BI–OBJECTIVE OPTIMIZATION OF COMPONENTS PACKING USING A GENETIC ALGORITHM

John R. Wodziak, Georges M. Fadel, Pierre Grignon
Design Methodology Group
Mechanical Engineering Department
Clemson University, Clemson SC 29634–0921 USA

ABSTRACT

This paper presents a computer based methodology to solve packing type problems, and applies it to the placement of goods in a rectangular volume such as a tractor trailer, in order to obtain a desired center of gravity. These goods are assumed to be in the form of rectangular boxes of sizes such that they individually fit in the enclosing volume. Note that we do not minimize the volume occupied by the boxes, rather we target the placement of the boxes in the container.

The methodology proposed uses Genetic Algorithms (GAs) to obtain a near optimal placement of parts. GAs use the principles of natural selection to solve this optimization problem. These principles, reproduction, crossover, and mutation, are used to construct feasible solutions from a set of candidate arrangements. Penalty methods complement the objective function to resolve the feasibility issue. The best of these candidate sets are selected and substituted in the space of possible arrangements until convergence to a solution occurs. The methodology finds a solution without any operator input, and is applied successively to one dimensional, two dimensional, and two and a half dimensional type packing problems.

INTRODUCTION

This research is the result of our interest in developing a methodology to deal with multi–objective problems encountered in complex assembly type designs. Research in the field of design methodology addresses relatively poorly the task of assembly of components. Design for Assembly [Boothroyd, 91] is a methodology that applies ease of assembly to the design of an individual component and the assembly it belongs to. The emphasis of the method is however more

on the manufacturing issues, promoting design changes to ease the burden of assembly operations. Our work assumes the component geometries are fixed, and the challenge is packing them in a volume, maximizing several objectives.

The assembly of individual components to form the final product is one of the most critical phases of the design process as its result is what the end user will see. The design of the assembly should be as optimal as possible regarding factors such as cost, time to assemble, heat, flow, electromagnetic interference and space utilization. Obtaining the best placement of components is a long and tedious task that could be significantly alleviated by computer methods. This methodology can be directly or partially applied to several other areas such as bin packing and computer chip placement on printed circuit boards. Our first attempt at the problem was to consider two objectives, the packing process, and the location of the center of gravity.

An example of the bin packing problem is the loading of tractor trailer trucks, cargo airplanes, and trains. Tractor trailer trucks are a main method of transport currently used in the United States for all types of goods. These trucks are subject to many laws and regulations that control the entire industry. One of these laws regulates the maximum weight per axle that can be carried by one of these trucks. Loading a trailer to maximize the load while balancing the per axle weight is a complex process due to the large number of possibilities that exist, even with a small number of boxes or objects. It is also important to consider the dynamic response of the truck as a whole to this loading. A balanced load provides better fuel efficiency and a more comfortable ride, thus reducing the driver's fatigue level. The same problem exists for obtaining the optimal packing of the truck that utilizes the maximum possible volume of the trailer. Heuristic methods using "rules of thumb" have been developed and are presently used in the industry REFXXX. An example of the savings that can be achieved by using these methods is illustrated by the US Army's invasion of Grenada in the 1980's. Logistics had originally estimated that ten transports would be needed to move the required equipment for the invasion. Utilizing a computer based heuristic for the packing

of the planes allowed the Army to eliminate one transport, and resulted in a cost savings of ten percent based solely on the volume of the materials to be transported.

BACKGROUND

In dealing with the truck packing problem, most approaches previously explored by researchers involve placing a box in the container, evaluating the next box, and then iterating toward a solution. Such procedures are computationally expensive and are not guaranteed to produce a global optimum as the solution. A better solution methodology is needed to ensure the global optimum is approached.

The truck packing problem is related to two classical operations research optimization problems, the knapsack problem and the Traveling Salesman problem (TSP). In the knapsack problem, the goal is to maximize the value of objects that can be placed into a sack contingent upon a weight constraint. In the truck packing problem, the objects are the boxes to be packed, and the "knapsack" is the trailer of a truck. The goal of the TSP is to minimize the distance traveled by a salesman making a comprehensive tour of a given set of cities. The Traveling Salesman Problem (TSP) is related to the order in which boxes are packed in the truck.

Amiouni et al. [Amiouni, 92] deal with a one dimensional bin packing problem. The authors develop a heuristic to balance the load in an airplane. The heuristic packs the boxes in order of non−decreasing density to minimize the moment that is being generated at the target center of gravity. It does this by packing the least dense box, then calculating a new target point for the next box. The problem is then repeated with one less box until all boxes are placed. The heuristic is applied to the one dimensional problem and our present work uses some of the ideas of Amiouni et al. as a basis for expansion to a higher dimensional space.

Aziz [Aziz, 91] presents a methodology for graphically representing boxes that are loaded into a truck using solid modeling techniques. A spatial occupancy enumeration technique is used to represent and test the boxes to see if they fit in the container. The technique allows for different sizes

and orientations of boxes. These techniques are considered during the implementation of the graphical display of the boxes once the heuristic is completed.

Gehring et al. [Gehring, 90] address the problem of packing rectangular boxes of different sizes into a shipping container in order to minimize volume wastage. They pack the boxes by order of decreasing volume in vertical slices. The spaces are filled by the best fitting group of boxes after the box which determines the size of the layer is packed. Other papers on packing to maximize space utilization include the works of [Haessler, 90][Dowsland, 93][Corcoran, 92] and [Szykman, 1994]. Corcoran uses Genetic Algorithms and shows that the solutions obtained are much better than those achieved through traditional methods. Again, the maximization of space utilization is the objective. Szykman and Cagan, and Dowsland use Simulated Annealing to closely pack objects and minimize wasted space.

In many of these cases, heuristics are used to drive the optimization process. These heuristics have to consider some physical constraints dictated by the problem, and thus must include constraints in the problem solution methodology. Unfortunately, heuristic based methods do not lend themselves easily to solving constrained optimization problems. Richardson et al. [Richardson, 1989] included penalty functions in genetic algorithms to solve constrained optimization problems. They recommend that penalty functions employ information about the degree of constraint violation instead of just the number of violated constraints, since some of the information concerning the particular solution needs to be captured.

This background provides the impetus for this research. The paper presents an overview of genetic algorithms, and then describes the particular solution methodology used to solve the problem in one, two, and two–and–a–half dimensions. The solution methods are applied to test problems, and conclusions are drawn.

GENETIC ALGORITHMS

OVERVIEW OF GENETIC ALGORITHMS

Genetic Algorithms (GAs) [Goldberg, 89] are search algorithms based upon natural selection or Darwinism. A GA is directly derived from the behavior of genes and chromosomes in nature. Although robust, a GA is a simplified model of the process that occurs in nature.

Each "Generation", or family of possible solutions, is made up of a set of strings or "chromosomes". Each chromosome is in turn made up of individual "genes". These genes are codings of design variables that are used to evaluate the function being optimized.

The GA calls a subroutine to compute the fitness value or normalized objective value for each chromosome in a population. These values are compared and evaluated relative to each other and the chromosomes with the best values "survive" and pass on to the next generation. In each generation, there is a small probability of each chromosome mutating (or changing) in one or more positions in the string(gene). There is a different, much higher probability that two strings will mate or crossover to produce a child. The children and mutations are placed into the next generation and the procedure iterates. The process of mutation, crossover, evaluation and reproduction are repeated until there is convergence to a suitable solution to the problem.

The use of Genetic Algorithms is based upon several general assumptions. The first assumption is that the problem environment can be represented by a fixed length string of symbols. "This string serves as the 'genetic material' with specific positions (loci) on the string (chromosome) containing unique symbols or tokens (genes) taking on values (alleles). " [Austin, 90]. It is assumed that each individual string represents a unique point in the search space. At each iteration in the search process, the system has a fixed population of strings that represent the current solutions to the problem. The only feedback to the GA is the evaluation (fitness) value of the individual strings. Iterations in the solution process are measured in discrete intervals of time called generations. No

deductive or auxiliary information is required by the GA. However, information which aides in the solution may be considered by the process.

GAs differ from traditional methods of search and optimization in four ways [Holland, 75]:

- GAs work with a coding of the parameters and not the parameters themselves.
- GAs do not search a single point but work from a population of points.
- GAs evaluate the function directly, and do not use derivatives or other auxiliary knowledge.
- GAs utilize probabilistic transition rules as opposed to the deterministic rules found in traditional methods of optimization.

These fundamental differences allow GAs to be more robust than "normal" search and optimization routines such as gradient based algorithms, point to point search and heuristic approaches.

The first step in using GAs is to code the problem space. This is done by mapping the design variables into chromosomes through the use of an arbitrary alphabet. The alphabet of choice for GAs is binary, or 1's and 0's. Use of binary codings allows the GA to associate high fitness with similarities in the string. Other codings, while viable, require a more expensive decoding process, and may not allow the GA to recognize similarities in "good" members of the population [Goldberg, 89].

GENETIC OPERATORS

Selection

Selection, or reproduction, is the process of creating a new generation. This is accomplished by copying strings or individuals from the last generation into the population of the new generation, based upon the evaluation of the individual's assigned fitness value. One method of selection for this methodology is called Rank Based Selection. This procedure sorts the population based on fitness value. From this sorted population, a number of offsprings are assigned to the new generation based on their position in the sort.

Crossover

Crossover is the operator responsible for introducing most new solutions to the population. This is done by selecting, at random, two parent strings. These strings are then crossed to produce two offspring. This process of sharing information between different strings gives GAs much of their power.

Different methods of crossover can be used. One such simple procedure is the one point crossover. Two strings are chosen at random from the population. The genetic information contained in the strings is then swapped on one side of the crossing points [Goldberg, 89] (Figure 1).

Initial strings
00000000
11111111

Cross point determined
00000 | 000
11111 | 111

Crossover completed
00000111

Figure 1. One Point Crossover

Mutation

Mutation is the process of randomly changing one bit of information in the string. The change is accomplished by changing a single gene from 0 to 1 or 1 to 0. Mutation prevents the GA from loosing potentially important information or stagnating during the solution process. Mutation has the effect of providing a push from a particular solution which may be a local optimum rather than the global one sought. Mutation is considered to be a secondary operator to crossover and reproduction that provides an insurance policy against premature convergence and loss of information [Goldberg, 89].

DEVELOPMENT OF THE ALGORITHM

The first step in the development of the algorithm is the selection of the appropriate coding for the definition of the variables. For the one dimensional case, boxes are initially assumed to be of fixed size and weight. This simplification allows a progressive increase in complexity from the one−Dimensional case to the two−and−a−half−Dimensionalcase with variable weights. With this assumption, the center of gravity of the entire load of boxes can be computed by the following formula:

$$COG_x = \frac{\sum_{i=0}^{N} X_i \; W_i}{\sum_{i=0}^{N} W_i}$$

where W_i is the weight of each box, and X_i is the centroid location of each box. The load from an individual box is assumed to be a point load at the centroid of the box. Since the size of the boxes has to be taken into account when loading the truck, the initial set of design variables selected is the distance between the boxes. Note that this selection eliminates the need to compute interferences between boxes, and results in significant savings in execution time. Also, since all boxes are of the same size and weight, the order of the boxes is irrelevant at this point. Thus the initial bitstring coding selected has to represent the space between each two boxes, and has to vary between 0 and some maximum distance selected based upon the number of boxes loaded, their size and the size of the trailer. This space between two boxes is rounded to the nearest inch to result in an integer representation which can be easily coded into the Genetic Algorithm. Note that this restriction is lifted with some GA codings [REF XXXX}. For a maximum spacing of 31 inches, a five bit binary gene coding can be used. This coding allows each gene to represent a design variable which can be between 0 and 31 inches in size.

Figure 2. Sample Bitstring for One−dimensional Problem.

Figure 2 illustrates a sample bitstring for a one dimensional problem restricted to five boxes.

10010	10011	11111	00000	00101
Box 1	Box 2	Box 3	Box 4	Box 5

Box 1 X Offset = 18”
Box 2 X Offset = 19”
Box 3 X Offset = 31”
Box 4 X Offset = 0”
Box 5 X Offset = 5”

Even though this problem might not be very realistic, the constraint that all the boxes have to fit within the bounds of the trailer is added to the fitness or optimization criterion to ensure valid results. The penalty imposed is a quadratic penalty of the form:

$$\text{if } distance_i > MAXLENGTH - BoxSize_i \text{ Then}$$
$$Penalty_i = Penalty_{i-1} + (distance_i + BoxSize_i - MAXLENGTH)^2$$

and the algorithm for the one dimensional loading can be described as follows:

Figure 3 Graphical Depiction of 1D algorithm.

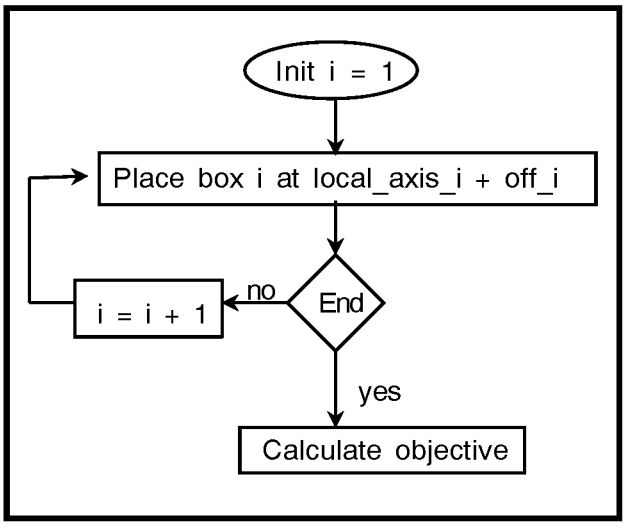
Note that various penalties functions were evaluated . Example such evaluations are linear, square and square root functions of the excess distance out of the trailer. The square function performed best. The algorithm illustrated in Figure 3 can be described as follows:

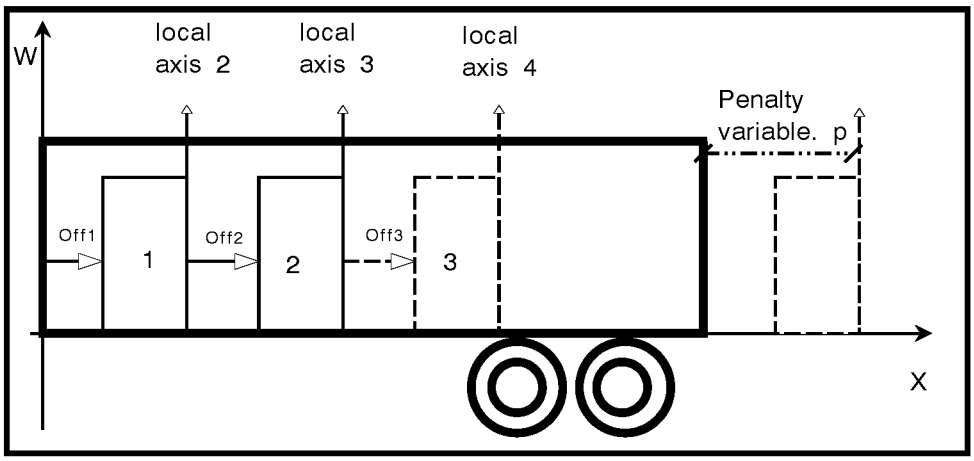
1. Take the first box and place it x_1 inches from the closed end of the trailer. x_1 is the distance calculated from the binary string.
2. Add the centroid location of the last box placed to the location summation.
3. Take the next box and locate it at x_n inches plus the box size from the last box.
4. Add the box weight to the total box weight.
5. Repeat steps 2 through 4 until all of the boxes are placed.
6. Calculate the center of gravity by dividing the location summation by the total weight of the boxes.
7. Find the absolute value of the difference in the desired center of gravity versus the one found in 6.
8. Establish the penalty for this solution (if any)
9. Set the evaluation of this solution to be the difference in centers of gravity plus the penalty.

When different sized boxes are considered, the formula is adjusted in step 4.

TWO DIMENSIONAL ALGORITHM

The first two−dimensional algorithm developed for this methodology involves packing the boxes on the flatbed of the truck. The boxes are considered to have length and width but their height is ignored. The boxes are packed on the X−Y plane, defined to be on the truck platform at Z = 0.0, and the desired center of gravity has X and Y components. The X dimension is defined along the largest dimension of the trailer, from back (closed end) to front, and Y is defined across the floor of the





trailer, from left to right looking into the open trailer. The desired center of gravity remains the same in the X direction, and is located at the center of the trailer in the Y direction. The two components of the COG have to be computed independently, and an offset for each box in each of the directions is needed. Thus a distinct five bit string is assigned to each box in each direction. The algorithm for the two dimensional case is listed below. The fitness function is the sum of the differences in the centers of gravity for both directions plus any penalty incurred. The algorithm used is as follows:

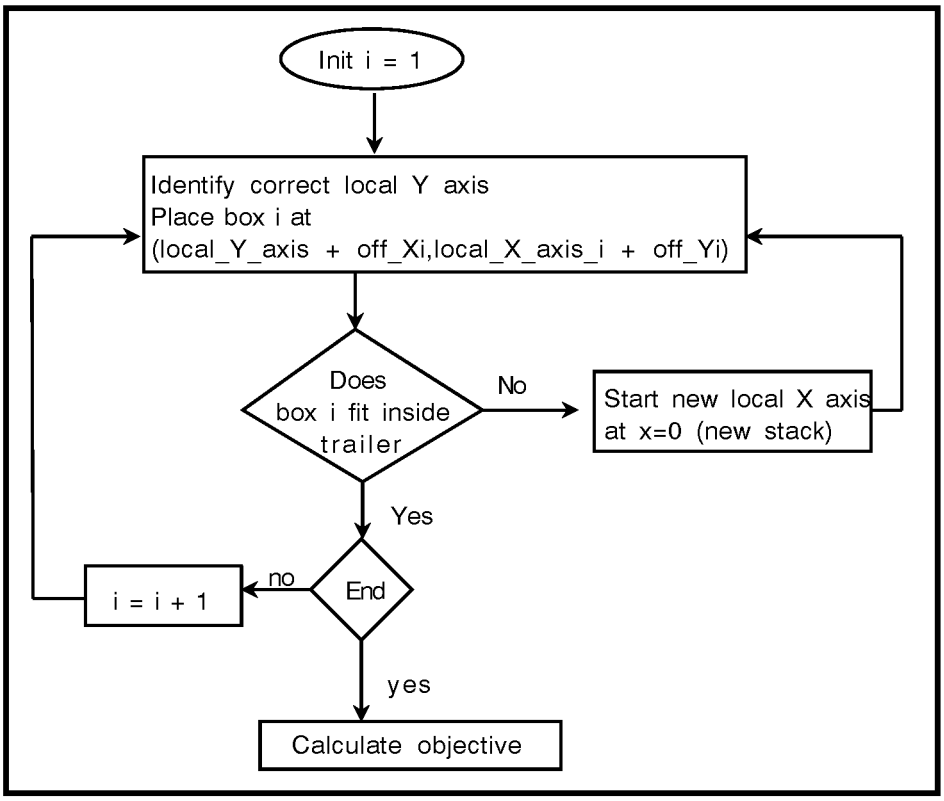
1. Take the first box and place it x_1 inches from the back of the trailer (closed end) and y_1 inches from the left wall. x_1 and y_1 are the distances calculated from the binary string.
2. Check that (y_{i-1} + y_i + the size of the box in the Y direction) fits within the trailer. It yes, go to 4.
3. Start a new row. Set y_{i-1} to 0 for the current box only.
4. Find the maximum X distance used in the range $y_i \le Y \le y_i$ + BoxsizeY_{i-1}.
5. Place the box at maximum X plus x_i and y_{i-1} + BoxsizeY_{i-1} + y_i .
6. Set the maximum value for the X range of the placed box to the actual placement plus Boxsize X_i.

7. Add the box weight to the total.
8. Add the X centroid position to the X location summation.
9. Add the Y centroid position to the Y location summation.
10. Repeat steps 2 through 9 until all of the boxes are placed.
11. Calculate the center of gravity for the entire load (X and Y components)
12. Compute the absolute value of the difference between the desired center of gravity and the one found in 11.
13. Establish the penalties for this solution (if any)
14. Set the evaluation of this solution to be the sum of the differences in centers of gravity for both directions plus any penalty incurred. For the 2D case, the penalty function is the sum of the penalties for the X and Y dimensions. The X penalty is the same as the 1D case, the Y penalty is also the same, except that it is applied to the width of the trailer.

The next step is to randomize the order in which the boxes are placed within the trailer and determine the optimum order. This problem is similar to the classical optimization problem of the Traveling Salesman (TSP). It is also similar to other scheduling situations encountered in various applications such as job shop scheduling, or computer processor time scheduling.

Most existing solutions of these problems in the literature put an emphasis on the crossover operator to obtain a solution. The theory behind this reasoning is that in nature, the information carried by an allele is independent of the allele's position in the chromosome. This is different in traditional GAs. In traditional GAs, the position of an allele in the chromosome determines part of its meaning. What is needed to reorder the boxes is a means of numbering the boxes, and then changing the position of that representation in the artificial chromosome without changing its meaning to the GA. In nature, the primary method of changing the position of an allele is inversion. Under this process, two points are chosen along the chromosome. A cut is made at these points, and the ends of the chromosome switch places. In GAs, this can be formulated through the use of the crossover operator [Goldberg, 89].

Use of the crossover operator is unfortunately not just an application of an algorithm. In fact, the traditional n−point crossover operators have a high tendency to produce non−viable solutions due to omission and duplication of strings [Fox, 91]. Much research in this field focuses on developing analogs of traditional operators to satisfy general principles and restrictions inherent in the problem of order [Fox, 87]. This research has led to the development of Order crossover, Edge recombination and other methods.



2D

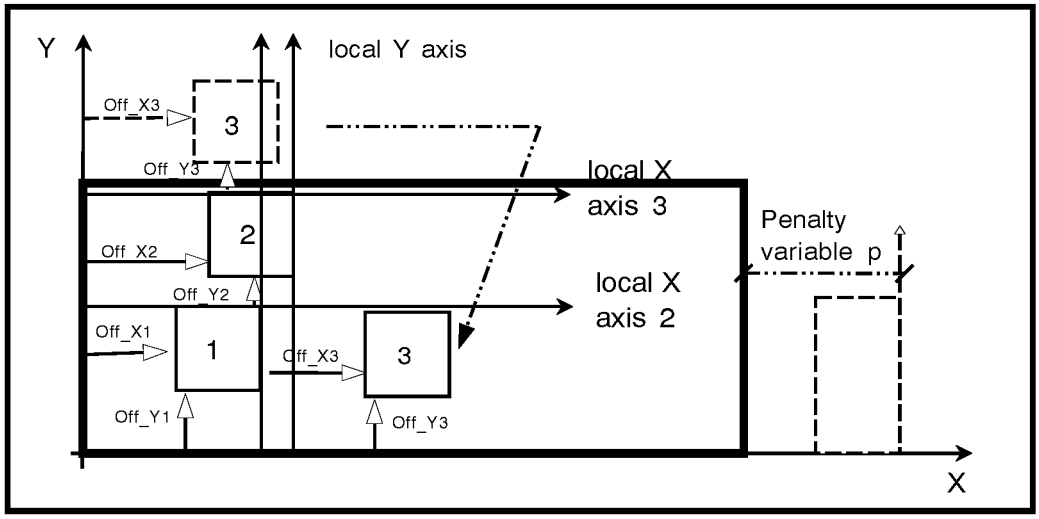


Figure 4. 2D Box Packing Illustration

The crossover method applied in this research is the Order Crossover (OX) [Corcoran, 92][Fox, 91] [Oliver, 87]. OX involves two parents creating two children at the same time. This operator allows the order in which the boxes are placed into the truck to be changed. After determining the parents for the process, the basic algorithm is as follows:

1. Two random cut points are determined.
2. Then the string positions between the cuts are placed directly into the child.
3. Starting after the second cut point and proceeding to the end of the string, the symbols are copied from the second parent into the first child, and from the first parent into the second child. In this process any symbols already present in the child are omitted from this copying operation.
4. When the end of the string is reached, the copying process is continued at the beginning of the string and continues until all positions are filled.

[Corcoran 93] showed that Order Crossover (OX) is the best crossover operator for this type of problem. OX is implemented to change the sequence in which the boxes are placed into the trailer. This is accomplished by dividing the bitstring into two portions. The first portion contains a string of sufficient length to represent the number of boxes under consideration. The second portion contains the offset distances between the boxes. The first portion of this string is mapped from a binary representation into a set of integers. These integers undergo order crossover to generate two new sequences and are re−mapped into binary format for further operations.

Box Rotation

The next stage of the algorithm development is to allow for the boxes to be rotated before they are placed into the truck which should result in a better packing. The rotation of the boxes is accomplished by evaluating a rotation bit assigned to each box. If this bit is "1" then the X and Y sizes of the box are swapped. This assumes that a box cannot be laid on its side. After this point the algorithm proceeds in the same fashion as the one described earlier for the two dimensional packing without rotation. This algorithm is given in a formalized manner below; it is described in more details in [Wodziak, 94].

1. Determine the next Box Number, n
2. Check if box is to be rotated. If it does, swap the X and Y size of the box
3. Check to see that $(Y_i + BoxsizeY)$ is within the imposed limits. If it is, go to step 5
4. Start a new row. Set $Y_i = 0.\ 0$
5. Find the maximum X distance used for the range $Y_i \leq Y \leq Y_i + BoxsizeY$
6. If the maximum X + BoxsizeX is outside the truck, mark the box as unused. Set X_i and Y_i equal to the last packed box. Go to Step 11

7. Place the box at maximum X and Yi. Increment the number of boxes packed by 1
8. Set the maximum X value for the range of the placed box to be equal to the placement plus the BoxsizeX
9. Add the box weight to the total.
10. Add the X centroid position to the X location summation.
11. Add the Y centroid position to the Y location summation.
12. Repeat steps 2 through 9 until all of the boxes are placed.
13. Calculate the center of gravity for the entire load (X and Y components)
14. Find the absolute value of the difference in the desired center of gravity versus the one found in 13.
15. Establish the penalties for this solution (if any).
16. Set the evaluation of this solution to be the sum of the differences in centers of gravity for both directions plus any penalty incurred.

TWO AND A HALF DIMENSIONAL ALGORITHM

The next phase of the project is the ultimate goal of this work, packing multiple layers of boxes into the truck. The boxes at this stage are packed in three dimensions, but the center of gravity is calculated and compared to the target center of gravity in two dimensions. The height difference in the desired and calculated centers of gravity is not considered. The boxes are considered to have height, length and width, and are of various weights. The sequence in which these boxes are placed is determined from the first portion of the bitstring which is manipulated by Order Crossover.

The boxes are packed along the X and Y axes in the same manner as the two dimensional algorithm. When the first level (Z=0) on the truck platform is filled, a new layer of boxes is started at (0, 0, h_max_k). h_max_k is determined as the maximum height of all the previously packed boxes below the current box. This is continued until all boxes are placed. Note that the layers of boxes are not of uniform height since only the maximum height under a specific box is considered at one time. Also, this algorithm will result in some boxes resting on parts of boxes, which might not be very stable.

Penalties are assessed for violations of the trailer dimensions in the X, Y and Z directions. However, only the Z direction violation results in a penalty since the algorithm guards against extension past the X and Y limits.

The algorithm for this packing is formalized below:

1. Determine the next Box Number, n_i
2. Check if $\{ Y_{i-1} + BoxsizeY_{Ni-1}\} + Y_i + BoxsizeY_{Ni}$ } is within the limits of the trailer. If the box fits and i>1, goto step 4.

3. Start a new row. Set $Y_{i-1} = 0$ and $BoxsizeY_{Ni-1} = 0$ for the current box only.
4. Find the maximum X distance used for the range $Y_i \leq Y \leq Y_i + BoxsizeY_{Ni}$.
5. If the maximum $X + BoxsizeX_{Ni}$ is outside the trailer, start a new layer. Set Y_i equal to 0, got step 3.
6. Find the maximum height used for the area under the current box in the range $X_i \leq X \leq X_i + BoxsizeX_{Ni}$ and $Y_i \leq Y \leq Yi + BoxsizeY_{Ni}$
7. Place the box at the maximum X, Y_i and maximum height.
8. Set the maximum X value for the range of the placed box to be equal to the placement plus the $BoxsizeX_{Ni}$.
9. Update the maximum Height for the area of the box.
10. Add the weight of the box to the total weight.
11. Compute the updated X and Y Centers of gravity.
12. Repeat steps 1−11 until all the Boxes are considered.
13. Calculate the difference between the desired Center of Gravity and calculated Center of Gravity in the X and Y directions.
14. Establish the penalties for this solution (if any).
15. Set the evaluation of this solution to be the absolute value of the differences in X and Y coordinates of the centers of gravity plus the penalties if any.

RESULTS

Applications of these algorithms to truck packing problems are presented below in increasing order of complexity.

ONE DIMENSIONAL EXAMPLE WITH BOXES OF VARIOUS WEIGHTS

In order to verify the algorithms developed, problems of known solution were first tested. These problems consisted in packing boxes of decreasing sizes in the truck. The original ordering was random, and after executing the algorithm the solution obtained was compared to the analytical solution. Figure

The first implementation of the problem involves packing 21 boxes in a Fruehauf trailer Model # FB−9+1.5−LP−NF2−48[Fruehauf, 1993]. The dimensions of the trailer are approximately 570 inches long by 96.5 inches wide and 109 inches height. The boxes used are randomly generated with sizes between 10 inches and 36 inches in all directions, the weights are also random, and range between 20 and 100 pounds. The desired center of gravity is computed for a particular position of the rear wheel axles in the X direction, and at the center of the trailer in the Y direction (Figure 5).

The best solution for this case was produced in 283 generations and took approximately 10 minutes to run on a Sun SPARC IPC. The solution found was 39.28 inches away from the desired

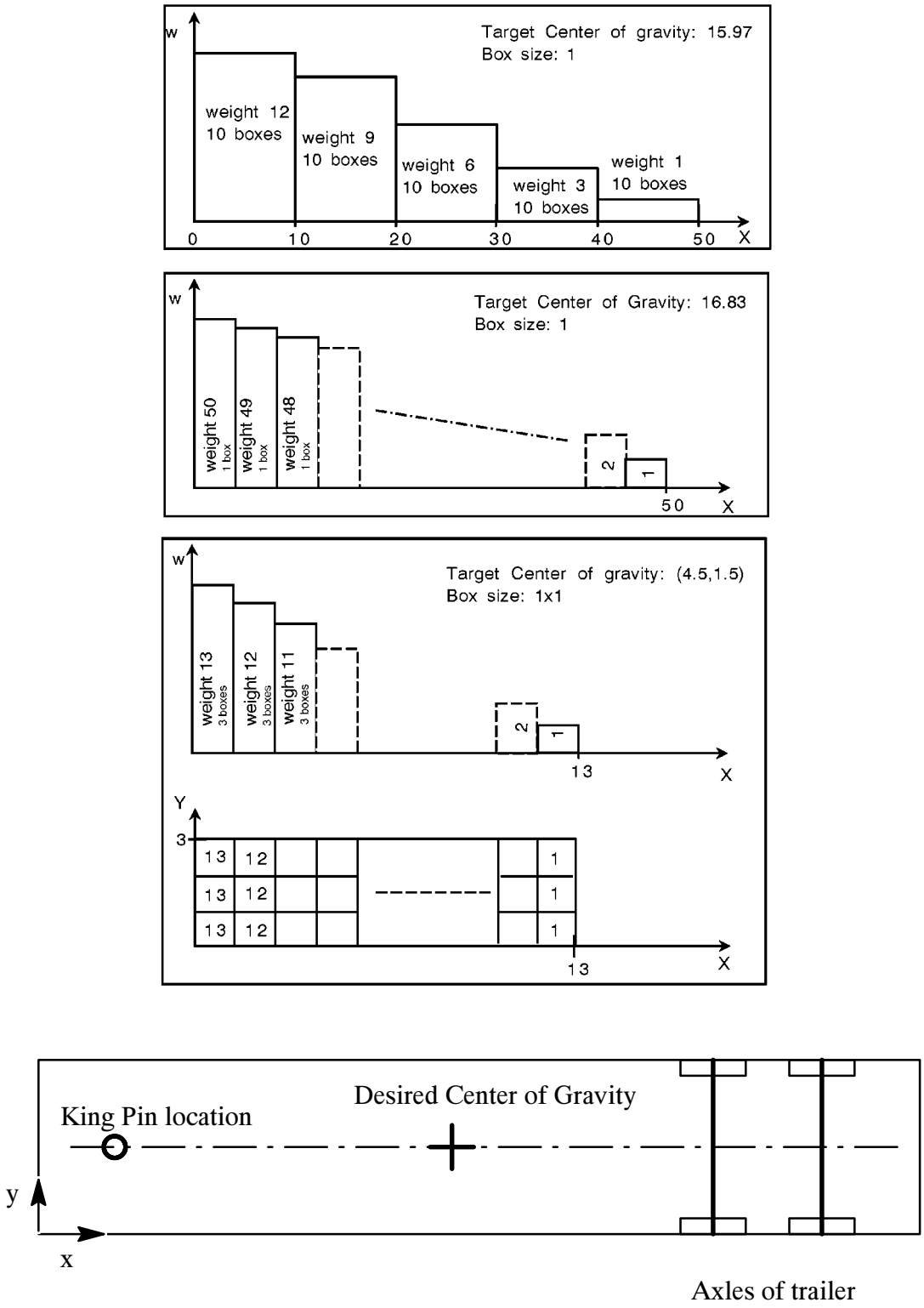


Figure 5. Top view of the Trailer

center of gravity. Figure 3 illustrates this result. As can be seen, because of the weight distribution of

the boxes, and because of the fixed relative location of the boxes, this is the best packing possible. In this figure, the height of the boxes is proportional to the their particular weight..

The second implementation allows reordering of the boxes, and the Order Crossover operator is used. The same problem as the one displayed in Figure 3. is solved, and Figure 4. illustrates the new result. As a result of the ability to move the boxes with respect to each other, the difference in centers of gravity (calculated and desired) was reduced to 0.048 inches. In this solution, the algorithm packed the heavier boxes (as shown by their height) towards the open end of the trailer which resulted in a much more accurate location of the center of gravity. These results prove the effectiveness of the re−ordering methodology and of the Order Crossover operator. The numbers below the boxes correspond to the order in which the data is read into the program.

Figure 4. One Dimensional Packing with Reordering

TWO DIMENSIONAL CASES

The first two dimensional case investigated packing boxes of the same size but different weight. The algorithm was used to pack 66 different boxes in the order they were read into the program. The best result obtained was off by 55.6 inches in the X direction and 4.8 inches in the Y direction, and no penalty was assessed. The second case tested relaxed the limitation on same size boxes and was able to bring the calculated center of gravity to within 33 inches from the desired COG in the X direction, and 6.4 inches in the Y direction. By reordering the boxes when placing them in the trailer, the algorithm achieved an accuracy of 0.22 inches in the X direction and 0.12 inches in the Y direction. This packing was achieved in 2004 generations and is displayed in Figure 5.

Figure 5. Two Dimensional Algorithm with Box Reordering

TWO AND A HALF DIMENSIONAL CASE

The last trailer packing problem presented involved packing the boxes in three dimensions, but ensuring the location of the calculated center of gravity matched the desired COG location in the X and Y coordinates. The boxes have different weights and sizes, and their packing order is determined by the algorithm through the Order Crossover operator. The results obtained are within 0.02 inches in the X direction and 0.04 inches in the Y direction. In this example, the height of the

boxes is read from the data file, and displayed accordingly. There is no relation between the height of a box and its weight. Figure 6 illustrates this result. Note that 40 boxes are used in this example, and 1066 generations were needed to achieve convergence.

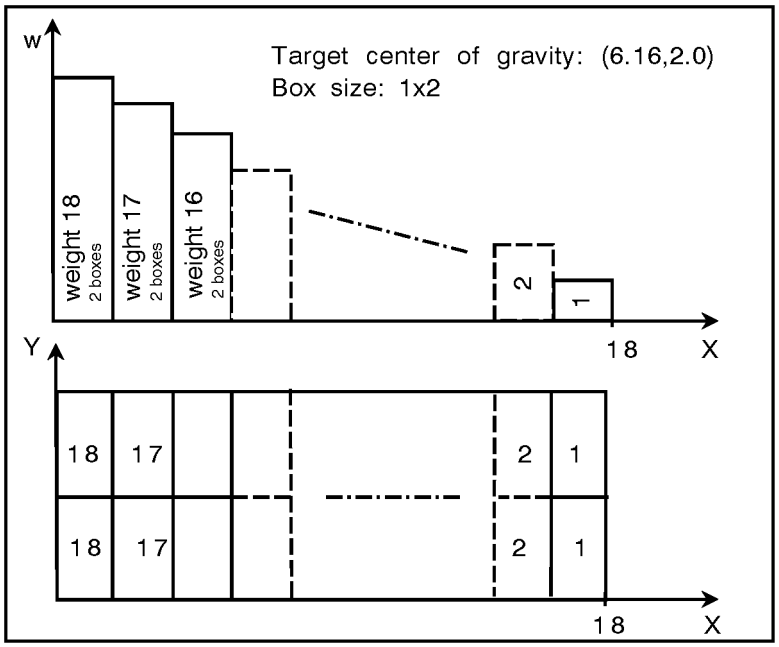


Figure 6. Two and a Half Dimensional Algorithm

CONCLUSIONS

The algorithms presented in this paper provide viable solutions to problems that are NP complete. These solutions are obtained through Genetic Algorithms. The solutions achieved are found from a solution space that can be impossible to search through traditional heuristic methods, especially if only a small percentage of the solutions are viable. The selection of the space between the boxes as the design variable alleviate the computational needs since no interference checking is required. The binary gene coding of the problem can become very long, especially if the number of boxes is high. This coding requires 5 bits per box per dimension considered plus an additional rotation bit and order number per box. This long string needs to be broken down to perform Order Crossover on one portion and regular crossover on the remaining part. The additional computational

burden is significant, but the most complex problems considered (66 boxes, 2 D) took around 2 hours and 44 boxes, 2.5 D took around 8 hours to converge to an acceptable solution.

BIBLIOGRAPHY

[Amiouni, 92] Amiouni, S., Bartholdi, J.J., Vande Vate, J.H., Zhang, J. "Balanced Loading" Journal of Operations Research, March, 1992, pp 238.

[Austin, 90] Austin, Scott. "An Introduction to Genetic Algorithms," AI Expert, March 1990, pp 49–53.

[Aziz, 91] Aziz, N. M., "A Computer–Aided Box Stacking Model for Truck Transport and Pallets", Computers in Industry, 1991.

[Boothroyd, 88] Boothroyd, G. and Dewhurst, P. "Product Design for Manufacture and Assembly", Manufacturing Engineering, April 1988, pp42–46.

[Corcoran, 92] Corcoran, Arthur L. and Wainwright, Roger L. , "A Genetic Algorithm for Packing in Three Dimensions," Proceedings of the 1992 ACM/SIGAPP Symposium on Applied Computing. Kansas City, Missouri, March 1992.

[Corcoran, 93] Corcoran, Arthur L. and Wainwright, Roger L. , "LibGA: A User Friendly Workbench for Genetic Algorithms," Proceedings of the 1993 ACM/SIGAPP Symposium on Applied Computing. Indianapolis, Indiana, February, 1993.

[Dowsland, 93] Dowsland, K., "Some Experiments with Simulated Annealing Techniques for Packing problems" European Journal of Operations Research, Vol 68, 1993, pp 389–399.

[Fox, 91] Fox, B. R. and McMahon, M. B. , "Genetic Algorithms for Sequencing Problems. " Foundations of Genetic Algorithms. ed. Gregory J. E. Rawlins. Morgan Kaufman Publishers, 1991.

[Fruehauf, 93] Fruehauf Corporation, personnal communication.

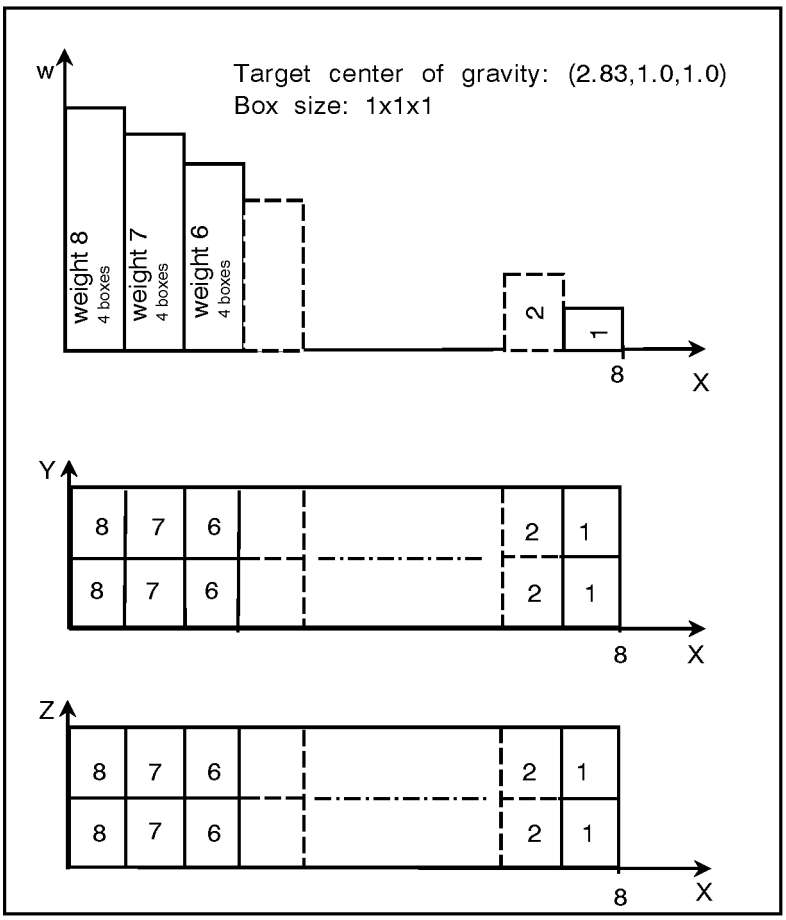
[Gehring, 90] Gehring, H., Menschner, K. and Meyer, M., "A Computer Based Heuristic for Packing Pooled Shipment Containers", European Journal of Operations Research, January 1990, pp 277–288.

[Goldberg, 89] Goldberg, David E. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison–Wesley, 1989.

[Haessler, 90] Haessler, R.W., and Talbot, F.B., "Load Planning for Shipment of Low Density Products", European Journal of Operations Research, January 1990, pp 289–299.

[Holland, 75] Holland, John H. "Adaptation in Natural and Artificial Systems", University of Michigan Press, 1975.

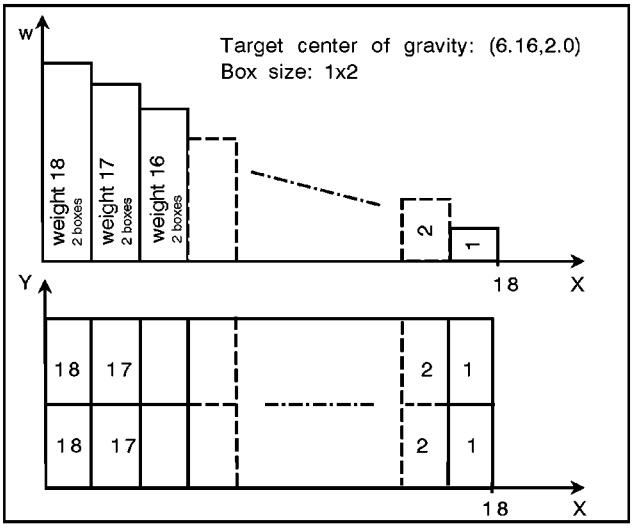
[Oliver, 87] Oliver, I. M. , Smith, D. J. and Holland, J. H. "A Study of Permutation Crossover Operators on the Traveling Salesman Problem. " Genetic Algorithms and their Applications: Proceedings of the Second International Conference. 1987



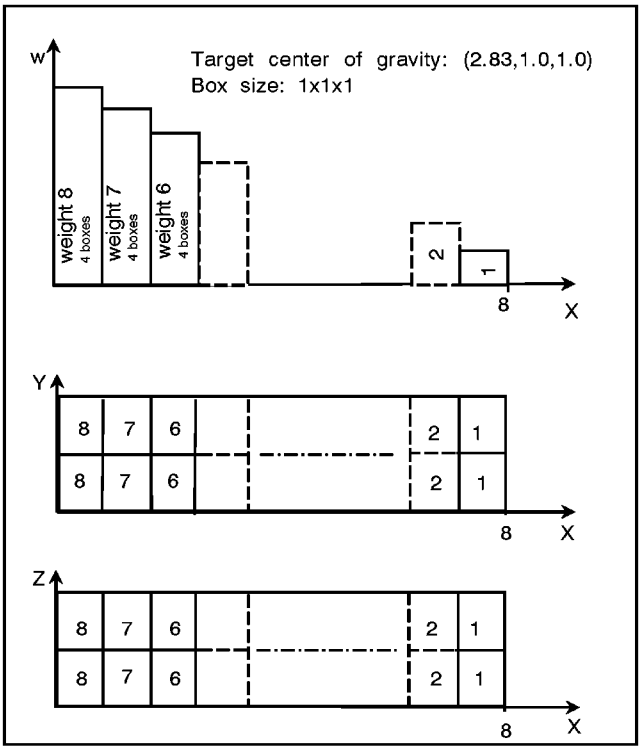
[Richardson, 89] Richardson, J.T., Palmer, M.K., Liepins, H., Hilliard, M. “Some Guidelines for Genetic Algorithms with Penalty Functions” Proceedings of the Third International Conference on Genetic Algorithms, 1989.

[Szykman, 94] Szykman, S. and Cagan, J., “Constrained Three Dimensional Component Layout using Simulated Annealing”, Paper presented at the ASME International Congress and Exposition, Chicago, November, 1994.

[Wodziak, 94] Wodziak, John R. , ”Optimal Packing Utilizing Genetic Algorithms. ” Master of Science Thesis, Clemson, SC, 1994.


w
Target center of gravity: (6.16,2.0)
Box size: 1x2
weight 18
2 boxes
weight 17
2 boxes
weight 16
2 boxes
2
1
18
X
Y
18 17 2 1
18 17 2 1
18
X

2.5D


w
Target center of gravity: (2.83,1.0,1.0)
Box size: 1x1x1
weight 8
4 boxes
weight 7
4 boxes
weight 6
4 boxes
2
1
8
X
Y
8 7 6 2 1
8 7 6 2 1
8
X
Z
8 7 6 2 1
8 7 6 2 1
8
X

3D