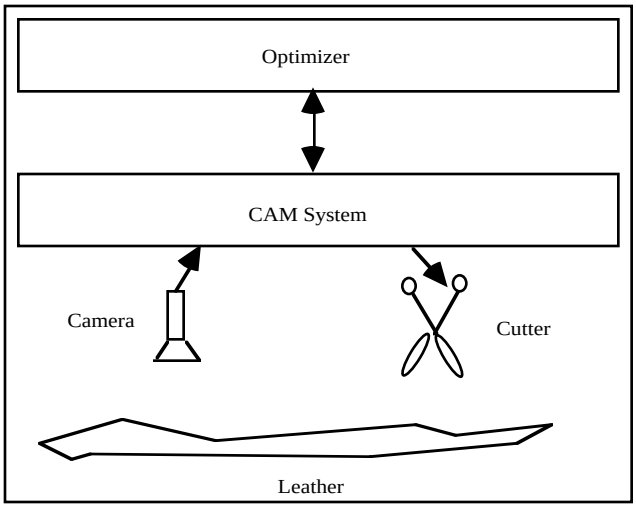


Figure 1. The Complete Leather System

The component of this system that this research proposal will address is the optimizer. We will discuss this in more detail in a later section.

Applications of Stock Cutting

Some examples of the Stock Cutting and Nesting Problem and the special complications regarding each include:

• *metal - die-cut from sheet-metal*

Sheet metal can be considered rather uniform. Although there are constraints on the placement of the die-cuts (stencils) due to the proximity of one stencil to another or the edge of the material. In fact, metal is more uniform than fabric, and the remaining metal is able to be melted to produce a new sheet.

• *fabric - removal of patterns from a bolt of fabric to produce clothing*

Fabric, however, has its own constraints. In the fashion industry, a pattern is overlaid on a bolt of fabric, from which sections are cut and are then sewn together to produce articles of clothing. A specific number of patterns must be cut in order to produce a particular garment (front, back, sides, lapel, etc.). Additionally, the cut directions are also important because the final garment must be coordinated and the stripes or other image in the fabric must properly align. All material remaining from the cut, the waste, cannot be used and is usually considered a loss for this production run. Therefore, the correct layout will lead to better use of resources and a more optimal final result.

• *leather - cutting from a hide*

We are interested in contributing to the leather nesting problem. In this problem, the surface represents an animal hide. Dealing with natural constraints, different parts of the leather may be of different color, have different quality, and may contain actual flaws. There may be actual holes or lines in the hide that prohibit a particular area of the surface from being used. The stencils must be placed in areas that correspond to the at least the required quality necessary in the final cut piece. As opposed to fabric and metal, due to the natural hide practical constraints, the markers placed on one hide are not generally transferable to another because no two are created alike. Furthermore, there are real-time implications to solving this problem, as all of the calculations need to be done without hampering the productivity of the leather-line (in other words, an algorithm that takes thirty minutes to run on a super computer solving a complex optimization problem is not even sub-optimal, it is unfeasible).

Automating layout in practice is a difficult task, requiring a fusion of techniques from engineering disciplines, computer science, operations research, and manufacturing. Cutting and packing problems occur in many branches of industry and involve multidimensional analysis for the different forms: one-dimensional (wood or steel), two-dimensional (metal, fiber), or three-dimensional (packing trucks, assembly of mechanical equipment, etc.).

Leather Complications

We have already started to explain how the stock-cutting problem applied to a leather surface is more complex than the cutting of other materials. There are more constraints in dealing with leather than other materials. Some of the system-wide problems and constraints are presented now:

• *the 'usual' problem of layout for machine-cut tools*

It is understood that there are certain stencil cutting issues that must be noticed immediately. In order to manufacture an item, a certain number of leather pieces must be cut. As an example, an order to manufacture shoes may require five pieces that must be cut from the leather. However, in this example, there are only three different types of stencils needed; two of the stencils are used twice to remove sections from the hide as a necessary prelude to shoe manufacture. Calling these stencils {A, B, C}, we know that each shoe requires one type-A stencil cut, 2 type-B stencils cut, and 2 type-C stencils cut. Essentially, the following equation will have to be satisfied:

$$H_{SIZE} = \sum_{i \in M} N(a_i + 2b_i + 2c_i), \tag{1}$$

where,

H = the leather hide size

M = the set of different types of shoes (and associated parts)

N = number of each shoe of type M

a,b,c = the part/stencil types

In the event that we do not have sufficient material in the hide to cut two type-B stencils, there is no point in cutting any of the other stencils for that particular shoe, as there will not be a sufficient stock to complete the process. Here, an extraneous cut will also be deemed waste.

• *irregular shape of hide*

The hide, although large, has an irregular shape and irregular boundaries. It is necessary to take this into account not only when performing the initial layout, but also during all subsequent movement of the stencils. They must be performed with a check to ensure the boundary of the hide has not been exceeded. The representation of the irregular shape requires algorithmic advances and consideration over the traditional packing problems in the literature. The formulations of equations (1)-(3) is clearly too simplistic for the irregular shape of a hide.

• *flaws*

Actual holes and lines of unusable material exist on a hide from a variety of reasons, including anatomical reasons, injury to the animal, or handling of the raw material prior to being processed. These flaws present obstacles to the stencil layout process and must be avoided in all but the most unusual cases. It is possible to use such material when it is not apparent to the consumer's eye, such as for the sole of a shoe.

• *pull directions*

Pull directions can be thought of as the pattern on a slice of fabric. Just as with fabric (but not with metal), there are additional constraints about pull-directions. This is a regional issue, as it deals with a particular region of the hide.

• *color and other regional issues*

Discoloration and colored regions of the hide must be taken into account when the final product must have matching color sections.

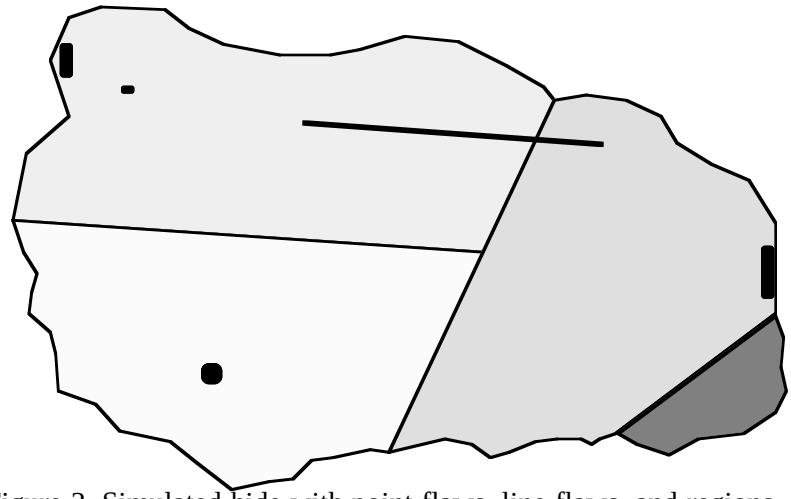


Figure 2. Simulated hide with point flaws, line flaws, and regions

II. PREVIOUS APPROACHES

The stock cutting problem has been of great interest to researchers for some time; (Sweeny and Paternoster, 1991) have identified more than 400 papers that deal with stock cutting and related problems and applications. Because even a small increase in the output of industry production where the waste of raw material is minimized can have large effects in profitability, the field of manufacturing and optimization remains open and continues to be highly explored. The optimal layout problem of irregular shapes is NP-complete on any medium and therefore does not allow us to examine all of the possible configurations.

There are classes of problems that are computationally difficult to solve, including stock cutting, nesting, assignment, bin packing and facilities layout, and VLSI layout, many different approaches have been applied and customized to each particular application. (Vance, 1994) presents an algorithm for the binary cutting problem employing column generation and branch-and-bound to obtain optimal integer solutions. With regard to nesting, (Prasad, 1991) develops a method of automatic allocation of sheet metal components to a given metal strip, but does so in a row-wise manner and does not deal with a general 2D

situation that is easily applicable to different shapes and irregular cutting material. In the solution of the two-dimensional knapsack problem, (Fayard, 1995) solves a sequence of one-dimensional problems and then is able to utilize a linear programming approach. (Prasad, 1991) reports on the design and implementation of a computer aided nesting system for two-dimensional nesting on metal, but they, too, employ a nesting structure that sub-models for multiple-row layout. (Roussel, 1993) recognizes that the methods they have studied follow three general methodologies: i) rectangular approximation followed by an approximation placement; ii) heuristics; and, iii) graph search and theories. They basically present an improvement in graph search with modified data structures, objective function, and research heuristics to combine the methodologies. In fact, (Saab, 1990) shows another search technique of stochastic evolution which is used to combine iterative techniques and biological evolution for combinatorial optimization. Demonstrating their work through the network bisectioning and traveling salesman problems (as (Dunlop, 1985) has for VLSI layout, another similar area), they show better results than typical with simulated annealing (also demonstrated in (Lam, 1988)). We would similarly use this as a method of comparison in further studies.

An interesting approach taken by (Dagli, 1990) uses neural networks to attempt matching between two irregular patterns. The information generated by the feature recognition network is then passed to an energy function and the optimal configuration of patterns are computed through a simulated annealing algorithm. This can be a cumbersome and time consuming algorithm, and should be considered for off-line processes only.

(Albano, 1977) proposed a method of two-dimensional layout that is almost completely user-interactive, and commands are manually entered (interesting and adaptable, but not practical), whereas (Adamowicz, 1976) presents a solution to a rectangular nesting problem using a constrained dynamic programming algorithm to lay out groups of rectangles in strips. Using this method on non-rectangular objects requires approximations, and may not yield good results for highly irregular objects. (Haims, 1970) also employs rectangular approximations to contain irregularly shaped objects, and uses dynamic programming to obtain results. (Reda, 1994) has an approach that assumes rectangular objects and takes a multiple-objective linear integer programming approach as follows: The mathematical formulation for this problem has been attempted in this way (Abd El-Aal, 1994):

Minimize $$\sum_{j \in J} T_j X_j \tag{2}$$

subject to:

$$\sum_{j \in J} a_{ij} X_j \geq N_i \qquad \text{with } \forall i \in n$$
$$X_j \geq 0 \qquad \text{and } \forall j \in J \tag{3}$$

where

T_j = the trim loss by cutting pattern j. The trim loss after cutting from a feasible combination - for a rectangular piece cut - is:

$$T_j = LW - \sum_{i \in n} a_{ij} l_i w_i \tag{4}$$

X_j = number of stock sheets to be cut using pattern j

a_{ij} = number of times that piece i appears in cutting pattern j

n = Number of pieces required

J = set of all feasible cutting problems for the problem

This was presented so the reader will immediately see that there are two problems in formation of the problem this way: i) When the pieces are non-rectangular, the calculations become non-trivial; ii) when J rises to a high order (n-pieces, each located at (x_i, y_i) with rotation r_i), this set becomes extremely large. The number of possible orientations and rotations for two objects with respect to each other is uncountable; for many objects, the space is not searchable for the optimal placement.

Sphere packing algorithms have been studied in (Ridenour Peternoster, 1992) from 1-,2-, and 3-D applications and are interesting for warehouse layout and transshipment. Combinatorial optimization has been studied for the general solution, including aspects such as Tabu Search (Areibi, 1993) and study of NP-hard problems, and many software implementations have been devised; however, the nesting problem is known to be NP-hard even in strongly restricted special cases [Fowler, 1981]. Of course, recently, genetic algorithms (GAs) and parallel GAs have been applied to these problems (Levine, 1994). That may, in fact, prove to be useful in future versions of our algorithm, as we need to perform a modified gradient-descent for many parallel objects. We also note the properties of the problem that are found in (Chowhurdy, 1989) to develop objective functions around which to maximize. Objective functions will

present an important part of our algorithm, however, we implemented them differently primarily for reasons of execution time.

Problems Unsolved

The methods we propose provide a general approach to provide real-time solutions to many large-scale optimization problems. We present a solution to this problem addresses issues not yet covered by the other approaches references above.

Specifically, there is a need to perform highly complex real-time optimization during manufacturing to solve the problem of maximizing the yield of production while minimizing the cost of materials. Although traditional linear and non-linear programming approaches have dealt with these problems, some problems are too complex and result in a combinatorial explosion for those methods. In addition, the proposed solutions to these problems often involve heuristics that are problem specific and do not tend to a general direction for the class of problems for which the true optimal solution is unable to be directly calculated. Many methods are too computationally intensive and time intensive to satisfy our requirements.

Although rectangle and sphere packing have been well studied and industry-available software for performing packing and nesting exists, there is no uniform algorithm for performing the same task when the objects in question are not geometrically-friendly. In fact, the automatic generation of nesting routines for non-convex objects to be packed or cut is an active area of research, as current algorithms have yet to adequately match human performance on these problems. Some solutions allow the translation, although not the rotation, of 2D objects, and many rectangular estimation or various heuristics have been applied. Even the translation problem alone, as shown above, is considered NP-complete.

Leather nesting, or cutting, presents a difficult subset to the traditional nesting problem. As discussed, the hide, although large, has an irregular shape and irregular boundaries; therefore, it is necessary to take this into account not only when performing the initial layout, but also during all subsequent movement of the stencils. They must be performed with a check to ensure the boundary of the hide has not been exceeded. The representation of the irregular shape requires algorithmic advances and consideration over the traditional packing problems in the literature. Actual holes and lines of unusable material, flaws, exist on a hide from a variety of reasons, including anatomical reasons, injury to the animal, or handling of the raw material prior to being processed. These flaws present obstacles to the stencil layout process and must be avoided in all but the most unusual cases. It is possible to use such material when it is not apparent to the consumer's eye, such as for the sole of a shoe. Pull directions can be thought of as the pattern on a slice of

fabric. Just as with fabric (but not with metal), there are additional constraints about pull-directions. Discoloration and colored regions of the hide must be taken into account when the final product must have matching color sections.

There is no attempt that provides a real-time workable solution: it is beneficial for the users of such algorithms and computer packages to work within the guidelines of the manufacturing floor; to interface with the people who need the solution hands-on. Totally or partial automation is not widely used because of a variety of problems with the software available.

III. MODEL FOR OUR PROPOSED SOLUTION

We feel the best solution to a large and complex problem such as this is to attempt to model the packing behavior that is already evident in nature; there are many examples in nature providing an optimal packing or nesting of items. Our algorithm will offer a better solution to problems of this class by presenting a control trade-off between computation time and regional optimality of the solution. Our goal is to develop an algorithmic scheme that will yield the best possible solution to this combinatorial optimization problem using parallel executed simple decision rules based on nearest neighbor (local decisions) parameters which achieve results that are competitive with or better than the results of a human nester, and to aid in manufacturing. In our model, each object to be placed will be a separate process in a parallel computing architecture, and will consider the parameters (location, orientation) of its nearest neighbors (other parts in the vicinity) to make translocation and re-orientation decisions. Although traditional parallel implementations of algorithms have been applied to problems before, we are not employing a divide-and-conquer approach to the resulting decision tree. Rather, as opposed to other attempts to run parallel algorithms, we distribute the objective function of the optimization itself. We intend to show that the fusion of existing algorithms and the establishment of uniform and 'simple' local decision rules into an optimization function will more quickly yield to a better optimum for the entire system.

Although such totally automated layout has not been widely utilized in industry, any amount of automation will result in reduced costs, decreased manufacturing time, and quality improvements. Additionally, a computer will be able to provide a layout much faster than a human could, decreasing the time for cutting. Currently, human experts spend time trying to solve this problem heuristically in order to find the tightest non-overlapping packing. A computer will also be able to more consistently follow the rules established for the cutout, and therefore will make the optimization more uniform for each application; new products and new uses for this algorithm will be quickly learned by an automation

system. The solution of cutting and packing problems in the textile and leather industries will play an important role in industrial practice. The result in the decreased cost of production will of course increase the competitiveness of the industry and will enable a quick turnaround for customer demands; the increase of profitability is clearly the result.

The environment under which we operate is the factory floor where real-time computing issues are very rigid. We therefore are forced to reject many existing solutions to this problem precisely because we are attempting to develop a real-time solution. When a leather hide is presented, there is very little time to perform an optimization, and time constraints must be balanced against our ability to actually perform the result. The initial layout stage will be accomplished using standard algorithms, as mentioned. The goals will be standard (as in, solutions exist) non-linear programming and sphere packing in 2D where we will be optimizing layout for quality and yield versus waste. Although some of those algorithms will be stated below, our focus will not be on standard 2D sphere packing problems, but on the optimization that we can perform thereafter. We will be measuring, during the initial stage, the performance index of the layout against the constraints given.

It should be further noted that what we wish to add to the solution of this problem by our research is the technology solution: we are not dependent *a priori* on the particular stencil types or hide geometry to solve this problem. Essentially, we envision creating a work-engine using our new methodology that can be incorporated within another package for solving similar problems. This proposed algorithm may prove invaluable to stimulating further research in a variety of the disciplines upon which we touch: engineering control, parallel computer science, operations research, and manufacturing research. Although we are tailoring the solution for now to leather stock cutting because of our interest in dealing with the constraints in a non isotropic and non homogenous environment, the work engine that we develop is applicable to other problems.

Methodology

Our goal is to develop an algorithmic scheme that will yield the best possible solution to this combinatorial optimization problem in a given time using parallel executed simple decision rules based on nearest neighbor (local decisions) parameters. It is our belief that the combination of traditional approaches to this problem as described in the literature, with the fusion of these methodologies executed on a massively parallel scale, will enable the system to achieve a *better* optimal than other methods where each object in the system is making simple and fast decisions (and learning) based on a pre-existing rules-set and local data only. We do not claim to have a global minima to the overall problem, but we are attempting to find a

local minima to the problem that is comparable to and rivals those solutions currently available. A "better" solution is one that yields a reduction in the trim loss.

We propose a solution that is using our model of:

• An intelligent controller modeled after expert systems.

• simple decision-rules: fundamentally, using distance and energy (of the system, as viewed locally by each part) calculations to perform a gradient descent down to the global optimum (example provided below).

• nearest neighbor shared information - each process in the parallel architecture makes its parametric decisions based on those objects physically closest to it in 2-space.

• multiple resolution levels - development of a resolutional representation of the objects being manipulated (here, the stencils on the hide) to reduce computation while preserving the ability to not only make rapid geometrical calculations from an object to its nearest neighbors, but effect the precision of such calculations as the algorithm requires.

The algorithm contains a two phase solution (see Figure 3) to the problem, where a multi-resolution approach is taken.

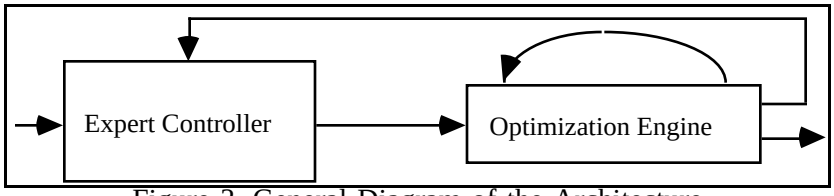
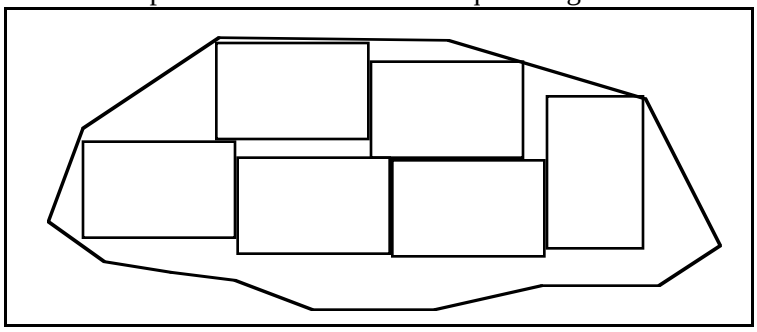


Figure 3. General Diagram of the Architecture

• Initial layout using *Expert-System Controller* (EC) methodologies and conventional sphere packing algorithms, where the stencils are represented by bounding approximations and that is sufficient to determine an initial, although sub-optimal solution. This is the low-resolution placement of the polygons.

The EC represents each of the stencils to be placed by a gross-approximation such as bounding rectangle, and places the rectangles on the hide in a feasible solution. At this point, the only requirements are that the rectangles do not violate any geometrical rules of the leather such as being out of bounds, and

that the rectangles do not overlap with each other. We now have a feasible solution, although the amount of loss if the marker cut would be performed now would be quite large.



.c9.Figure 4. Layout at Low Resolution;

• Optimization parallel model *Optimization Engine* (OE) where the algorithm we develop will move from sub-optimal to the optimal solution. This is the high-resolution approach; we do not use a gross approximation of our polygons for the calculations during this phase.

In the OE phase, each stencil (polygon) is represented by a program running on an individual processor in a multi-processor environment. In this large processor network, the processors communicate with each other through a multicast transmission system and share location and orientation data for their respective polygons, but each processor is responsible for moving and rotating only one polygon. Operating asynchronously, each processor makes the decision to move or to rotate based on a function (objective function) that uses the geometrically-based nearest neighbors as parameters of the function. This architecture is known as Single Program Multiple Data (SPMD) and is a type subset of the multiple-instruction multiple data (MIMD) case. Each processor is running the same instruction set but with only the nearest neighbor data. The goal of each processor is to move and rotate its polygon to minimum value of the corresponding objective function, where the objective function value is very small when the polygon and its nearest neighbors are not tightly packed, and the objective function value is very high when the polygon and its nearest neighbors are densely packed. Therefore, gradient descent (or 'ascent', in our case) is used by each polygon's processor to control its translocation and rotation in achieving this goal.

When the objective function is carefully selected, the overall goal of determining a tightly-packed leather hide with polygons will be achieved when, on a local scale, each polygon has locally determined that it is in its most densely possible configuration. The function is carefully selected to correspond with the problem under study, as other types of optimization problems will require different objective functions.

The OE will discontinue its run when the polygons can no longer be move to achieve, locally, higher values for their respective objective functions.

• Feedback will be presented to the controller when the OE has completed. The EC may then have additional room to place more polygons on the hide, and the OE will be again called to tighten the polygon pack. Alternatively, the user may decide to accept the solution presented and discontinue the optimization.

The Natural Model

We feel the best solution to a large and complex problem such as this is to attempt to model the behavior that is already evident in nature. As a result, we have nicknamed it the FLOCKING ALGORITHM. There are many examples of nature providing an optimal packing or nesting of items, where each of the subjects to the optimization is performing a local optimization for its part while yielding a global optimal for the entire system, and only making local decisions:

1. The Traveling Birds Problem

During the migration of birds, it is evident that schools of birds are able to perform extremely complex navigation and motion in real-time without there being an apparent global controller for the bird-school system. The birds seems to be able to simultaneously perform the following activities:

- Add and delete bird members from their school structure
- 3D patterning in real-time
- velocity changes
- global navigation
- with no accidental collisions!

In an attempt to understand this behavior, we will further examine models of human behavior and attempt to relate them appropriately:

2. Theater exit

If a movie theater full of individuals the need for a rapid exit became apparent (i.e. someone yelling "Fire!"), there are a series of 'rules' that the individuals will follow while making an egress. Simultaneous, people will head for the door and will begin queuing their way through the door as the bandwidth

capacity of the door is rather limited. As the queue backs up, people will be unable to leave this seats as quickly, and the lines in the aisles form. The line will build and will, barring any unforeseen events, exit as appropriate. Overlap between peoples' bodies is not allowed, and orientation is important for motion as well as people-packing.

3. Fighter pilot/formation airplane flying

With one squadron commander performing the global navigation task, the wing elements present will not perform any navigation other than maintaining a particular distance and bearing from the closest plane to their aircraft. While each member of the formation is solving this identical problem, all dependent on the actions of the leader, the formation is maintained.

Thus, simple decision rules applied to a global problem are evident in forming good solutions to those problems.

Multiobjective Criteria

Our optimization problem is a multiobjective problem. For each pair of objects, the objects are trying to be as close to each other as possible. We can define a function to represent this that is of the form:

$$f(Xc_a Yc_a, Xc_b Yc_b) = \frac{k}{\sqrt{(Xc_b - Xc_a) + (Yc_b - Yc_a)}} \tag{5}$$

Where the arguments to the function are the X and Y center positions of polygons *a* and *b*, and *k* is a scaling parameter. This function reaches a maximum when the objects are overlapped directly one-to-one, and minimum at the objects placed at a distance of infinity. Alternatively, we can use a convex hull around two polygons to determine the same thing:

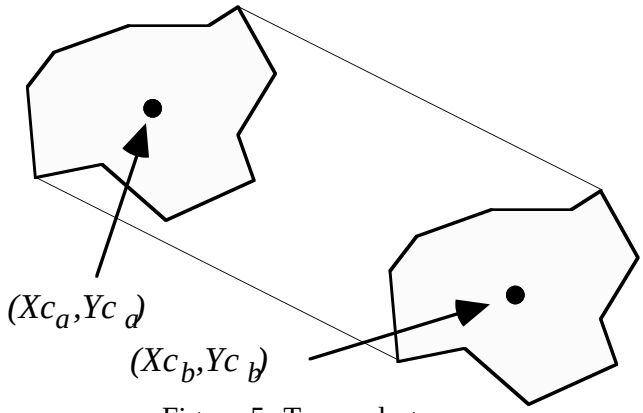


Figure 5. Two polygons

It should be clear that the minimum distance that two polygons can occupy when we consider their convex hull is simply a sum of their areas:

$$sm\_area(P^a_{xyr}, P^b_{xyr}) = Ar(P^a_{xyr}) + Ar(P^b_{xyr}) \tag{6}$$

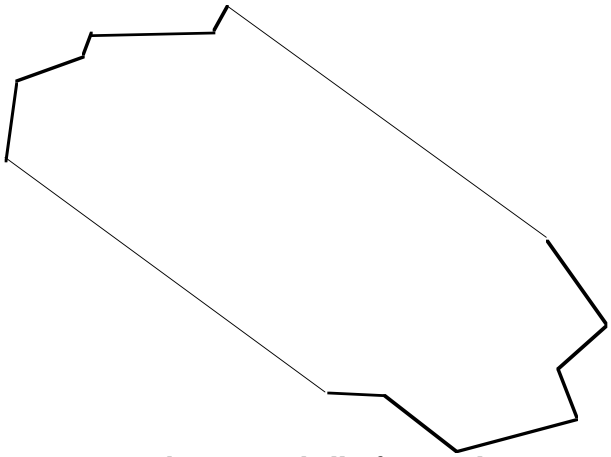


Figure 6. The convex hull of two polygons

$$area\_usage(P^a_{xyr}, P^b_{xyr}) = \frac{sm\_area(P^a_{xyr}, P^b_{xyr})}{convhull\_area(P^a_{xyr}, P^b_{xyr})} \tag{7}$$

The convex hull (*convhull_area*) is presented in figure 6, and the *area_usage* function is a nearness function between two polygons, bounded ≤ 1.

Additionally and simultaneously, we must ensure that the two polygons are non-overlapped. Although the algorithms for this calculation are more complicated than equation (5), we need to check if i) any of the

polygon vertices of *a* are contained inside polygon *b*; ii) the two polygons are identical and are flush on top of each other; iii) polygon sides cross the other polygon's sides or cross a vertex. We can define a functional procedure for this as well, and in our simulations, we use MATLAB routines. In order for us to find a solution to this problem, we must perform a scalarization of our criteria functions into a unified function. When we find the efficient scalarization of the functions we wish to optimize, we can then perform the necessary operations to find the optimal value of that generated objective function. We show the development of this function below.

There will be many possible solutions to the minimum trim-loss case, and we seek to find a solution to the problem from the set of best solutions, similar to a Pareto Optimal solution. Each polygon is trying to obtain its best position, and therefore, there is a competition as each polygon vies for position. In multiple-objective problems, we seek to obtain the set of Pareto optimal solutions. As presented in [Lin], we seek to

$$\begin{aligned} &\text{maximize } z_1 = J_1(x_1,\ldots,x_n),\ldots,z_N = J_N(x_1,\ldots,x_n) \\ &\text{such that } (x_1,\ldots,x_n) \in X. \end{aligned} \tag{8}$$

Where $x_1,\ldots,x_n$ are variables under control, and X is the constraint set, and $J_1,\ldots,J_n$ are real-valued objective functions. The n-tuple $x^* = (x_1^*,\ldots,x_n^*)$ is a Pareto optimal solution iff $x^* \in X$ and there exists no n-tuple $x \in X$ such that (i) $J_i(x) \geq J_i(x^*)$ for all i's; and, (ii) $J_j(x) > J_j(x^*)$ for at least one j. In general, there are many Pareto-optimal solutions, but any one of them is by definition a good alternative to any other so far as the given multiple objectives are concerned. In many cases, it is useful to obtain the set of Pareto optimal solutions. We consider Pareto-optimality because of the competing objectives of each polygon in our architecture.

The Decision Rules

In the examples presented we can begin to understand the rules that are applied to these types of problems. Each element of the system has a very simple rule: in the case of the pilots it can be seen as select the nearest target object, and maintain an objective function, here one combining distance and bearing, to that object. The people exiting the theater will execute a similar algorithm, including not getting too close to others, not trampling others (optimally), velocity, and orientation. People in this situation also apply learning algorithms. When children practice fire-drills when elementary school students, they are learning the rules that make such evacuations possible. This is the beginning of the development of the decision rules that are used to solve this problem. When each element in the system continues to perform according to a rather simple algorithm and the variables for the algorithm's

equations are solely dependent on only the local neighbors (physically co-located), it is possible for an optimal *global* solution for the system to be the result. We would like to demonstrate that given a fixed amount of time and computation, this algorithm will be able to provide a better local solution to a complex optimization problem.

In our model, each object to be placed will be a separate process in a parallel processing architecture, and will look at it's nearest neighbors (other parts in the vicinity) and will make re-orientation and re-positioning decisions based on those local parameters.

Application

We now apply the objective function components developed above. Each processor element (PE) has to make the following calculation, where its polygon is labeled P^1. After we have determined the polygons that are closest to P^1, we identify them as $P^{2..N}$,where N is number of nearest neighbors in which we are interested. Note that N is a variable that changes based on the type of simulation we are performing. We have used an objective function that is of the following form and is equal to:

$$Obfn(P^{pc}_{xyr}, P^1_{xyr},\ldots,P^{N}_{xyr}) = \frac{Ar(P^{pc}_{xyr}, P^1_{xyr},\ldots,P^{N}_{xyr})}{Conv(P^{pc}_{xyr}, P^1_{xyr},\ldots,P^{N}_{xyr})} \bullet ins(P^{pc}_{xyr}, P^1_{xyr},\ldots,P^{N}_{xyr}) \quad (9)$$

where

$$Ar(P^{pc}_{xyr}, P^1_{xyr},\ldots,P^{N}_{xyr}) = \alpha(P^{pc}_{xyr}) + \sum_{i=1}^{N} \alpha(P^i_{xyr}) \quad (10)$$

represents the area (alpha function) of our polygon (P^{pc}_{xyr})at its current location and rotation (x,y,r), and for each of the nearest neighbors.

The $Conv(P^{pc}_{xyr}, P^1_{xyr},\ldots,P^{N}_{xyr})$= function is the smallest convex hull area of all of the polygons P^{pc}_{xyr} and its nearest neighbors $\sum_{i=1}^{N} P^i_{xyr}$. (11)

We realized that this is the core of our function, in that objects that are very distant, will increase the value of the denominator, and reduce the value of the objective function. When objects are close, the denominator is smaller, and the function is thereby maximized. However, we note that objects are not allowed to overlap, and

$$ins(P^{pc}_{xyr}, P^1_{xyr},\ldots,P^{N}_{xyr}) \quad (12)$$

will return a zero if overlap between any of the nearest neighbor polygons is indicated. At each time step, if no overlap occurs, the configuration is stored; if during the next time step we discover an overlap, the previous feasible solution is returned. This is for the user-interface purpose that a feasible solution is presented any time the operator desires one.

The SPMD Model and Simulations

We have noted that research using MIMD models (of which, SPMD is a sub-type) often uses such architecture in one of the following ways: segmentation of a problem decision tree, or parallel execution of multiple rules in a competing environment. In the former problem, an example of cryptology is best. If attempting to find the precise 16-bit code to decrypt a message, we can divide and subdivide the problem between parallel processors where each processor is brute force attacking a subset of the possible 16-bit codes. We can envision the decision tree of possible solutions being distributed between processors; each processor has a unique N most significant bits (MSB), but each cycles through all possible 16-N least significant bits (LSB).

In the latter problem of a competing environment, we may observe problems in the fields of finite element analysis, computational fluid dynamics, crash testing, electrical design simulation, and computer aided design and manufacturing (CAD/CAM) as just a few examples. In these examples, the nature, properties, and interaction of each individual unit of the simulation is determined and known. During the course of the execution of the program, we allow the units to interact, and we study the global nature of the system.

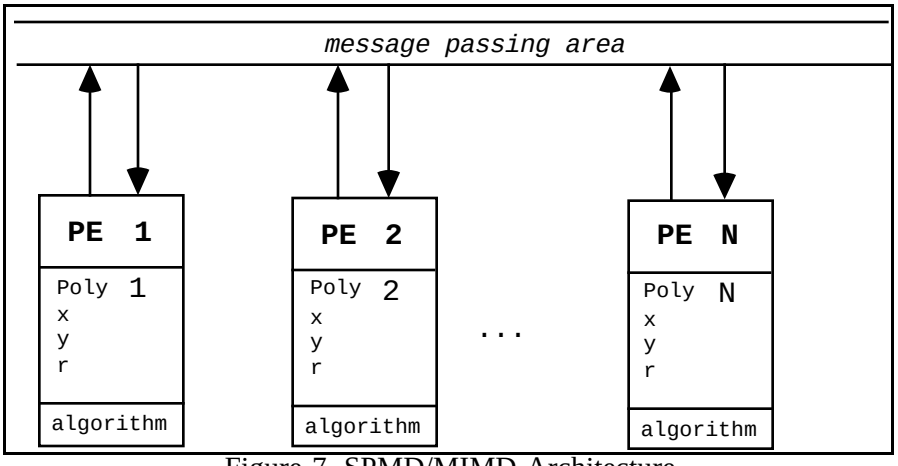


Figure 7. SPMD/MIMD Architecture

We are using our SPMD architecture in a different way, and our load-balancing problem is handled in a way that recognizes that each sub-task in our problem is still potentially NP-complete. We do not face the same issues that are considered in [Chakrabarti] for load balancing, because our *function* is distributed, and we are also not concerned with the typical synchronization as is [Krishnamurthy] because of our asynchronization of processors. We have as our goal the known global optimization desired of the entire system, and we are developing rules and objective functions to be executed at the lower levels by each computational unit in order to achieve those goals. Whereas in the cryptology example the task tree of possible solutions is distributed among independently operating and probably synchronous processors, in our algorithm we instead distribute the objective function and allow the processors to function asynchronously. In the competing environment examples, the local behavior is known and the overall system behavior is to be determined, we know *a priori* the general behavior of our desired system, and we seek the objective function for each processor to achieve our goal.

The objective function derived above is executed in parallel for each object on each processor. Each PE broadcasts the location of its polygon, and each PE listens to the broadcast to find the nearest neighbors. Each PE calculates the value of the objective function independently, and decides based on descent techniques how to increase the value of the local objective function, when the only variables allowed for it to change are those associated with the PE's polygon. When that maximum is determined, we have reached a steady-state of the search engine.

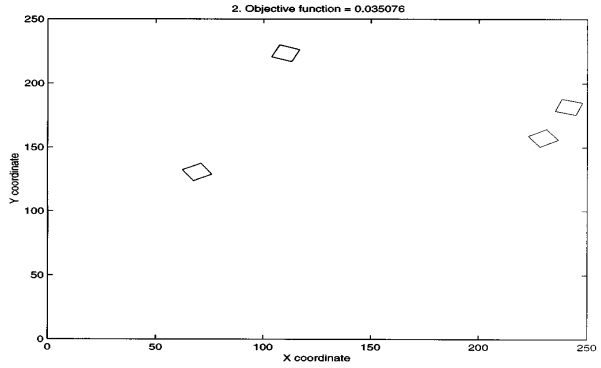
IV. SEARCH ENGINE SIMULATION RESULTS

We have conducted sample runs using rectangular objects to determine if it is possible for our search engine to operate in parallel, execute a simple decision rule with our defined objective function, and to reach a tight geometrical fit. We viewed this as the most important first step in determining if our algorithm would achieve our goals. We use a simple distribution (random placement of objects into non-overlapping configurations apart) rule to express our control intelligent controller, and do not use the feedback proposed. We are looking for optimization to occur when we are only performing 'local' optimization for each object, and we will study the global nature of the system. In addition, we have performed simulations in which we vary the number of nearest neighbors allowed for each calculation performed. Written in Matlab 5.1 under Solaris 2.5 and running on a SPARC Workserver, we have defined an experiment with rectangular objects in an unconstrained environment to test the SPMD architecture and the search engine, as the first challenge to this architecture. The simulations are Monte-Carlo in

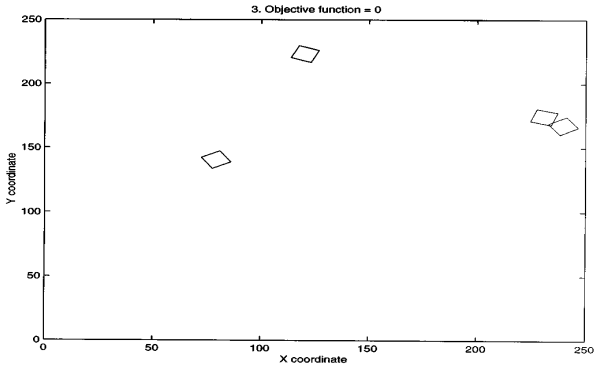
nature, and our software provides a geometrical view for each time step (shown below), as well as a plot of the objective function value for the system when the run is completed.

Figure 8 begins the demonstration with a typical run with four polygons. In this run, we see how the objects move from feasibility to 1-nearest neighbor optimality. It is important to note that the number of nearest neighbors selected is paramount in the final outcome, as we observe here. Each plot represents an advance in time (some omitted), and the values of the *objective function* (*obfn*) are provided. Plots are either the in geometric domain (polygon configurations) or objective function domain (and values).

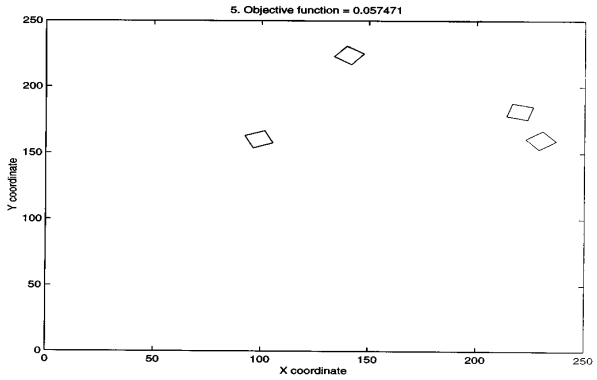
Figure 8. Four polygon, 1- nearest neighbor simulation (a)-(i)



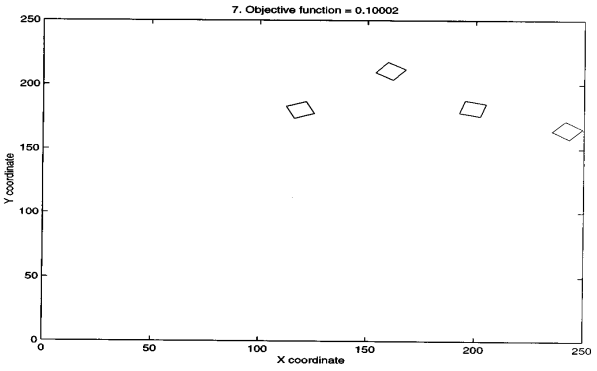
(a) *Obfn = 0.035076*



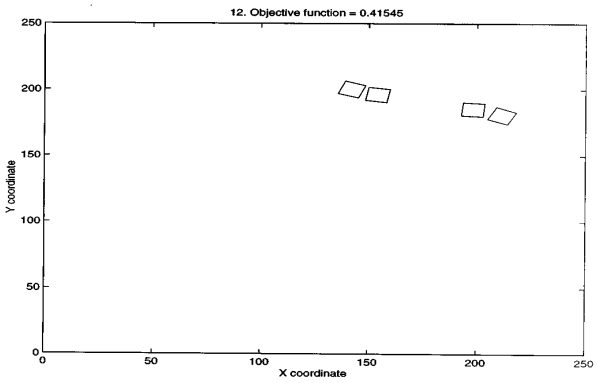
(b) *obfn = 0* . Overlap occured



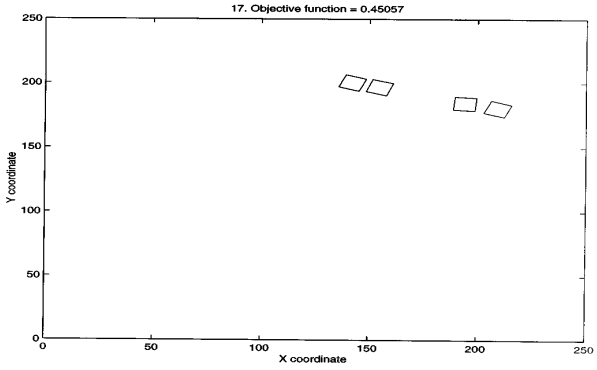
(c) *obfn = 0.057471.*
Overlap corrected



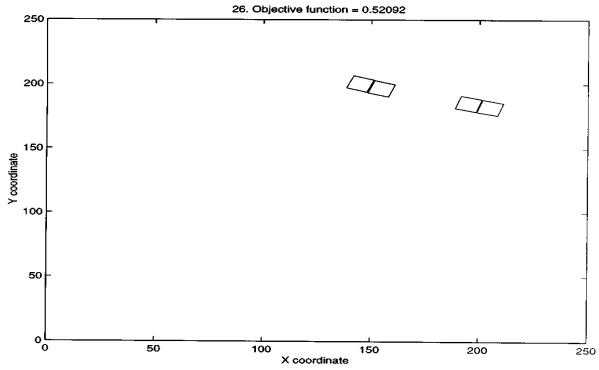
(d) *obfn = 0.10002*



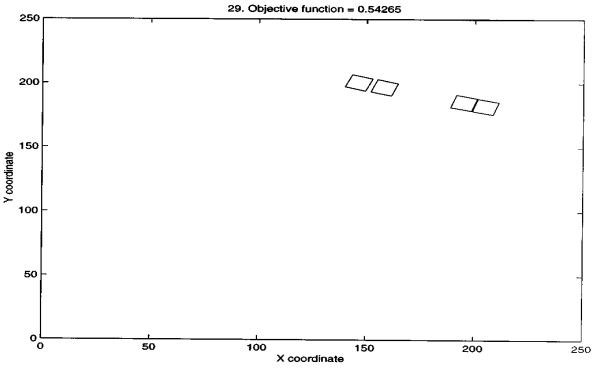
(e) *obfn = 0.41545*. Closed pattern energes



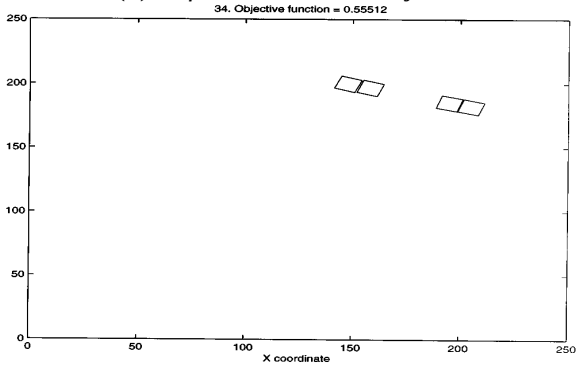
(f) *obfn = 0.45057*. Subtle rotations improve obfn



(g) *obfn = 0.52092*



(h) *obfn = 0.54265*. Objects closer

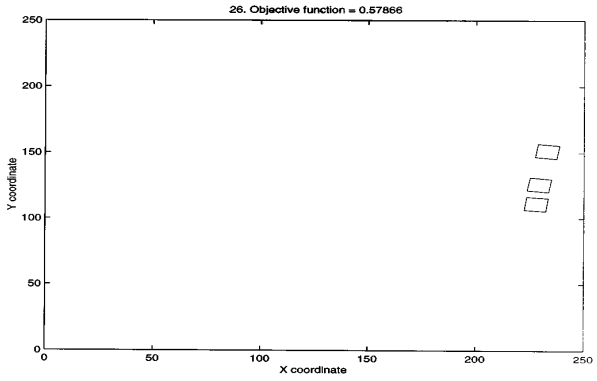


(i) *obfn = 0.55512*. Close to completion.

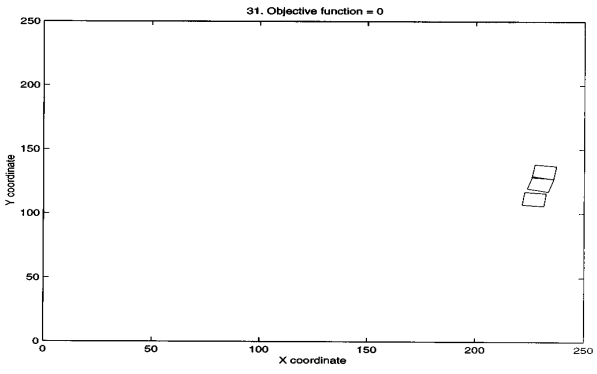
Note that we are providing slides of the more remarkable images. Each plot advances in time, and we observe the motion of the objects, but we do not provide here all of the time incremental slides. In those not presented, we see the objects coming closer to cluster, and rotating to align with their neighbors.

Another example with two objects is presented below, and the subtle rotations used to increase the objective function are quite visible.

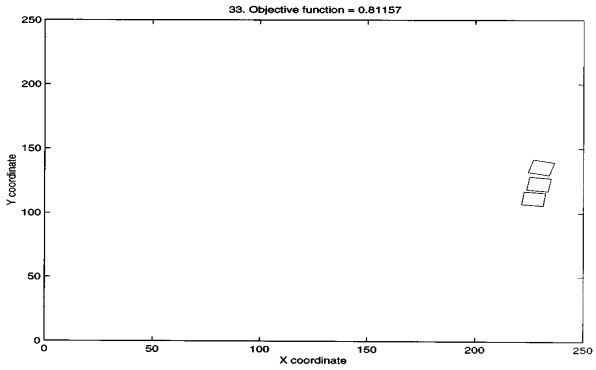
Figure 9. 3-Body Simulation (a)-(d)



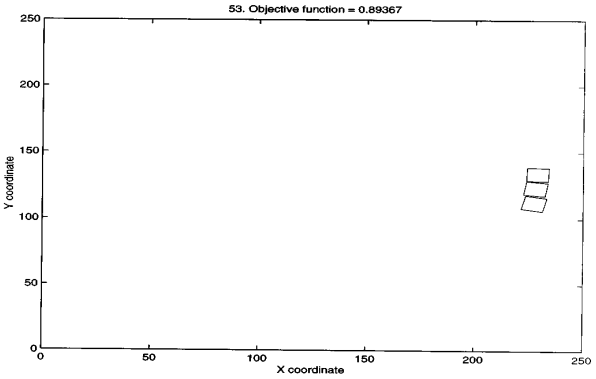
(a) Feasible solution during a run



(b) Overlap occured



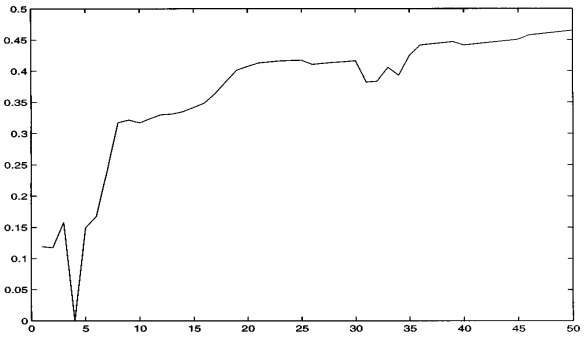
(c) Overlap corrected and *obfn* continues to climb



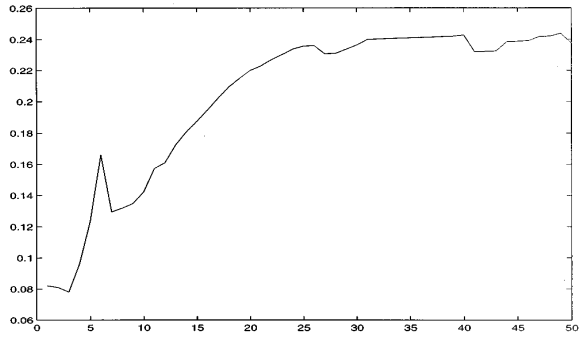
(d) Close to completion

The following series of figures are for different runs of the simulation algorithm. In each case, we are observing the objective function value (ordinate) plotted against the time step in the simulation (abscissa).

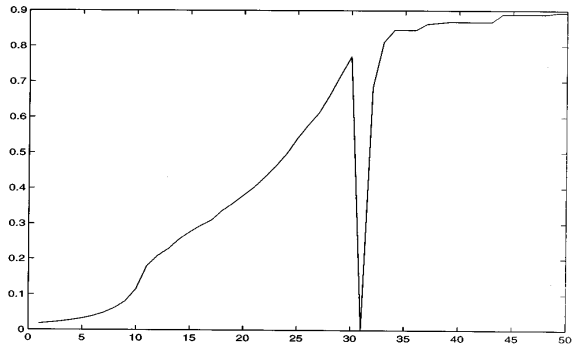
Figure 10. Sample Objective Functions (obfn) vs time (a)-(e)



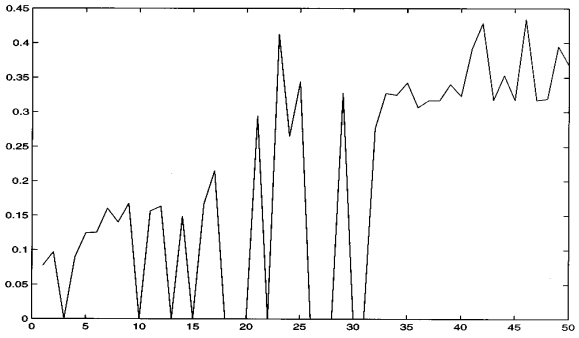
(a) We observe one drop to zero occuring indicating a overlap during the run



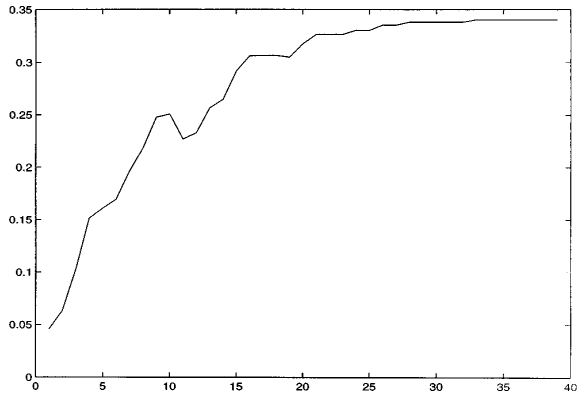
(b) No overlap occurs, but we observe subtle fluctuation in the function as the objects move



(c) Overlap occurs as we approach the end of the run but corrects and configuration continues to improve



(d) Although overlap occurs repeatedly, the general slope of the function is positive



(e) Nice smooth climb to tightness of objects

V. CONCLUSIONS

DISCUSSION OF RESULTS

We observe in the geometric-domain figures of section IV that the algorithm closes to more optimal results and tighter packing than originally provided by our placement routine (using simple rules), and gives promise to approaching a global optimum with the specifications for this problem. It is noticed in figure 8 the whole-system objective functions as they change with respect to time, that there is a notable positive derivative to these functions when executed in parallel. Even with a gross initial placement (low value of objective function), our algorithm provides for an increase in the *obfn*, and therefore the compactness of the associated polygons. We notice that our algorithm is more dynamic than a simple gravity-function of attraction between neighboring polygons. Rotation of bodies to increase *obfn* value is the significant difference. From the figures of the three body case, we see that there are slight rotational corrections between one time step and another, as the relative alignment between the two bodies is considered and handled to yield a tighter configuration.

We observe oscillations as polygon objects will occasionally overlap, move apart to correct for the overlap, and then move toward each other again, but the adaptive step-size we have introduced is the compensation, and we note that even when our objective function is quickly changed to infeasible solution, our correction continues at our last feasible solution and continues to optimize the polygons. Additionally, although we note that we do not claim to achieve a global optimum in these simulations, as no global optimum is able to be found in any closed form, we do observe solutions that show our SE gives promise to solutions for a variety of problems. This architecture enables us to deliver a feasible solution at any time during the run, another feature of the algorithm. Due to the asynchronous communication between PEs,

polygons can be added (or even deleted) without affecting the other computations occurring with other, non-related polygons.

CONTINUING WORK

The experiment performed is an important and necessary step in determining the efficacy of the FLOCKING ALGORITHM as presented in Section III. We realize that this experiment does not completely describe the benefits of our architecture as compared to other methods, but we have determined from this experiment that parallel optimization does occur as we theorized when the objective function is distributed across processors, the function is simple and repeated, and the architecture supports asynchronous communication. As we mature this algorithm, we shall begin adding more and complex constraints - and with the data structures we have developed, constraints are simply polygon objects that do not move. They are represented as objects in the global data structure, and do not need individual PEs. We expect that clustering will occur around fixed-location constraints.

We must continue to study the implication of the objective function and the oscillations that occur; in fact, it was determined that the nature of the function is stochastic and therefore, conventional gradient descent methodologies will not yield the maxima or minima of the function. Two objects will begin to move toward each other, but may quickly develop an overlap, from which they back-off. They then attempt to move again, resulting in the same occurrence. This is one oscillation; we simply modify our stepping size to counter-effect this oscillation. Another equivalent problem is that of how to find the highest peak of an ocean wave moving with respect to two dimensions and with respect to time is quite similar; our objects do not know the nature of the motion of other objects: although object P^1 sees object P^2 as its nearest neighbor, P^2 actually sees P^3 as its nearest, and makes its calculations accordingly. In other words, once the swimmer has observed a higher crest at a different location and has moved toward it, the wave has already 'moved' away. We now modify our adaptive step size to account for the apparent randomness in the process when examined on a local level, but in reality, a full model of expected course for each object would need to be charted; this clearly is counterproductive to our real-time considerations for a solution.

It is also necessary to continue to study the dependence of our algorithm on the number of nearest neighbors selected. We have observed clustering in our simulations of a small number of polygons, and in large-polygon simulations, the number of nearest neighbors selected for calculation affects the steady state greatly. Further study is necessary to determine the ideal cluster size, as the clustering of polygons takes place around the number of polygons selected for the nearest neighbors. Although this number is a parameter of our simulations, we require all polygons to cluster with the same number of nearest

neighbors, but do not provide the ability for polygons to adaptively change this value. The nearest neighbor parameter is another area of exploration.

BIBLIOGRAPHY

Abd El-Aal, RMS, "An Interactive Technique for the Cutting Stock Problem with Multiple Objectives," European Journal of Operational Research, 1994, p.304-317.

Adamowicz, M., and Albano, A. "A solution of the rectangular cutting-stock problem," IEEE Transactions on Systems, Man, and Cybernetics, vol. SMC-6, no. 4, April 1976, pp. 302-310.

Adamowicz, M. and Albano, A. "Nesting two-dimensional shapes in rectangular modules," Computer Aided Design 2, 1976, 27-33.

Albano, A. "A method to improve two-dimensional layout," Computer Aided Design, vol. 9, no. 1, January 1977, pp. 48-52.

Albano, A. and Sappupo, M. "Optimal allocation of two-dimensional irregular shapes using heuristic search methods," IEEE Trans. Systems Man and Cybernetics, SMC-10, 1980, 242-248.

Areibi, S., and Vannelli A. "Circuit Partitioning Using a TABU Search Approach," Proc. IEEE Int. Symp. Circuits and Systems, IEEE, PISCATAWAY, NJ, (USA), 1996, vol. 4, pp. 671-674

Bailleul, j, Soenen, R and Tiaibia k. "Nesting Two Dimensional Irregular Shapes in Anisotropic Material," Advances in CAD/CAM - Proceedings 5th Int. IFIP/IFAC Conf. on Programming Research and Operations Logistics in Advanced Manufacturing Technology, Amsterdam, 1983.

Cai, Yuzu, Liu ,Lujun, and Wang, Wei "An expert system for automatic allocation of 2D irregular shapes," in Expert Systems in Computer Aided Design, ed by John Gero, Elesvier Science Publishers, B.V. (IFIP 1987) 407-423.

Chowdhury, S., "Analytical Approaches to the Combinatorial Optimization in Linear Placement Problems," IEEE Transactions on Computer-Aided Design, Vol. 8, No. 6, June 1989, p. 630.

Chakrabarti, S., and Yelick, K. "Implementing an irregular application on a distributed memory multiprocessor," Computer Science Division, University of California, Berkeley (under DOD/ARPA, AT&T, and NSF contract)

Chakrabarti, S, Randae, A., and Yelick, K. "Randomized load balancing for tree-structured computation," IEEE Scalable High Performance Computing Conference, Knoxville, TN, 1994.

Dagli, CH. "Neural networks in manufacturing: possible impacts on cutting stock problems," Proc. of Rensselaer's Second Int. Conf. on Computer Integrated Manufacturing, 1990.

Dowsland K. and Dowsland W. "Solution Approachs to Irregular Nesting Problems," European Journal of Operations Research, vol. 84, 1995, p. 506-521.

Dunlop, A and Kerighan, B. "A procedure for placement of standard-cell VLSI circuits," IEEE Transactions Computer-Aided Design, vol. CAD-4, no. 1, January 1985, pp. 92-98.

Duque-Anton, M., Kunz, D., and Ruber, B. "Channel Assignment for Cellular Radio Using Simulated Annealing," IEEE Transactions on Vehicular Technology, Vol 42, No. 1, February 1993, p.14.

Fayard, D., and Zissimopoulos, V. "An approximation algorithm for solving unconstrained two-dimensional knapsack problems," European Journal of Operational Research, 84, 1995, pp. 618-632.

Fowler, R., Paterson, M., and Tanimoto, S., "Optimal packing and covering in the plane are NP-complete," Information Processing Letters, Vol. 12, No. 3, 1981, p. 133-137.

Haims, M., and Freeman, H. "A multistage solution of the template-layout problem," IEEE Transactions on Systems Science and Cybernetics, vol. SSC-6, no. 2, April 1970.

Heistermann, J., and Lengauer, T. "The Nesting Problem in the Leather Manufacturing Industry," ...

Krishnamurthy, A., and Yelick, K. "Optimizing Parallel Programs with Explicit Syncronization," Computer Science Division, University of California, Berkeley (under DOD/ARPA and NSF contract).

Lam, J., and Delosme, J. "Performance of a New Annealing Schedule," 25th ACM/IEEE Design Automation Conference Proceedings, 1988, p 306.

Levine, D. "A parallel genetic algorithm for the set partitioning problem," Mathematics and Computer Science Division, Argonne National Laboratory, September 1994.

Lim, JG. "Three methods for determining Pareto-optimal solutions of multiple-objective problems," Department of Electrical Engineering and Computer Science, Columbia University, NY.

Minnich, R. "ZOUNDS" software library for Distributed Parallel Computing, Sarnoff Laboratories and personal communication, 1996.

Nielsen, Jakob and Maurits Faber, Jan, "Improving System Usability Through Parallel Design," IEEE Computer, February 1996, p. 26.

Poshyanonda, P., Bahrami, A., and Dagli, C., "Two Dimensional Nesting Problem: Artificial Neural Network and Optimization Approach," IEEE, 1992, pp. IV-572.

Prasad, YKDV and Somasundaram, S. "CASNS - a heuristic algorithm for the nesting of irregular-shaped sheet metal blanks," Computer Aided Engineering Journal, 1991, p. 69-73.

Raoot, A., and Rakshit, A. "A fuzzy approach to facilities lay-out planning," Int. J. Prod. Res., vol. 29, no. 4, 1991, pp. 835-857.

Reda, M.S., and Abd, El-Aal. "An interactive technique for the cutting stock problem with multiple objectives," European Journal of Operational Research 78, 1994, p 304-317

Ridenour Peternoster, E., and Sweeney, P.E., "Cutting and packing problems: a categorized application-oriented research bibliography," Journal of the Operational Research Society, 1992, pp. 691-706.

Roussel, G "Improvements about automatic lay-planning for irregular shapes on plain fabric," 16th IFIP conference on System Modeling and Optimization, Compiegne, France, July 1993.

Sastry, S. and Pi, J. "Estimating the Minimum of Partitioning and Floorplanning Problems," IEEE Transactions on Computer-Aided Design, Vol. 10, No. 2, February 1991, p. 273.

Saab, Y., and Rao, V. "Combinatorial Optimization by Stochastic Evolution," IEEE Transactions on Computer-Aided Design, Vol. 10, No. 4, April 1991, p. 525.

Saab, Y., and Rao, V. "Stochastic Evolution: A fast effective heuristic for some generic layout problems," 27th ACM/IEEE Design Automation Conference, 1990.

Sweeney, PE., and Paternoster, ER. "Cutting and packing problems: an updated literature review," Working paper no 654, School of Business, University of Michigan, 1991.

Vance, P., Barnhart, C., Johnson, E., and Nemhauser, G. "Solving binary cutting stock problems by column generation and branch-and-bound," Computational Optimization and Application, 3, 1994, pp. 111-130.

Yang, D. and Jiang, H. "A Production Planning Expert System in Manufacturing," Dalian University of Technology, China.