The Dissertation Committee for John Michael Harwig
Certifies that this is the approved version of the following dissertation:

AN ADAPTIVE TABU SEARCH APPROACH TO CUTTING AND PACKING PROBLEMS

Committee:

J. Wesley Barnes, Supervisor

James T. Moore

Elmira Popova

John J. Hasenbein

Erhan Kutanoglu

Bruce W. Colletti

An Adaptive Tabu Search Approach to Cutting and Packing Problems

by

John Michael Harwig, B.S., M.S.E.

Dissertation

Presented to the Faculty of the Graduate School of

the University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

December, 2003

Dedication

To Monica, Michael and Anne Marie

Acknowledgements

I would like to thank my advisor, Dr. J. Wesley Barnes, whose insightful guidance and unwavering support made this dissertation possible. I would also like to thank all of my friends and study partners from the "War Room" at UT for their help and support during the completion of this work. I am grateful to my parents, Dennis and Mary Ann Harwig and my in-laws, Jerry and Mary Johnson for their encouraging words and a helping hands when I needed it most. I am extremely grateful to be blest with the world's two best children, Michael and Anne Marie. Thank you for understanding when you were asked to temporarily leave your friends, pets and school in Texas and move to Virginia. I wish to extend a sincere heartfelt thank you to my loving wife, Monica, who selflessly took our children with her while on active duty fighting the war on terror; I shall never forget your sacrifice.

An Adaptive Tabu Search Approach to Cutting and Packing Problems

Publication No._____

John Michael Harwig, Ph.D.

The University of Texas at Austin, 2003

Supervisor: J. Wesley Barnes

This research implements an adaptive tabu search procedure that controls the partitioning, ordering and orientation features of a two–dimensional orthogonal packing solution, details an effective fine-grained objective function for cutting and packing problems, and presents effective move neighborhoods to find very good answers in a short period of time. Results meet or exceed all other techniques presented in literature for a common test set for all 500 instances.

These techniques have natural extension into both two dimensional arbitrarily shaped and three dimensional orthogonal packing heuristics that use rules based on given partitions, orders and orientations. Techniques to extend this research into those new horizons and methods that might improve the search are also presented.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The goals of this research are to develop a new framework which can be used to model and analyze the general class of cutting and packing problems and to demonstrate the usefulness of this framework by implementing it to solve a 2 dimensional bin packing problem using adaptive tabu search (ATS).

1.1 MOTIVATION

A group theoretic perspective on tabu search has shown promise for solving problems related to the Traveling Salesmen Problem (TSP) and Vehicle Routing Problem (VRP). Recent applications of "group theoretic tabu search" (GTTS) include the Aerial Fleet Refueling Problem (Wiley 2001), the Theater Distribution Vehicle Routing and Scheduling Problem (Crino 2002), and the Crew Scheduling Problem (Combs 2002). These types of problems are critical in the efficient movement of items, but a key piece of the movement puzzle is still lacking. In addition to being concerned with the movement of vehicles (trucks, aircraft, and ships) practical logistics must consider the efficient and effective packing of materiel. From a military perspective, combat aircraft and warships 'self-deploy' and everything else must be packed. Frequently in current practice, packing occurs in hierarchical levels. First, small items are loaded onto pallets that are then loaded into larger containers or vehicle cargo areas. Then these larger containers or vehicles may be loaded into ships and/or aircraft along with

larger, irregular shaped items that did not require or allow pre-packing. Two possible military goals could be to ship everything in the fewest possible containers (a multi-dimensional bin packing problem) or ship the highest priority materiel with the vehicle assets available (a geometric multi-dimensional multi-knapsack problem). These two problems are present both in the deployment phase of an operation and in the everyday shipment of materiel at every level of the military transportation system.

The armed forces have automated the process for ordering and designating items for shipment. Thus, at any one time lists of items exist that must be shipped from one location to another. However, the automation that supports the loading of these items does not provide a standard strategy to obtain effective and efficient packing solutions. Current load planning is performed either by ad hoc methods or by trial and error. An automated system that provides high quality packing solutions within very short implementation horizons is greatly needed.

With fewer military personnel stationed overseas and the rise in operations tempo (OPTEMPO), a key piece to the National Military Strategy depends on Force Projection. The headquarters responsible for projecting our military forces is the United States Transportation Command (USTRANSCOM), whose mission is "to provide air, land and sea transportation for the Department of Defense both in time of peace and in time of war". The only way to improve projection capabilities without purchasing more strategic lift assets is to improve efficiency. The desire for systems to achieve high efficiency is evident in the USTRANSCOM vision for the future:

"*USTRANSCOM, providing timely, customer-focused global mobility in peace and war through efficient, effective, and integrated transportation from origin to destination."* (http://www.transcom.mil/missions/mission.html)

This research proposes a S_n representation of the solutions to multi-dimensional cutting and packing problems. The intended application arena is military materiel loading. More specifically, it offers a S_n perspective for an ATS approach for the packing of two dimensional (2-D) orthogonal items, addresses extensions for (1) the packing of 2-D arbitrary shaped polygons, (2) three dimensional (3D) orthogonal packing, and (3) the complexity added by constraints due to hazardous cargo, weight and balance, and load stability. The packing of 2-D arbitrary shaped polygons items is applicable to Aircraft Loading. The 3-D rectangular case is applicable to pallet and container loading problems.

1.2 CUTTING AND PACKING PROBLEMS

1.2.1 CUTTING AND PACKING PROBLEM DESCRIPTION

Cutting and Packing problems have essentially the same logic structure. Both problems require the efficient use of collections of large spaces, *objects,* that must include a much more numerous heterogeneous collection of smaller *items*.

Dyckhoff and Finke (1992) emphasized the duality of cutting and packing problems. For packing problems, the larger objects are empty and must be filled with smaller items. For cutting problems, objects are divided into items. One-, two-, or three-dimensional cutting and packing problems have essentially the same objective and structural constraints. The intersection of the space that any two items occupy must be zero and the union of all items must be a subset of the union of all objects. Most of the literature for these two areas has been on problems with 2D rectangular, or 3D rectangular items. However, recent efforts in two dimensions have focused on the cutting or packing of a set of heterogeneous irregular shaped items.

Dyckhoff and Finke (1992) classify cutting and packing problems with five characteristics. *Dimensionality* defines the minimum number of dimensions necessary to describe the geometry of a feasible solution. Problems of more than three dimensions may be obtained if the problem is expanded to non-spatial concerns (e.g. weight or time). *Type of assignment* describes whether all items or all objects are included in the assignment. *Characteristics* of items and objects include considerations about the items' shape and orientation. *Pattern restrictions* place conditions on the assignment and spatial arrangement of items. This includes geometric characteristics, operational characteristics, and allocation dynamics. The final area to consider is the *objective*, i.e., the metric used to determine solution quality.

1.2.2 Problem Statement

The following information is known:

The physical attributes of all items including:

- Length , width, height or *x, y, z* coordinates for each 'corner'
- Weight
- Location of the center of gravity
- Packing restrictions on orientation (e.g. "this end up")
- Restrictions on mixing of hazardous materiel (e.g. corrosives and explosives are not in the same container)
- A value or utility associated with the item or some prioritization

The physical attributes of all objects (containers) which includes:

- Spatial dimension
- Weight restrictions
- Center of Balance restrictions

The desired metric, such as:

- Maximize the "utility" of what is packed in a set of containers (Knapsack)
- Minimize the number of containers (Bin Packing)
- Minimize the amount of unusable space (Trim Loss)

Thus a modeling formulation for cutting and packing problems normally takes the following form:

Optimize the selected metric

Subject to:

Container Width, Length, Height, Weight, Balance and Stability limitations

Hazardous Materiel considerations

Specialized Constraints

Since length, width, and height are interdependent, it is difficult to formulate these problems using classical mathematical programming methods. For example, the heights added depends on the *x, y* location of the boxes and whether or not they are aligned vertically. The following relationships generalize feasibility requirements in the spatial realm:

Let S_{Total} be the space bounding the volume of object *i* in R^3 and let $S(x_{ij})$ and $S(x_{ik})$ be the space occupied by the *j* and *k* items in object *i* where binary variable x_{ij} has value 1 if item j is assigned to container i. A solution is feasible if $S(x_{ij}) \subseteq S_{Total}$ and $S(x_{ij}) \cap S(x_{ik}) = 0$, i.e., all items to be packed in a particular container must fit in the container and no two items can occupy the same space. These constraints are extremely difficult to express in a mathematical programming context and are best controlled using rule based methodology.

Weight and balance must also be considered to ensure feasible loads. In some instances, such as military aircraft loading, weight and balance may be more important than spatial requirements. First, the total weight of the items must not exceed the aircraft's allowable payload. This can be easily represented as a knapsack constraint. When loading very heavy items such are armored fighting vehicles, the weight limit will be reached well before the spatial limits. Balance constraints for aircraft loading include lateral, longitudinal, and in some cases vertical limits on the aircraft's center-of-gravity (CG). These must be within specified limits.

Trucks, trailers and boxcars also have weight and CG limits. Weight limits include total weight, axle weight, and trailer tongue weight. The last two are determined by computing the longitudinal CG and evaluating the balance of forces. Most trucks and trailers have vertical CG limits to reduce the chances of rolling the vehicle during transport. These vertical CG limits could easily be the first constraint reached with open bed trucks and trailers hauling moderately heavy, stackable items. A potential loss of $1.92 million occurred during Desert Storm when one HEMAT trailer carrying Hellfire missiles rolled onto its side. The pallets on the trailer, like other missile pallets in the 12th Aviation Brigade, had been stacked one layer higher in an attempt to haul additional ammunition.

Stability is defined as how well an item is supported to prevent damage and to prevent it from shifting as a result of forces caused by the normal movement of the container. Stability is achieved by providing (1) sufficient

support from below to prevent damage and movement caused by gravity and (2) sufficient lateral support to avoid shifting caused by forces associated with acceleration or inclination of the container during transport.

Shipment of certain combinations of cargo is prohibited. For example, materiel that is explosive can never be packed with materiel that is corrosive.

There are obviously many specialized constraints that can be included in the problem. These constraints may include requirements for subsets of items to be packed in the same container or requirements for items to be packed at a certain location and/or orientation within the container. An example of a specialized constraint in the Aircraft Loading Problem is the loading of heavy equipment such as armored vehicles. Current regulations require that these items be 'center loaded' in the cargo area.

1.3 A NEW SOLUTION FRAMEWORK FOR CUTTING AND PACKING PROBLEMS

"Group theory can powerfully enhance the study and understanding of metaheuristic search neighborhoods. This observation is particularly apparent when metaheuristic search is applied to P|O (partitioning and ordering) problems." (Colletti and Barnes 99-02) This research will provide additional evidence to support that claim.

A *group* is any set and binary operation, which together satisfy the properties of closure, associativity, identity, and inverse. (Herstein 1975)

The integers under addition form an Abelian (commutative) group. The real numbers {1, -1} under multiplication form an Abelian group of order two

(Herstein 1975). A group that has been shown to provide a unifying framework in P|O problems is the symmetric group on *n* letters, S_n (Colletti and Barnes 1999). Groups that do not satisfy the commutative property are called *non-abelian*. S_n is a non-abelian group that consists of all possible *permutations* of n distinct letters. A permutation is a *one-to-one* mapping of a set of n letters *onto* itself. Additional detail on S_n is available in Colletti (1999).

An essential component of any local search method, like tabu search (TS), is a move neighborhood. A *move* is defined as an operation on an incumbent solution that changes it to a new solution. The *move neighborhood* is defined as the set of solutions (with possible duplicates) that are reachable in a single move from the incumbent solution. (Barnes, 2001)

If packing can be completed based on a set of packing rules, S_n can be used to represent a packing solution, and GTTS techniques using only S_n can be employed. For example, consider a 2–dimensional packing problem with rectangular items and objects. If the items have fixed orientation and we use a popular Bottom-Left (BL) heuristic (Baker et al. 1980), then a packing sequence can be represented by S_n that consists of a single cycle. When considering arbitrarily shaped items, the spatial aspects of two- and three- dimensional cutting and packing problems suggest other types of algebraic groups that could provide analogous efficiency in neighborhood search when implementing reactive tabu search.

However, only special packing problems require all items to have fixed orientations. If rectangular items can be packed in any orthogonal orientation, the solution space is much larger and more efficient packing solutions will exist. However, allowing multiple orientations requires a more complex solution representation. If we are primarily concerned about the space that is occupied in a particular orientation, a 2D rectangle could be represented by S_2. The first letter would represent which side is parallel to the x-axis and the second letter could represent which side is parallel to the y-axis. Thus, there would be n different groups that use S_2 that represent the orientation of each box.

We now have two different group representations that represent the sequence and orientation of the rectangles to be packed. This concept may be directly extended to packing 3D rectangular objects with S_n representing the packing sequence and S_3 representing the possible orientations of a rectangular box. This is shown in Figure 1-1.

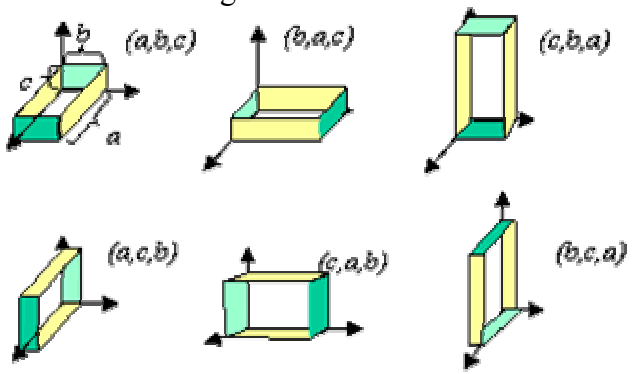


Figure 1-1 Group of letters representing the volume occupied by a three dimensional box

This research combines these two representations into a single group. However, in order to simplify the representation, it is easier to allocate 1-letter for each orientation so that (in three dimensions) there would be one active letter and five inactive letters. The inactive letters are treated like other one-cycles S_n and can be left off our representation. However, we have a special move class that goes beyond the basic symmetric group permutations that will target swapping these inactive letters with their active counterparts whenever we need to physically rotate an item.

1.4 STATEMENT OF THE PROBLEM

This research implements an ATS approach to solve the 2D orthogonal bin packing problem with homogeneous bins. The research also offers a group theoretic representation of solutions to this problem which provides a framework for later work in more intensive use of abstract algebra concepts for approaches to this type of problem.

The specific problem addressed is concerned with $j = 1,2, \ldots, n$ rectangular items of *width* w_j and *height* h_j. An unlimited number of identical rectangular bins of width W and height H are available. The objective is to pack all items into the minimum number of bins, in such a way that no two items overlap and the item edges are parallel to those of the bins. This research solves the problem where items may be packed in either of their two allowable orientations. It is assumed without loss of generality that all input data are positive integers and that any item will fit into a bin in at least one of its orientations.

Lodi, Martello, and Vigo (1999-1) use the notation 2D|R|F (two dimensions |rotation allowed | free packing allowed) for this problem.

Chapter 2

Background and Related Work

This section contains an overview of packing problems and solution approaches, to include heuristic, exact and metaheuristic approaches to 2D rectangular, 2D irregular shaped and 3D orthogonal problems. It highlights several instances where Group Theory has been used in a spatial context. Essential implementations that use GTTS and TS as a method for attacking combinatorial problems are also presented.

2.1 SOLUTION APPROACHES TO PACKING PROBLEMS:

If one considers the weight and space aspects of what is being packed, the problem is a multidimensional knapsack problem, which is known to be NP Hard. (Chu and Beasley, 1998). Chazelle provides the first $O(n^2)$ implementations of the BL heuristic, but offers no detail on how to keep the data structure updated. Multidimensional packing problems are more complicated because of the geometric aspect of the items being packed. Thus, when considering reasonably sized problems in 2 or more dimensions, classical exact methods are insufficient for practical sized problems. This literature review will focus on methods to solve packing problems in 2 or more dimensions. Most of the work has investigated

heuristic and metaheuristic approaches. Hopper and Turton (2001) present a fairly comprehensive survey of these algorithms.

2.1.1 2D rectangular items/ rectangular objects

Lodi, Martello and Toth (2001, 2002) present surveys on 2D packing problems and advances in 2D packing. The first paper covers models, approximation algorithms, lower bounds and exact algorithms. The second paper discusses bounds, exact methods, heuristic approaches, and metaheuristic methods for the various classes of 2D problems. Some traditional mathematical optimization techniques have also been proposed. Lodi, Martello, and Vigo (1999-2) present approximation algorithms for the 2D bin packing problem (BPP). Martello and Vigo (1998) present a two-level decomposition principle to obtain an exact solution to the 2D BPP.

2.1.1.1 HEURISTIC APPROACHES:

There are several popular heuristic approaches for packing 2D rectangular items. The most basic of the 2D heuristics rely on a version of the bottom left (BL) heuristic (Baker et al. 1980). Starting from the top right corner, each piece is pushed as far down as possible until it makes contact with another item and then as far to the left as possible. These bottom-left movements are repeated until the item can no longer be shifted further to the bottom or left. Jakobs (1996) used

this in a hybrid Genetic Algorithm (GA). A packing pattern is represented by a permutation, π, of the items. For example π = (3,2,4,5,1) first loads item 3, then items 2, 4, 5 and 1. This is illustrated in Figure 2-1a.

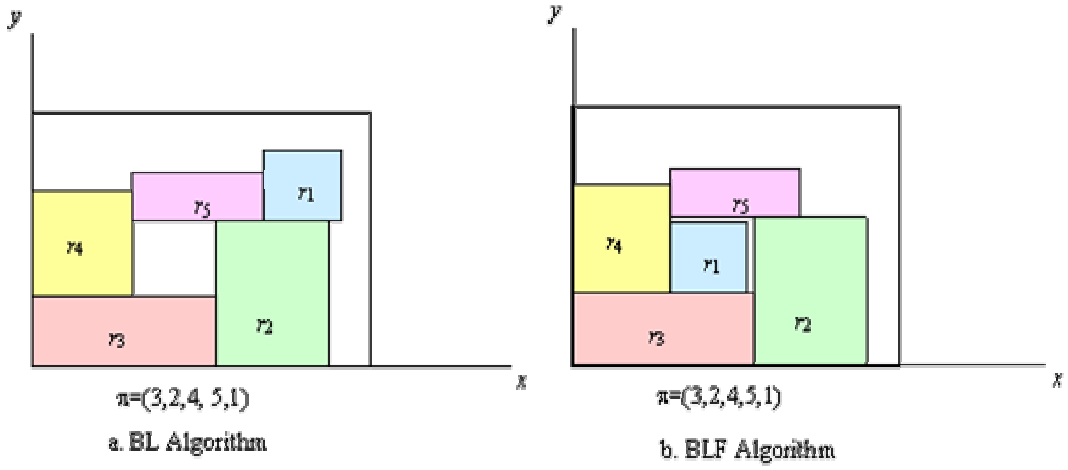


Figure 2-1 Examples of the BL and BLF Algorithms

BL requires $O(n^2)$ operations and tends to create packing solutions with relatively large dead spaces. The $O(n^3)$ Bottom-left-fill (BLF) heuristic (Chazelle 1983) attempts to fill the gaps created by BL. The strategy is to left-justify items at the lowest available position. Figure 2-1b shows the improved packing BLF obtained by using the identical π. BLF is capable of obtaining a denser packing, but requires more computation.

A minor permutation of π will yield identical packing with either of the two heuristics. For example, both heuristics yield the pattern in Figure 2-1b when π = (3,2,4,1,5).

2.1.1.2 Exact Approaches

Gilmore and Gomory (1965) implemented the first math-programming model for packing problems in two or more dimensions, a column generation approach based on enumerating all subsets of items (patterns) that can fit into a single bin.

Fekete and Schepers (1997-1, 1997-3) use a graph theoretic framework to enumerate feasible packing solutions with an exact tree search algorithm and present results for test problems for single bins with up to 97 boxes.

Lodi, Martello, and Vigo (1998) analyze a simple lower bound, the sum of the item areas to be loaded divided by the container area, and determine its worst case performance. They also use new lower bounds in a branch-and-bound algorithm for problems of up to 120 items.

2.1.1.3 Metaheuristic Approaches

Unlike heuristic approaches, few metaheuristic approaches have been applied to the 2D problem. The most commonly applied metaheuristics to the problems considered here are GA, simulated annealing (SA) and TS.

Genetic algorithms (Barnes, 2001) emulate the evolutionary process. Jakobs (1996) presents a hybrid GA that uses the BL heuristic combined with GA. A weakness of many BL variants is that item sets can exist where no π will yield an optimal packing. Liu and Teng (1999) use an improved BL with a GA which they claim overcomes this problem.

Lodi, Martello, and Vigo (1998) present a TS algorithm for problems with *guillotine cuts* and fixed item orientation. Guillotine cuts requires all packing

patterns “cuts” to go completely across the object. In Lodi et al. (1999-1), they generalize their approach to include non-guillotine cuts and item rotations. Their TS algorithm controlled the outer loop with several “inner heuristics” and found it to be effective in most of the evaluated cases with instances up to 100 items. Their TS controlled the partitioning while the inner heuristic controlled π by sorting, placement and possible orientation.

2.1.2 2D rectangular items / irregular shaped items

Many practical problems involve packing non-rectangular items within a rectangular container. This type of problem is known as the ‘nesting problem’ or ‘polygon placement problem.’ Early implementations would find the rectangle of minimal area that enclosed these items and used techniques for 2D rectangular objects. This is obviously sub-optimal for cutting problems and could yield unstable solutions when stacking items. This literature review focuses on methods that do not ignore the complexity caused by irregular shapes.

2.1.2.1 Heuristic Approaches

Bennell and Dowsland (1999) preprocess calculations using a “no fit polygon” (NFP) with Minkowski sums. Bennell, Dowsland and Dowsland (2001) present a new method to compute the NFP. Dowsland, Vaid and Dowsland (2002) present an algorithm that uses a bottom left placement policy for packing irregular shaped polygons which could be implemented using various metaheuristic approaches. Gomes and Oliveira (2002) present a two-exchange neighborhood heuristic that utilizes a low-level placement strategy to convert a

sequence into a feasible layout. Their technique allowed for non-convex shaped items and used the NFP to manage overlap.

2.1.2.2 Exact Approaches

Daniels and Milenkovic (1999) present a mathematical programming approach for minimizing the space occupied by translating polygons in a rectangular surface. Not allowing rotation is applicable when the surface has a grain. For example, in cutting pieces of wood for furniture, the orientation of the parts are fixed so that the patterns have a particular look (i.e. vertical grain). These pieces would be allowed to translate, but not to rotate.

2.1.2.3 Metaheuristic Approaches

An implementation by Theodoracatos and Grimsley (1995) use SA with polynomial cooling schedules to pack simple polygons in a bounding rectangle. Their approach allowed translation and rotation of the items, but not reflection. Jakobs' (1996) permutation based GA extended a technique previously used for 2D rectangular objects to polygons.

Chocalaad (1998) used a two-stage algorithm that relied on simple tabu thresholding (STT) and applied it to the Air Force's Aircraft Loading Model. STT consists of alternating two phases: an improving phase and a mixing phase. In both phases neighborhood moves are divided into subsets, and only one subset is considered at each iteration. Subsets are searched using a block random order scan (BROS). Swap, translate and rotate moves were allowed but only with 90-, 180-, and 270-degree rotations. Intersection of items was checked by Manhattan geometry for rectangular items and by the Theodoracatos and Grimsley (1995)

method for non-rectangular shape items. Romaine (1999) extended the work of Chocalaad (1998) and showed that combining the two stages was more efficient. He also used multiple non-homogeneous containers (i.e. aircraft types) and tested the problem with a sample of actual air loads to show the improvement over existing software used by the U.S. Air Force.

Bennell and Dowsland (1999) present a STT approach to the irregular stock cutting problem similar to the approaches of Dowsland (1996) and Chocalaad (1998). They argue for STT instead of traditional deterministic TS because of the "spiky solution landscape."

2.1.3 The 3D orthogonal packing problem

In recent years, the 3D orthogonal packing problem has received much attention. This problem is also known as the distributor's or multi-pallet loading problem (Terno et al. 2000), the "3-Dimensional Bin Packing" and the "3-Dimensional Knapsack Problem".

2.1.3.1 Heuristic Approaches

Bischoff, Janetz and Ratcliff (1995) present a heuristic that constructs layers from the bottom up with two box types at a time or two different orientations of the same box type. Terno, Scheithauer, Sommerweiß, and Riehme (2000) present the Parallel Generalized Layer-wise Method (PGL). Their approach consists of a layer-wise loading strategy that utilizes optimal 2D loading patterns and produced results that were comparable to a 1997 GA implementation and a 1998 TS implementation by Bortfeldt and Gehring.

Baltacioglu, Moore and Hill (2001) present a human intelligence based heuristic that builds layers, fills gaps within layers, and examines various box orientations. Their heuristic produced results comparable to Bortfeldt and Gehring (1997,1998) and Terno et al. (2000) on the Bischoff/Ratcliff test problems while requiring less computational effort. Pisinger (2002) discusses a new heuristic that decomposes the problem into layers that further decompose into strips.

2.1.3.2 Exact Approaches

Martello, Pisinger and Toth (2000) present an exact approach to the bin-packing problem for a *single bin*. In the paper, they present some new lower continuous bounds that improved the branch and bound approach. They solve problem instances with up to 90 items.

2.1.3.3 Metaheuristic Approaches

Bortfeldt and Gehring (1997) developed a GA that generates a set of disjunctive box towers which ensure stability and then arrange the box towers on the floor of the container according to a given objective. Bortfeldt and Gehring (1998) develop a TS implementation that solves this problem for a weakly heterogeneous set of boxes. This approach included stability constraints, orientation constraints, stacking constraints, and weight constraints. Their results showed improvement over previous implementations, particularly implementations with weakly heterogeneous boxes. Gehring and Bortfeldt (2001) utilized a Hybrid GA and used test problems BR1 to BR15 by Bischoff, Janetz

and Ratcliff (1995). Like Gehring and Bortfeldt (1997), this approach built vertical, cuboidal shaped layers.

Faina (2000) reduces the 3D BPP to a finite enumeration scheme and attacks it using SA. Faina's "method of zones" uses information about the vertices of a box to track possible packing corners and to check potential intersection of boxes. Faina (2000) provides pathological examples for which the method will not work, but claims that his technique will be successful in most practical cases.

Faroe, Pisinger, and Zachariasen (1999) present a Guided Local Search (GLS) implementation to the 3D BPP. GLS is a new metaheuristic that is similar to TS. GLS is based on a set of attributes, which characterizes a solution to the problem in a natural way. An indicator function is set to one or zero depending on whether or not the solution has that feature. A feature is defined such that its presence has a direct contribution to the value of the solution. For the case of the 3D BPP, the indicator function is used to indicate pair wise overlap between two items. Since overlap between large boxes is more undesirable than overlap of small boxes, the penalty for overlap includes the amount of overlap and the volume of the two pieces that are overlapping. They also implemented the concept of Fast Local Search (FLS), a technique that partitions the neighborhood into sub-neighborhoods and visits them in a specified order. Sub-neighborhoods are revisited as long as they contain improving moves. A good feature about the GLS approach is that it does not require repacking after making neighborhood

moves. The approach presented did *not* allow items to be rotated, but the authors claim that this feature could be added.

2.1.4 Weight, balance, hazardous cargo and other considerations

Practical constraints concerned with such things as weight and balance restrictions and mixing of incompatible materials are usually ignored in the literature. However, a realistic solution to any practical loading problem must consider factors other than the spatial dimensions of the items being loading. There can be restrictions on both the items being loaded and on the container to be loaded. Items may have a fixed orientation such as predetermined bottom and top. Items may also be restricted to the top or bottom layer.

Considering such factors adds complexity to a quite formidable multi-dimensional spatial problem. For example, when loading a military cargo aircraft, in addition to spatial constraints, constraints associated with "compartment limitations, axle and contact limitations, pallet weight limitations, upper and lower deck considerations, door opening size/angle loading, available troop seats, cargo/passenger location restrictions and allowable cargo load (ACL)" must also be considered. (ALPS OCD 4.2 28 Feb 02) If the items are to be loaded are trucks, factors such as total weight, axle loads, center-of-gravity, and compatibility issues for various types of hazardous cargo are also important. Pallet loading and bin packing problems (BPPs) should consider both load stability and the load that items can sustain without damage to their contents.

Bischoff, et al. (1995) address stability issues for the packing of 3D rectangular items. Terno et al. (2000) address stability with a heuristic that sorts

by height and then optimizes a 2D rectangular area problem one layer at a time with 2, 4, or 8 different box types. Davies and Bischoff (1999) provide an overview of recent literature for weight distribution considerations in container loading. The 3D packing technique they use builds wall-to-wall/floor-to-ceiling 'blocks" of items. For stability, the heuristic only loads on totally flat rectangular surfaces to ensure that all boxes are fully supported from below. To achieve balanced loading they consider both permutations of the blocks within the container for longitudinal balancing and replacing blocks with their mirror image for lateral balancing.

Chocalaad (1998) and Romaine (1999) include many practical considerations in applying a rudimentary TS algorithm to the U.S. Air Force's Air Loading Model (ALM). Chocalaad implemented a two-phase heuristic to solve the ALM for a single aircraft. In the first phase he solved a 2D knapsack for weight and volume. The solution was passed to the second phase, which implemented a 2D cutting stock TS algorithm that considered spatial conflicts, weight and balance, floor loading, and hazardous cargo. Romaine showed improved results can be achieved by combining the two phases into one. He also allowed for multiple aircraft and multiple aircraft types.

Bortfeld and Gehring (1997) considered weight distribution in their GA. Their approach builds walls and they propose replacing a wall by a mirror image or moving walls to different locations in order to balance the load. Their 1998 TS implementation considered four non-spatial constraints: (1) they required complete support from below to ensure stability, (2) they considered orientation

constraints by restricting which side measurement(s) might be used as the height measurement, (3) they implemented stacking constraints that prohibit certain items from having other items stacked on top of them, and (4) they restricted the total weight loaded into a container.

2.2 GROUP THEORETIC REFERENCES

2.2.1 The symmetric group on n-letters, S_n

Herstein (1975), Fraleigh (1976), and Baumslag and Chandler (1968) are good sources for background knowledge needed to understand the group theory used in GTTS. Colletti (1999) presents S_n as the natural setting to characterize P|O problems and implement GTTS. Wiley (2001), Crino (2002), and Combs (2002) are recent applications of GTTS that used S_n to structure moves and move neighborhoods. Their contributions are discussed in Section 2.3.2.

2.2.2 Euclidean group

Lyndon's book (1985) on Groups and Geometry provides insightful explanations of the Euclidean group, E, but its primary focus is on two dimensions. However, the matrix representation shown in two dimensions has an analogous representation for three dimensions. E has numerous sub groups. E^+ is the group of Euclidean movements that prohibit reflections. This includes the product of subgroups that allow all possible translations T, and all possible rotations R. Lyndon represents rotations and translations in terms of matrices. $M(\theta)$ represents a rotation $\theta \in R$ (where θ is the angle of rotation) .

$$M\ (\theta)=\begin{bmatrix}\cos\theta & -\sin\theta\\ \sin\theta & \cos\theta\end{bmatrix}$$

.

A translation $\tau \in T$, moving from (0,0) to (a,b) is represented by

$$M'(a,b)=\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ a & b & 1\end{bmatrix}$$

The isomorphism of E^{+} to M'' is the set of all matrices of the form:

$$M''(a,b,\theta)=\begin{bmatrix}\cos\theta & \sin\theta & 0\\ -\sin\theta & \cos\theta & 0\\ a & b & 1\end{bmatrix}$$

Neumann, Stoy, and Thompson (1994) present numerous examples associated with group theory, Euclidean geometry, and the relationships that join them.

2.2.3 Related Group Theoretic Applications

Popplestone, Liu, and Weiss (1990) considered a group theoretic approach to assembly planning and developed a Euclidean group description that allows a robot to understand surface contact. Liu (2001) discusses how to use group theory for solid surface contact and provides a good background discussion of the Euclidean Group including many of the subgroups. Feng, Wang, Wang and Teng

(1999) present a group theoretic/graph theoretic method to solve layout problems. They use isomorphism and orbit properties to define and partition the solution space and then use several small examples to illustrate a math programming/heuristic hybrid optimization algorithm. Stoyan and Pankratov (1999) present methodology for optimal packing of congruent polygons and refer to the use of Fyodorov's crystallographic groups (1885) for packings (exact coverings).

Kreher and Stinson (1999) discuss combinatorial optimization, heuristic and metaheuristic search, and group theory, including permutation groups, orbits, and isomorphism. Lins, Lins and Morabito (2002) present a recursive '*n-tet'* graph approach for packing n-dimensional boxes into an n-dimensional container. An n-tet is a representation of a feasible packing as a depth assignment, which has a sequence of n-directed graphs. They use the *reflexive group*, which is a reversal ordering of the vertices in a directed graph G_i. There are 2^n such reflections in the reflexive group, R_n. They also discuss *rotational symmetries* (the set of $\pi/2$ rotations of an item) using S_n.

2.3 TABU SEARCH

Glover and Laguna (1997) provided the initial summary treatise on TS. Since that time, TS theory and methodology have undergone an explosive growth. Newer aspects of tabu search seen in this growth include *scatter search* and *path relinking* (Glover, Laguna, and Marti, 2000 and 2002a&b), *multistart techniques*

and *strategic oscillation* (Glover, 2000), *ejection chains* (Rego, 2000), and parallel methods such as *cooperative search* (Ouyang, Toulouse, Thulasiraman, Glover, and Deogun, 2000). Many of these techniques provide empirical evidence for their success. However, a theoretical explanation for this success is perhaps best explained by the landscape theory work of Barnes and associates developed over the last three years. (See Section 2.3.2.)

The research documented here focuses on deterministic TS methods that intelligently react to the search history and use logical rules to adjust parameters to diversify and intensify the search. This focus will lead to the development of an ATS using a S_n solution representation to attack the loading problems discussed above.

2.3.1 Reactive Tabu Search (RTS)

Battiti and Tecchiolli (1994) were the first to develop a truly reactive version of TS and to demonstrate the superiority of RTS. Battiti (1996) presents a comprehensive overview of self-tuning heuristics and the fundamental techniques they employ. Carlton (1995) implemented RTS to solve the general vehicle routing problem. This includes the multiple Traveling Salesmen Problem with Time Windows (mTSPTW) and the Vehicle Routing Problem with Time Windows (VRPTW).

2.3.2 Group Theoretic Tabu Search (GTTS)

Colletti (1999) presents the first detailed use of local search and group theory on partitioning and ordering problems. Colletti's dissertation is the 'text book' for anyone learning and implementing GTTS. His techniques using elite

subpath neighborhoods may provide useful hints combining the S_3 representation of the orientation of a 3D rectangular item with the S_n representation of its packing sequence. Colletti and Barnes (1999) reinforce the elite subpath neighborhood concept.

Wiley (2001) and Barnes, Wiley, Moore and Ryer (2004) describe the application of GTTS to the Aerial Fleet Refueling Problem, the assigning and scheduling of tanker aircraft to algorithmically determined refueling points during an aerial deployment of other aircraft from the Continental United States to an overseas theater. This work demonstrated the ability of GTTS to solve extremely complex VRP related problems. Crino (2002) and Crino, Moore, Barnes and Nanry (2004) present a GTTS for the Theater Distribution Vehicle Routing and Scheduling Problem where orbits and "orbital planes" are used to both partition the search neighborhood and to intensify and diversify the search. Combs (2002) presents a GTTS/Set Partitioning approach to solve the aircrew-scheduling problem. Jones (2002) provides empirical computational evidence on the efficiency gained by using group theory to compact the evaluation of large, complex, multi-graph neighborhood structures.

Several efforts have been reported that investigate the *landscape*, i.e., the solution space topology of combinatorial optimization problems related to a selected search neighborhood. These include Barnes, Colletti, and Neuway (2000 a&b) who perform extensive analysis of neighborhood adjacency matrices. Others include Barnes and Colletti (2000), Barnes and Colletti (2001), Barnes, Colletti and Neuway (2002), Barnes, Dokov, Acevedo, and Solomon (2002 a, b,

and c). These papers have led to a "theory of landscapes" which shows promise in the quest to develop a methodology for comparing search neighborhoods and for determining the best neighborhood for a specific problem or class of problems.

This chapter provided an overview of solution approaches to packing problems, related group theory, and TS/GTTS approaches that have potential application to packing problems. As evident by the lack of published work, a tabu search implementation for packing that controls order, orientation and partitioning of items is greatly needed for all 2D and 3D problem types. The recent work in GTTS shows that S_n offers an effective method to represent and control such an implementation.

The next chapter discusses the implementation of a 2D heuristic that allows us to represent a packing solution as a partitioning and ordering problem and maintains the capability to represent any feasible layout.

Chapter 3

An Implementation of the Bottom Left Heuristic

One method to address the complexity of orthogonal packing is to use packing solutions which maintain bottom-left (BL) stability. BL stability ensures that every item is supported on both its left and bottom edges. BL heuristics take an ordered list and pack each item into a BL stable position based on a particular set of packing rules. There are several methods that have been used to choose BL stable positions. Chazelle (1983) used the lowest possible BL stable location and broke ties by taking the leftmost, Jakobs (1996) began at the top right and then alternated movements, first moving as far as possible toward the bottom then as far as possible to the left until the item was BL stable. Liu and Teng (1999) let the item move down whenever possible and then slide left upon contact until either it was able to fall again or BL stability was achieved. Figures 3-1 a, b, c show how the different rules can be used to place a particular item into a BL stable position. The dashed rectangle in (a) shows the 3 candidate positions and (b) and (c) show the motion of the rectangle as it follows the placement rule.

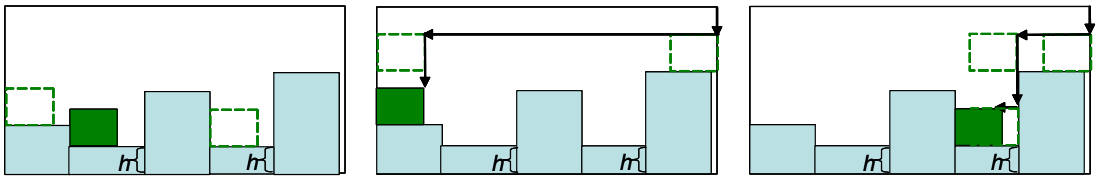


Figure 3-1 Comparison of three BL placement rules

Each of these methods can be implemented as $O(n^2)$ algorithms. Prior to this dissertation, there was no publicly available BL Heuristic code available. Fortunately, Chazelle's paper offers a fairly robust representation of the data structure and includes pseudocode for finding BL stable positions. This research uses Chazelle's method to find the set of feasible locations. Liu and Teng's technique was selected for choosing the packed location because of its capability to represent any physical packing solution as a permutation of letters in our symmetric group. (Other rules such as Jakobs (1996), or Lodi et al. (1999-1) could be implemented with little overhead but do not have the ability to represent any physical packing solution.)

Chazelle (1983) stated that the structure update could be achieved in $O(n^2)$ effort, but did not provide detail on the data structure update procedure used after packing each item. His belief that there are only two key cases to be considered in the update was incorrect. Effective and efficient ways to identify and implement the *several* required update cases are discussed later in this chapter. This chapter presents details of Chazelle's method for placing an item, provides a method for updating the data structure so that any permutation of items can be packed in $O(n^2)$, and discusses adaptations required to use this heuristic for packing multiple bins with items that are allowed to rotate.

This chapter has five core sections. The first section explains the data structure used in Chazelle's method in order to lay a foundation for the next two sections. The second section explains how to find the feasible BL stable packing

locations for a single item in a given hole. It significantly augments Chazelle's presentation by providing both a narrative and detailed explanation of the pseudocode. This includes finding bottom and top traces based on the item's width, and combining the two traces based on item height. The third section provides an overview of the *new* update procedure that keeps Chazelle's data structure current. It also identifies and corrects some of his paper'deficiencies. Since, the possible situations that can occur for the update are numerous, this section presents only a narrative description (with pictorial examples) of how to identify the cases, and perform updates for the data structure for each of these cases. For details on the detailed logic involved, readers are referred to the embedded documentation in the Java computer code, available upon request from the Graduate Program in Operations Research and Industrial Engineering at The University of Texas at Austin. The first three sections limit consideration to the placement of a single item with a given orientation.

The fourth section explains the data structure and additional algorithmic architecture required to adapt Chazelle's heuristic for a multiple BPP that allows items to be freely rotated. The final section presents a performance comparison of this heuristic with two others implemented by Lodi et al. (1999-1) on two different test sets.

3.1 INDICES, VARIABLES AND DEFINITIONS:

The indices, variables and definitions presented below are key to understanding both Chazelle's method for finding feasible locations and the data structure update procedure. A BLH must ensure that every item maintains the BL

condition, i.e., each item is supported on both the left and bottom by either the container or another item. There is no requirement for this support to be 100% in contact with the placed item, but the supporting structures must make contact at more than one point.

A *Hole* is defined as an empty polygonal space with horizontal and vertical boundaries or *edges*. Any hole with BL stability is a *BL Stable Hole*. When packing a container (excluding the case of a completely full container), there is at least one hole, H_0, open to the top of the container. The technique developed in this research uses (for H_0 *only*): (1) Chazelle's method to find the bottom left stable positions and (2) Liu and Teng's rules for placing the items into the left stable position.

The following terminology is adapted from Chazelle (1983). An edge is a *notch* if it is "sealed" at each of its endpoints. An example of each of the four kinds of notches are shown by the hole pictured in Figure 3.2 (which is not BL stable). The interior of the diagram is the hole, while the exterior boundary is formed by edges of packed items.

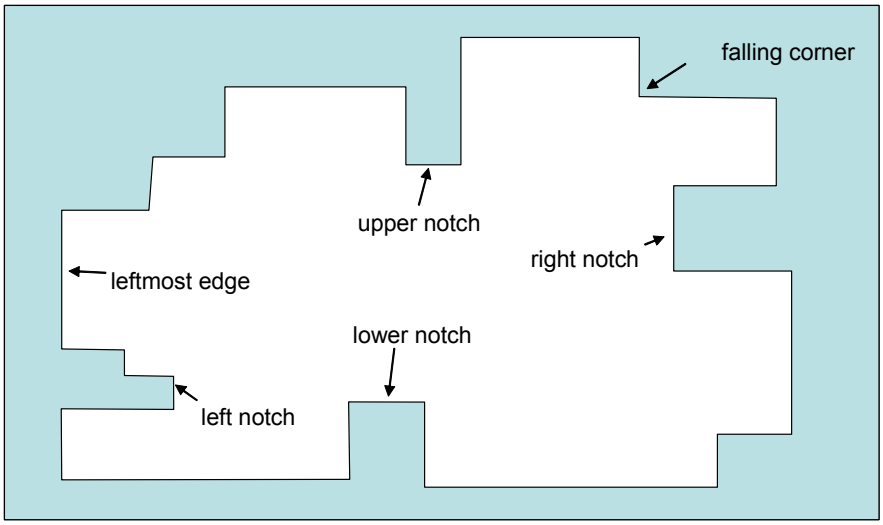


Figure 3-2 Chazelle (1983): Special edges of a hole

Chazelle (1983) proves that a packing pattern that ensures BL stability can neither have right notches nor upper notches and can have, at most, one falling corner. If an item formed an upper notch it would not have bottom support. Likewise if an item formed a right notch it would not have left support.

Trapped holes, i.e. completely enclosed holes, are not addressed in the initial procedure discussed here but information about trapped holes are used in the tabu search procedure to intensify a search in a particular bin. *Only* trapped holes can have a falling corner. Our current interest is in packing H_0, which can not have a falling corner. Hence, our major concern in implementing an $O(n^2)$ BL heuristic is dealing with left notches.

Chazelle referred to holes without left notches as *nice holes* and his method of finding feasible positions only works on nice holes. To overcome this problem Chazelle chose to partition his holes into *nice subholes*. The purpose for

this partition was to "allow us to search a hole for packing locations without any duplication of effort." (Chazelle, 1983, p. 700)

A *subhole*, *s*, results from partitioning H_0 along vertical lines extending from the *top right corner* of each left notch. The hole shown in Figure 3-3 has two left notches and can be partitioned into three nice subholes.

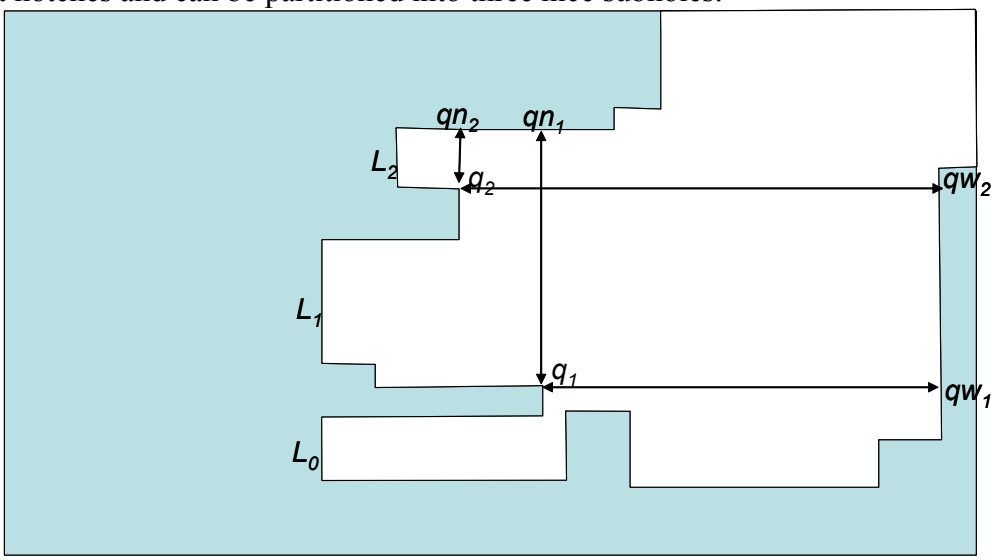


Figure 3-3 Special links for a hole

Each hole uses special links to partition the hole into nice subholes. Designate the top right corner for each left notch *s* as q_s. The point at which the vertical partitioning line originating at q_s contacts a higher boundary point of the hole is designated qn_s. Define qw_s to be the first boundary point directly to the right of q_s. Figure 3-4 shows how these points are used to define the space included in each subhole.

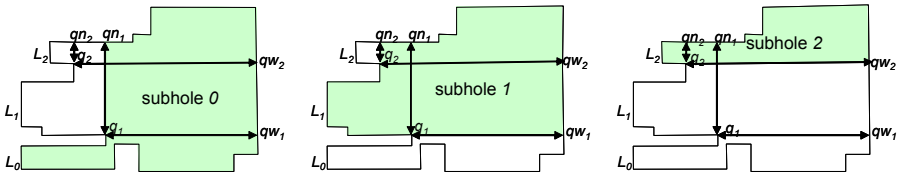


Figure 3-4 Partitioning of hole with special links

Line (q_s ,qn_s) is a vertical partition for subhole *s-1*, while line(q_s ,qw_s) is a horizontal partition for subhole *s*. These partitions do not create mutually exclusive areas. However, the set of BL-stable positions are formally defined, i.e., horizontal-vertical edge combinations that could support an item in a BL stable position appear in only one subhole. Once the "not-nice" hole is cast into nice subholes, each subhole can be treated as a separate hole for packing purposes. Each subhole *s* is defined by a set of coordinates that detail its boundary and form a *top* and *bottom trace*. Figure 3-5 shows the coordinates used to define the trace for a particular subhole

FB_s details subhole *s*'s bottom trace. In Figure 3-5, $FB_s = \{b_0, b_1, ..b_7\}$ where each b_j constitutes a two dimensional coordinate, (b_{jx}, b_{jy}). The next section details how all possible BL-stable packing locations for an item of a given width are found. This is accomplished by tracing the lower left corner of the item as it "*slides*" to the right across the horizontal edges of the bottom trace, rising when it hits a vertical edge on the right side and falling to the tallest horizontal edge below when the object's left corner clears the right corner of a supporting horizontal edge.

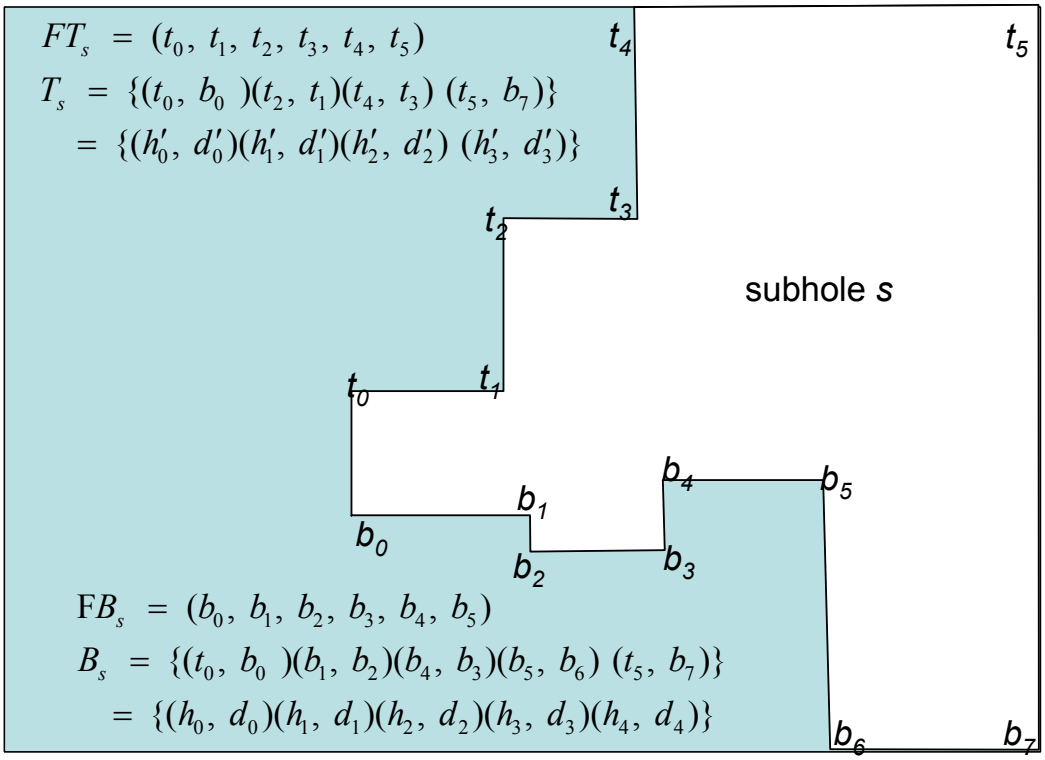


Figure 3-5. Top and bottom trace for subhole *s*

Exactly analogous to FB_s, FT_s comprises subhole *s*'s top trace which starts at the top end of the leftmost vertical edge, t_0, and moves to the right until it reaches the top of the right-most vertical edge. In Figure 3-5, $FT_s = \{t_0, t_1, \ldots, t_5\}$.

This section defined indices, variables, and parameters that are essential in partitioning holes into nice subholes so that Chazelle's O(n) method can be used to find the set of all possible left stable positions for a particular item. The next section describes how this method is implemented using Chazelle's terminology.

3.2 FINDING ALL BL STABLE POSITIONS

This section explains how Chazelle's method finds the set of feasible BL stable positions for a given item. *Only one* subhole is considered. In the actual implementation, the subholes of a particular bin are checked sequentially beginning with the lowest subhole. A three step process is employed: (1) the algorithm finds the set of potential lower left corner points considering *only* the item's width and FB_s, (2) potential upper left corner points are found considering only the item's width and FT_s, and (3) the information from steps 1 and 2 is combined while considering the item's height to determine the set of feasible BL stable positions.

Chazelle's algorithm uses two more sets that describe the top and bottom traces. By using the first and last t_j of FT_s with FB_s, B_s forms the set of vertical edges in the bottom trace. In Figure 3-5, $B_s = \{(t_0, b_0), (b_1, b_2), (b_4, b_3), (b_5, b_6), (t_5,b_7)\}$. For notational convenience and coding efficiency, the edges of B_s are relabeled indexed from left to right as (h_i, d_i) where h_i is the top node and d_i is the bottom node. Thus, $B_s = \{(t_0, b_0), (b_1, b_2), ..., (t_5,b_7)\} = \{(h_0, d_0), (h_1, d_2), ..., (h_7,d_7)\}$where the (h_j, d_j) have the same horizontal coordinate (by definition as a vertical edge pair).

Exactly analogous to B_s, T_s is the set of vertical edges in top trace FT_s. In Figure 3.5,

$$T_s = \{\{(t_0 , b_0), (t_2 , t_1), ..., (t_5 , b_7), s\}$$
$$=\{(h_1', d_1'),(h_2', d_2'),(h_3', d_3'),(h_4', d_4'), fc)\}.$$

fc is a boolean variable that is set to 1 if *T* has a falling corner.

The next step is to find the *width* feasible BL corner points, $C_s = \{ c_0, c_1, \ldots c_i, \ldots c_{|C_s|-1} \}$ where each $c_i = (c_{ix}, c_{iy})$ is a point in the subhole's bottom trace, i.e., C_s defines all possible positions where an item of width *w*, i.e., horizontal or "x-dimension" *w*, would be supported from the bottom. Thus, Chazelle's heuristic, *BOTTOM*(*B*), uses B_s and considers only the item's width to find C_s.

Before considering the detail given in the pseudo code for the *BOTTOM(B*) procedure presented in Figure 3-7, we provide a summary overview with some clarification of the notation employed. In *BOTTOM(B),* a horizontal bar, (e_1,e_2), of width *w* (with *no* height) starts with its right end, e_2, placed at the left-most point in *B*. (e_1, e_2) begins its slide across *B*. As it slides, the left end, e_1, leaves a track whose end points and corner points form C_s. The notation: $(e_1) \leftarrow (d_0 \ -_x \ w)$ is shorthand for $e_1 = (d_{0x}\text{-}w, d_{0y})$, i.e., set the coordinates of e_1 to be the same as d_0 but subtract *w* from the horizontal coordinate. Any *operator* with a right subscript, like $-_x$, implies that the operation is limited to the indicated coordinate. In Java programming, this notation is efficient since e_1 and d_0 are Java objects with attributes for horizontal (x) and vertical (y) components. In addition the following notation will be used: *x(a):* the x-coordinate of point *a* and *y(a):* the y-coordinate of point *a.* As presented in the example of Figure 3-6, *BOTTOM(B*) takes

$$B_s = \{(h_0,d_0), (h_1,d_1), (h_2,d_2), (h_3,d_3), (h_4,d_4)\}$$

and creates

$$C_s = \{d_0, h_1, d_1, d_2 -_x w, h_2 -_x w, h_3, d_3, d_4 -_x w)\}.$$

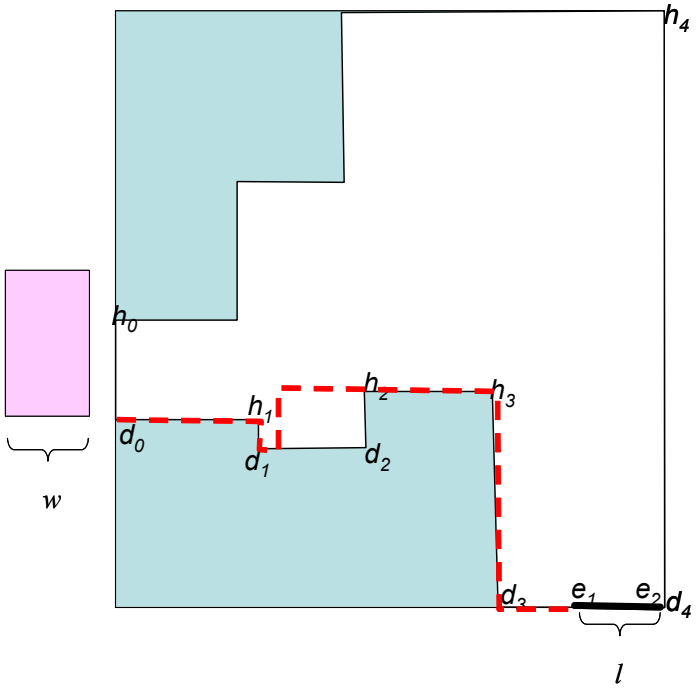


Figure 3-6 Finding C_s from B_s with (e_1, e_2) of width w

Now let us consider the pseudo code of *BOTTOM(B)*. The first four lines initialize the problem of placing (e_1,e_2) as shown in Figure 3-8(a), i.e., e_2=d_0, e_1=($x(d_0)$-w, $y(d_0)$), e_1 becomes the first point in C_s, and the horizontal edge queue, Q, is initialized to a null list. After appropriate updates have occurred (as detailed in *SLIDE2*), Q contains an ordered list of the horizontal edges that have been traversed by e_2 but still might have an effect on the final components of C_s. *line*(e_1, e_2) denotes a line passing through (e_1, e_2) and extending to the right. *support* denotes the right point of the current horizontal edge on which (e_1, e_2) is sliding.

The first support is the point where *line*(e_1, e_2) intersects vertical edge h_1d_1. This intersection can occur *only* at h_1 or d_1. *SLIDE2(start)*, with *start* ≡ *1*, is called to determine the rest of the trace, C_s. After returning from *SLIDE2(1)*, all c_i to the left of h_0d_0, and all c_i to the right of q_s are deleted. An example of such deletions are shown for subhole F_1 (with point q_1) in Figure 3-8(f). Right end deletions are required only in upper subholes to prevent including locations which do not have bottom support.

Procedure BOTTOM(B) (Chazelle 1983)
$(e_1, e_2) \leftarrow (d_0 -_x w, d_o)$ // See Figure 3-8a
$C \leftarrow e_1$
$Q \leftarrow \varnothing$
$support \leftarrow h_1d_1 \cap line(e_1, e_2)$
*SLIDE*2(1)
Remove from C all points c_i ; $c_i \leq_x d_0$ or $q_i \leq_x c_i$

Figure 3-7 - Pseudo Code for *BOTTOM(B)*

Next we provide an overview of *BOTTOM*'s core procedure *SLIDE2*. Chazelle's procedure, *SLIDE*, was recursive in nature. *SLIDE2* is a simpler non-recursive O(n) method which yields identical results. Figure 3-8 presents key events of *SLIDE2* and Figure 3-9 details its pseudo code. This subroutine moves *(e_1,e_2)* from left to right while maintaining non-penetrating contact with at least one of the vertical h_id_i edges in B_s. Three things can happen as *(e_1,e_2)* slides:

(1) *(e_1,e_2)* can move to the right while sliding on a support. In Figure 3-8 (b) the "support" is the upper right corner of the horizontal edge on which (e_1, e_2) is sliding. (In the less common case, where more than one horizontal edge

supports (e_1, e_2), the support is associated with the leftmost horizontal edge.) The vertical edge, h_1d_1, in Figure 3-8(b) and six vertical edges in Figure 3-8(d) are passed over since the support is unchanged and no additions to C_s were required.

(2) Figure 3-8(b) shows that (e_1,e_2) may impact a vertical edge. If this occurs, the bar must then rise to the top of the impacted edge and resume moving to the right.

(3) As shown in Figures 3-8(c&d), e_1 can clear *support* before making contact on the right with the next vertical edge either (i) by reaching the next horizontal edge at the *same* height as the current support (Figure 3-8(c)) or (ii) by allowing (e_1,e_2) to fall to the highest supporting horizontal edge. After determining a new *support,* (e_1,e_2) will continue to slide to the right. Figure 3.8(c) shows how a gap can be traversed between two horizontal edges at the same height. When e_1 slides past the current support, h_1, a "fall" of zero height results in the support being transferred to h_4. This special case requires no additions to C.

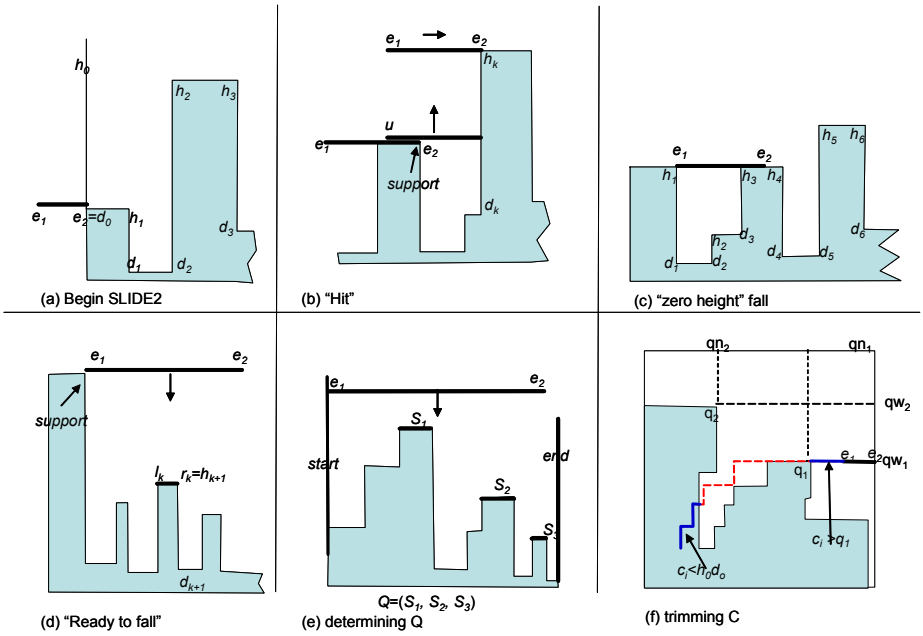


Figure 3-8 Determining *C* (Chazelle 1983)

Let us now consider the logic of *SLIDE2* as presented in Figure 3-10:

The $h_i d_i$, $i = 0, \ldots, |B_s|\text{-}1$ are the set of vertical edges that must be considered and the procedure argument, *start*, is the index of the first vertical edge that is considered for support. *tightfit* is a variable used in the consideration of a special case detailed below.

(1) The first segment of code deals with (e_1,e_2) impacting an $h_i d_i$. If the horizontal coordinate of $h_i d_i$ is less than w units from the current support, ($h_i d_i <_x support +_x w$) will be *true*. Further if the vertical coordinate of h_i is greater than the height of (e_1,e_2), ($y(h_i) > y(b_2)$) will be *true*. If both conditions are *true*, $h_i d_i$ is impacted, i.e., the next edge is reached before e_1 reached the *support* and the top of $h_i d_i$ is higher than (e_1,e_2)*'s* current height, as in Figure 3-

8(b). If a new point to be added is the same as the last point in C_s, it is normally not added to prevent duplicate points in the trace. However, if a special "tight fit" case (to be explained later) occurs, the new point is added to maintain left right pairing. Two points *w* units to the left of the impacted edge, first *u* and then the new e_1 are added to C_s. Next, if more vertical edges need to be considered, a new *support* is found, *Q* is emptied, and the index, *i*, is incremented for consideration of the next $h_i d_i$.

(2) If $h_i d_i <_x support +_x w$ and $y(h_i) \le y(b_2)$, the bar slides above the next $h_i d_i$ and $h_{i+1} d_{i+1}$ is considered. No additions to C_s are required.

(3) If neither condition in (1) or (2) holds, $h_i d_i \ge support +_x w$ and (e_1,e_2) falls to the height of the tallest horizontal edge between *support* and $h_i d_i$. Two key procedures used to find the edge on which (e_1,e_2) lands are *SETUP* and *MERGE*. These procedures result in a height prioritized queue of horizontal edges. *SETUP*, beginning at the edge i=*end* and working back to *start*, builds a queue *Q'*, of *new* height prioritized edges. If the preceding move was not an impact with an edge, there may be edges remaining in the previously constructed *Q*. *MERGE* takes any edges remaining in *Q* and consolidates them with the edges in Q' to form the new *Q*. This is performed in linear time. *MERGE* and *SETUP* will be discussed in more detail later. The top edge in *Q*, i.e, the edge with the lowest index, is where (e_1,e_2) will land. The point where (e_1,e_2) began to fall (e_1 = support), and the point where e_1 finished are added to C_s.

A *special* case was ignored by Chazelle. It occurs when the width of (e_1,e_2) is precisely equal to the distance from the current support to the next vertical edge. We explicitly consider this special case (where $h_i d_i = support +_x w$) and appropriate information is added to the trace. The fall procedure is essentially the same. As illustrated in Figure 3-9, the item falls into this tight fitting depression, point c_6 is added to C_s by the "fall" section of *SLIDE2*. This situation is "flagged" by setting *tightfit* = *true*. This allows redundant points to be added to C_s and is only present in the "hit" portion of *SLIDE2*. Upon return to the "hit" section, since the fall procedure did not increment i, the new support, d_i, satisfies the first "hit" condition. In Figure 3-9, i=6 and support = d_6, which clearly satisfies the hit condition since $h_6 d_6 <_x d_6 +_x w$. Since h_6 is higher than (e_1,e_2), a hit occurs. The *tighfit* = *true* condition allows a duplicate entry to be added to C_s which maintains the left–right pairing required for future algorithmic use. Two points w units to the left of the impacted edge, first u=c$_7$ and then the new e_1=c_8 are added to C_s.

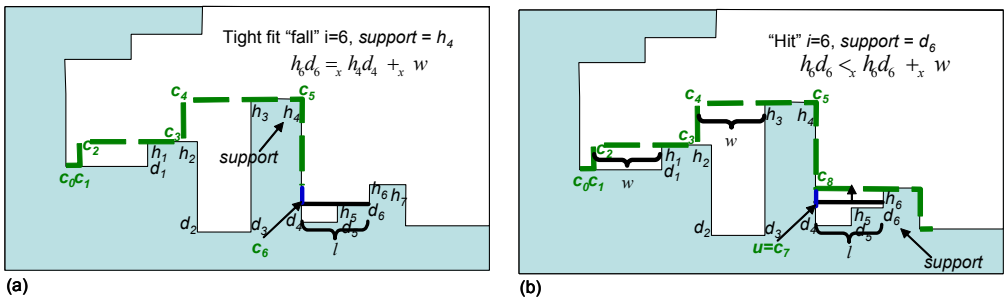


Figure 3-9 (a) A tight fitting "fall" allows (b) redundant point u=c_7

```
Procedure SLIDE2(start) //(Harwig 2003): non-recursive version of SLIDE (Chazelle 1983)
tightfit = false
while start ≤| B | −1 {//
  i ← start
    //(a) "hit" h_i d_i see figure 3-8(b)
    //Notation comment: for vertical edge h_i d_i both h_i and d_i have same x-coordinate
    if (h_i d_i <_x support +_x w and (y(h_i) > y(e_2))) {
        (e_1, e_2) ← (h_i −_x w, h_i)
        u ← e_1 −_y [y(h_i) − y(support)]
      if (u ≠ c_|C|-1 or tightfit = true){ C ← C ∪ (u)}
        if (e_1 ≠ c_|C|-1 or tightfit = true){ C ← C ∪ (e_1)}
        if(i <| B | −1) {
            support ← h_{i+1} d_{i+1} ∩ Line(e_1 e_2)
            Q ← ∅
            i ← i +1
            start ← i
            tightfit = false
      }//end if i<m
    }//end if hit
    //(b)"slide above"
    else if((h_i d_i <_x support +_x w and (y(h_i) ≤ y(e_2)))) {
        tightfit = 0
        i ← i +1                    //note: support and start do NOT change
     }//end else if "slide above"
    // (c)"fall" since h_i d_i ≥_x support +_x w see figure 3-8(d)
    else {
        tightfit = false
        e_1 ← support
        e_2 ← e_1 +_x w
        if (e_1 ≠ c_|C|-1 ){ C ← C ∪ {e_1} } // add the point before the fall
        Q' ← SETUP(start, i)
        Q ← MERGE(Q, Q')
        a ← POPl(Q)                         //l_k ( r_k ) is the left ( right) endpoint of segment a
        if (y(e_1) − y(l_k) > 0){           //if not a "zero height fall"
            (e_1, e_2) ← (e_1, e_2) −_y [y(e_1) − y(l_k)]   //update the bar position
            if (e_1 ≠ c_|C|-1 ){ C ← C ∪ e_1}               //add the new point
            if (h_i d_i =_x support +_x w ){   //special case "perfect fit"
            tightfight = true}                  // allows "hit" to add duplicate to retain left-right paring of C
        }//end if not a
        else { C ← C \ c_|C|−1 }            //"zero height fall" remove point just added from the end of C
        start ← i
        support ← r_k
    }//else "fall"
  }//end while
```

Figure 3-10 Pseudo Code for *SLIDE2*(*start*)

The above discussion did not detail how the highest edge for each fall is determined. *SETUP* and *MERGE* yield a height prioritized queue, Q, of horizontal edges. These comprise potential edges upon which (e_1,e_2) might land if a "hit" does not occur in the slide to the right. Q' is used to update Q. These queues are double ended, i.e., items may be added or removed from either end. The following actions are used in these queues: If $Q = \{l_0r_0, l_1r_1, \ldots, l|_{Q|-1}r_{|Q|-1}\}$, TOP1($Q$) = l_0r_0, TOP2(Q)= $l|_{Q|-1}r_{|Q|-1}$. POP1(Q) removes l_0r_0 from Q, POP2(Q) removes $l|_{Q|-1}r_{|Q|-1}$ from Q. MERGE(Q,Q') appends Q to the top of Q'.

Procedure SETUP(*start,end*), portrayed in Figure 3-8(e) and described in pseudocode in Figure 3-12, determines which new edges to add to Q'. The arguments *start(end)* indicate the left (right) most vertical edge that has not been considered for use in building horizontal edges. *end* is the last vertical edge from B_s checked when the fall occurred. *SETUP* checks all edges between "*start*" and '*end*'. *SETUP* starts at vertical edge *end* and builds a horizontal edge with the left adjacent vertical edge. Given *end* = h_kd_k, there are four possible ways that the vertical edges can define horizontal edges

($d_{k-1}d_k$, $d_{k-1}h_k$, $h_{k-1}d_k$, and $h_{k-1}h_k$)

SETUP determines which kind of horizontal edge is present and denotes it as

$l_{k-1}r_{k-1}$. $l_{k-1}r_{k-1}$ is added to the top of Q' if it is as high as the current *TOP1(Q)*. *SETUP* continues to work to the left until *k=start*. Once the new edges are determined Q' is merged with Q as pictured in Figures 3-11 (b) and (c).

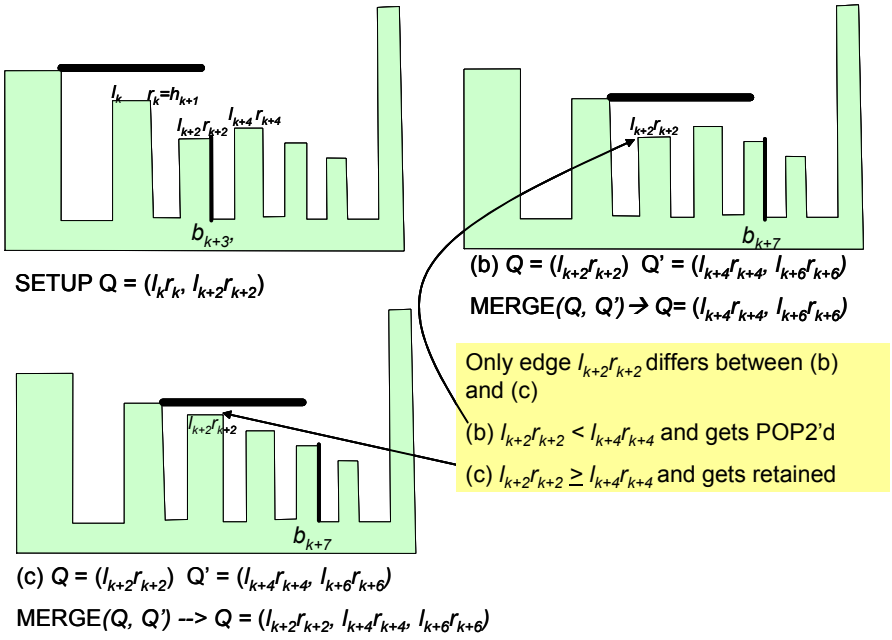


Figure 3-11 Updating *Q*. (a) SETUP, (b)MERGE, with POP2 (c) MERGE with no POP2

If a "hit" did not occur since the last fall, Q may have edges in it. The subroutine MERGE (*Q,Q'*) combines *Q'* and *Q* into a single queue. MERGE removes any element of Q that is lower than TOP1(Q'). Figure 3-11(b) shows the case where an element of *Q* is lower than TOP1(Q') and $l_{k+2}r_{k+2}$ is removed. Figure 3-11(c) shows the opposite case where the remaining element of *Q* is retained. The pseudo code subroutines SETUP and MERGE are presented in Figures 3-12 and 3-13.

```
Procedure SETUP (start, end) Chazelle(1983)
//"by convention, if Q=∅, then TOP1(Q) ≤_y l_k r_k for any edge in the structure.
//Q is here a local variable"
Q ← ∅
i ← end − 1
while (i ≥ start){
   if (TOP1(Q) ≤_y l_i r_i){
      Q ← l_i r_i ∪ Q        //l_i r_i ∪ Q appends l_i r_i to the top of Q
      }
   i--;
}
return Q
```

Figure 3-12 - Pseudo Code for SETUP(*start,end*)

```
Procedure MERGE (Q, Q') Chazelle(1983)
if (Q ≠ ∅ and Q' ≠ ∅){
     s ← TOP1(Q')              // TOP1(Q) gives the value of the first element of the queue Q
     while TOP2(Q) ≤_y s {     // TOP2(Q) gives the value of the last element of the queue Q
        POP2(Q)                //POP2 removes the last element of the queue Q
     }
}
Q ← Q ∪ Q'                     //Q ∪ Q' appends Q to the top of Q'
```

Figure 3-13 - Pseudo Code for MERGE(Q, Q')

Once *BOTTOM(B)* procedure is finished, the output is a set of horizontal edges that represent locations where the leftmost corner of an item of the given width, *w*, will fit on the bottom trace. The future procedures that check height need C_s to be listed in left right pairs, i.e., $C_s = \{(l_0 , r_0), (l_1 , r_1), ...(l_m , r_m) \}$, where *m* is the index of the last horizontal edge in C_s. For an item of width *w*, the

lower dashed (green) line in Figure 3-14 shows the loci of points for C_s. Note: Left stability and height requirements have not yet been determined.

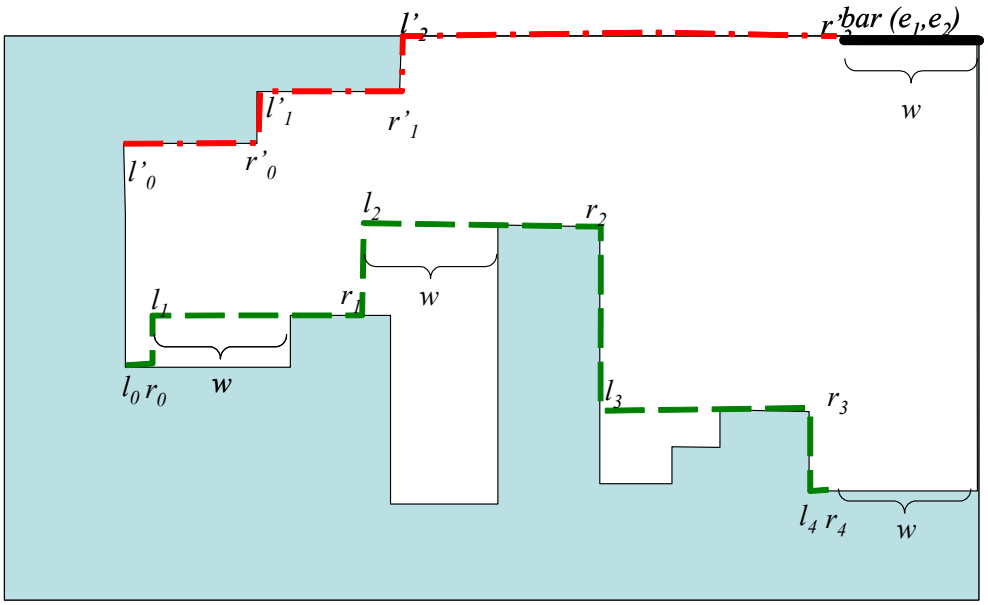


Figure 3-14 Final traces C (bottom green line) and D (top red line)

The second step in finding feasible packing locations is to find potential upper left corner points considering only the item's width and the top trace (FT_s), Finding the feasible locations top trace is much simpler than finding the bottom trace since there are no notches and at most one falling corner. However, recall that only trapped holes can have falling corners. Since there are no upper notches for all open holes, the edges of FT_s form an inverted stairway. The only remaining task is to determine where e_2 makes contact with the right side of the container and then to trim off all points to the right of e_1.

Let D_s be a set of horizontal edges that define the top trace's feasible top-left corner points for given item width. The relevant portion of Chazelle's heuristic *TOP(T),* presented in Figure 3-15, finds the points that define D_s. Notice that since this implementation only involves the case where "*fc*=0", this procedure is very simple. *TOP(T)* initially sets D_s equal to the top trace. i.e. $D_s = \{h_0, d_1, h_1, d_2 \ldots, h_{|T|-1}\} = \{f_0, f_1, f_2, f_3, \ldots f_{|D|-1}\}$. It then removes all the points that are less than w units from $h_{|T|-1}$. Prior to use in the future "placing" procedure, D_s is redefined in left right pairs: $D_s = \{(l'_0 , r'_0), (l'_1 , r'_1), \ldots(l'_p , r'_p) \}$, where p is the index of the rightmost horizontal edge in D_s.

```
Procedure TOP(T)  //(Chazelle 1983)
if (fc =0) { //"then no falling corner"- this is only case for open holes
   D ← {h0 , d1 , h1 ,..., h|T|−1}
   Remove from D all points fj ∈ D : fj > h|T|−1 −x w
}
else{//"then falling corner"
//This is only can happen to trapped holes.  See Chazelle (1983)
```

Figure 3-15 - Pseudo Code for TOP (T)

After determining C_s and D_s based strictly on the items width, the two are combined with the item's height in a heuristic to determine the feasibility of all potential BL-stable positions on the surface of C_s. Chazelle's heuristic *PLACING*(*h,C,D*) determines E_s. E_s is a set of potential "places" for packing an item. A *place* is an (x, y) coordinate combined with a Boolean indicator for feasibility. Figure 3-16 is used in this general overview of PLACING(*h,C,D*) These potential packing locations are determined by considering the x-position of

all vertical edges from either C_s or D_s and horizontal edges from C_s. This procedure moves left to right and determines the height feasibility of possible BL positions. Whenever the height from *C* to *D* changes, a new place is added. If the place is height feasible *PLACING*(*h,C,D*) sets a Boolean indicator to *true*. There is one place (at r_1) in Figure 3-16 that is infeasible for height.

PLACING(*h,C,D*) begins by checking the difference in height between the left ends of the first horizontal edges of each of the width based feasibility traces (C_s, and D_s) to determine if the vertical distance between these points is as least the height of the item (*h*) . The pseudo code for procedure Placing is presented in Figure 3-17. If $y(l'_0)$- $y(l_0) \geq h$, the Boolean indicator for height is set to *true* and the place at l_0 is added. It then increments to the next edge from C_s and begins a procedure which moves to the right determining height feasibility between the traces at each possible x-position for which this value may change. The procedure switches between incrementing along the top and bottom edges depending on which trace is changing. It consists of two coprocedures that continue checking the distance between the horizontal edges from each trace until all combinations have been checked.

These coprocedures perform symmetric roles. The "top" ("bottom") procedure works while the right end of the current bottom (top) edge is still a valid measure based on vertical alignment with the top (bottom) edges being considered. Hence, the top (bottom) coprocedure increments *j(i)*, as long as $r'_j <_X r_i$ ($r_i <_X r'_j$) and while edges still remain to be checked, i.e. $j<p$ ($i<m$). If $y(l'_j)$- $y(l_i) \geq h$, the place is marked *true* for height feasibility. The only

nonsymmetrical action between the two coprocedures are the places added to E_s. The bottom procedure adds two places corresponding to r_{i-1} and l_i. However, due to the inverted stairway structure of D_s, the top coprocedure adds only single places, $m_k \leftarrow (x(l'_j), y(l_i))$.

Once all interior edges have been checked, the procedure checks the right ends of last edge in each trace, sets the appropriate height feasibility indicator for this point and adds the final place corresponding to the location of C_s's r_m.

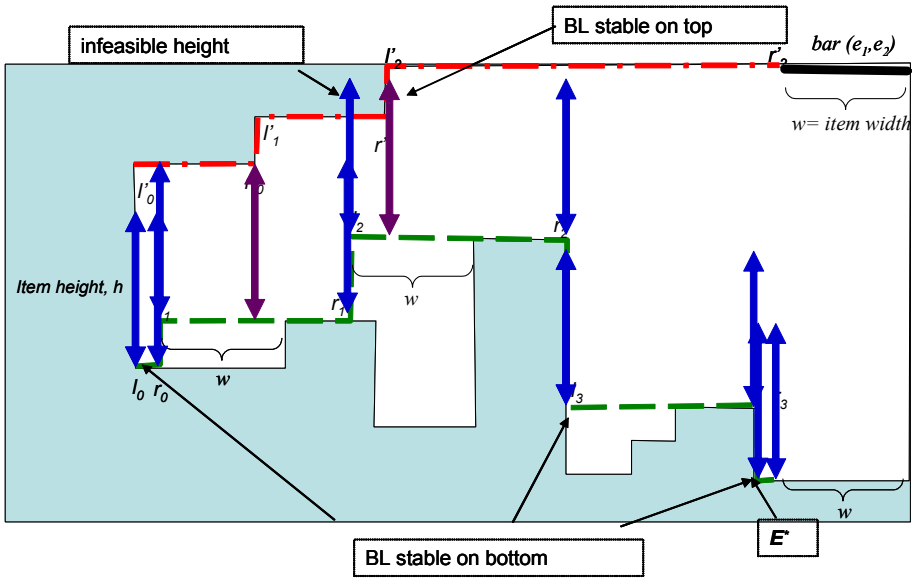


Figure 3-16 – Finding E

Procedure PLACING(h, C, D) //(Chazelle 1983)

$i \leftarrow j \leftarrow 0$

if ($y(l_0')$ - $y(l_0) \geq$ h) { //"then no falling corner"- this is only case for open holes

$E \leftarrow \{l_0, true\}$

}

else{//"then falling corner"- this is only can happen to trapped holes

$E \leftarrow \{l_0, false\}$

} $i = 1$

while (($i <| C |$) or ($j <| D |$)) {

//"top" coroutine increments the top edges for a fixed bottom

while (($r_j' <_X r_i$) and ($j <| D |$)) {

$j \leftarrow j + 1$

$m_k \leftarrow (x(l_j'), y(l_i))$

$if\,(y(l_j') - y(l_i) \geq h)\{$

$E \leftarrow E \cup \{m_k, true\};\}$

else{

$E \leftarrow E \cup \{m_k, false\};\}$

} //end while $(r_j' < r_i)$ and $(j <| D |)$

//"bottom" coroutine inrements the bottom edges for a fixed top

while ($r_i <_X r_j'$ and $i <| C |$) {

$i \leftarrow i + 1$

if$(y(l_j') - y(l_i) \geq h\{$

$E \leftarrow E \cup \{(r_{i-1}, true), (l_i, true)\};\}$

else{

$E \leftarrow E \cup \{(r_{i-1}, false), (l_i, false)\};\}$

} //end while $(r_i < r_j')$ and $(i <| C |)$

}// end while $(i <| C |)$ or $(j <| D |)$

if$(y(r_p') - y(r_m) \geq h)\{$

$E \leftarrow E \cup \{(r_m, true)\}\}$

else{ $E \leftarrow E \cup \{(r_m, false)\}$

p).

Figure 3-17 - Pseudo Code for PLACING(h,C,D)

Figure 3-17 marks the last of the psuedocodes that are given in this dissertation. From this point on, algorithmic methods are presented in detailed narrative descriptions with pictorial presentations as required. For those readers interested in the detailed coding of the methods, the source code of JAVA is available upon request from the Graduate Program in Operations Research and Industrial Engineering at The University of Texas at Austin.

The final result is $E_s = \{m_0, m_1, m_2,..m_{|E|-1}\}$ composed of trace C_s with augmented places corresponding to trace D_s's left ends and indicators at each of these places for height feasibility. Each point where the blue arrows touch trace C in Figure 3-16 correspond to an $m_k \in E_s$. The location where the top arrow of blue line is above D (designated "infeasible height") represents a place where the height indicator m_k is FALSE.

Chazelle's procedure stops once E_s is found and ignores the left stability of each of these places. Of course, left stability can not be ignored. In the example in Figure 3-16, the BL stable positions are indicated. There are two ways for an item to have left stability. It either makes contact on the left with a vertical edge from FB_s (points generated by the bottom coprocedure) or it can make contact with a vertical edge from FT_s (places generated by the top coprocedure). Places generated by the bottom procedure ($m_k \in E_s$) have left stability if $y(m_{k-1})>y(m_k)$. Places generated by the height feasible top procedure E_s have left stability if place m_{k-1} *is not* height feasible while $y(m_{k-1})=y(m_k)$. This indicates contact with a top vertical edge. When checking m_k the procedure does not know which coprocedure generated the point. This requires that these conditions be

checked in sequence. The Left Stability flag changes from its default value of *false* to *true* when one of these conditions is met. As indicated in Figure 3-16, there are three bottom procedure generated positions which are BL stable. It also has two top generated places (indicated by the purple lines), both are feasible for height, but only the rightmost is bottom left stable.

Now that left stability is known, let E^* be the packing position to be packed based on the selected rule. Chazelle uses the "lowest" point in the bin as E^* and breaks ties by taking the left most. Liu and Teng's (1999) method 'stair steps' by first moving down along the right wall of the container until it makes contact and then slides to the left until it can move down again. This repeats until it is settled in a position with both left and bottom contact. In Figure 3-16, E^*, is the same for both rules. However, Lodi et al.'s (1999-1) "touching perimeter" method would select the point furthest to the left on the bottom (with full left and bottom support).

This section detailed Chazelle's method of finding all feasible BL stable positions in O(n) time. Chazelle (1983) also claims that the update can be achieved in O(n) time and is a matter of "straight forward graph manipulation". However, he considers only the cases where the subhole splits or does not split. He ignores the numerous cases where subholes can merge back together. Thus the heuristic for finding the location is much easier than the task of updating the structure, merging and splitting subholes, updating special links, and considering the potential effect of left notches on the top trace in *all* lower subholes. The next section provides an overview on how these tasks are performed.

3.3 UPDATING THE DATA STRUCTURE:

Although the primary goal of this research was to implement a GTTS for packing, it quickly became apparent that the development of a *generic* method for efficiently and effectively updating the data structure was a necessary precursor to that effort. Many of these heretofore unknown special cases were discovered during the initial implementation of several different $O(n^2)$ Adaptive Tabu Search Bin Packing (ATS-BP) neighborhoods on two benchmark problem test sets that combined for a total of 500 sample problems. It is possible (and probable) that future investigations will yield additional previously unknown special cases.

When an item is packed into a container, two major updates must be performed:. (1) updating the H_0 parameters, i.e., updating the top (*FT*, *T*) and bottom traces (*FB*, *B*), and the special points *q*, *qw*, and *qn*, for all affected subholes (2) creating information on any new trapped holes. The trapped hole data is used in the ATS-BP procedures discussed in Chapter 4.

The update task requires determination of how the item effects various aspects of the data structure. All items must be supported on the left and bottom. These supports are identified first and this can be viewed as an alternate method to defining the location of the bottom left corner that corresponds to *E**.

The information cited in Sections 3.1 and 3.2 is not sufficient to update the data structure. The corner points of the item are also needed to determine which update cases are present. Figure 3-18 presents an example of these corner points and the supporting edge labels used to update the data structure.

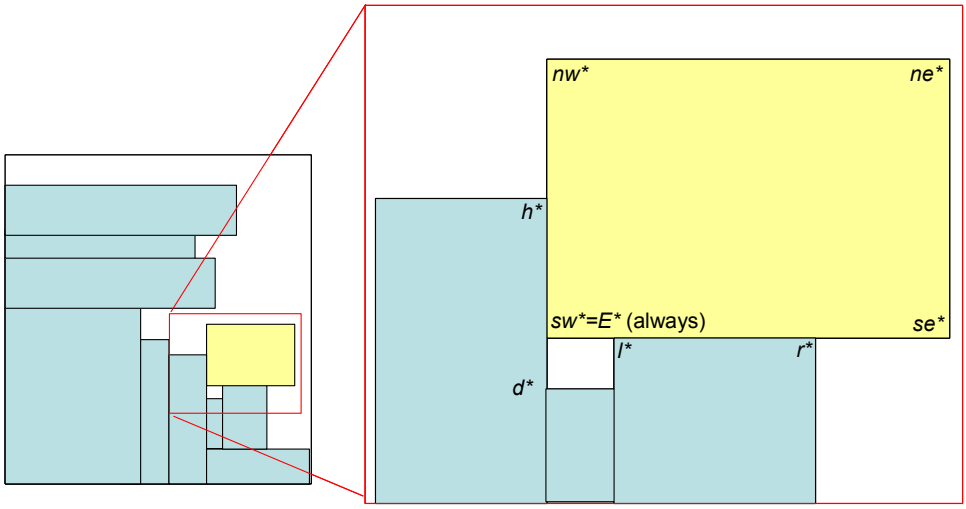


Figure 3-18 Notation used for update procedure

Figure 3-18 illustrates some important notation used in updating the data structure:

*h** : the top of the left edge support for *E**.

*d** : the bottom of the left edge support for *E**.

*l**: the left of the bottom edge support for *E**.

*r** : the right of the bottom edge support for *E**.

*ne**: the top left corner of the item after being packed at *E**

*nw**: the top left corner of the item after being packed at *E**

*se**: the bottom right comer of the item after being packed at *E**

*sw**: the bottom left corner is by definition equal to *E**

For H_0, the update requires determining if subholes are to be merged and/or split. Figure 3-18 provides examples of some of the cases which must considered:

a. *simple placement*, One that requires only updates to FB for the subhole being packed and traps no deadspace

b. *left side trapped hole*, Requires updates to FB and has deadspace to the left of the bottom support

c. *left side trapped hole with merge*, Has deadspace to the left of the bottom support and contacts a top edge that coincides with a higher subhole *q*

d. *right side trapped hole*, The item extends past the bottom support, but makes contact with FB creating deadspace and not requiring the subhole to split

e. *right side trapped hole with merge,* Similar to d. above but the right edge contact coincides with a higher subholes *qw*

f. *special link and subhole adjustments.* Placement of the item is on top of *q* requiring *q*, *qw* and probably *qn* to change positions.

These are not the only cases that must be considered in updating. Another common case is a placement creating a left notch and thus requiring a subhole to split. Figure 3-19 is intended to introduce and graphically portray some of the cases with their required actions. These actions are discussed in detail in Subsections 3.3.2, 3.3.3, and 3.3.4. Several actions can be required by a placement. A method to sequentially determine and implement required actions has been developed in the research documented here.

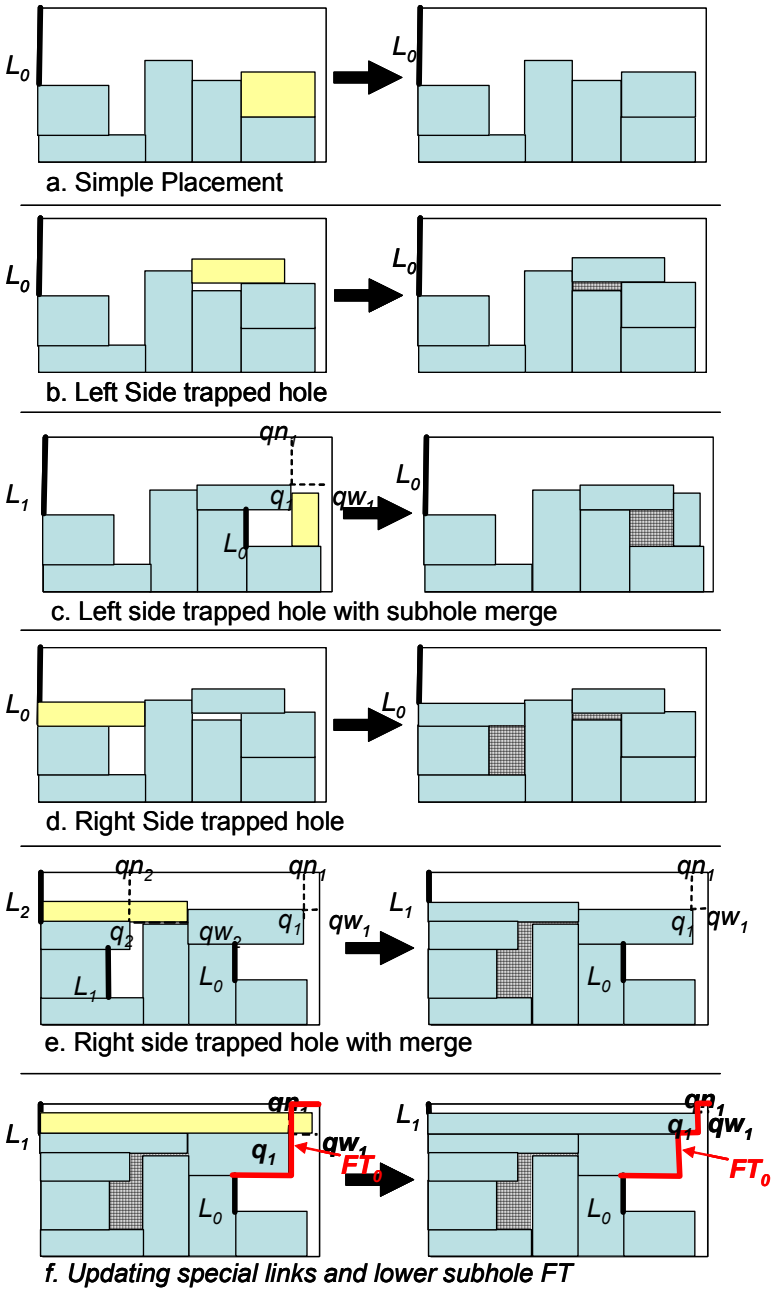


Figure 3-19 Examples of item placement and data structure interactions

This section is composed of four subsections. Subsection 3.3.1 presents some key data elements used in the update procedure and describes the data structure that allows sequential changes to the left and right of the place's bottom support. This eliminates the need for a potentially much more complex process that would consider all combinations of the possible cases. Subsection 3.3.2, Left Side Cases, presents update actions required by *incomplete* BL corner support at sw*, where a *complete* BL corner support is present if and only if $d^* = l^* = sw^*$. Subsection 3.3.3, Right Side Cases, presents update actions required when $se^* > r^*$. Finally, Subsection 3.3.4 presents updates for special cases such as *incidental top contact*,i.e., contact with the packed item above h^* on the item's left boundary and/or anywhere on the item's top boundary, and the removal of redundant points to nice fitting items. Each subsection considers a single bin and how the placement of one item will affect the bin's data structure.

3.3.1 The BL Packing Data Structure

The BL Packing Data Structure is determined from the chosen E^* for the bin and includes the current subhole where the new item is to be placed. Information associated with additional subholes within the main hole of the bin may also be required. To pack an item into a BL stable position and update the structure requires the bin's current data structure and information used to determine the location where the item will be packed.

3.3.1.1 BASIC CONCEPTS OF JAVA CLASSES :

The BL heuristic documented here was implemented in Java, an object oriented language. Some basic concepts about Java are covered here to aid in understanding both the data structures used in the update methods and the techniques employed in the Adaptive Tabu Search (ATS) implementation discussed in Chapter 4.

This section also describes how information is nested within " objects" in Java's Objected Oriented Programming structure.

(Wĕbopēdia)^{TM} defines *Object Oriented Programming* (OOP) as: "A type of programming in which programmers define not only the data type of a data structure, but also the types of operations (functions) that can be applied to the data structure. In this way, the data structure becomes an *object* that includes both data and functions. In addition, programmers can create relationships between one object and another. For example, objects can inherit characteristics from other objects." An *object* "is any item that can be individually selected and manipulated". (http://webopedia.internet.com).

Classes are templates from which objects can be created. Classes describe both the properties and behaviors of objects that belong to it. Classes are not objects, but instead are used to instantiate (i.e., create) objects in memory. (http://webopedia.internet.com).

In OOP is it common to build classes that consist of more primitive classes thus making new classes inherit all the features of these classes plus any additional features that need to be added. For example, the next subsection

discusses class *Bin*, which has primitive features such as *H* and *W* (height and width) and inherited classes such as Main Hole (class Hole) and Trapped Holes (an ArrayList of objects from class Hole).

3.3.1.2 THE DATA HIERARCHY FOR A BIN:

As shown in Figure 3-20, the data structure to define a "bin" has multiple echelons. A *Bin* is composed of classes for its Main Hole, Trapped Holes, Items Packed, additional primitive information such as bin dimensions, the location of the center of gravity (CG), the percent fill, and other bin specific information that assists in packing.

Class *Bin*

- *H*
- *W*
- *Main hole* (Class *Hole*)
  - Class *Subhole*
    - *FT* (ArrayList of Class *Coordinates*)
      - "Coordinates"
        - » *x*
        - » *y*
    - *FB* (same sub classes as FT)
    - *T* (ArrayList of Class *Vertical Edge*)
      - Vertical Edge
        - » *h* (Class *Coordinates)*
        - » *d* (Class *Coordinates)*
    - *B* (same sub classes as T)
- *Trapped Holes*(ArrayList of Class *Hole* same sub classes as shown for main hole above)
- *Percent Fill*
- Max *height*
- Max *width*
- *CG* (Class *Coordinates)*
  - *x*
  - *y*
- *Items Packed* (ArrayList of Class *Item*)
  - *Item*
    - *h*
    - *w*
    - *orientation*
    - S_n
    - *cg*
    - (Left Corner) (Class *Coordinates)*
      - *x*
      - *y*

Figure 3-20 Data hierarchy of Class *Bin*

The Main Hole and the elements in the *ArrayList* of Trapped Holes are essentially the same as the holes discussed in Section 3.1. An *ArrayList* has methods that allow adding and removing of items at any point within the array. In

this research all arrays, such as *FT* and *FB*, and the queues *Q* and *Q'*, make use of the ArrayList class.

Each Main Hole consists of a list of *Subholes* which are numbered from the bottom right. If you begin at the bottom of the right most edge of the hole, each leftmost edge corresponds to a starting point for a subhole's data structure. As shown in Figure 3-21, subhole "0" is rooted at edge L_0 which may be higher than succeeding L_i.

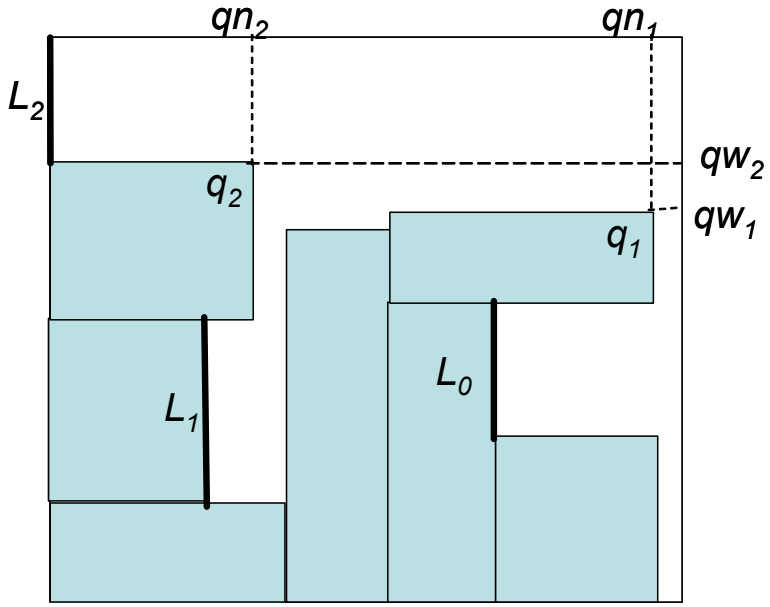


Figure 3-21 Subhole positioning and indexing

To record when a trapped hole was created, the algorithm stores the letter, in the permutation packing representation, of the packed item that created the trapped hole, the *Trapping Item.* Information about whether an item is either above and/or to the left of the trapped hole is valuable for structuring candidate moves in tabu search.

As previously mentioned, each subhole has a top trace *FT* and a bottom trace *FB*. All subholes, F_i, i > 0, will have the special links *q, qn,* and *qw*. A packed item may require updates associated only with a single subhole. However, packing of a single item can affect the data structure within multiple subholes.

A complete understanding of the data structure of a hole is essential for the following discussions. Key additional concepts beyond the trace update, detailed earlier, are subhole splitting, subhole merging, and the partial and complete trapping of subholes. Examples of each of these four concepts are presented in Figures 3-22, 3-23, and 3-24. Algorithmic details on the logic used to detect these situations are presented in the following sub-sections.

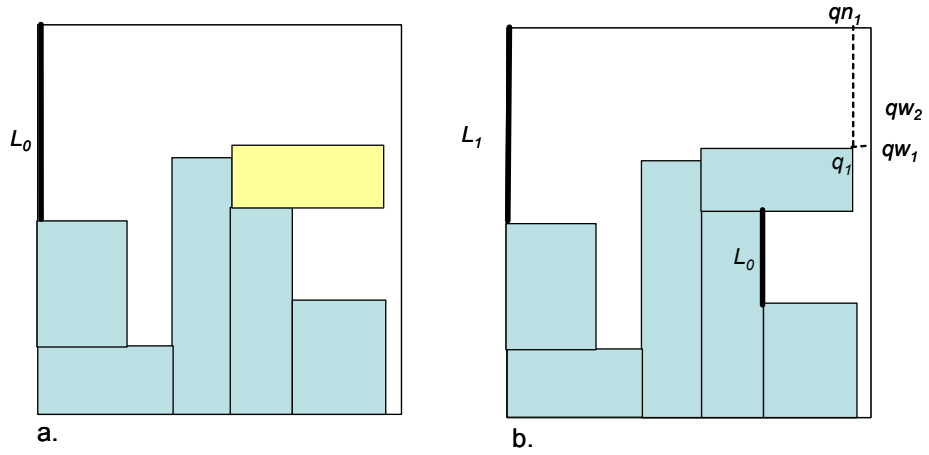


Figure 3-22 Subhole splitting action

Figure 3-22a shows the single hole, H_0, split into two subholes by the creation of a new left notch. All actions taken when the item extends past its bottom support to the right are discussed in Subsection 3.3.3, *Right Side Cases.*

Also included in Subsection 3.3.3 are details on special cases involving the partial and complete trapping of subholes. A partial trapping (Figure 3-23 a, b) converts part of a subhole into a trapped hole, while the remaining points of the trace that are accessible from the top are merged with the current subhole where the item was placed. Figure 3-23 c and d converts an entire subhole, subhole 0, into a trapped hole.

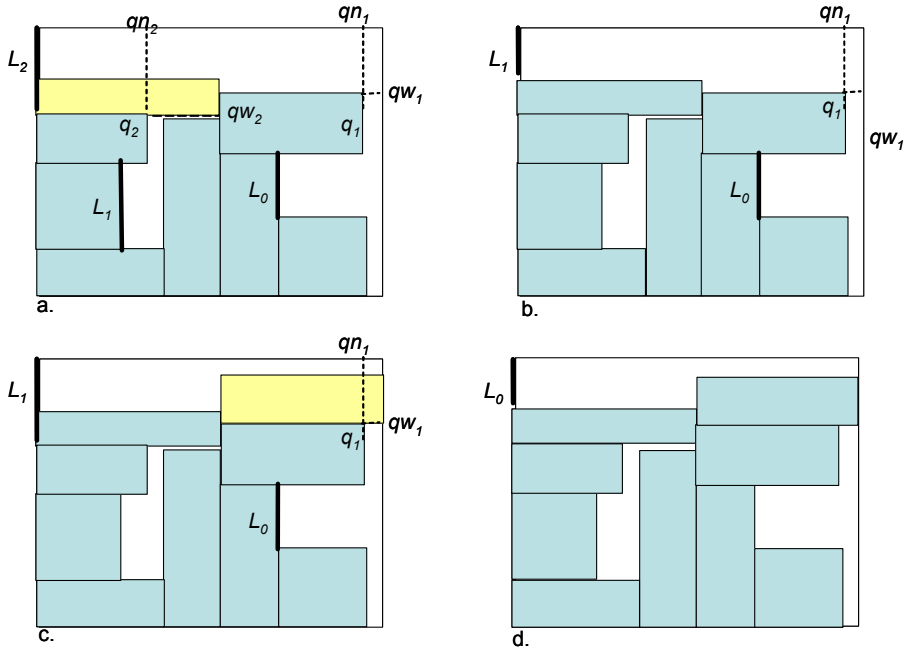


Figure 3-23 Partially (a & b) and completely (c& d) trapped subholes

Figure 3-24 shows the placement of another item such that the left notch no longer exists in the main hole and the two subholes, 0 and 1, must now merge into a single subhole. The logic and cases that require merging are presented in Subsection *3.3.2, Left Side Updates*. Techniques to adjust for other cases such as trapped holes that occur below the item are also presented in that subsection.

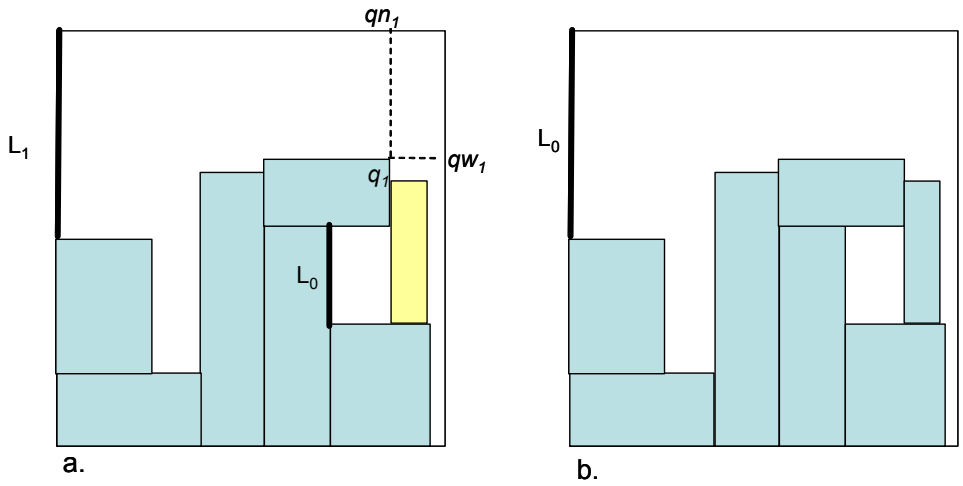


Figure 3-24 Subhole merging action

This subsection gave a brief overview of key update actions. All of these updates require the development of tests that categorize cases that can occur along the top, left side and right side of the item. Subsection 3.3.1.3 describes additional data needed so that common terminology can be used with the cases in Subsection 3.3.1.4.

As shown in Figure 3-25, the simplest type of update occurs when an item has complete BL corner support and is "fully" supported on its bottom side. In this case, $b_j = sw^* \equiv E^*$ and updating begins from point j. The procedure now sets $b_j = nw^*$, $b_{j+1} = ne^*$ and $b_{j+2} = se^*$ in the *FB* array. There are numerous variations of this simplest case that can occur. For example, if $nw^* = h^*$, nw^* becomes redundant and needs to be removed from the data structure. For this reason, nw^* is always checked. However, in this simple case, ne^* need be checked only when $w = r^* - l^*$, i.e., only when there may be a fit tight along the right side.

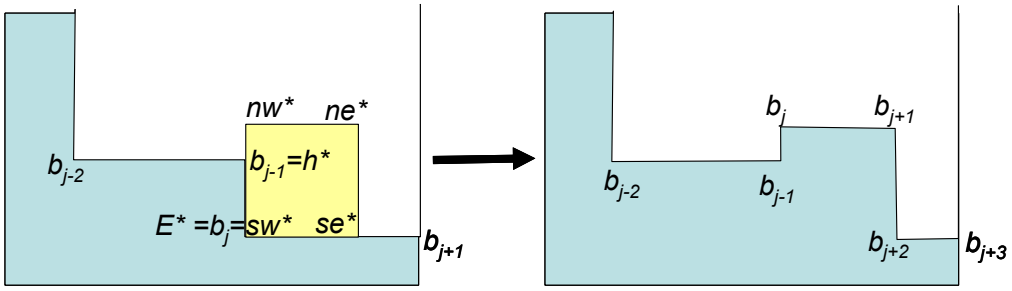


Figure 3-25 Simple update

As illustrated in Figure 3-26, *x(E*) may be less than x(l*)*, causing a trapped hole to the lower left. In this case the data structure for the *trapped hole* is obtained and then the data structure for the main hole is updated.

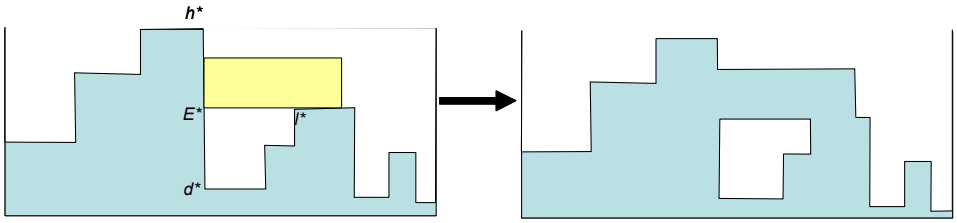


Figure 3-26 Trapped hole

A *new* and important finding allows update considerations between *h** and $b_{|FB|-1}$ (the last element of FB) to be greatly simplified by assuring that left side updates may always be independently performed before right side operations and any other special operations that might be required.

As discussed earlier, left actions include trapping holes between *h** and l* and possibly merging a subhole with a higher indexed subhole. There are special cases where the item makes contact with *FT* at locations other than on edge *h*d**. These special cases are considered after the left side and right side updates and are

presented in Subsection 3.3.4. Right actions address data structure changes to the right of r* and include updates associated with newly trapped holes, splitting a current subhole into two new subholes, adjusting for partially and completely trapped subholes and merging two current subholes.

To justify why left operations can be performed before other operations, define G ={g_0 , ... $g_{|G|-1}$ } to be the interior corner point "trace" of the main hole from the top right corner, scanning counter-clockwise, to the bottom right corner as illustrated in Figure 3-27. Partition G into three ordered subsets [*GH*|*GL*|*GR*]. Let $g_{|GH|-1}$ be the point prior to *h** so that three ordered subsets are *GH*={g_0, ... $g|_{GH|-1}$} *GL*=={*h*, ..., l**}. and *GR*={*r*,..,* $g_{|G|-1}$}. Set *GL* may contain portions of multiple subholes from the current data structure for H_0. However, once the left side updates are complete it will have *only* 2 elements: GL_{new}={*h*, d*=sw*= l**}. All other points, including those from subholes between *h** and *l*,* become part of a trapped hole or are removed because they are not needed to define any structures. Excluding sw*, *no* data points from the item are added in the left side update. The left update is independent of *GH* or *GR*. The only changes to the current data structure are updates of the indices associated with GR. Figure 3-27 shows an example of this situation. Figure 3-27a shows the H_0 structure and the new item to be placed at E*. In Figure 3-27a, there are three subholes that compose H_0 and superscripts on the top and bottom traces designate the subhole indices. Figure 3-27b shows the interior description *G={GH|GL|GR}* before left side adjustments. Notice that the special links in Figure 3-27a are not associated with corner or end points and have no counterpart in 3-27b. Figure 3-27c shows

$G=\{GH|GL_{NEW}|GR_{NEW}\}$. All points from the old d* to the old l* are now represented by a single point d*=l*=sw*=E*. Figure 3-27d shows the new H_0 after the left side adjustments have taken place. Now there are only two subholes and subhole 0 is much smaller.

At this point, the operations associated with correcting for an incomplete BL corner in the data structure are complete. Note that the item has not yet been inserted. Updates required for insertion are performed by later actions.

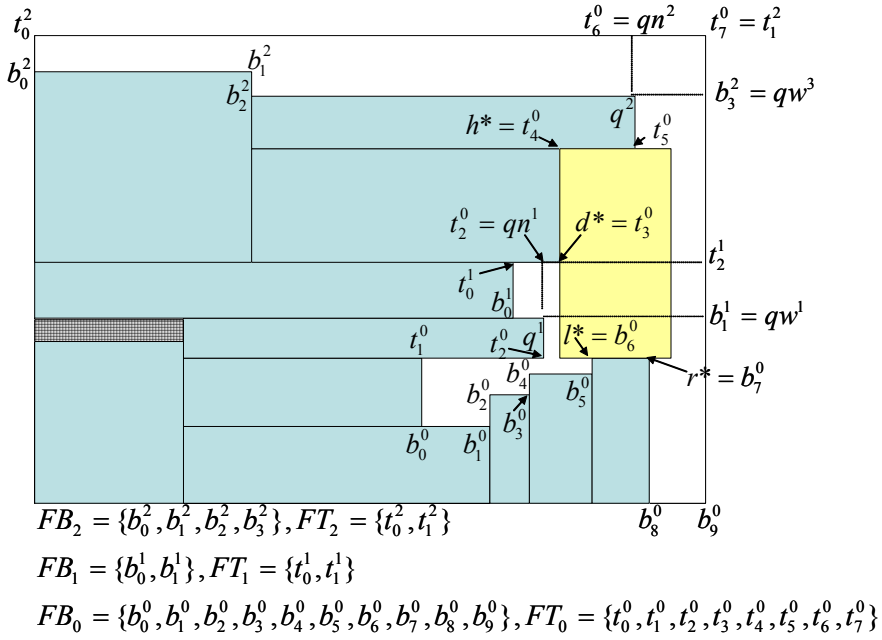


Figure 3-27a *F* representation

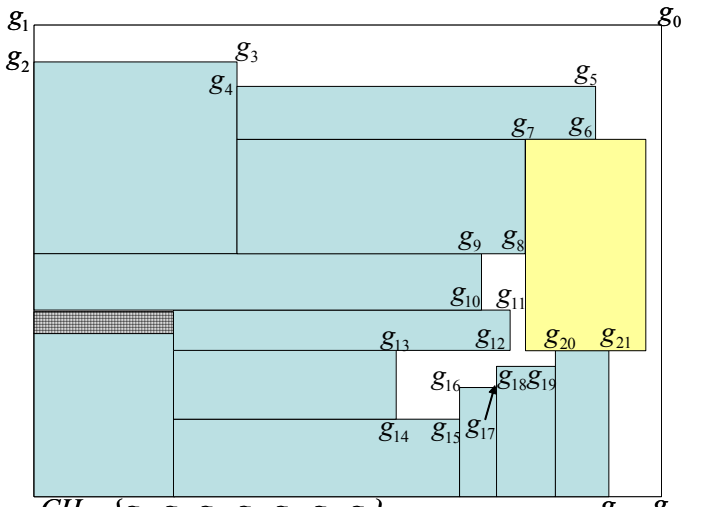


Figure 3-27b *G* representation

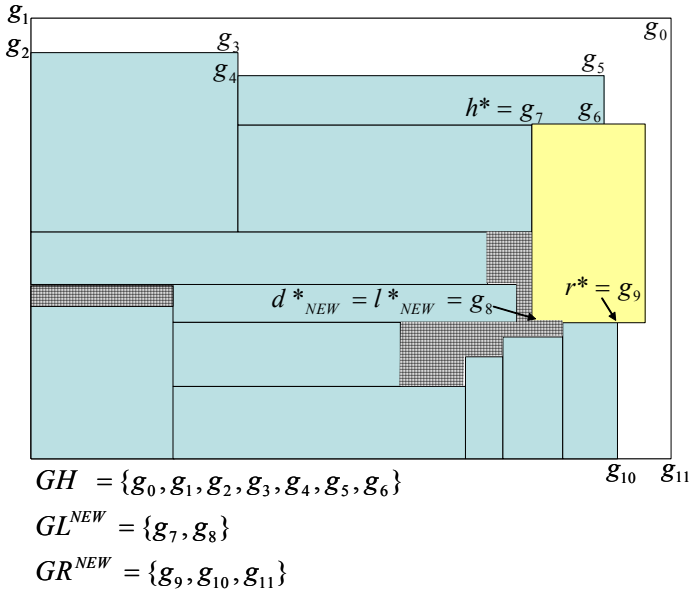


Figure 3-27c G^{new} representation

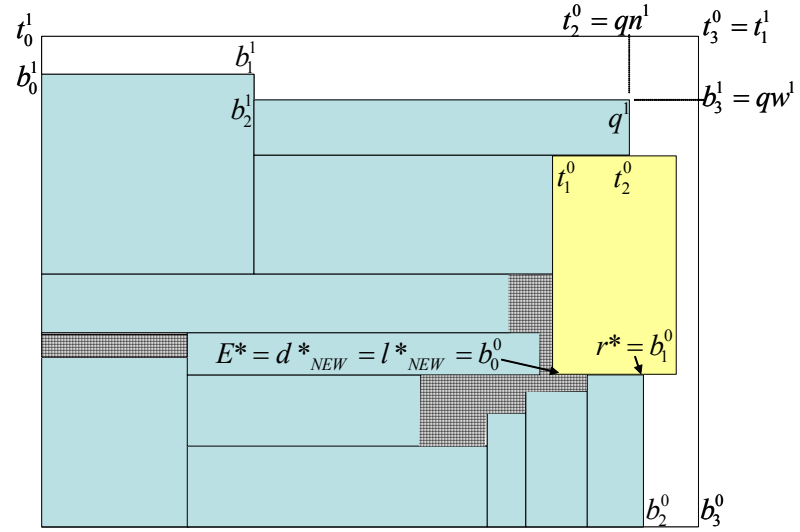


Figure 3-27d F^{new} representation

The number of subholes with higher index than the subhole embracing E^* may decrease but these changes are independent of remaining updates. This fact allows the updates to occur in the order, left side, right side, and top contact updates and removes the need for other more complex approaches.

The methodology described above greatly enhances the efficiency of identifying and updating new sub-cases.

3.3.2 Left Side Cases.

When $d^* \neq l^*$, the item placement requires that some portion of the trace to the left of l^* be removed from the new H_0 data structure. The trace that is removed becomes part of the description of the newly created trapped hole. Additionally, the remaining, untrapped, portion of the current subhole, s, may merge with another subhole, $u > s$, if contact is made with a vertical edge containing q_u. This section provides narrative detail on how these adjustments are performed.

When $d^* \neq l^*$, additional checks determine which subcase exists. The relative position of l^*, E^*, d^*, and any q_j (the upper right corner of a left notch) of a higher subhole are key features for discriminating between the presence of a simple left side trapped hole or a situation requiring a trapped hole/merge combination.

Let s be the current subhole, i.e., the subhole containing E^*. The simplest left side case occurs when $d^* \in FB_s$. If this occurs, d^* will be below E^*, l^* will be to the right of E^*, and there will be a trapped hole below and to the left of l^*. This case is shown in Figure 3-28. This case requires the removal of trapped points from FB_s and adding them to the data structure for trapped hole, t. A trapped hole that is formed exclusively from points from FB_s is a *simple* trapped hole.

This removal is straight forward. Let j be the index of d^* and k be the index of l^*. The procedure removes points j through k from FB_s and adds points j through $k-1$ to FB_t. E^* and l^* are added to FT_t, the trapped hole's top trace.

Finally, point $E^*=sw^*$ is added to FB_s at position j. The new FB_s is passed to the next section of code as a smooth left corner for item placement.

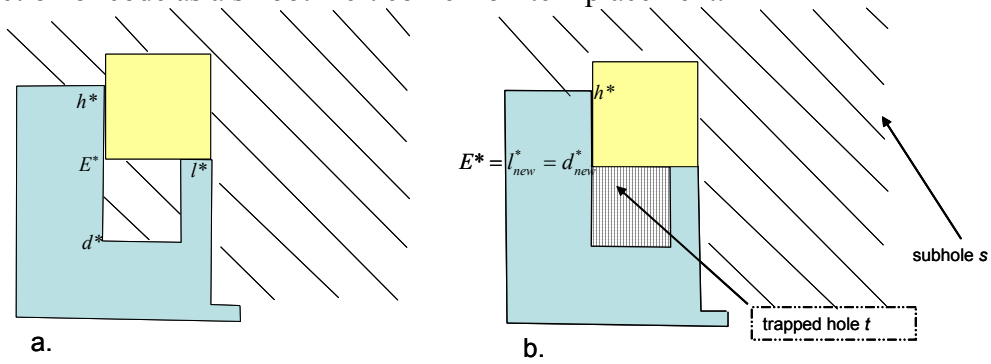
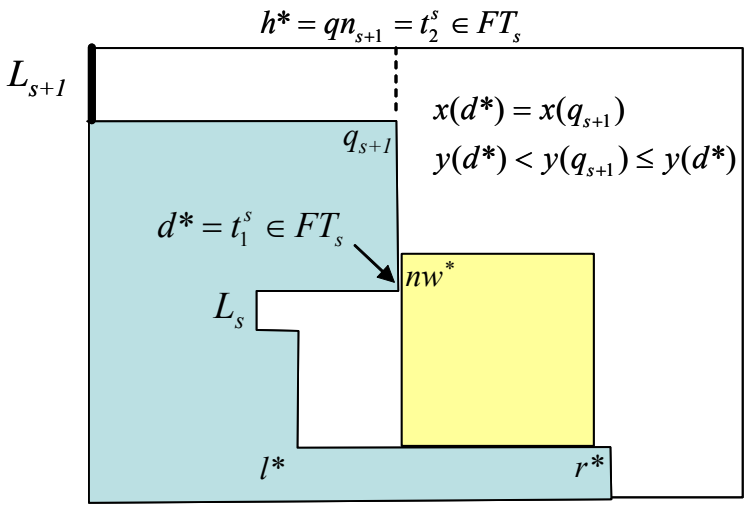
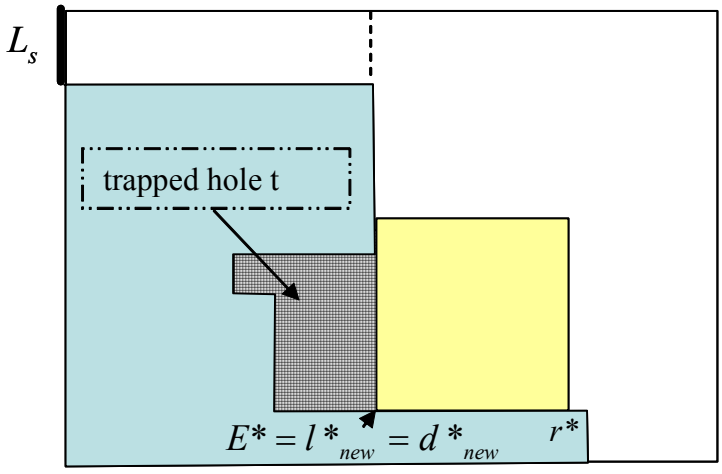


Figure 3-28. Simple (i.e. no merge) hole on lower left

If $d^* \in FT_s$, there will be a trapped hole to the left and it is necessary to check for merge conditions. For clarity of explanation, consider Figures 3-29 and 3-30 which picture two cases of adjacent subholes s and s+1. Figure 3-29a illustrates when a left side merge is required, i.e., whenever $x(d^*) = x(q_{s+1})$ and $y(d^*) < y(q_{s+1}) \leq y(h^*)$. After the item is inserted, subholes s and s+1 are merged to become the *new* subhole s as presented in Figure 3-29b. However, as shown in Figure 3-29b, if the lower end of the edge passing through q_{s+1} does not touch the item's right side, then the subholes do not merge and subhole s is redefined.

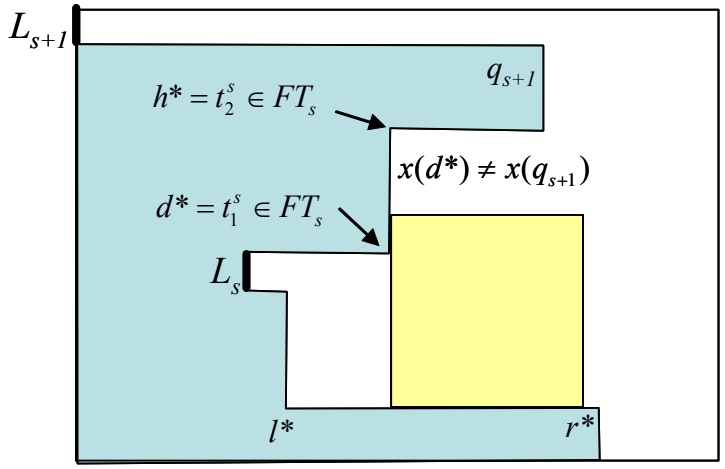


a. Subholes *s* and *s*+1

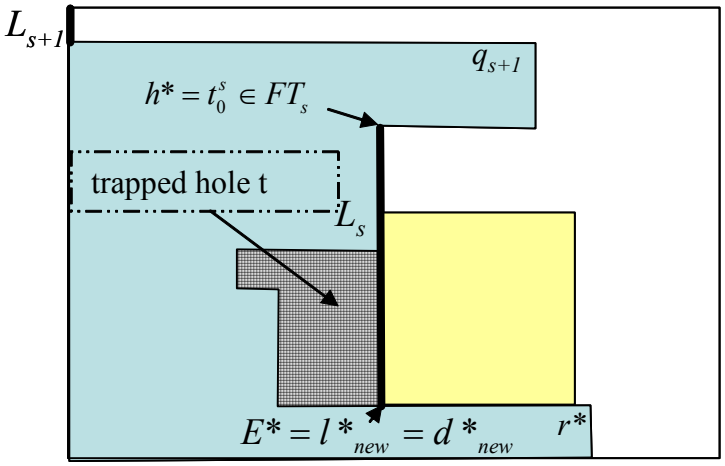


b. New subhole *s*

Figure 3-29 Top contact with hole on left and merge



a. Subholes s and s+1



b. Adjusted subhole s.

Figure 3-30 Top contact with hole on left but no merge

If no merge is required, the update is simplified. The trapped portion of FT_s is removed and becomes FT_t. Similarly, all trapped points in FB_s are added to

FB_t. Finally, the point E^* is added as the last point in FT_t and the first point in the new FB_s.

Now suppose that either of the conditions shown in Figures 3-29 and 3-30 are satisfied for non-adjacent subholes, u and s where $u > s+1$. The capturing procedure will be complicated by the presence of one or more additional subholes on the left which will be trapped. An example of this is shown in Figure 3-31 where subholes 2 and 0 are merged.

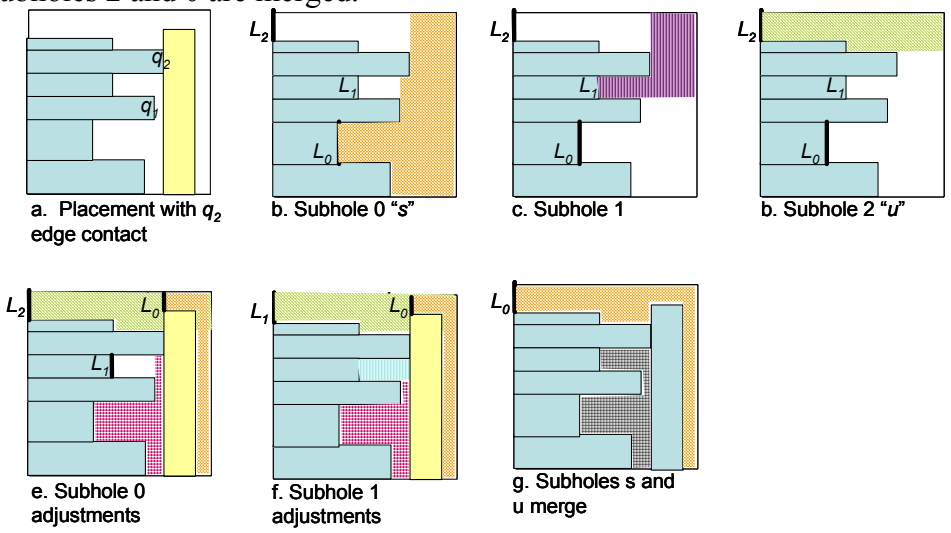


Figure 3-31 Adjustments for left side trapped subholes and left side merge

This section's final case concerns a trapped hole with a "falling corner" which occurs if $d^* \in FT$ and $E^* < l^*$. Updating procedures for this case are similar to the case just described. However, the trapped subholes top trace, FT_t, is updated differently since the falling corner introduces two additional points, as shown in Figure 3-32.

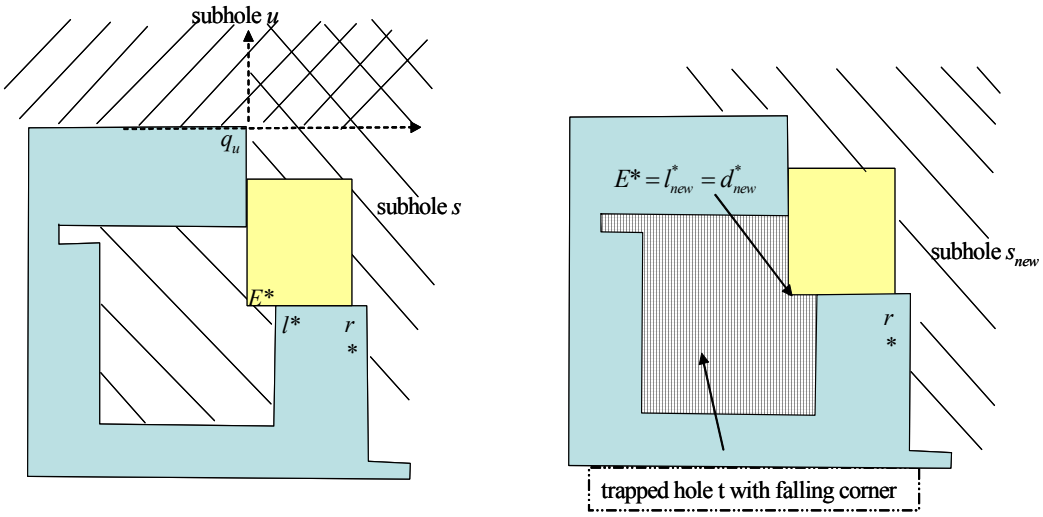


Figure 3-32 Trapped hole with falling corner case

This section considered cases where the item's lower left corner would have incomplete BL support. This left side update creates a complete corner where all structural points d^* through l^* are replaced by a single new point $d^*=l^*=sw^*$ and required merge operations are performed.

3.3.3 Right Side Cases

If $x(se^*) \leq x(r^*)$, no left notches can be created and the simple update presented at the start of Section 3.3 is used. As shown in Figure 3.33(c), a left notch is created if and only if $x(se^*) > x(r^*)$ and se^* is not horizontally or vertically supported.

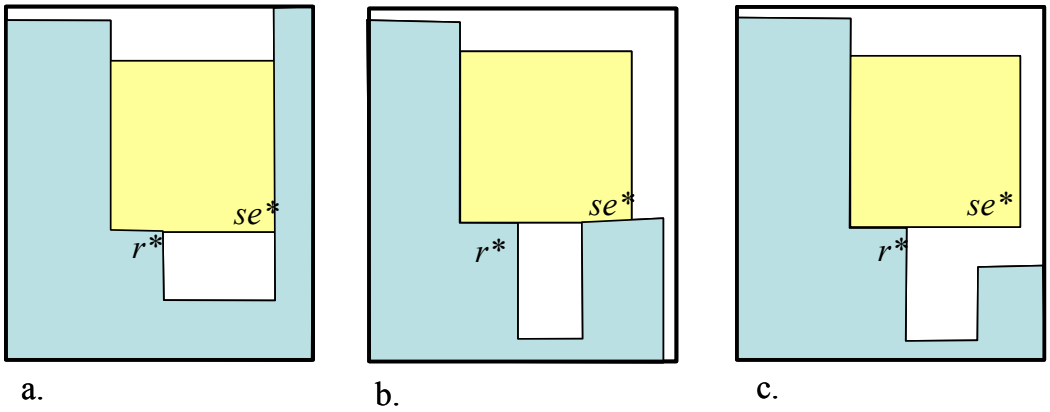


Figure 3-33 (a) Right hand support (b) horizontal support (c) no support at *se**.

There are two major right side cases: Either the item does *not* touch the far right vertical edge of the current subhole (Case 1) or it does (Case 2). Case 1 has no trapping or merging action with lower indexed subholes, but includes the subcases that require splitting the current subhole. Case 2 is more complex since it involves the possibility of complete trapping and/or partially trapping and merging with lower subholes. The following outline presents a taxonomy of the various subcases for Cases 1 and 2.

Case 1: $x(se^*) \neq x(b_{|FB|-1})$

1a: Horizontal or vertical support at *se** from FB

(No new subholes and one or more simple trapped holes)

1a(1): Right side support

1a(2): Bottom support

1b: No horizontal or vertical support at *se** from FB

(Overhang on right)

1b(1): New left notch and subhole

1b(2): No new subhole

Case 2: $x(se^*)=x(b_{|FB|-1})$

2a: one or more simple trapped hole(s) One or more points between r^* and $b_{|FB|-1}$ and

2a(1): only simple holes are trapped

2a(2): one or more trapped subholes

2a(2)i: no partially trapped subhole

2a(2)ii: a partially trapped subhole

2b: trapped subhole(s)

(No trapped simple holes, i.e., no points between r^* and $b_{|FB|-1.}$)

2b(1): no partially trapped subhole

2b(2): a partially trapped subhole

Case 1 and case 2 examples are provided in Figures 3-34, 3-35 and 3-36

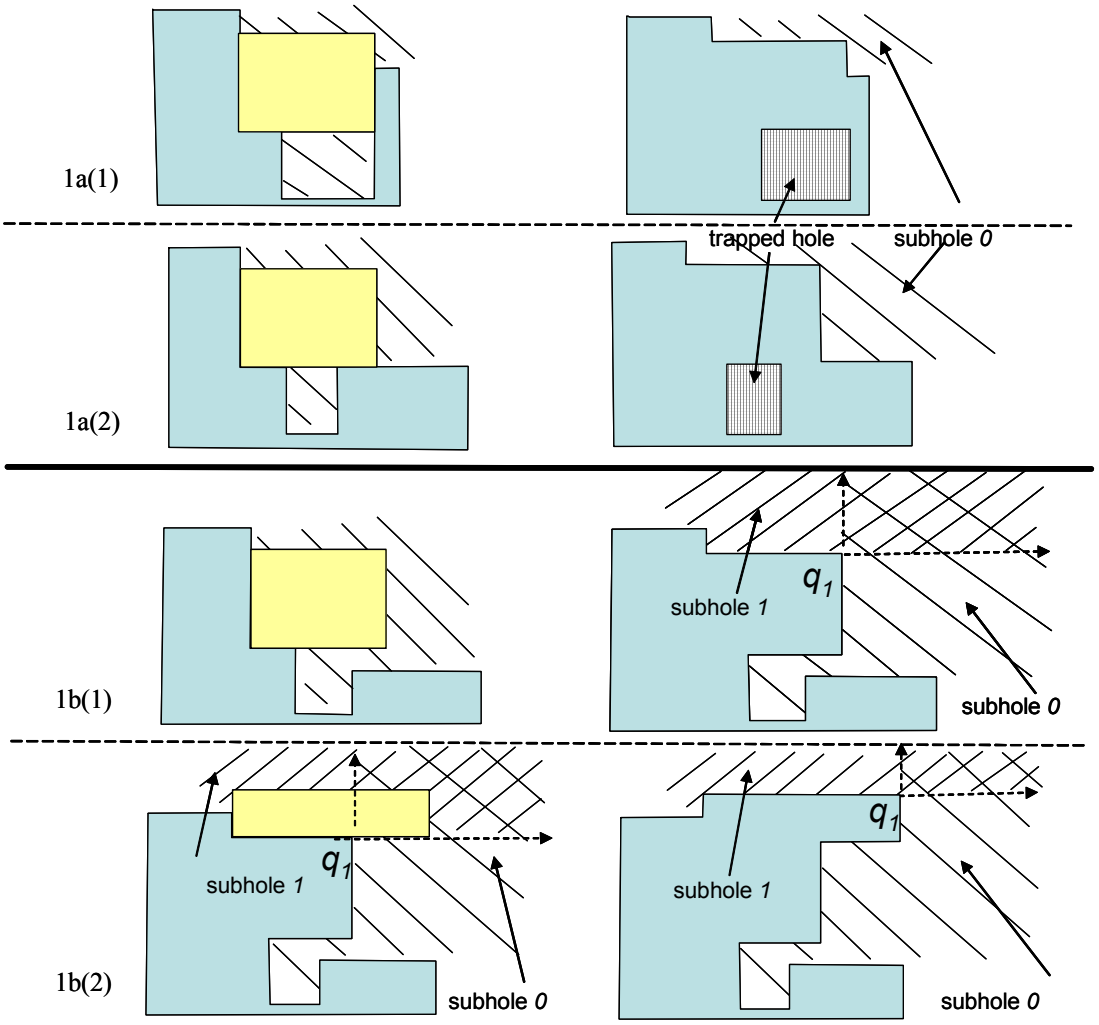


Figure 3-34 Case 1 a(1):Right side support, a(2): Bottom support, 1b(1): Overhang, New left notch and subhole, 1b(2): Overhang, No new subhole

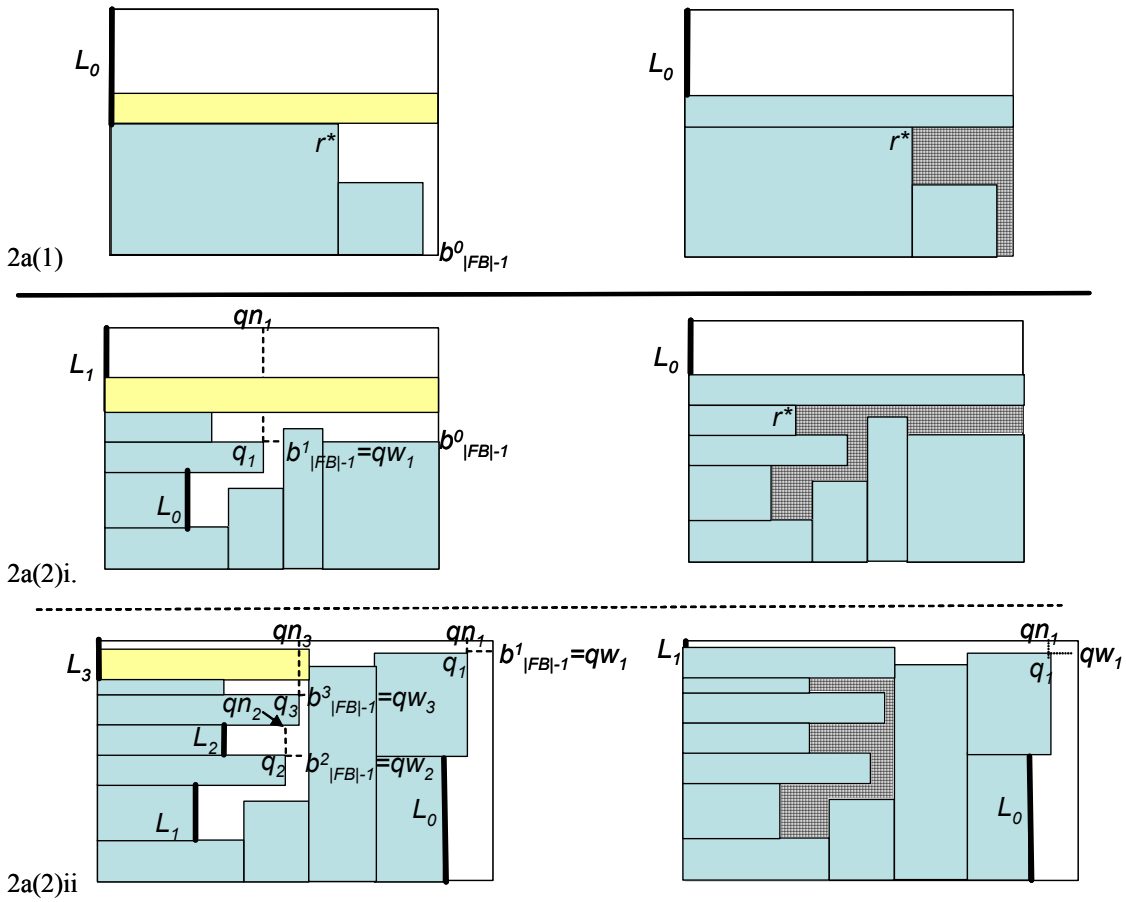


Figure 3-35. Case 2a subcases. 2a(1): only simple holes are trapped, 2a(2)i: no partially trapped subhole, 2a(2)ii: a partially trapped subhole

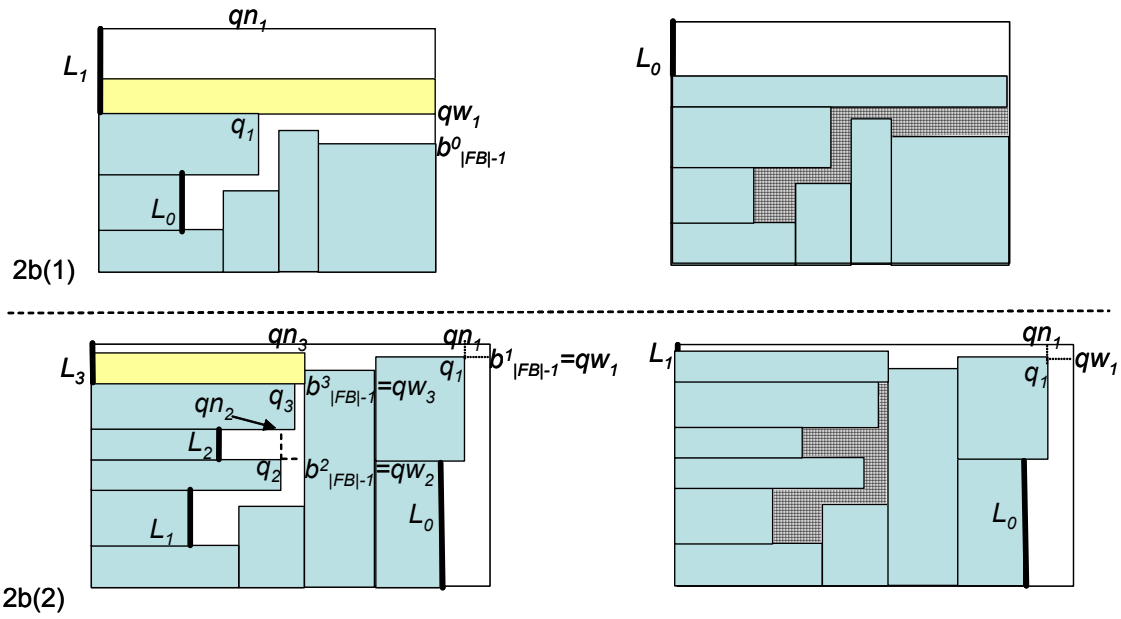


Figure 3-36. Case 2b subcases. 2b(1): no partially trapped subhole
2b(2): a partially trapped subhole

Case 1 ($x(se^*) \neq x(b_{|FB|-1})$) invokes a search of the $b_i \in FB_s$ from r^* to the right until $x(b_i) > x(se^*)$ to determine whether or not se^* is supported. If support at se^* is found, case 1a is present and one or more simple trapped holes are created. If se^* has no support , case 1b is present. Figure 3-37 presents an example of case 1a(2) where more than one simple trapped hole is created.

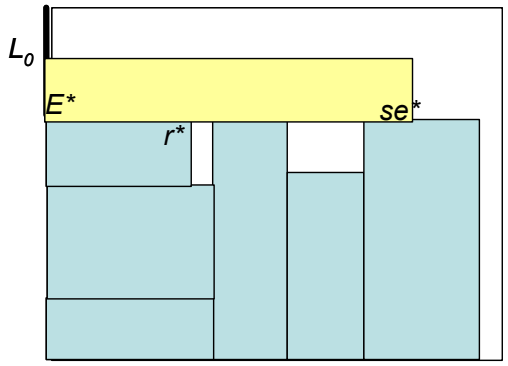


Figure 3-37 Multiple simple trapped holes right of *r**

In a somewhat more detailed view, the search traverses the bottom trace to the right of *r** until one of three conditions occur:

Case 1a(1)
$$b_i: \ x(b_i) = x(se^*),\ y(b_i) \geq y(se^*)$$
$$b_{i-1}: \ x(b_{i-1}) = x(se^*),\ y(b_{i-1}) \leq y(se^*)$$

Case 1a(2)
$$b_i: \ x(b_i) \geq x(se^*),\ y(b_i) = y(se^*)$$
$$b_{i-1}: \ x(b_{i-1}) \leq x(se^*),\ y(b_{i-1}) = y(se^*)$$

Case 1b
$$b_i: \ x(b_i) > x(se^*),\ y(b_i) < y(se^*)$$
$$b_{i-1}: \ x(b_{i-1}) < x(se^*),\ y(b_{i-1}) < y(se^*)$$

When case 1b is present, the algorithm determines which subcase, 1b(1) or 1b(2), exists, as illustrated in Figure 3-34. In case 1b(1), the current subhole is split into two subholes and the data structure is updated. In case 1b(2), the current subhole is not split and the data structure is updated including the required new special links associated with the new left notch. (Readers desiring greater detail

on these cases are referred to the JAVA source code available upon request from the Graduate Program in Operations Research and Industrial Engineering at The University of Texas at Austin

Figures 3-35 and 3-36 illustrate case 2 where $x(se^*) = x(b|FB|-1)$. The two main subcases are (2a) with simple trapped holes, with possible completely trapped lower subholes, and/or a partially trapped lower subholes and (2b) with no simple trapped holes but with completely trapped lower subholes, and/or a partially trapped lower subhole.

The existence of a simple trapped hole (case 2a) is determined if one or more points exist between r^* and $b_{|FB|-1}$ in FB_s. If case 2a exists, additional logic determines if only simple trapped holes are present (case 2a(1)) or if the trapped holes are accompanied by one or more trapped subholes (case 2a(2)). If case 2a(2) is present, the presence of a partially trapped subhole is ascertained.

Figure 3-35a(1) shows a simple trapped hole. Figure 3-35a(2)i shows a simple trapped hole with a completely trapped subhole below. Figure 3-35a(2)ii illustrates a simple trapped hole with one completely trapped subhole and one partially trapped subhole below. Partially trapped holes *always* require merging. The situation characterized by Figure 3-35a(2)ii could have multiple completely trapped holes, but there can only be one hole that is "partially trapped."

The situation of Figure 3-35a(2)ii embodies the two other examples in Figure 3-35 as special cases. In order to update the data structure for H_0 in this most difficult case, the partial subhole that is trapped by the inserted item must be identified. This is achieved by traversing the subhole array list, removing all

completely trapped subholes until the partially trapped subhole is identified. The bottom trace of H_0 is then updated by deleting all points associated with the new trapped hole and appending all remaining points above and to the right of *se**. The deleted points are stored in a data structure that describes the new trapped hole.

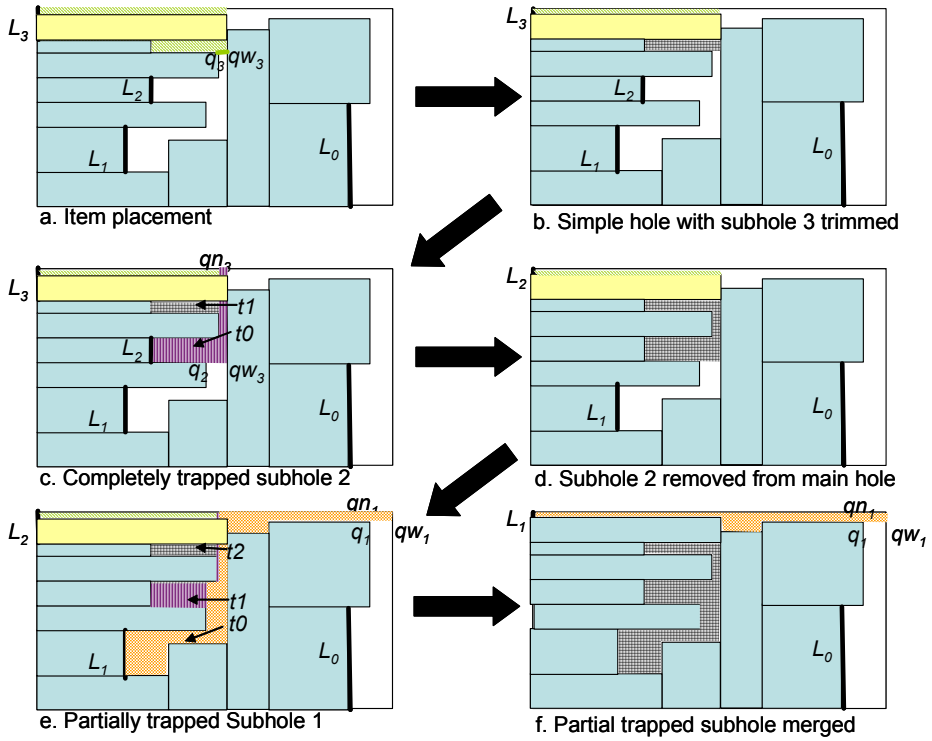


Figure 3-38 Case 2(a)ii update sequence

Figure 3-38 pictures this process for a typical example of case 2(a)ii. Case 2(b) shown in Figure 3-36, is managed in an exactly analogous fashion. (Once

more, readers desiring more algorithmic detail are referred to the associated JAVA code.)

Once this process is complete, the data structure is ready for the final special case checks in Subsection 3.3.4.

This section presented an overview of the right side cases and provided detailed descriptions of representative cases. The next section covers some additional special cases that have to be considered after correcting for the left side/right side cases.

3.3.4 Additional Checks

The ability to implement complicated left side and right side changes sequentially greatly simplifies the update procedure. However, there may be additional required updates. This final subsection discusses three actions not considered by left side or right side updates. First, we consider when an item contacts the top trace and how this may trap a hole on the left and/or invoke a merge condition not detected by previous updates. Second, we explain how changes in one subhole can affect the special links and top traces in adjacent subholes. Finally, we consider the detection and removal of redundant points from the top and bottom traces.

The left side updates focused on the interaction between the supporting edges (h^*d^* and l^*r^*) with the corners of the item. It did not place the item but provided a complete corner where $d^*=l^*=E^*$. The right side update placed the item and, if appropriate, saved a portion of the data structure associated with trapped holes and merged the current subhole with a subhole of lower index. To

this point, we have not considered the possible effects of the top of the item contacting the top trace of the current subhole. If such a "top" contact occurs, the following test for a merge condition must be conducted: If the $h_k d_k$ edge passing through q_u for any subhole u>s touches the top of the item, or *side* of the item, above h^*, the subholes must be merged into one subhole. Figures 3-39a and 3-39b both show top contact. However, only Figure 3-39a meets the merge condition. This merge condition may not indicate the existence of a hole since 'nice fits' could meet these merge condition. The holes shown in Figure 3-39 are above h^* and would not be detected by the Left Side actions. Thus, a hole-capturing feature is required by the top contact merge procedure.

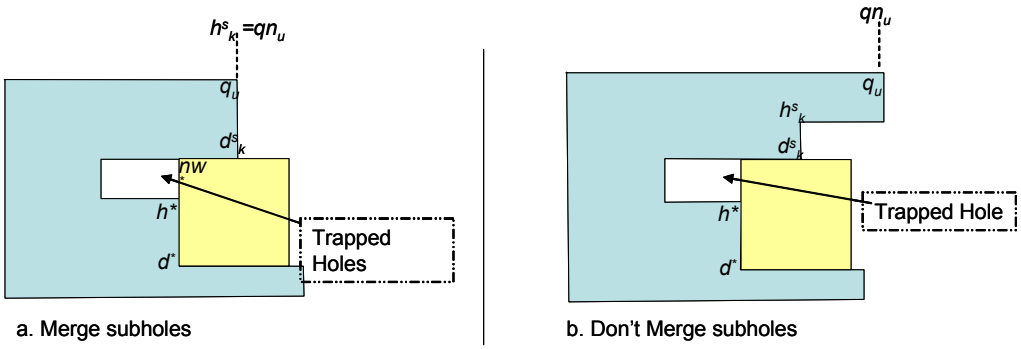


Figure 3-39 Incidental top contact

If the merge condition is not met, as in Figure 3-39b, adjustments are made to the current subhole so that the bottom trace begins at the first d_k from array T that makes contact with the top of the items. All FT_s (FB_s) points not right of nw^* are retained in the trapped hole FT_t (FB_t). Finally point nw^* is added to FT_t.

The merge procedure's trapped hole subprocedure is similar to the left side case. The merged *FB* replaces the last point in FB_u , qw_u (for subhole u>s), with q_u and appends all the remaining points from FB_s to FB_u. FB_u is resaved as FB_s. FT_u is the new top of the merge FB_s and is resaved as FT_s. q_s, qw_s, and qn_s are retained as the special links. Finally, subhole u is removed from the main hole's subhole ArrayList.

An item placement may affect subhole special links. For example, qw_t, *for* $t>s$, can change when the new FB_s has points above q_t ,i.e., when $x(b^s_i)>x(q_t) \cap y(b^s_i)>y(q_t)$. For this reason, when lower subholes are packed, qw_t may undergo a change that requires FB_t and FT_t to be modified as shown in Figure 3-40.

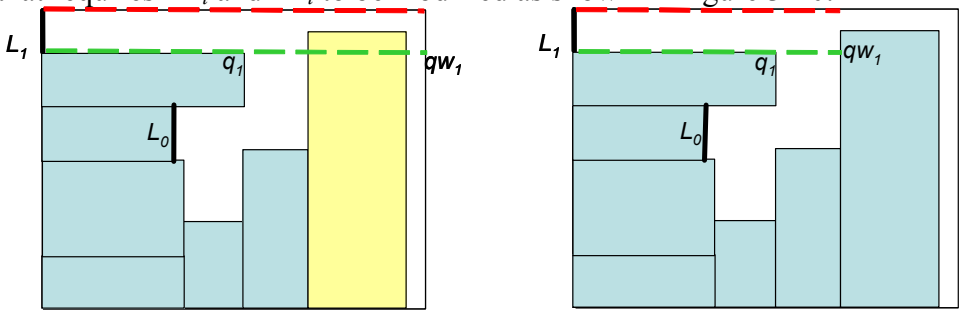


Figure 3-40 Updating qw_t, FB_t, and FT_t when lower subhole is packed

If an upper subhole u is packed and there is an overhang condition or top contact which changes FT_u, special links may need to change for one or more lower subholes s.

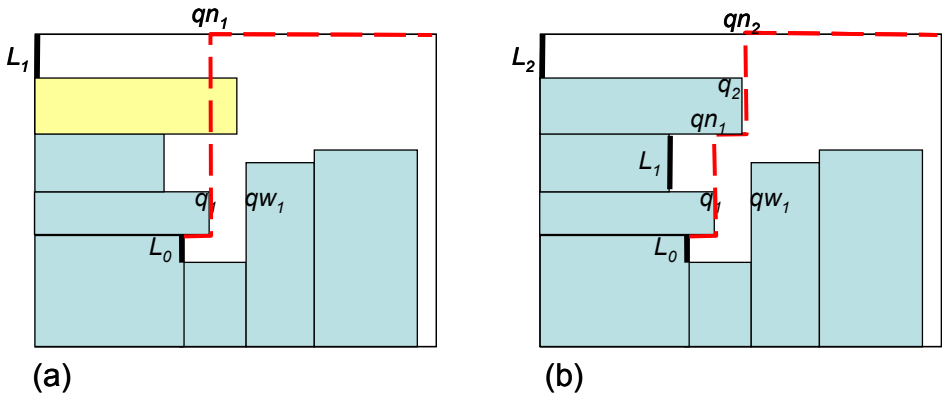


Figure 3-41 Updating q_n and FT when a lower subhole is packed.

An example of this general case is shown in Figure 3-41. Figure 3-41(a) shows the incorrect FT_0 and qn_1 that result after the left and right update actions for this special case. After the special update is performed, FT_0 and qn_1 are corrected as shown in Figure 3-41(b).

The final special considerations involve keeping the packed subhole's bottom and top traces free of redundant points, i.e., points not defining a corner or end point in either trace. The presence of such points can cause errors in the update procedure.

Redundant points may be created when (1) the item's top edge precisely matches $y(h^*)$, and (2) when the item's right edge precisely matches $x(r^*)$. These conditions are necessary and sufficient to have a pair of identical redundant points. Condition (2) also requires further investigation to determine if the item's top edge precisely matches $y(b_k)$, where b_k is the next point in the *FB* trace.

Condition (1) implies $y(h^*) = y(nw^*)$ which requires 2 identical redundant points be removed from the new *FB* trace as shown in Figure 3-42.

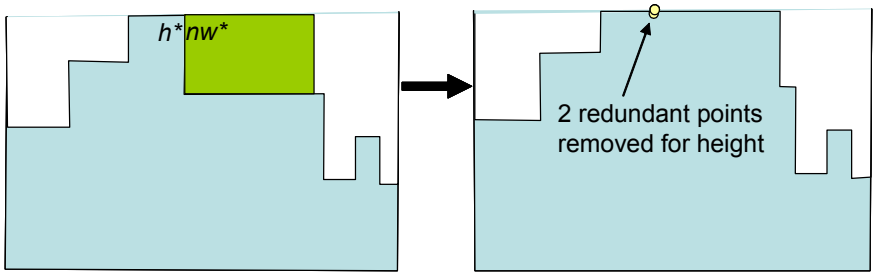


Figure 3-42 Remove of (2) redundant points

For condition (2), there are two identical points removed "for width" at the lower right corner of the item as shown in Figure 3-43a and b. If the $y(ne^*) = y(b_k)$, as in Figure 3-43b, the two additional redundant points are removed for height match.

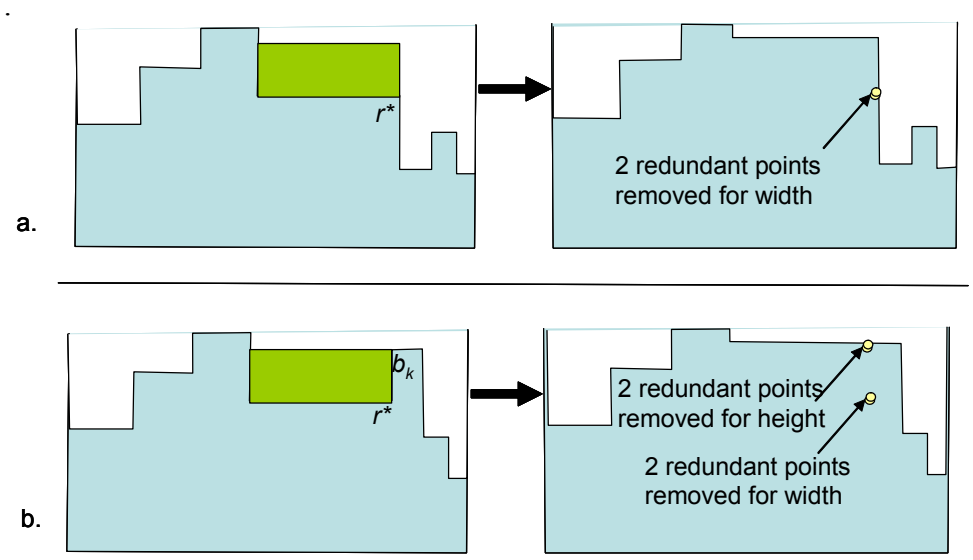


Figure 3-43 Redundant points due to right side edge contact.

There are two special cases for matching height in either the top left or top right corner of the subhole: (1) $h^* = t_0$, and (2) $ne^* = t_{|FT|-1}$.

Case 1 has two subcases:

1a - If $x(t_1) \le x(ne^*)$ as shown in Figure 3-44a, set $b_0 = t_1$ and remove t_0 and t_1 from *FT*.

1b - If $x(t_1) > x(ne^*)$ as shown in Figure 3-44b, and set $t_0 = ne^*$ and remove b_0 and b_1 from FB.

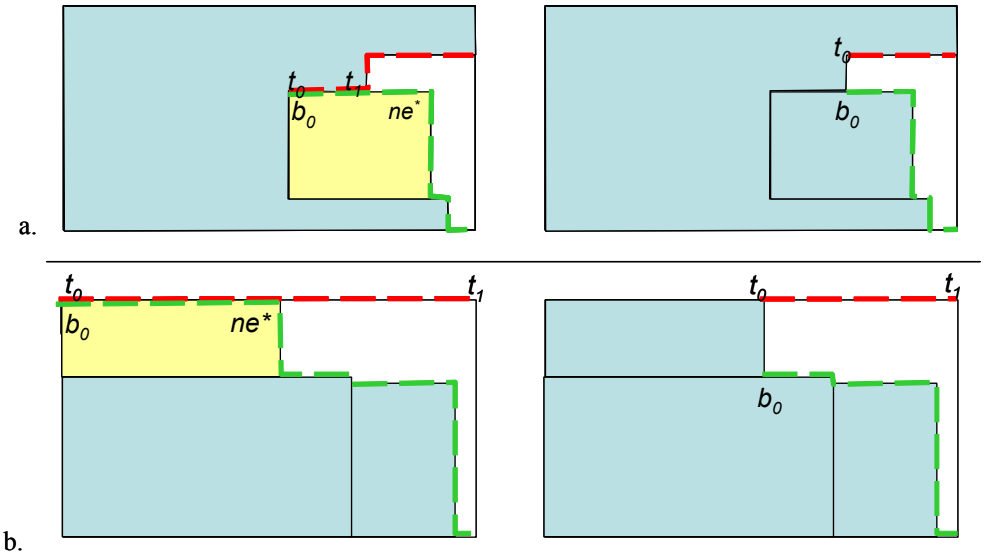


Figure 3-44 Top contact requiring trimming of *FT* and *FB*.

In case (2), the item reaches the top of the hole along the right side, i.e. $ne^* = t_{|FT|-1}$. This is the reverse of the example pictured in Figure 3-44b where the *last* point in FT, $t_{|FT|-1}$, is set to *nw** and the *last* two points in *FB* (corresponding to the *nw** and *ne*) are removed.

Another related special case can occur if the item contacts the subhole top trace between the ends of that trace. In this situation, the subhole is not split and a portion of the new subhole has zero height, i.e., the top and bottom traces coincide

as shown in Figure 3-45a. If a later nice fit is placed that touches the top trace at either t_0 or $t_{|FT|-1}$ and it also has equal height with a point on the bottom trace on the right or left respectively, the update makes appropriate adjustments to FT and F_s so that there is no unusable space on the end of the trace. This adjustment is portrayed in Figure 3-45b.

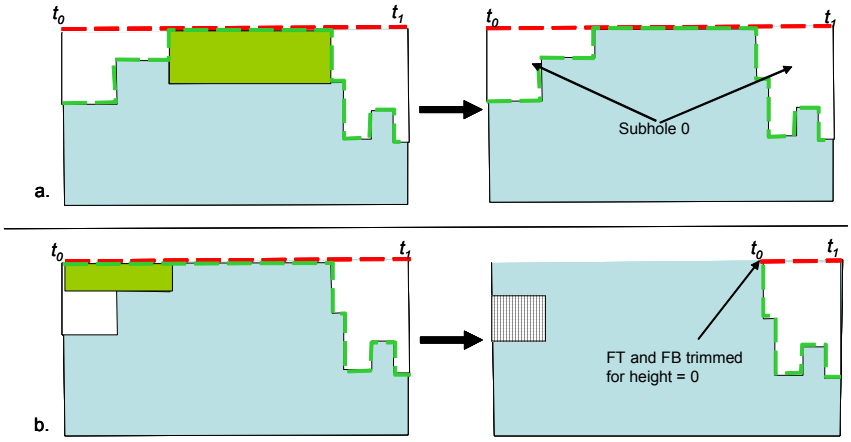


Figure 3-45 Trimming top and bottom trace of sections with no usable height

This subsection covered required special updates that must be performed after the left side or right side actions are completed. These final adjustments assure that the data structure is ready for the next item to be placed.

The next section details a new one-pass heuristic based on a perimeter contact factor (pcf), the pcf-BL heuristic, for creating an initial solution to the 2D|R|F bin packing problem.

3.4 THE PCF-BL HEURISTIC

Sections 3.2 and 3.3 described how to determine E^* and how to update the data structure once the item is placed at E^*. This section details a new one-pass heuristic, the pcf-BL heuristic, for creating an initial solution to the 2D|R|F bin packing problem.

Prior to this section, we have limited consideration only to a known item with fixed orientation being placed in a single bin. We now expand our considerations to include selecting one of the two possible orientations of a specific item and choosing which of the identical bins receives the item. Thus, the procedure repeats the actions of Section 3.2 and determines E^* for each allowed bin/orientation combination. It then selects the "best" E^*, and executes the update actions covered in Section 3.3. This subsection discusses the greedy measure used to select this best E^*.

While there is no limit to the number of bins available in a classical bin packing problem, a practical upper bound is the number of items. Each bin is indexed for later reference and a specific indexing scheme related to S_n is described in Chapter 4.

Before packing any bins, a lower bound on the number of bins, *LB*, and an *item-to-bin ratio*, *IBR*, are determined. Define

$$LB = \frac{\sum_i^n h_i w_i}{HW}$$

where n is the total number of items, h_i is the height and w_i is the width of item i. H and W are the common height and width of the set of identical bins. *LB* is used to determine when to open a new bin. The item to bin ratio, IBR = *n/LB*, is an

upper bound on the average number of items that can be packed in a bin and is used for bin selection.

We first focus on using the pcf-BL heuristic to generate an initial solution to the bin packing problem. Figure 3-20 introduced the bin characteristics "*Percent Fill*", "*max width*", and "*max height*." *Percent fill* is the percentage of the bin area occupied by packed items. *Max width* and *max height* are the maximal horizontal and vertical dimensions still available for packing. These features are used to exclude bins where the items will obviously not fit. Every *open* bin, i.e., any bin designated as available for packing, is screened to determine if its available space is consistent with the item to be packed. A *placement* is defined as the triplet, (item, orientation, bin). If no feasible placement exists in the current open bins, a new bin is opened.

The initial solution heuristic first orients the rectangular items so that the maximal dimension is set to the height, h_i. The heuristic then orders the items using a two level hierarchy. First the items are sorted in descending order by area, ($w_i h_i$), and then within identical areas by descending h_i. Initial solution placements are performed in this order with each item being considered in either of its two orientations.

For all placements considered, a perimeter contact function, *pcf,* is evaluated and the placement with the maximal pcf is implemented. (This maximal pcf *may* occur in a newly opened bin.) The pcf is motivated by Lodi et al.'s(1999-1) measure that considered the item's touching perimeter for each BL position in the entire bin. The measure implemented for this research similarly rewards item

contact with the current traces on all sides. However, we penalize the perimeter of all simple trapped holes associated with the current subhole and, for cases with $r^* < se^*$, we penalize either the next lower trapped subhole r perimeter or the "shadowed" perimeter beneath the newly created overhang (as illustrated in Figure 3-46a and b).

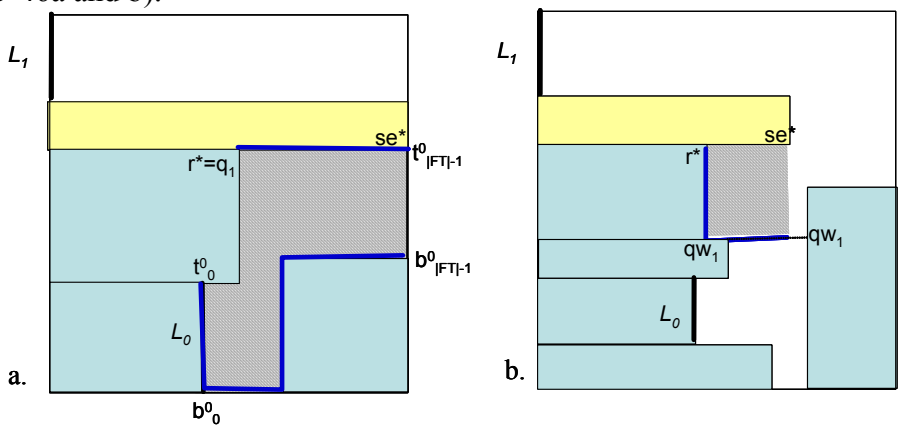


Figure 3-46 a. trapped subhole penalty b. "shadowed" perimeter penalty

Thus, we first obtain the six components of the *pcf* as shown in Figure 3-47. (left, top, lower left holes, bottom, right, and right side overhang/trapped subholes) and then compute their sum. The following narrative descriptions are provided for each component.

(1) Left side component. We differentiate between "low" and "high" IBR: IBR ≤30 and IBR > 30. (The dividing value of 30 was empirically obtained.)

If IBR≤30, the left side component is equal to the perimeter contact with the left support.

Steps for determining the *pcf*

1. **Left side comp.=edge contact between edge(*nw*sw**) and edge(*h*d**)**
   - **Is there high IBR? Check for adjustments to left side factor**
2. **Height comp.: Does *h**=*nw**?**
   - **height comp = w_i**
3. **Lower left hole comp.: Does *d** ? *l**?**
   - **lower left hole comp = sum all edge distances between d* and l* for current subhole ? (negative number)**
4. **Bottom comp. = edge contact between edge(*sw*se**) and edge(*l*r**)**
5. **Right side comp.: Does *se** contact FB?**
   - **If r* the top of a vertical FB edge? Right side comp = h_i**
   - **Else right side comp=edge contact between edge(*se*ne**) and trace FB**
6. **Right side overhang comp.: Is *r*<se**:**
   - **If there a trapped subhole- penalize trapped subhole perimeter (negative number)**
   - **Else– penalize trace perimeter from r* to the point FB leaves the 'shadow' of se* (this includes simple trapped holes right of r*) (negative number)**

Add components 1 through 6 to determine *pcf*

Figure 3-47 Steps for determining the perimeter contact function

For IBR> 30, this research showed that a trace with *nw* slightly* higher than *h** is desirable on lower layers and positions close to the right side of the bin. To enforce this preferential type of placement, the following procedures were employed: The full item height for this portion of perimeter is credited if (a) the item rests on the bin bottom and also has at least 75% contact between *h** and *d** or (b) if the item rests on another item and (i) has at least 95% contact between *h** and *d** and (ii) E* lies within .4*H* of the bin bottom and within .6*W* from the right side of the bin. In Figure 3-48, where IBR > 30, items 10 through 15 receive full credit under condition (a) and items 16 and 17 receive full credit under condition (b). Items not satisfying either condition are credited with only their actual left edge perimeter contact.

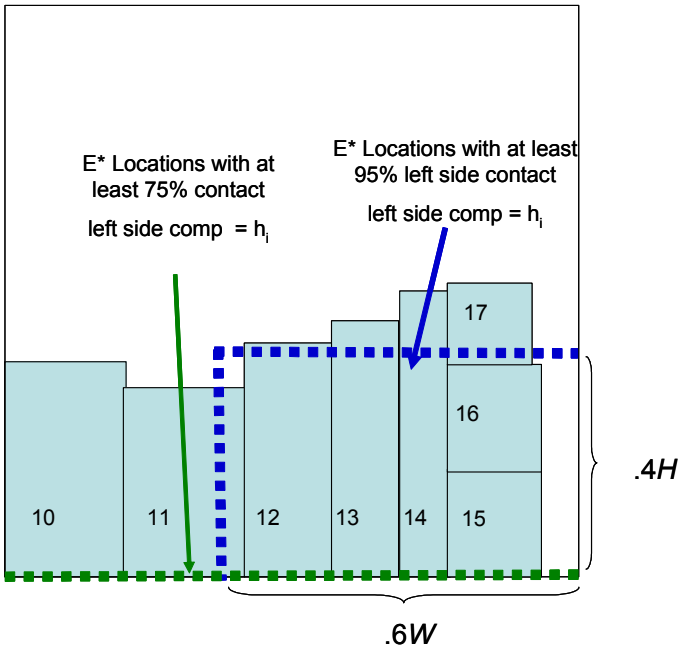


Figure 3-48 High IBR E* locations with adjusted criteria for the left side factor

Figure 3-49 shows how the items in Figure 3-48 are packed under the low IBR left side factor. The first difference occurs when item 12 is packed in its alternate orientation.

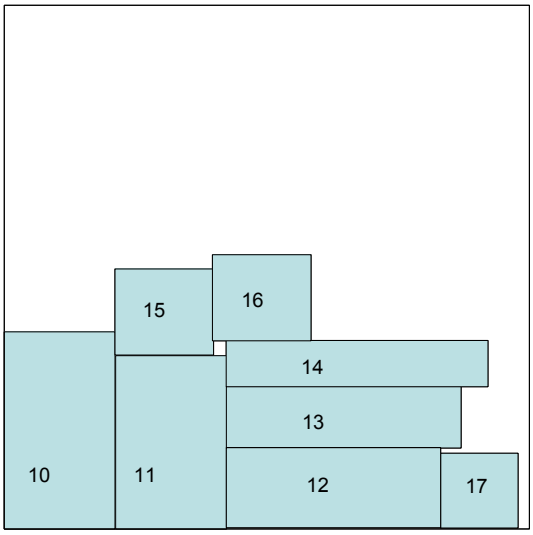


Figure 3-49 Resulting packing solution with of set in Figure 3-48 low IBR left side perimeter rules.

(2) top component. The top component is credited with the item width whenever $y(h^*) \equiv y(nw^*)$, *i.e.,* in this special situation, the new item provides a wider base for subsequent item packing.

(3) left side hole component. A penalty for the trace perimeter of a left trapped hole within the current subhole is assessed if $d^* \neq l^*$.

(4) bottom component. The bottom component credits item perimeter contact with the bottom support.

(5) right side component. The right side component rewards for right item perimeter contact. As pictured in Figure 3-50, an alternate reward of h_i is given if $r^* = se^*$ and r^* is the top of a vertical edge in the bottom trace because this provides a smoother left wall for subsequent packing.

(6) right side overhang component. This was discussed earlier in association with Figure 3-46.

The six components are combined (rewards minus penalties) to determine the *pcf* for the placement.

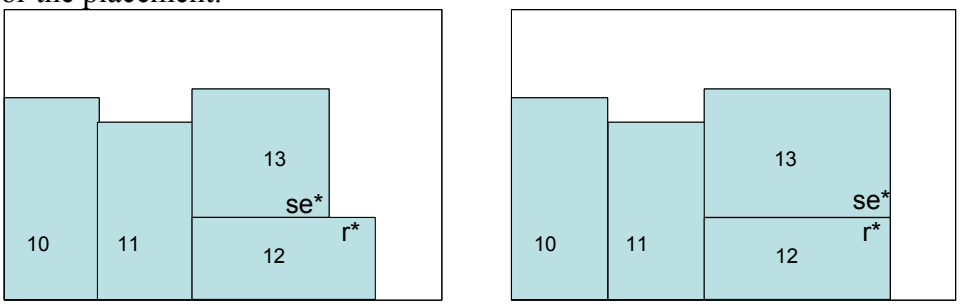


Figure 3-50 Alternate right side component

Given the best placement among the current open bins, we must decide whether to implement that placement or open a new bin. The logic employed in this research is similar to that presented by Lodi et al. (1999-1) for their TP heuristic. There are four situations to consider:

(1) If pcf $\geq h_i + w_i$ or *Open Bins* $\geq LB$, the placement is implemented.

(2) If *Open Bins* $< LB$ and pcf $< h_i + w_i$, a new bin is opened and the item is packed in the BL corner of that bin. (For the placement of an item in a new bin, pcf $\geq h_i + w_i$.)

This section describes the pcf-BL heuristic for an initial solution for multiple bins. Section 3.5 presents the results from this heuristic when applied to two standard sets of benchmark problems.

3.5 TEST PROBLEM RESULTS FOR MULTIPLE BIN PACKING

The results for the pcf-BL Heuristic described in this section comprise ten different bin packing problem classes. The first six classes are from the Berkey

and Wang (1987) test set. The last four are those proposed by Martello and Vigo (1998).

Each class considers five values of n: 20, 40 ,60, 80, and 100. Each n has 10 instances for a total of 500 problems.

3.5.1 The Berkey and Wang Test Set

The six classes of the Berkey and Wang test set are decribed below:

Class I: w_j and h_j uniformly random in [1,10], W=H=10;

Class II: w_j and h_j uniformly random in [1,10], W=H=30;

Class III: w_j and h_j uniformly random in [1,35], W=H=40;

Class IV: w_j and h_j uniformly random in [1,35], W=H=100;

Class V: w_j and h_j uniformly random in [1,100], W=H=100;

Class VI: w_j and h_j uniformly random in [1,100], W=H=300;

Table 3-1 shows the Berkey and Wang results for three one-pass heuristics. The results in Table 3-1 are given in terms of an *average* ratio across the 10 problem instances in each subclass where the numerator is the number of bins used by the one-pass heuristic and the denominator is the improved lower bound on the required number of bins from Dell'Amico, Martello, and Vigo (2002). The first column is for the "Floor Ceiling" (FC) approach (Lodi et al. 1999-1). The second column is for "Touching Perimeter" (TP) method (Lodi et al. 1999-1) and the third column is for the pcf-BL heuristic. The final column presents the computational time required by the pcf-BL heuristic.

BERKEY AND WANG TEST SET (1987) (2BP|R|F)					
		Lodi, Martello, and Vigo (1999)			
Class	*n*	FC LB	TP LB	pcf-BL LB	average time
I	20	1.06	1.05	1.03	0.13
	40	1.08	1.06	1.06	0.20
	60	1.09	1.05	1.06	0.30
	80	1.09	1.06	1.07	0.41
	100	1.07	1.03	1.04	0.47
	Average	**1.078**	**1.050**	**1.052**	**0.302**
II	20	1.00	1.00	1.00	0.13
	40	1.10	1.10	1.10	0.21
	60	1.05	1.00	1.05	0.28
	80	1.03	1.07	1.07	0.36
	100	1.03	1.00	1.03	0.44
	Average	**1.042**	**1.034**	**1.050**	**0.284**
III	20	1.18	1.06	1.06	0.17
	40	1.16	1.11	1.11	0.28
	60	1.19	1.11	1.10	0.40
	80	1.15	1.10	1.08	0.49
	100	1.13	1.08	1.08	0.67
	Average	**1.162**	**1.092**	**1.086**	**0.402**
IV	20	1.00	1.00	1.00	0.14
	40	1.00	1.00	1.00	0.25
	60	1.10	1.10	1.10	0.31
	80	1.10	1.07	1.10	0.43
	100	1.07	1.03	1.07	0.56
	Average	**1.054**	**1.040**	**1.054**	**0.338**
V	20	1.08	1.06	1.06	0.18
	40	1.10	1.11	1.08	0.27
	60	1.11	1.08	1.11	0.42
	80	1.11	1.08	1.08	0.53
	100	1.10	1.08	1.07	1.10
	Average	**1.100**	**1.082**	**1.080**	**0.500**
VI	20	1.00	1.00	1.00	0.17
	40	1.40	1.40	1.40	0.27
	60	1.05	1.05	1.05	0.32
	80	1.00	1.00	1.00	0.42
	100	1.07	1.07	1.10	0.48
		1.104	**1.104**	**1.110**	**0.332**
Set Average		1.090	1.067	**1.072**	**0.360**

Table 3-1 Berkey and Wang Test Set Heuristic Results

3.5.2 The Martello and Vigo Test Set

Lodi et al. (1999-1) constructed the following classes where they state that "a more realistic situation is considered." They conjecture this based on the fact that not all items are generated on the same interval. They present four types of items which are combined in varying proportions as described below.

Type 1: w_j uniformly random in [2/3W, W] and h_j uniformly random in [1,1/2H]

Type 2: w_j uniformly random in [1, 1/2W] and h_j uniformly random in [2/3H,H]

Type 3: w_j uniformly random in [1/2W, W] and h_j uniformly random in [1/2H,H]

Type 4 w_j uniformly random in [1, 1/2W] and h_j uniformly random in [1,1/2H]

Class VII: *Type 1* with probability 70%, *Type* 2, 3, 4 with probability 10% each.

Class VIII: *Type 2* with probability 70%, *Type* 1, 3, 4 with probability 10% each.

Class IX: *Type 3* with probability 70%, *Type* 1, 2, 4 with probability 10% each.

Class X: *Type 4* with probability 70%, *Type* 1, 2, 3 with probability 10% each.

The results for this data set are shown in Table 3-2 (formatted identically to Table 3-1). pcf-BL performed comparably to the TP Heuristic for this test set averaging 1.083 for its ratio to the lower bound and TP averaging 1.086.

MARTELLO AND VIGO TEST SET(1998) (2BP|R|F)					
		Lodi, Martello, and Vigo (1999)			
Class	*n*	FC LB	TP LB	pcf-BL LB	average time
VII	20	1.19	1.13	1.15	0.16
	40	1.17	1.10	1.12	0.26
	60	1.18	1.12	1.11	0.39
	80	1.17	1.11	1.11	0.54
	100	1.17	1.11	1.10	0.72
	Average	**1.176**	**1.114**	**1.118**	**0.414**
VIII	20	1.16	1.16	1.12	0.14
	40	1.19	1.16	1.12	0.26
	60	1.18	1.11	1.10	0.43
	80	1.16	1.11	1.10	0.56
	100	1.17	1.12	1.09	0.64
	Average	**1.172**	**1.132**	**1.106**	**0.406**
IX	20	1.00	1.01	1.01	0.16
	40	1.01	1.02	1.02	0.28
	60	1.01	1.01	1.01	0.53
	80	1.01	1.01	1.01	0.68
	100	1.01	1.01	1.01	1.91
	Average	**1.008**	**1.012**	**1.012**	**0.712**
X	20	1.15	1.20	1.15	0.16
	40	1.09	1.08	1.09	0.29
	60	1.09	1.09	1.10	0.40
	80	1.06	1.06	1.06	0.57
	100	1.07	1.06	1.07	0.70
	Average	**1.092**	**1.098**	**1.094**	**0.426**
Set Average		1.112	1.089	1.083	0.490
Grand Average		1.099	1.076	1.076	0.412

Table 3-2 Martello and Vigo Test Set (1998) Heuristic Results

3.5.3 Summary of Results

Overall Tables 3-1 and 3-2 show that the pcf-BL yields comparable results to the TP heuristic. Since, the *only* results available were the consolidated average ratios as presented in Lodi et al. (1999-1), computational times for the FC and TP methods are not provided.

Originally, it was not the goal of this research to develop a new one pass heuristic for the BP problem. We would have greatly preferred using a previously

created heuristic both to generate an initial solution from which to begin the ATS and to assist in implementing the ATS moves. However, such algorithms cited in recent literature (Liu and Teng 1999; Lodi et al. 1999-1) either no longer exist or were not publicly available (Lodi 2003; Liu 2003). Thus the construction of this heuristic was required before the ATS development could proceed.

Fortunately, the pcf-BL heuristic developed in this research provides starting solutions and intermediate solutions that are structurally superior for the ATS technique and are consistent with the S_n representation used. (It is interesting to note that Lodi et al. (1999-1) did not use their TP one pass algorithm to provide a starting solution for their tabu search methodology.)

This chapter covered all actions needed to implement the pcf-BL Heuristic. This is the first such implementation made public. As demonstrated, Chazelle's point of view that updating the data structure "involves only straightforward graph manipulation" was a gross understatement. However, this $O(n^2)$ implementation enhances the ATS where it becomes the engine for computing the move value of repacked bins. The next chapter covers the ATS implementation.

Chapter 4

An Adaptive Tabu Search for Bin Packing (ATS-BP)

This chapter presents an ATS for solving the 2D orthogonal packing problem viewed as a partitioning and ordering (P|O) problem. This chapter has six sections. The first section presents an objective function that allows for differentiation of intermediate moves which may not change the total number of bins being used. The second section describes how an order based heuristic, such as the pcf-BL, allows an S_n representation of the bin packing problem. The third section presents essential move concepts and neighborhoods implemented in this research. The fourth section explains a dynamic move strategy that integrates the different move sets into an ATS. The fifth section discusses data structure efficiencies that improved the speed of the search. Finally, the sixth section presents the results for the two benchmark test sets previously discussed in Chapter 3.

4.1 A FINE GRAINED OBJECTIVE FUNCTION FOR THE BP PROBLEM

In BP problems, the primary goal is to minimize the number of bins while ensuring that every item is packed. If a search method uses only the 'number of bins' as a relative measure for a set of moves, the great number of equal valued moves would thwart the search process. This section presents a finer-grained objective function that allows effective relative evaluation of competing moves in support of the primary goal.

Moves that increase the likelihood of reducing the number of bins should be favored. There are 3 kinds of moves that fit this definition.

(1) excess bin moves. An *excess bin* is a designated candidate for achieving this primary goal by moving its items to other bins. Moves that reduce the total space occupied by items in an excess bin should have high relative value. When such moves reduce the same amount of space, the move that results in the greater number of items in the excess bin is preferred.

(2) intra-bin moves. If two moves yield different packing arrangements of the same set of items in a bin, the one with a more compact arrangement should have higher value. A "compact arrangement" will be fully discussed in Section 4.1.2

(3) inter-bin moves. Given two bins, a move yielding greater aggregation of the dead space should have higher value because the chances of packing a large item from the excess bin (or other bins) increases.

The first step in developing the finer-grained objective function is to have a framework that will allow the integration of the effects of excess bin, intra-bin, and inter-bin moves into a single objective function. This is achieved by using a *potential energy* based approach. Li and Milenkovic (1995) used a potential energy based objective function for 2D strip packing of polygons with their Position Based Physics (PBP) technique. This research uses a similar objective function with specific factors that account for the desired features of the moves described above.

In Newtonian physics, an item's potential energy (*PE*) is *mgd* where *m* is the item mass, *d* is the distance of the item's center of gravity (cg_i) relative to a specified reference plane and *g* is the gravitational constant. In a 2D domain, the reference planes are the *x* and *y* axes of the imputed Cartesian system. All items possess equal, homogeneous density implying that item *i*'s mass is $m_i = h_i w_i$ and its $cg_i = [cg_{ix}, cg_{iy}]$ is at its geometric center. In the current context, following Li and Milenkovic (1995), we define *PE* relative to both the *x* and *y* axes. Let μ and ρ be the vertical and horizontal gravitational constants, respectively. As pictured in Figure 4-1, $PE_x = m_i \rho d_x$ and $PE_y = m_i \mu d_y$.

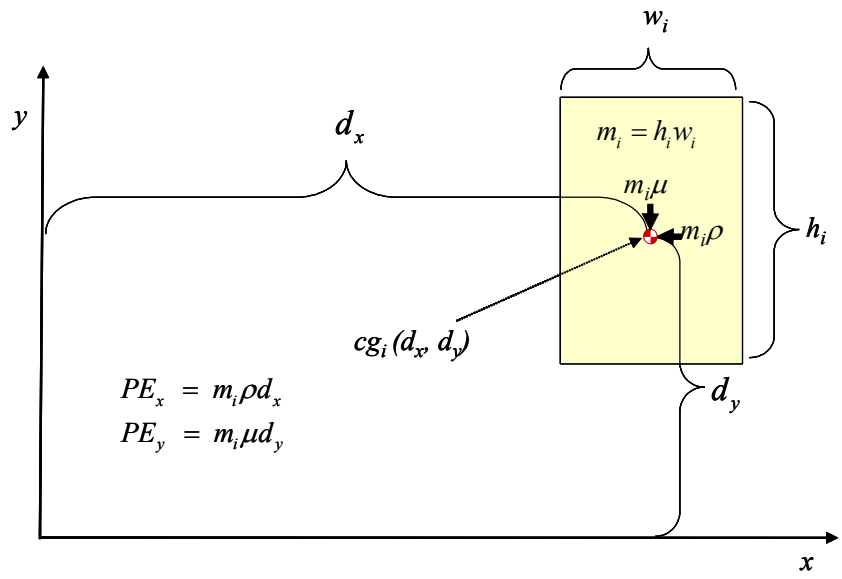


Figure 4-1 Determining the PE of item i

As shown in Figure 4-2, $CG_k = [CG_{kx}, CG_{ky}]$ is the *composite* CG of bin *k* and the items packed within bin *k*. CG_k is computed as

$$CG_{kx} = \frac{HW(\frac{1}{2}W) + \sum_i m_i d_{ix}}{HW + \sum_i m_i} \text{ and } CG_{ky} = \frac{HW(\frac{1}{2}H) + \sum_i m_i d_{iy}}{HW + \sum_i m_i}$$

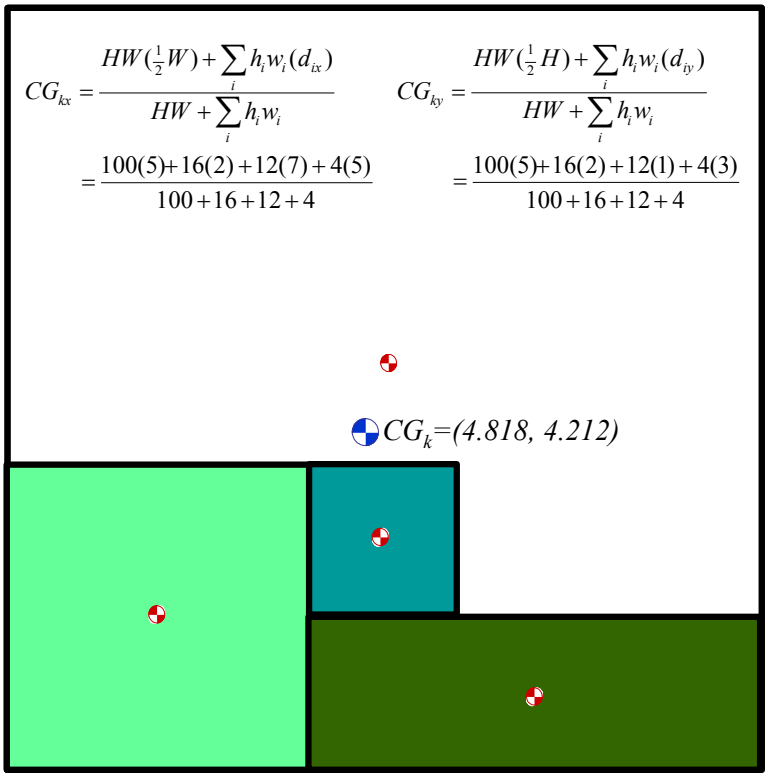


Figure 4-2 Determining CG_k

The PE_k of a bin k is defined analogously with the exception that *only* open bins have $PE_k > 0$.

As discussed below, the fine-grain objective function of "minimizing the total *PE* of all bins" assures that the effects of the excess bin, intra-bin, and inter-bin moves will drive the total packing configuration toward minimizing the number of packed bins. The next three subsections discuss the interaction of

objective function components associated with these moves and the final subsection describes how the pcf-BL incrementally determines the new CG_k for all bins so that the fine-grain objective function is efficiently evaluated.

4.1.1 Evaluation of Excess Bin Potential Energy

This subsection discusses how an excess bin's *PE* is assessed. This research penalizes excess bins by translating their bottom left corner to (2.5W, 2.5H) before determining its *PE* where the multipliers of value 2.5 are always sufficient to cause the excess bin's *PE* to be large relative to all other bins.

Since this implementation makes no moves to improve the packing arrangement in an excess bin, each item in an excess bin is assigned a cg_i equal to the geometric center of the excess bin as indicated in Figure 4-3.

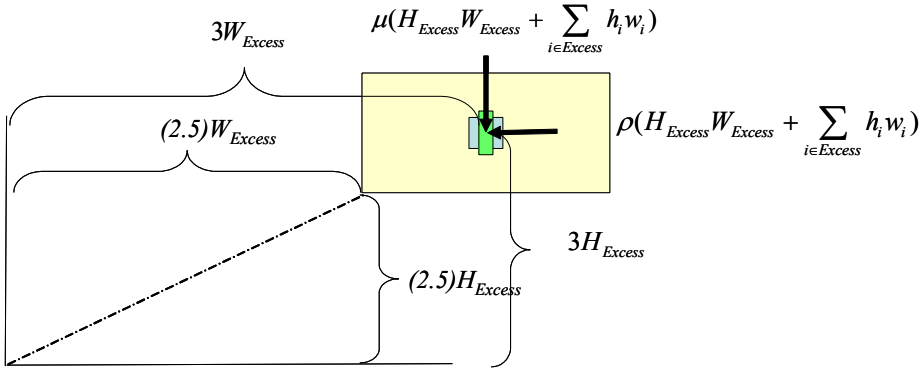


Figure 4-3 Excess bin *PE* evaluation

A discriminant is included in the objective function term for the excess bin to differentiate between moves that yield the same total space occupied in the excess bin. When this occurs, the move that leaves more items in the excess bin is preferred because smaller items are easier to place in currently open bins. This

discriminant is defined as $-\frac{|S_i| h_{\min} w_{\min}}{2HW}$, where h_{min} and w_{min} are the dimensions of the smallest item. (Lodi et al. (1999-1) used a similar term in their "filling function.") The final equations that define the *x*- and *y*- values of the excess bin's *PE* are

$$PE_{Excess_x} = [3\rho W_{Excess}](H_{Excess}W_{Excess} + \sum_{i \in Excess} h_i w_i - \frac{|S_i| h_{\min} w_{\min}}{2HW})$$

$$PE_{Excess_y} = [3\mu H_{Excess}](H_{Excess}W_{Excess} + \sum_{i \in Excess} h_i w_i - \frac{|S_i| h_{\min} w_{\min}}{2HW})$$

4.1.2 Setting the gravitational constants

In the context of this research, arrangement A is more "compact" than arrangement B if $PE_A < PE_B$. The subsection discusses how μ and ρ are used to weight the PE of competing arrangements to encourage layouts that are favorable for future placements associated with intra-bin rearrangements.

It would be natural to assume that μ should equal ρ. However, setting μ marginally greater than ρ can better consolidate dead space when packing in nearly full bins. This concept is illustrated in Figure 4-4.

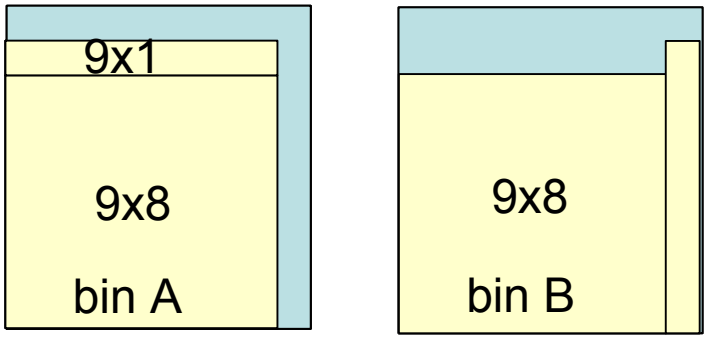


Figure 4-4 Adjusting μ and ρ.

If $\mu = \rho$, then bin A would be preferred over bin B ($PE_A < PE_B$). However, bin B is superior, i.e., any item packable in bin A is packable in bin B but the *converse* is false. Consider the following mathematical relationship for Figure 4-4. The *PE* relationship required to favor bin B over bin A is:

$$\rho(4.5(72)+4.5(9))+\mu(4(72)+8.5(9) > \rho(4.5(72)+9.5(9))+\mu(4(72)+4.5(9)$$

which implies $\rho/\mu > 0.889$. Bins A and B are 81% full; fuller bins would require a higher ratio. This research uses μ=1.0 and ρ=0.9.

4.1.3 Bin Dead Space Penalty

This section details how dead space consolidation caused by inter-bin moves is rewarded.

The *PE* is indicative of how compactly the items are placed. However, in some cases, this can encourage inferior configurations. With two or more partially filled bins, the *PE* measure tends to equally distribute the items. For example, in Figure 4-5, the *PE* objective function would favor moving an item from bin A to bin B (as shown in Figure 4-5b). The following translation was developed to overcome this tendency.

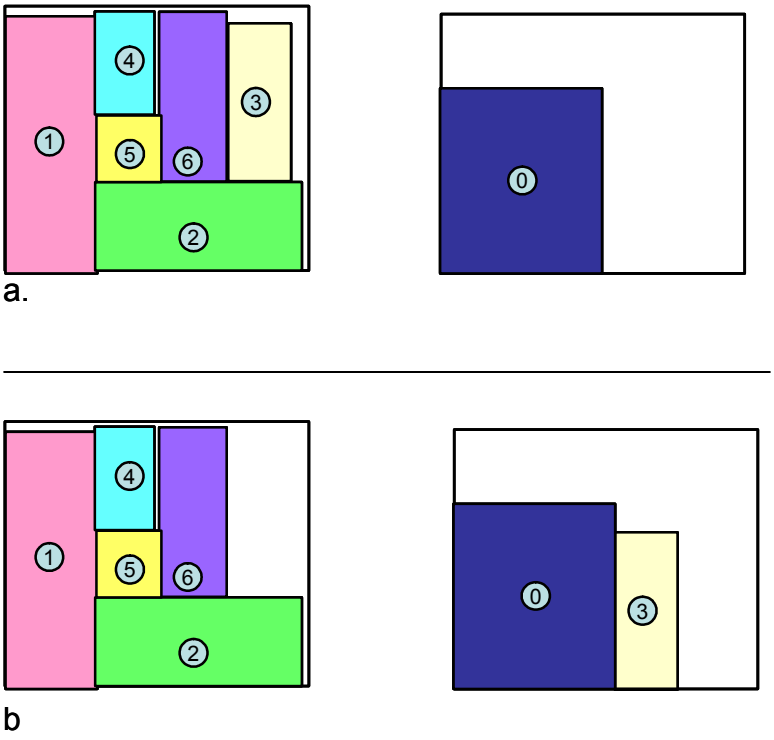


Figure 4-5. *PE* cross-leveling dilemma

The 'dead space' in a bin is any area that is not filled by an item. Translating the lower left corner of a bin, X_k and Y_k, by a distance proportional to the square root of the dead space, in both the *x*- and *y*- directions, corrects this shortcoming. The additional scaling factor of 1.5 favorably weights inter-bin moves relative to intra-bin moves and ensures that the top right corner of a non-excess bin is always at lower *PE* than the bottom left corner of an excess bin. X_j and Y_j are defined as:

$$X_j = 1.5W_j\sqrt{1-\frac{\sum_{i\in j} h_i w_i}{H_j W_j}} \qquad Y_j = 1.5H_j\sqrt{1-\frac{\sum_{i\in j} h_i w_i}{H_j W_j}}$$

Figure 4-6 pictures the relative dead space translation distances of non-excess bins compared to that of excess bins. Full bins' lower left corners are at the origin. *Near* empty bins' lower left corners would be near (1.5*W*, 1.5*H*).

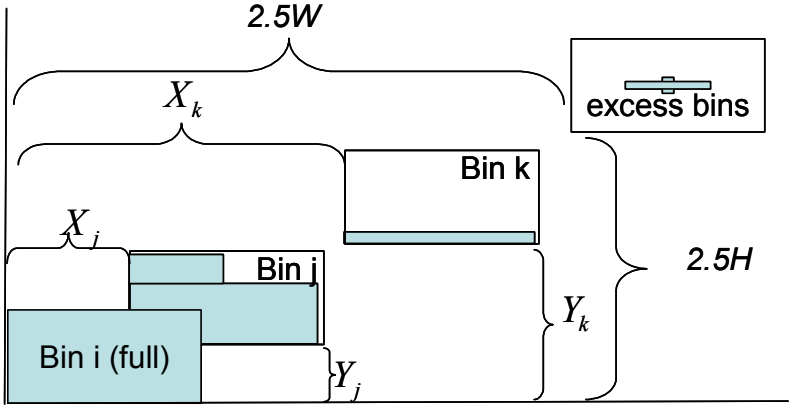


Figure 4-6 Translation of lower left corner based on dead space

After determining the new translation penalties, the new PE_k is calculated as

$$PE_k = [\rho\left(CG_{kx} + X_j\right) + \mu\left(CG_{ky} + Y_j\right)](1 + \frac{\sum_i h_i w_i}{HW})(HW)$$

Moves that reduce the contents of an excess bin also reduce the dead space penalty of the destination bin. The total bin *PE*s yield the fine-grained objective function.

Although this objective is tailored to the 2D bin packing problem, the use of *PE* has potential for all types of packing problems including arbitrary shaped items in both 2D and 3D and both bin packing and geometric knapsack variants.

This subsection covered the dead space penalty component of the objective function. A partially full bin will have either an excess bin factor or a dead space factor, but not both.

The next section discusses the ATS-BP solution representation.

4.2 SYMMETRIC GROUP REPRESENTATION FOR BIN PACKING

In this section, we describe an order-based solution representation for the BP problem. S_n solution representations have been used by several earlier researchers. Wiley (2001) allocated letters for each tanker-trip combination for the Aerial Fleet Refueling Problem and Crino (2002) allocated numbers to each vehicle-trip in the Theater Distribution Vehicle Routing and Scheduling Problem (Crino 2002). Both of these implementations represented the 'carrier' as the first letter.

One method of using S_n to represent BP solutions is to allocate a unique letter to represent each bin. A unique bin letter is fixed as the first letter in its permutation cycle and is accompanied by the unique letters associated with the items in the bin. This implementation used in the research reserves the numbers 0 to n-1 for the maximal number of bins. Each of the n items has two associated unique letters. One letter represents the orientation where the item's longest axis is in the horizontal plane ("orientation 0"). The other orientation has the longest axis in the vertical plane ("orientation 1"). The letters n to $2n$-1 represent the orientation 0 for the n items in the original list order. Letters $2n$ through $3n$-1 represent orientation 1 for the items. Thus, given i, the item's original order

number, and *o*, the orientation, then the item's letter, *i*,* for orientation *o*, would be:

$i^* = i + n(o+1)$

This S_n representation has a large set of permutations that have no defined solution and are subsequently forbidden. One and only one orientation for each item must be in a cycle of two or more elements. A letter leaves (enters) a one-cycle by a transposition (swap) with a letter of equal modulo(n).

In symmetric group notation, one-cycles are implicit. Thus, the bins not being used and the orientations not being used would not be shown. For example, suppose there are 20 items, numbered 20 through 39, with items 20 through 30 packed in order in bin 0 in orientation "0" excepting item 25 and items 31-39 packed in order in bin 1 in the orientation "0" excepting item 32. The corresponding S_n representation would be:

(0, 20, 21, 22, 23, 24, *45*, 26, 27, 28, 29, 30) (1, 31, *52*, 33, 34, 35, 36, 37, 38, 39)

Additional examples of how the S_n representation changes for various moves are described in the next section.

4.3 ATS-BP MOVES

The four subsections of this section discuss the moves contained in the ATS-BP. The first subsection describes ejection chains, a fundamental concept for all move types. The second, third and fourth subsection detail the excess bin, intra-bin, and inter-bin moves. For all moves, define a *packing position* to be a BL placement in a non-excess open bin.

4.3.1 Ejection Chains

Let us consider a *single* bin and the effects of a single move on that bin. All items that are not *directly relocated* as part of the move definition retain their current orientation and relative order regardless of any required absolute change in the packing sequence. However, a direct result of a move may be that all items no longer fit in the bin. The pure pcf-BL manages any bin overfill by opening a new bin. The ATS-BP utilizes an excess bin for this purpose. This is achieved through a slight modification of the pcf-BL.

Laguna and Glover (1997) state "An *ejection chain* is an imbedded neighborhood construction that compounds the neighborhoods of simple moves to create more complex and powerful moves." An ejection chain begins with selecting certain elements of the current solution to change. This directed change results in at least one other element being forced to change or be "ejected" from its current state. Ejection chains have been used in tabu search techniques for several classes of problems with significant success. Approaches include Barnes and Chambers (1995) for the Job Shop Scheduling Problem, Dorndorf and Pesch (1994) for the Clustering Problem, Laguna et al. (1995) for the Multilevel Generalized Assignment Problem. Rego (1998) for Vehicle Routing Problems, Glover (1992) and Glover and Pesch (1997) for the Traveling Salesman Problem, and Gonzalez-Velarde and Laguna (2002) for the Coloring of Graphs. This research is the first to incorporate this technique to solve packing problems.

Whenever the ATS-BP moves an item for an insert move or a pair of items in a swap move, it *directly relocates* the selected items. However, the relocation

may cause other items to no longer fit in the bin. In this situation, the ATS-BP implements an ejection chain through the pcf-BL that feasibly places non-fitting items into an excess bin.

In terms of the S_n representation, consider the following three cycles from the 20 item example presented in Section 4.2 (where bin 2 is the excess bin):

(0, 20, 21, 22, 23, 24, **45**, 26, 27, 28, 29, 30) (1, 31, **52**, 33, 34, 35, 36, 37) (2, 38, 39)

If the ATS-BP swaps item 45 and 52 without ejection chains, the new solution would be:

(0, 20, 21, 22, 23, 24, **52**, 26, 27, 28, 29, 30) (1, 31, **45**, 33, 34, 35, 36, 37) (2, 38, 39)

Since item 37 no longer fits in bin 1, it is ejected to the "excess" bin yielding:

(0, 20, 21, 22, 23, 24, **52**, 26, 27, 28, 29, 30) (1, 31, **45**, 33, 34, 35, 36) (2, 37, 38, 39)

The only items that permitted to eject are those that follow 45 and 52 in the cycle. i.e. if 52 or 45 eject the move is infeasible.

4.3.2 Excess Bin Moves

In general, larger items are harder to place. At each iteration, ATS-BP identifies the excess bin's *Big-Item*, i.e., the item with maximal $h_i w_i$. Excess bin moves seek to place the Big-Item and other relatively large items into packing positions by checking both orientations. No items smaller than a stated threshold (relative to the Big-Item) are moved. The threshold function will be detailed in Subsection 4.4.2.

4.3.2.1 Excess-Bin-Insert

An excess bin insert move extracts a single item from the excess bin and places it in a packing position. This move is the primary move for emptying an excess bin. At each iteration, the ATS-BP attempts to insert large items into all possible packing positions.

Figure 4-6 shows an example of an excess bin insert, with ejections, where item 10 is inserted after item 12 in bin 0. Since item 10 fits in its current orientation, the move is feasible. However as a result, items 15 and 16 no longer fit in bin 0, and are ejected into excess bin 1.

Figure 4-6 is the first of several figures that picture solution configurations. In such figures, for clarity of presentation, the item labels in the bin "pictures" will correspond uniformly to the orientation "0" letters regardless of the actual orientation presented. The symmetric group representation will identify each item by its correct S_n letter.

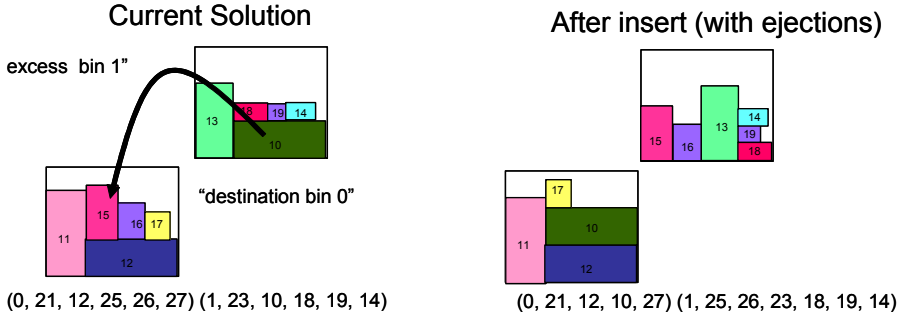


Figure 4-6 Excess-bin-insert with ejections

4.3.2.1 Excess-Bin-Swap

Excess-bin-swap moves swap *only* the Big-Item with candidate items in packing positions and may cause ejections. (Both Big-Item orientations are considered.) Figure 4-7 shows an excess-bin-swap where item 15 is swapped with item 10 and item 16 is ejected.

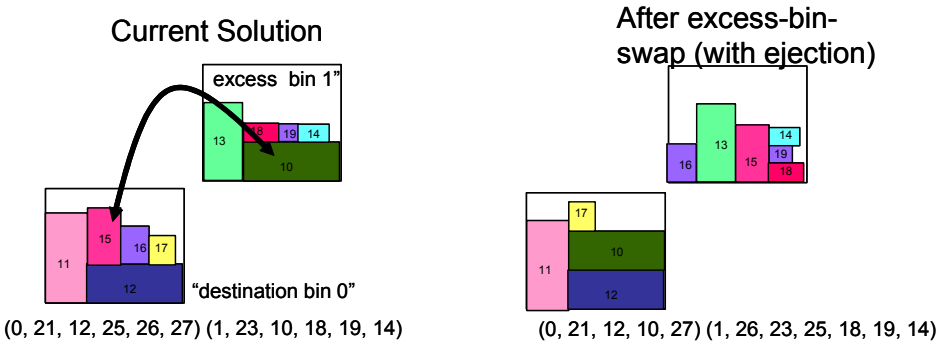


Figure 4-7 Excess-bin-swap with one ejection.

Candidates to be swapped with the Big-Item are screened based on lower and upper values proportional to the Big-Item. These values are detailed in Subsection 4.4.2 Big-Items are not allowed to swap with items of identical size. These screening methods reduce the neighborhood size without sacrificing effectiveness. Excess-bin-swap moves consider only the Big-Item to enhance its likelihood of placement.

4.3.3 Intra-bin-insert Moves

Intra-bin insert moves take a single item from one packing position to another in the *same* bin. Since intra-bin insert moves make small or null changes

in the fine-grained objective function, they are prone to cycling. ATS-BP prohibits consecutive intra-bin-insert moves.

Because the pcf-BL heuristic implements a placement in the first BL stable packing position encountered, moves that change the order of items to be placed can result in a new placement in a highly desired packing position. A special case of the intra-bin-insert move, the *trapped-hole-delay* move, attempts to make such packing positions available.

A trapped-hole-delay move directly relocates an item, that "seals" a trapped hole, to a later position in the packing order. Trapped-hole-delay moves are less prone to cycling and are allowed to be performed up to 2 consecutive times. Only bins with more than 5 items are considered for this move.

Figure 4-8 illustrates how the *trapped-hole-delay* move works. Consider the configuration pictured in Figure 4-8a where the trapped hole of interest is indicated by the orange cross-hash. Item 51, the 27^{th} item in the packing order, is the associated "hole sealer" that needs to be moved later in the packing order. Big-Item 99 is currently in the excess bin (pictured separately here for simplicity). A superior new configuration would place item 99 into the trapped hole sealed by item 51. Unfortunately, the dense configuration of Figure 4-8a causes all excess-bin-insert moves to be unfavorable. For example, the configuration pictured in Figure 4-8b is the result of inserting item 99 into packing position 27 thus shifting item 51 to position 28. In this case, the pcf-BL "first BL stable packing rule" causes item 99 to be placed left adjacent to item 78.

This leads directly to the consideration of the trapped-hole-delay moves which begin checking all items that seal a hole.

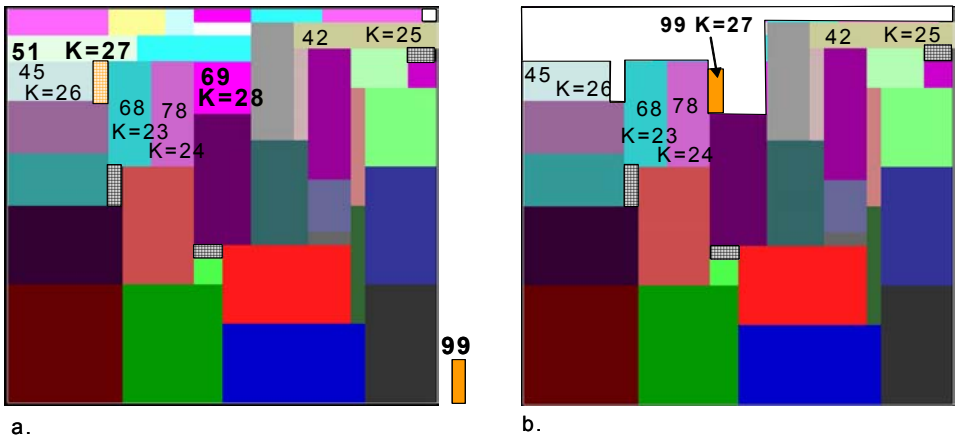


Figure 4-8a, b Trapped-hole-delay insert example

Returning to the initial configuration of Figure 4-8a, the result of one such trapped-hole-delay move is shown in Figure 4-8c where item 51 is delayed one position to k=28 in the packing sequence, i.e., item 51 swaps with item 69. This trapped-hole-delay move, with a move value of zero, is implemented. On the next iteration, the excess–bin-insert neighborhood places item 99 at position k=28 as shown in Figure 4-8d. To complete the move, the remaining items, beginning with item 51 in position k=29, are placed by the pcf-BL heuristic.

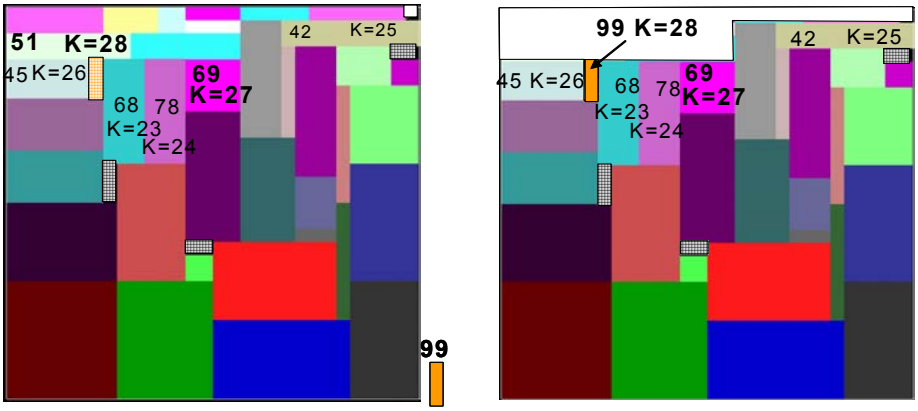


Figure 4-8c, d Trapped-hole-delay insert example

Other trapped-hole-delay move combinations could have resulted in other appealing placements of item 99 such as in the trapped hole below item 68.

Previous tabu search applications have also seen the value of intermediate "zero valued moves". (Barnes and Chambers, 1995)

4.3.4 Inter-Bin Moves

The inter-bin moves, inter-bin-insert or inter-bin-swap, allow the ATS-BP to consolidate dead space by moving at least one item from a position in one bin and placing it into another.

4.3.4.1 Inter-Bin Insert

This move inserts one item, considering both orientations, from one packing position to a packing position in another bin. The departure bin is repacked without the relocated item. The heuristic attempts to repack the destination bin with the inserted item. The move is not considered if the inserted

item ejects from the destination bin. However, items that follow the inserted item are allowed to eject.

This move only considers items that either have $h_i < h_{Big}$ or $w_i < w_{Big}$. This is the most powerful of the non-excess bin moves since it can empty an open bin.

4.3.4.2 Inter-Bin Swap

This move swaps the packing positions of two items in different bins. Ejection of either of the moved items is not allowed. Many items too large for inter-bin-insert moves are included in the inter-bin-swap moves. Identical items are not considered for swap moves.

Inter-bin-swap moves embody a search intensification context which preserves the orientation of the swapped objects associated with the bin, not the object. Figure 4-9 shows an example of this concept. In Figure 4-9a, item 12 in bin A (in orientation 0) and item 13 in bin B (in orientation 1) are swapped. In the resulting configuration, item 13 assumes its orientation 0 and item 12 assumes its orientation 1. The S_n representation of this new solution replaces letters from 12 and 23 with letters 13 and 22.

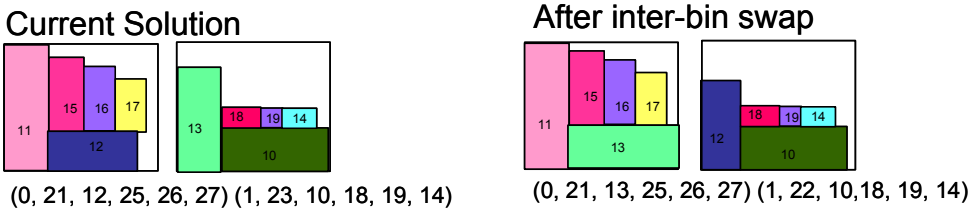


Figure 4-9 Swap move where orientation is retained.

4.3.5 Alternate Initial Solution for High IBR problems

When the initial solution generated by the pcf-BL heuristic described in Section 3.4 yields an excess bin with 10 or more items, a possible alternate initial solution is investigated if IBR >30. This alternate solution may take advantage of intrinsic features of the pcf-BL heuristic.

Figure 4-10a shows a part of the initial solution for a specific example yielded by the original heuristic. Figure 4-10b pictures part of the corresponding alternate initial solution. When placing items, layouts that build a "stairway" that ascends from left to right form "insert friendly" sequences, i.e., the sequences correspond to distinct positional layouts, in the subsequent layer. There is only one ordering yielding the second layer in the bin in Figure 4-10b, while the second layer in the bin in Figure 4-10a has several orderings.

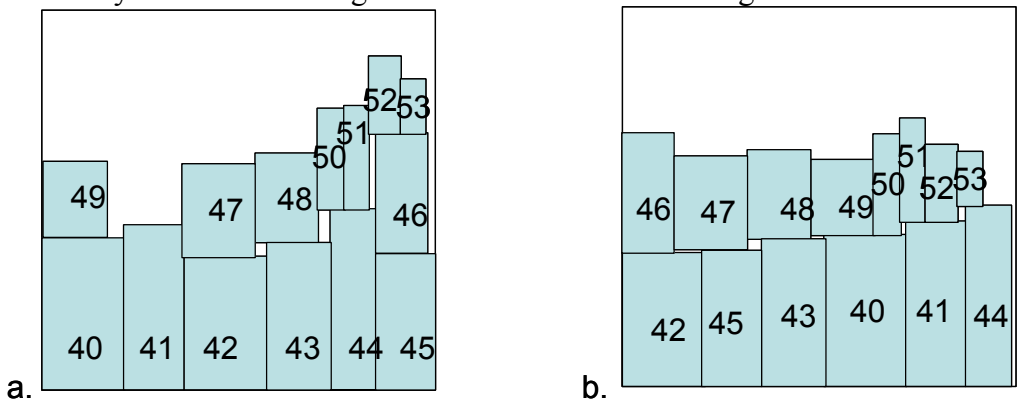


Figure 4-10 (a) Sort by size (b) First layer reverse height sort

The alternate initial solution procedure conducts the same two tiered sort as the regular initial solution. Next, a subset of the size sorted items are collected for a reverse height sort. Items are accumulated until one of two limiting

conditions are reached: (1) the $\sum_i h_i \leq 1.6W(LB)$ or $\sum_i h_i w_i \leq 0.5\ HW(LB)$. The resulting subset is sorted in reverse h_i order redefining their packing order. These items are packed in their new order while the remaining items are packed in the original two level hierarchy order.

The alternate initial solution is used in lieu of the original initial solution if *both* the number of bins does not increase, *and* if either one of the following conditions hold (a) the number of items does not increase in the excess bin by more than 40%, or (b) the Big-Item is reduced in size.

When used, the alternate initial solution greatly accelerates the search process.

This section detailed the moves used in ATS-BP. The next section discusses the dynamic integration of these moves implemented in ATS-BP.

4.4 THE ARCHITECTURE OF ATS-BP HEURISTIC

The following two subsections overview the ATS-BP architecture.

4.4.1 Tabu Structures

Tabu search is concerned with imposing *restrictions* to guide the search process through difficult regions and to allow the search to escape from local optima. *Tabu restrictions* are constraints imposed by move *attributes* to prevent the reversal of recent moves. A *tabu tenure* stipulates the number of subsequent iterations that a tabu restriction remains in force. *Tabu attributes* are solution components that change with a move and are used to distinguish future solutions

from recently visited solutions. ATS-BP imposes two types of tabu restrictions: *Tabu Bin* and *Tabu Order*.

Tabu Bin prohibits an item, in either orientation, from returning to an excess bin within tabu tenure iterations, i.e., either orientation letter associated with an item is prohibited from returning to a particular cycle in the S_n solution representation. The tabu bin attributes consist of a bin letter and an item's letters. *Tabu Order* prohibits an item-orientation from repeating a particular packing order associated with a bin, i.e., prohibits an item-orientation letter from returning to a specific position in a bin cycle within tabu tenure iterations. Tabu order is used for all moves not involving excess bins. A triplet of tabu attributes is used for this restriction, (bin letter, item letter, item position). The following examples illustrate how the tabu order restriction defines whether or not a move is tabu.

Suppose the current *adaptive tabu tenure*, detailed below, is 7 iterations and the current solution is

(0, 20, 21, 22, 23, 24, **45**, 26, 27, 28, 29, 30) (1, 31, **52**, 33, 34, 35, 36, 37, 38, 39)

If an inter-bin swap of letters 45 and 52 is performed, no move that will result in placing letter 45 in bin 0 at position 6 AND letter 52 in bin1 at position 2 may be performed before iteration 9.

Using the same current solution, suppose that an insert move selects letter 45 to be relocated. Four tabu attributes are used to enforce the associated tabu order restriction, the bin letter, preceding item letter, relocated item letter, and item position. In our example, letter 45 cannot return to position 6 in bin 0 if it will be right adjacent to letter 24 before tabu tenure iterations have been

completed. In the special case that the relocated letter was in the bin's first position, it may not return to that first position within tabu tenure iterations.

Moves satisfying an *aspiration criterion* can override a tabu restriction. ATS-BP uses a single aspiration criterion, only moves that achieve a new best global objective function value are allowed to override a tabu restriction.

The tabu bin and tabu order restrictions use the same *adaptive tabu tenure structure*. The initial tenure is set to a minimum value. If the selected move does not improve the objective function, the tabu tenure increases by 1. If the selected move improves the objective function it decreases by one. The increase (decrease) is never allowed to be above a maximum (minimum) tenure. Any time the aspiration criteria is met the tenure is reset to the minimum. ATS-BP has two minimum and maximum tenure settings:

(1) IBR≤7: minimum 7 and maximum 12.

(2) IBR> 7: minimum 9 and maximum 14.

These difference limits reflect the fact that the higher IBR problems favor intra-bin moves which tend to be more prone to cycling than other moves.

This subsection detailed the tabu restrictions used in this research. The next section discusses how moves are dynamically selected.

4.4.2 The Organization and Logic Flow of the ATS-BP Heuristic

In this subsection, we outline the conceptual framework for the dynamic move selection strategies and discuss the logical flow of the heuristic. Key to implementing a successful move strategy is defining high and low influence moves. Influence "…measures the degree of change induced in solution

structure..." (Laguna and Glover, 1997) The high influence moves in ATS-BP are excess bin moves. All other moves have low influence. If a non-ejecting excess-bin-insert move is found or if any excess bin move finds a new global best solution relative to the fine grain objective, the best high influence move is implemented. ATS-BP allows a maximum of four consecutive low influence moves.

ATS-BP rewards low influence moves for arrangements favorable to future high influence moves. It also limits, independently, both the number of consecutive inter-bin moves and intra-bin moves.

The dynamic move selection strategies described in this section vary the frequency of use and the size of both high and low influence move neighborhoods in order to achieve intensification/diversification effects.

ATS-BP changes which move neighborhoods are implemented and the neighborhood sizes are based on the following measures:

(1) Number of items, n

(2) Item to Bin ratio, IBR

(3) Aspiration Criteria count (AC): number of iterations since the most recent global best solution was found

(4) Non-improving moves count (NIMC): number of iterations since last improving move

(5) Non-excess bin moves count (NEBMC): number of consecutive low influence moves.

(6) Inter-bin moves count (InterMC): number of consecutive inter-bin swaps or inserts.

(7) Intra-bin moves count (IntraMC): number of consecutive hole-top inserts or intra-bin inserts.

(8) Every 50 iterations the search strategy oscillates between an intensification and diversification mode, starting with the intensification mode. These modes are discussed below.

Intensification and diversification modes in the ATS-BP are implemented for problems with IBR<7. The ATS-BP uses *only* the intensification mode (the default setting) for problems with IBR $\geq$ 7.

Figure 4-11 shows the pseudo-code for ATS-BP which is now discussed in detail. At initialization, the ATS-BP obtains the initial solution, identifies the initial excess bin and Big-Item, and sets counters and required parameters. There are 3 optional termination conditions: (1) achievement of a known lower bound on bins, (2) performance of a maximal number of iterations and (3) achievement of a maximum limit on computation time. The procedure then begins examining each neighborhood presented in Section 4.3.

```
Obtain  initial solution, initialize counters, and other parameters
Identify excess bin  & Big-Item
While termination conditions are not satisfied, do
{          // Find best move
             Examine excess bin moves, record BEST_MOVE
                    // (Excess-bin-insert neighborhood restrictions determined by AC, NIMC, NEBMC, |S|)
                    // (Excess-bin-swap neighborhood restrictions are not dynamic )
             If  insert non-ejection move found or aspiration satisfied, exit to FOUND BEST MOVE
             Else
             If  NEBMC < 4
             {      // Trapped-hole-delay neighborhood
                    If  IntraMC < 2 I  IBR ≥ 5
                    {      Examine trapped-hole-delay moves for bins that have > 5 items, update  BEST_MOVE
                           If aspiration satisfied take best trapped-hole-delay move, exit to FOUND BEST MOVE
                    } Else
                    // Intra-bin-insert neighborhood
                    If  IntraMC < 1
                    {   If  (diversification I  NEBMC < 3) U  (In intensification mode)
                       {        Examine intra-bin moves, update  BEST_MOVE
                                //(Neighborhood definition restrictions determined by NIMC, AC, ** )
                                If aspiration satisfied take best intra-bin move, exit to FOUND BEST MOVE
                       }Else
                    }
                    // Inter-bin neighborhoods
                    If  (InterMC = 3) U  (diversification I  n≤60 I  InterMC = 2), exit to FOUND BEST MOVE
                    Else
                    {           Examine inter-bin insert moves, update BEST_MOVE
                                   // (Inter-bin insert neighborhood definition restrictions determined by NEBMC, AC**)
                                If aspiration satisfied take best inter-bin insert move, exit to FOUND BEST MOVE
                                Examine inter-bin swap moves, update BEST_MOVE
                                  // (Inter-bin swap neighborhood definition restrictions determined by NEBMC, AC**)
                                If aspire take best inter-bin swap move, exit to FOUND BEST MOVE
                    } End Else
            } End If NEBMC < 4
            FOUND BEST MOVE:  Implement BEST_MOVE
            Update solution, tabu tenure, cntrs, store best solution found, new excess bin and Big-Item
} End while termination conditions

** IBR < 7 varies neighborhood size for diversification setting:  Iters Mod (100) ≥50.
IBR ≥ 7 remain in the intensification settings at all times
```

Figure 4-11 ATS-BP pseudo code

All excess-bin moves are evaluated at each iteration. The *excess-bin-insert neighborhood* incorporates a dynamic *functional* candidate list mechanism to screen items considered for extraction from the excess bin, i.e., only items that satisfy

$$(h_i > \varphi\, h_{Big} \text{ U } w_i > \varphi\, w_{Big}) \text{ I } (h_i w_i \geq f(\varphi) h_{Big} w_{Big})$$

are considered, where $f(\varphi)$ is a discrete function of φ as detailed below.

Of course, the Big-Item is *always* a candidate for excess bin extraction. The logic supporting this pair of relations is that a high value of φ will strongly restrict the candidate list of items available for insertion from the excess bin and will favor larger items *dissimilar* from the Big-Item. Small values of φ will have the opposite effect, admitting more items to this candidate list while still prohibiting the smallest items. Figure 4-12 details the four values of φ used in ATS-BP and the decision process that invokes each value. $\varphi \equiv 1$ is the default setting for ordinary search conditions. Upon occasion, numerous items can be present in the excess bin. $\varphi \equiv 2$ is applied in this situation to prevent the larger items from being ignored in lieu of numerous smaller items which would hinder the progress of the search process. $\varphi \equiv 0.8$ and $\varphi \equiv 0.6$ are increasingly liberal values allowing for a larger candidate list. φ is set to 0.8 when moderate search stagnation has occurred as evidenced by three simultaneous circumstances where a moderate number of iterations have seen no aspiration criterion satisfaction, the previous move was not improving and was not a high influence move. φ is set to 0.6 when a greater number of iterations have seen no aspiration criterion satisfaction and the last two moves were not high influence moves.

$\phi = 1$ // Default value

If $(|S| > 12)$ $\phi = 2$

Else If $((2 \le AC \le 7)\,\mathrm{I}\ (1 \le NIMC)\,\mathrm{I}\ (1 \le NEBMC))$ $\phi = 0.8$

Else If $((7 < AC \le 35)\,\mathrm{I}\ (2 \le NEBMC)$ $\phi = 0.6$

Figure 4-12 Excess-Bin-Insert φ

The discrete function, $f(\varphi)$, was created for the following reasons. When the search is not in the diversification mode and $\varphi \neq 2$, $f(\varphi) = 0.5\varphi$, which achieves the standard area threshold for candidate list membership. When the search is in the diversification mode and $\varphi \neq 2$, $f(\varphi)$ is set to 0.65φ to be more restrictive on small items becoming part of the candidate list for excess-bin-insert moves. When $\varphi = 2$, $f(\varphi)$ is set to 0.375φ to relax the restriction on area, since $f(\varphi) = 0.5\varphi$ would force candidate items to be of identical area.

The *excess-bin-swap neighborhood* uses a single static equation to dynamically screen candidates for swapping positions with the Big-Item in the excess bin,

$$\frac{1}{16} h_{Big} w_{Big} \leq h_i w_i \leq 4 h_{Big} w_{Big}$$

where all items, i, in any packing position are considered. The screening method is dynamic because the Big-Item may change with each iteration. Because this neighborhood is evaluated at every iteration, this restriction is present to lessen the associated computational effort without lessening the power of the search.

Having considered the high influence excess bin moves, the algorithm proceeds to the low influence moves. If the last 4 iterations have used low influence moves, the best high influence move is immediately implemented. Otherwise, if the last 2 moves have *not* been intra-bin moves (which include intra-bin-insert and trapped-hole-delay moves) and IBR ≥ 5, the trapped-hole-delay neighborhood is evaluated.

In this neighborhood, bins with more than 5 items are checked for trapped holes and the move is conducted on all items which seal trapped-holes. The IBR

threshold prevents this type of move computation when associated improvements are unlikely.

Intra-bin-insert moves are examined only if the previous move was not an intra-bin move. In addition, when the search is in the diversification mode (which only occurs when IBR < 7), the additional restriction of the last 3 moves not being low influence moves is imposed.

In evaluating the intra-bin neighborhood, candidate bins are limited to those bins satisfying the following relation (which requires a minimal amount of dead space in any candidate bin):

$$\gamma h_{Big} w_{Big} \leq (HW - \sum_{i \in k} h_i w_i)$$

γ is a dynamic parameter used to expand the candidate list of the intra-bin neighborhood as the number of nonimproving moves (NIMC) increases. Table 4-1 details the possible values of γ and the conditions to which they apply.

Diversification	NIMC	AC	γ	|M|
No	NIMC≤1		1	4< |M|
No	1<NIMC≤3		0.8	4< |M|
No	3<NIMC		0.4	4< |M|
No	3<NIMC	20<AC	0	4< |M|
Yes	NIMC≤1		0.3	2< |M|
Yes	1<NIMC≤3		0.24	2< |M|
Yes	3<NIMC		0.18	2< |M|
Yes	3<NIMC	20<AC	0	2< |M|

Table 4-1: Determining the bins considered for inter-bin move candidates

First, consider the case where the search is *not* in the diversification mode. In this case, only bins with 5 or more items (4<|M|) are considered. If an

improving move was just executed $\gamma \equiv 1$, limiting the search only to the most promising bins for future Big-Item placement. If NIMC is 2 or 3, $\gamma \equiv 0.8$ slightly relaxes the list restriction. When NIMC $\geq$ 4, $\gamma \equiv 0.4$. If NIMC $\geq$ 4 and AC > 20 (no aspiration satisfaction has occurred in the last 20 moves), the dead space restriction is removed by setting γ to 0.

During search diversification, all bins with at least three items are considered. Additionally, all of the γ values in the previous paragraph are reduced 70%.

If the best move from this neighborhood achieves the aspiration criterion, the remaining neighborhoods are not checked.

We will now discuss the inter-bin neighborhoods. If the inter-bin move count (InterMC), which includes both inter-bin-insert and swap moves, is equal to 3, the inter-bin neighborhoods are not evaluated. This criterion controls an over application of inter-bin moves while allowing essential consolidation of dead space. Given that InterMC < 3, one addition condition can preclude evaluation of the inter-bin neighborhoods:

diversification mode active I n$\leq$60 I InterMC=2

This additional restriction is present because the smaller problems more quickly achieve over application of this move type.

Presuming that the inter-bin neighborhoods are evaluated, the first considered are the inter-bin-insert moves. Inter-bin-insert moves consolidate dead space, with the goal of making enough space available in an open bin to accept the Big-Item in the excess bin. When NEBMC < 3, candidate moves are required

to create dead space in the departure bin that is at least 50% of the area of the Big-Item. When NEBMC = 3, candidate moves are required to create dead space in the departure bin that are greater than or equal to the area of the Big-Item. When AC > 10 and NEBMC < 2, the dead space requirement is no longer enforced with the goal of enabling a short term diversification. If the inter-bin-insert neighborhood finds a new global best solution, the related move is implemented without further investigation. Failing a new global best solution, the algorithm continues with the evaluation of the inter-bin-swap neighborhood.

The number of candidate moves for the inter-bin-swap neighborhood depends on the NEBMC, the AC, and whether or not the search is in the diversification mode. Candidate bin pairs, (*j,k*), *combined dead space* must exceed a defined proportion of the Big-Item area as given in the following expression:

$$\tau h_{Big} w_{Big} \leq (HW - \sum_{i \in j} h_i w_1) + (HW - \sum_{i \in k} h_i w_i)$$

τ is a dynamic parameter that depends on the NEBMC and the AC.

When NEBMC < 3, candidate moves are required to have combined dead space that is at least 50% ($\tau = 0.5$) of the area of the Big-Item. When NEBMC = 3, candidate moves are required to have combined dead space greater than or equal to the area of the Big-Item ($\tau = 1$). When AC > 10 and NEBMC < 2, the combined dead space requirement is no longer enforced with the goal of enabling a short term diversification ($\tau = 0$).

During intensification and when the NEBMC ≥1, candidate moves must create a dead space area in one of the bins exceeding the size of the Big Item , i.e., the following relation must be satisfied:

$$h_{Big} w_{Big} \leq \max((HW - \sum_{i \in j} h_i w_1)^{\text{Predicted}}, (HW - \sum_{i \in k} h_i w_i)^{\text{Predicted}})$$

The goal of the inter-bin-swap neighborhood is to produce a bin that will have enough space for the current Big Item. For problems with high IBR, this is easily achieved. For lower IBR problems this equation greatly reduces the candidates that must be examined.

If the search is in the diversification mode and NEBMC = 0:

(1) the maximum dead space constraint is not applied

(2) candidate moves are required to have combined dead space that is at least 50% ($\tau = 0.5$) of the area of the Big-Item.

(3) When AC > 10, the combined dead space requirement is no longer enforced with the goal of enabling a short term diversification ($\tau = 0$).

After consideration of all possible moves, the "best move" is implemented, the tabu structures are updated, the tabu tenure and all counters are adjusted. If required, a new excess bin and/or Big-Item is determined. The ATS-BP program continues iterations until a stopping criterion is reached.

This section described the dynamic move selection strategies used in this research to control the move neighbors visited in the ATS-BP.

One final comment about the ATS-BP structure involves an efficient bin data structure that appears to have been overlooked in previous research. The pcf-BL heuristic is an efficient 2D packing heuristic requiring $O(m^2)$ operations for each move evaluated, where m is the number of items that are in bins to be repacked. Since the low influence moves have up to $O(n^2)$ combinations, the

number of low influence moves per iteration grows rapidly with problem size. Techniques to reduce the number of moves considered will enhance computational efficiency. In addition to the basic screening criteria detailed in the previous section, this research implements two novel approaches to improve efficiency that can be adapted to any direct search method using an order-based heuristic applied to multidimensional bin packing.

Dowsland [1996] states that order based heuristics require a complete repacking of the container at each move. This is true only in a very rare worst case scenario. The ATS-BP saves the bin data structure as each item is packed so that future moves do not have to completely repack a bin. The bins are only repacked from the point of change. This is achieved by storing the "Bin History" for each item as it was packed (A numerical Item Data list is also stored that saves where each item is packed). The *Bin History* saves all crucial arrays and data needed to continue the pcf-BL from any point in the packing sequence. Whenever a new move is accepted, the bin history for all accepted items is updated. Whenever an item is inserted into a sequence, the ATS-BP identifies its immediate predecessor and retrieves the Bin History for the item. The expected computational savings associated with this technique is 50%.

The next section details specific data structure efficiencies that allow time saving in this search.

4.5 APPLICATION OF ATS-BP TO BENCHMARK TEST SETS

This section presents an analysis of the ATS-BP results for the Berkey and Wang (1987) and Martello and Vigo (1998) test sets, as described in Section 3.5,

compares those results with those of Lodi et al. (1999-1), and provides pictorial solution representation of selected instances.. Detail for the number of bins used and ATS-BP solution times are provided in Appendix A for each of the 500 instances.

BERKEY AND WANG TEST SET(1987) (2BP|R|F)						
Lodi, Martello, and Vigo (1999)						
Class	*n*	TP TS LB	average time	ATS-BP LB	average time	time to match
I	20	1.05	18.00	1.03	0.13	0.13
	40	1.04	30.02	1.04	4.11	4.11
	60	1.04	34.00	1.04	4.27	4.27
	80	1.06	48.08	1.06	0.97	0.97
	100	1.03	47.92	1.03	1.20	1.20
	Average	**1.044**	**35.60**	**1.040**	**2.14**	**2.14**
II	20	1.00	0.01	1.00	0.13	0.13
	40	1.10	0.01	1.00	1.46	0.21
	60	1.00	0.01	1.00	1.14	1.14
	80	1.03	6.10	1.00	22.61	2.76
	100	1.00	0.01	1.00	4.25	4.25
	Average	**1.026**	**1.23**	**1.000**	**5.92**	**1.70**
III	20	1.06	18.00	1.02	11.35	0.17
	40	1.09	42.17	1.09	0.38	0.38
	60	1.08	54.15	1.08	64.85	64.85
	80	1.07	60.07	1.07	9.94	9.94
	100	1.07	60.18	1.06	68.05	0.98
	Average	**1.074**	**46.91**	**1.064**	**30.91**	**15.26**
IV	20	1.00	0.01	1.00	0.14	0.14
	40	1.00	0.01	1.00	0.25	0.25
	60	1.10	0.09	1.05	28.80	0.31
	80	1.07	12.00	1.07	36.16	36.17
	100	1.03	6.00	1.03	9.19	9.19
	Average	**1.040**	**3.62**	**1.030**	**14.91**	**9.21**
V	20	1.04	12.01	1.04	1.31	1.31
	40	1.07	42.00	1.06	41.21	18.72
	60	1.06	45.23	1.06	74.59	74.59
	80	1.07	54.14	1.06	143.01	14.63
	100	1.07	60.12	1.05	196.23	1.58
	Average	**1.062**	**42.70**	**1.054**	**91.27**	**22.17**
VI	20	1.00	0.01	1.00	0.17	0.17
	40	1.40	0.03	1.10	5.97	0.27
	60	1.05	0.05	1.00	25.92	0.32
	80	1.00	0.01	1.00	0.42	0.42
	100	1.07	12.00	1.07	7.64	7.64
	Average	**1.104**	**2.42**	**1.034**	**8.02**	**1.76**
Test Set Average		**1.058**	**22.081**	**1.037**	**25.528**	**8.707**

Table 4-2 Berkey and Wang test set consolidated results

MARTELLO AND VIGO TEST SET(1998) (2BP|R|F)						
Lodi, Martello, and Vigo (1999)						
Class	*n*	TP TS LB	average time	ATS-BP LB	average time	time to match
VII	20	1.11	30.00	1.11	0.77	0.77
	40	1.08	48.06	1.07	99.28	36.93
	60	1.06	59.45	1.04	613.98	115.98
	80	1.10	60.12	1.08	259.78	54.06
	100	1.08	60.36	1.07	231.66	30.08
	Average	**1.086**	**51.60**	**1.074**	**241.09**	**47.56**
VIII	20	1.10	30.01	1.10	1.34	1.34
	40	1.10	54.22	1.08	77.91	0.28
	60	1.07	56.17	1.07	51.74	51.74
	80	1.08	60.11	1.07	529.40	10.69
	100	1.09	60.14	1.08	232.88	0.72
	Average	**1.088**	**52.13**	**1.080**	**178.65**	**12.95**
IX	20	1.00	0.06	1.00	0.26	0.26
	40	1.01	18.85	1.01	1.25	1.25
	60	1.01	18.03	1.01	2.54	2.54
	80	1.01	30.51	1.01	1.65	0.65
	100	1.01	36.86	1.01	52.69	1.39
	Average	**1.008**	**20.86**	**1.008**	**11.68**	**1.22**
X	20	1.12	6.01	1.12	0.36	0.36
	40	1.06	24.01	1.06	3.93	3.88
	60	1.06	30.44	1.06	235.17	235.17
	80	1.05	39.04	1.05	78.27	78.27
	100	1.05	43.38	1.04	269.68	12.25
	Average	**1.068**	**28.58**	**1.066**	**117.48**	**65.99**
Test Set Average		**1.063**	**38.292**	**1.057**	**137.227**	**31.931**
Grand Average		**1.060**	**28.565**	**1.045**	**70.208**	**17.996**

Table 4-3 Martello and Vigo test set consolidated results

The consolidated results (where all problems are weighted equally) are highlighted in blue in Table 4-3. Overall, the ATS-BP achieved an average $\frac{ATS-BP}{IBL}$ ratio of 1.045. This improvement of 0.015, compared to the TS-TP result of 1.060 reported by Lodi et al. (1999-1), is a 25% reduction in the previous best lower bound gap. This subsection highlights some of these improvements.

The largest improvements occurred in the three test sets with the highest IBR. Classes II, IV, and VI had IBR's that ranged from a minimum of 20 to a

maximum of 40. For these three classes, the ATS-BP found results within 2.1% of the lower bound, while TS-TP was with 5.7% of the lower bound. Part of this difference is explained by the TS-TP's inability to improve an initial solution to a single bin problem, and its limited search neighborhood when there are only a few bins.

ATS-BP also performed well on data sets with low IBR. ATP-BP was within 5.5% of the lower bound while TP-TS was within 6.1% of the lower bound. Classes I, III, V, VII, VIII, IX, X have IBR values ranging from 2 to 10. However, of those classes only Class X had problems where the IBR exceeded 5.

To illustrate the quality of the results obtained from the ATS-BP, a selected subset of specific packing solutions is now provided. As stated in Chapter 3, the results given by Lodi, et. al. (1999-1) presented only consolidated results. The solutions for the seven individual problem instances that *appear* to have experienced the greatest improvement over the results of Lodi et al. (1999-1) based on the $\frac{ATS-BP}{IBL}$ ratio are provided below in Figures 4-13 through 4-19. The boundaries of each container pictured are represented by a thick black line. Unfilled space *within* the container is also given in black. These figures were created with the software described in Rippert (2003).

Figures 4-13 through 4-19 demonstrate the quality of solutions that may be obtained when explicit control of partitioning, ordering and orientation is conducted within an adaptive tabu search methodology.

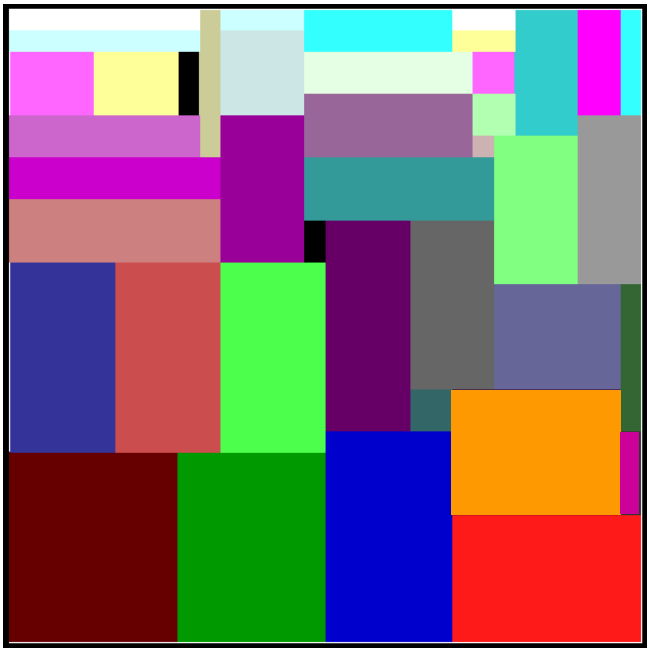


Figure 4-13 Berkey and Wang Class II Size 40 Instance 1. (99.44% fill)

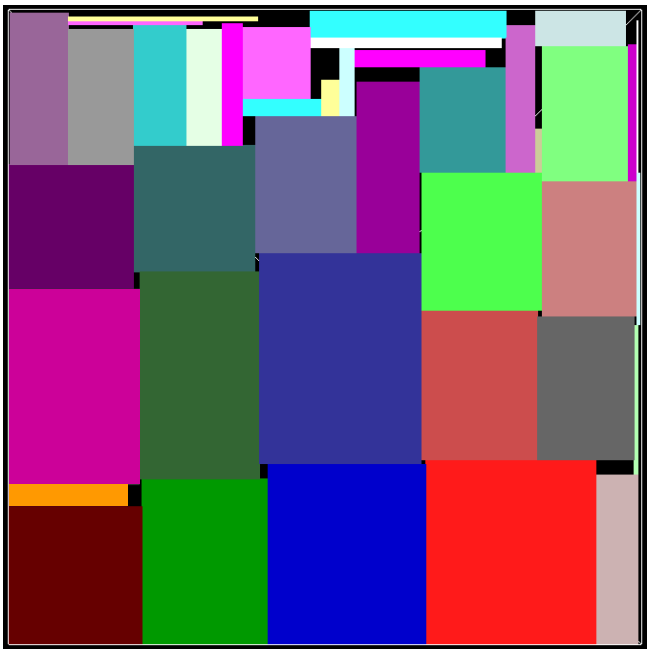


Figure 4-14 Berkey and Wang Class VI Size 40 Instance 6. (97.52% fill)

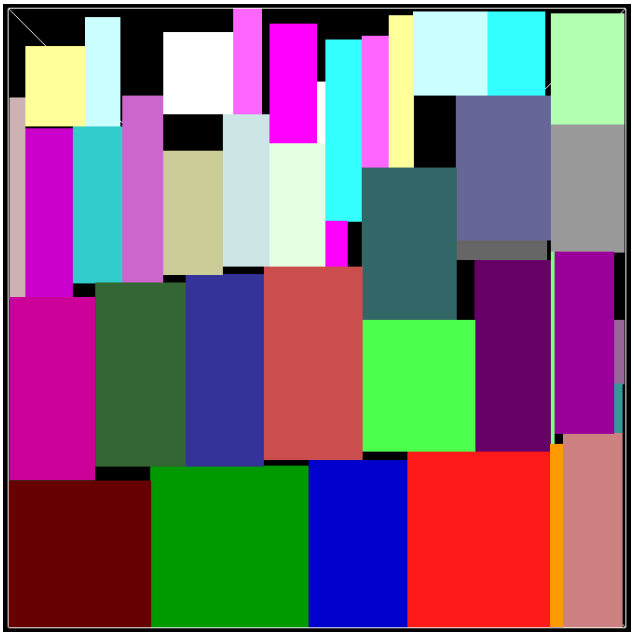


Figure 4-15 Berkey and Wang Class VI Size 40 Instance 9. (93.31% fill)

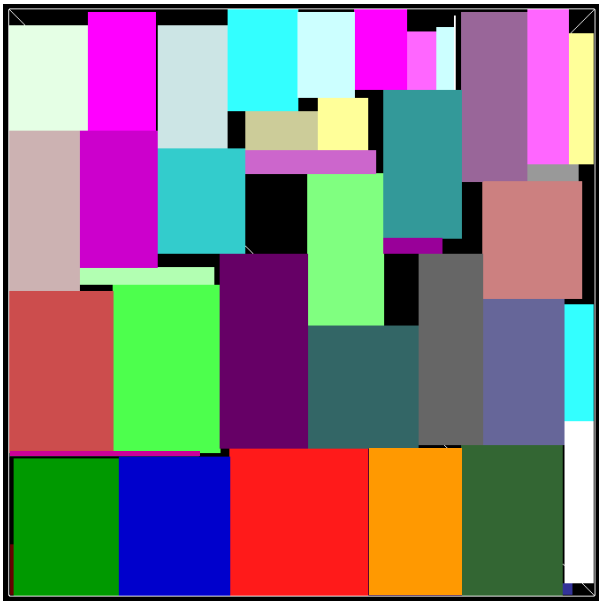


Figure 4-16 Berkey and Wang Class VI Size 40 Instance 10. (93.92% fill)

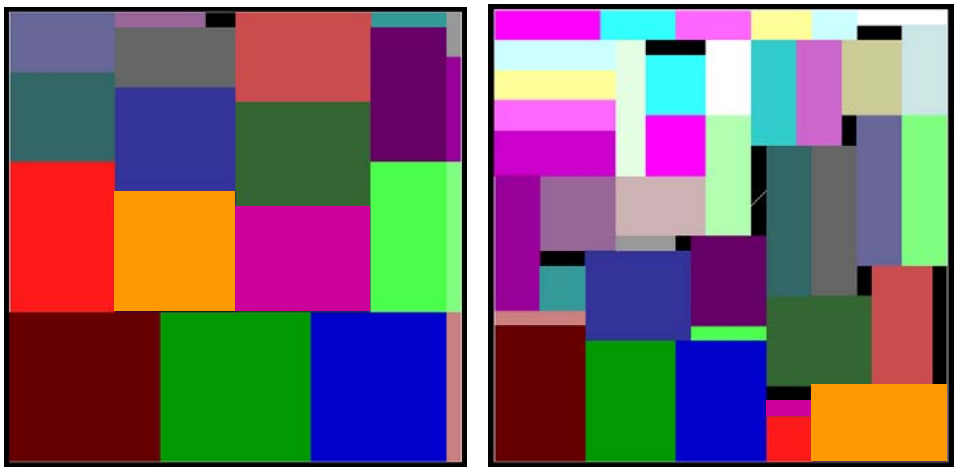


Figure 4-17 Berkey and Wang Class II Size 60 Instance 2. (99.77% and 96.44% fill

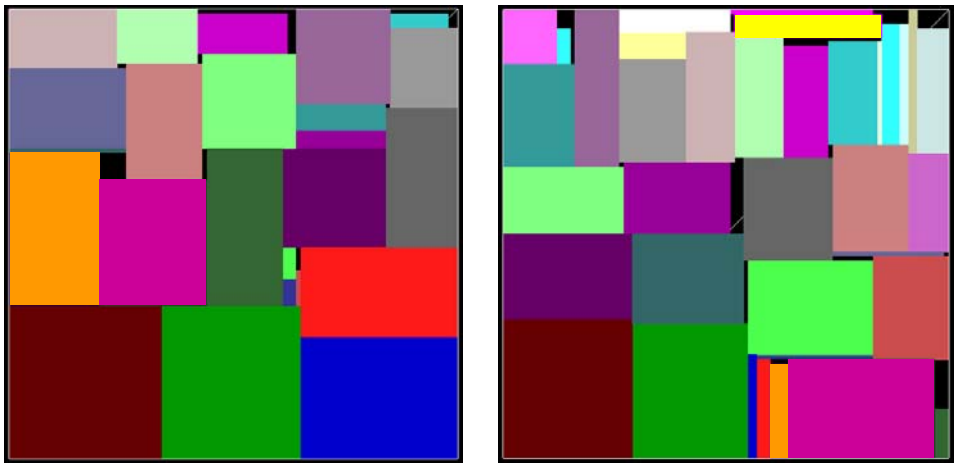


Figure 4-18 Berkey and Wang Class IV Size 60 Instance 4. (97.77 and 98.86% fill

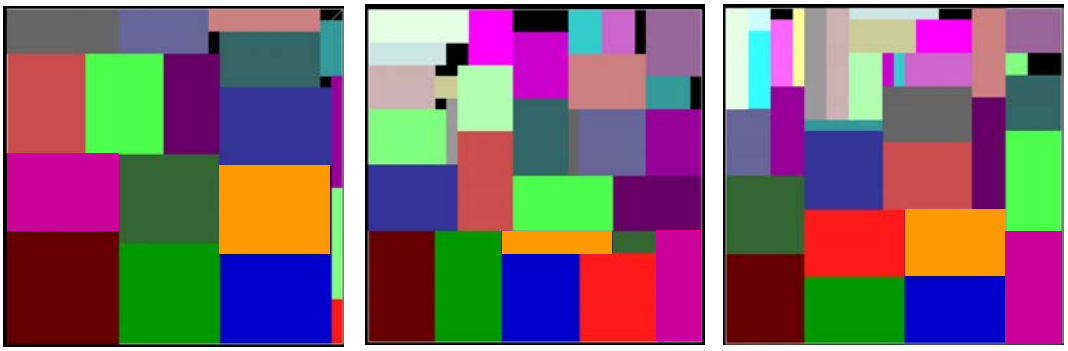


Figure 4-19 Berkey and Wang Class II Size 80 Instance 7. (99.44%, 97.33% and 98.78 % fill)

ATS-BP did take longer to find the answer to some of the problems that TS-TP discovered rather quickly. This is not surprising. Methods that sort by size tend to perform well in many cases. If a 'sorted' good solution exists, these methods in many cases quickly find the partitions needed to solve it. In general there are $m!2^m$ orderings for a set of m items to be ordered for packing in a single bin. Of the $m!2^m$ solutions, TS-TP determines one. However, in many instances that one solution is all that is required. Thus, there are several insights gained by this comparison. First is that ATS-BP could benefit from a move neighborhood that sorts and repacks similar to TS-TP. Future implementations could consider this move set. Second, ATS-BP might also benefit from moves that use move aggressive $O(n^3)$ heuristics such as TP, map them to Euclidean space, and map them back to a S_n representation in the pcf-BL solution-space. Additionally, a conceptualized powerful data structure change, which saves moves that do not change, could close this occasional time gap and is presented in Chapter 5.

Information about the techniques that is not presented in the summary tables of this section is the ability of ATS-BP, with its fine grained objective function, to discriminate between two or more solutions that use the same number of bins. The ATS-BP considers how full the bins are packed and how compact the items are within the bins. Thus, ATS-BP is readily adaptable to practical considerations such as when the number of bins available is strictly limited, a characteristic of the 2D orthogonal knapsack version of this problem.

This chapter detailed an adaptive tabu search method for bin packing that represented solutions in a S_n framework. The results of ATS-BP are substantial. ATS-BP's results met or outperformed the previous best technique on all 50 subsets from the Berkey and Wang (1987) and Martello and Vigo (1998) test sets. Unfortunately, individual problem results were *not* available for comparison. Overall, ATS-BP reduced the gap with the lower bound by 25%.

The next chapter discusses extensions to this problem.

Chapter 5

Directions for Future Research and Research Contributions

The first three sections of this chapter propose directions for future research. The first section proposes enhancements to ATS-BP. The second section suggests extensions of the ATS-BP to address other packing problems. The third section discusses issues involving practical considerations such as weight, balance, and hazardous materials.

The fourth section details the research contributions documented in this dissertation.

5.1 Extensions to ATS-BP

This first subsection details improvements to make ATS-BP more computationally efficient. The second and third subsections discuss insights into vocabulary building techniques and intensification-diversification strategies for this problem. The final subsection suggests approaches to extending the ATS-BP into a robust GTTS framework.

5.1.1 Move value matrix.

It would be beneficial to develop a move evaluation structure that preserves move values that do not change from one iteration to the next. With the current move neighborhoods, only two or three bins actually change each iteration. Moves could be stored in a (m+n) square move value matrix (where m is the number of bins and n the number of items) or similar data structure. After a

move, only elements in the affected row and columns associated with the affected bins moves would have to be updated. Implementing this could save significant amounts of computational effort when using the ATS-BP for problems with numerous bins.

5.1.2 Vocabulary Building

Vocabulary Building seeks to "identify meaningful fragments of solutions, rather than focusing on full vectors, as a basis of generating combinations." (Laguna and Glover, 1997) This approach has been successfully used for several problems including the Traveling Salesman problem (Glover, 1992), Vehicle Routing (Rochat and Taillard, 1995) and (Kelly and Xu , 1995). A recent approach that used Vocabulary Building in conjunction with Adaptive Tabu Search was presented by Combs (2002). He used a partitioning optimizer to provide a vocabulary building mechanism.

The vocabulary building method must be tied to the heuristic being used. One important discovery in this research is that the pcf-BL has many nice features if items are packed in ascending height from left to right, as discussed in Section 4.3.5. This initial solution reverse height sort technique could be used as a move that takes existing bins and repacks the bottom layers in this manner to build vocabulary words.

5.1.3 Intensification - Diversification Strategies

The intensification and diversification strategies discussed in this dissertation employed threshold setting conducted on a cyclic basis to

accommodate the situation where problems may be caught in inferior regions of the solution space. Future research could focus on more aggressive and reactive intensification and diversification strategies in order to improve the search process.

Moves that recombine bins like the base moves suggested by Lodi et al. (1999-1) may be an effective diversification move neighborhood strategy. ATS-BP could take small groups of bins, empty the contents, and allow a heuristic to sort and repack the bins.

5.1.4 GTTS-BP

The S_n representation presented in Section 4.2 provides a sufficient foundation to support the development of a true group theoretic approach to the bin packing problem. The ATS-BP did not require the advanced group operations related to S_n to achieve superior results. However, for more challenging packing problems, moves based on previous successful research could render enhanced search techniques.

5.2 PACKING EXTENSIONS

This section highlights insights on other 2D rectangular bin packing variations and 3D rectangular implementations.

5.2.1 2D Rectangular variations

This subsection highlights insights into implementing ATS-BP for the 2D free packing problem where item orientations are fixed, and issues to consider before implementing this approach on the 2D guillotinable cuts variations.

5.2.1.1 2D Oriented Free Packing (2BP|O|F)

The S_n representation for this problem would require only half as many letters. The group would not require special control of one-cycles as needed with the $2n$ letters in this implementation. Thus, any element of S_n may describe a feasible layout. This may be a good candidate for the first GTTS-BP implementation.

The pcf-BL already has the ability to explicitly control orientation. Thus, this problem could be implemented in ATS-BP with modifications to the existing JAVA code. Key changes would likely include how the initial solution is determined, move neighborhood definitions, and move strategies.

5.2.1.2 Guillotinable Cuts

An essential requirement for implementing ATS-BP to solve the guillotinable cuts variation of this problem is to identify or develop a heuristic to replace the pcf-BL. Lodi et al. (1999-1) present heuristics from Berkey and Wang and two new heuristic that they developed for this class. They present the following heuristics that can be used in this class: Finite Best Strip (FBS), Finite First Fit (FFF), Floor Ceiling (FC), Knapsack Problem (KP) (Lodi et al. 1999-1)

However, all of these heuristics, as described, require an initial sort. The existing heuristics ignore the ordering flexibilities in this problem. Thus, the first effort to implementing ATS-BP for this class could be to modify one of these heuristics to take advantage of reordering.

5.2.2 3D orthogonal Packing

5.2.2.1 Extending the Symmetric Group to 3D.

If extending to three dimensions, the additional orientations of an item would each be represented with letters exactly *n* units apart. When rotations or items are allowed, the extensions of the *i** formula of Section 4.2 to 3D should be straight forward. The restrictions to have one and only one letter per item not in a one cycle still would apply.

5.2.2.2 Extending the objective function to 3D

This objective function has a logical extension to both 3D orthogonal packing and when packing arbitrarily shaped polygons. Li and Milenkovic (1995) implemented a potential energy objective to compress arbitrarily shaped items for the strip packing problem using linear programming. They applied this method to the garment industry with successful results.

For three dimensions, the heuristic would have to directly consider the third dimension and the dead space penalties would need to be reevaluated. While the penalties proportional to the square root work well for area-based dead space, the volumetric-based dead space might require penalties proportional to the cubic root.

5.3 PRACTICAL CONSIDERATIONS

There are several practical considerations that would need to be added to packing heuristics in order to yield usable solutions in some real environments.

Some of these constraints would need to be included within the search process to make certain moves forbidden, others might incorporate penalties that would allow traversal of neighborhoods with slightly infeasible loads.

5.3.1 Hazardous Materials

The first step concerned with hazardous materials would be to classify and index all such items. Once a hazardous item is packed the bin could carry an index that prohibits other noncompatible hazardous materials and build an ArrayList of currently packed hazardous items. Primitive data structures already exist in the Java objects for the ATS-BP, but no consideration for hazardous materials is present.

5.3.2 Weight , Balance and Load Stability

Weight adds a new feasibility dimension to the packing problem. Enforcing feasibility could be incorporated into the logic of moves before making the pcf-BL move. Another option would be to allow penalized infeasible moves.

Balancing the load so that the bin CG is within specified limits is one more dimension that could be added. The logic used for the CG computation based on homogenous items could be readily adapted to compute a real CG. With the pcf-BL, post-processing could be conducted to determine if shifting items up and/or to the right would aid in the balancing loads.

Load stability is normally required when packing problems move into three dimensions. The implementations of Bortfeldt and Gehring (1997, 1998) required 100% support from below and may have been so restrictive that they compromised the ability to obtain a portion of good answers. Some items may be

structurally sound and would require analysis of dynamic stability during worst case accelerations when not completely supported. In some 3D heuristics, techniques that improve the density of the packing solution also improve the load stability.

5.4 RESEARCH CONTRIBUTIONS

This dissertation documents the following primary contributions:

(1) a new efficient and effective one pass heuristic for the bin packing problem, the pcf-BL heuristic, that provided superior starting solutions for subsequent efforts. The quality of those solutions is documented in Chapter 3 and is further evidenced in the final results presented in Chapter 4.

(2) a new and superior method for the solution of the bin packing problem, the ATS-BP. The quality of the ATS-BP is documented in Chapter 4 where the ATS-BP is shown to significantly improve on the results of the best previous technique.

(3) a framework for future work was constructed in the description of a symmetric group representation of the solution to the 2D and 3D bin packing problems. This was presented in Chapters 2 and 4.

A corollary contribution, embodied in the ATS-BP, was the development of a new fine-grained objective function for the 2D bin packing problem based on the concept of potential energy. The extension of this idea to 3D bin packing problems is straight forward.

APPENDIX A

This section contains the results for the two test sets described earlier. The first three columns show the number of bins determined by improved lower bound of Dell'Amico, Martello, and Vigo (ILB), the perimeter contact factor bottom left (*pcf*-BL) heuristic results, and the ATS-BP respectively. The next two columns show the ratio of both the *pcf*-BL heuristic results and the ATS-BP results to the ILB. The last two columns show the time of the *pcf*-BL initial solution and the time of the ATS-BP solution achieve the number of bins shown in columns two/three.

A.1 CLASS I

Class I had items h_i and w_i uniformly random [1, 10] with bins W=H=10.

Berky and Wang Class 1 Size 20				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	pcf-BL/ILB	ATS-BP/ILB	Time	Best
Cl1 20-01	7	7	7	1.00	1.00	0.13	0.13
Cl1 20-02	5	5	5	1.00	1.00	0.14	0.14
Cl1 20-03	7	7	7	1.00	1.00	0.14	0.14
Cl1 20-04	5	5	5	1.00	1.00	0.17	0.17
Cl1 20-05	6	6	6	1.00	1.00	0.11	0.11
Cl1 20-06	8	9	9	1.13	1.13	0.14	0.14
Cl1 20-07	6	6	6	1.00	1.00	0.11	0.11
Cl1 20-08	6	6	6	1.00	1.00	0.14	0.14
Cl1 20-09	7	7	7	1.00	1.00	0.13	0.13
Cl1 20-10	7	8	8	1.14	1.14	0.14	0.14
Averages	**6.4**	**6.6**	**6.6**	**1.03**	**1.03**	**0.13**	**0.13**

Table A.1.1 Berkey and Wang Class I, Size 20, *pcf* BL and ATS-BP Results

Berky and Wang Class 1 Size 40				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	pcf-BL/ILB	ATS-BP/ILB	Time	Best
Cl1 40-01	9	10	9	1.11	1.00	0.17	38.14
Cl1 40-02	11	12	12	1.09	1.09	0.17	0.17
Cl1 40-03	15	15	15	1.00	1.00	0.22	0.22
Cl1 40-04	13	14	14	1.08	1.08	0.20	0.20
Cl1 40-05	14	15	15	1.07	1.07	0.20	0.20
Cl1 40-06	13	14	14	1.08	1.08	0.27	0.27
Cl1 40-07	11	12	11	1.09	1.00	0.19	1.14
Cl1 40-08	16	17	17	1.06	1.06	0.24	0.24
Cl1 40-09	11	11	11	1.00	1.00	0.19	0.25
Cl1 40-10	11	11	11	1.00	1.00	0.20	0.31
Averages	**12.4**	**13.1**	**12.9**	**1.06**	**1.04**	**0.20**	**4.11**

Table A.1.2 Berkey and Wang Class I, Size 40, *pcf* BL and ATS-BP Results.

Berky and Wang Class 1 Size 60				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	pcf-BL/ILB	ATS-BP/ILB	Time	Best
Cl1 60-01	21	22	22	1.05	1.05	0.31	0.31
Cl1 60-02	18	19	18	1.06	1.00	0.27	37.27
Cl1 60-03	19	21	21	1.11	1.11	0.28	0.28
Cl1 60-04	20	22	22	1.10	1.10	0.34	0.34
Cl1 60-05	17	17	17	1.00	1.00	0.33	0.33
Cl1 60-06	17	17	17	1.00	1.00	0.25	0.25
Cl1 60-07	15	16	15	1.07	1.00	0.30	1.66
Cl1 60-08	19	21	21	1.11	1.11	0.26	0.26
Cl1 60-09	18	19	18	1.06	1.00	0.31	1.74
Cl1 60-10	23	24	24	1.04	1.04	0.31	0.31
Averages	**18.7**	**19.8**	**19.5**	**1.06**	**1.04**	**0.30**	**4.27**

Table A.1.3 Berkey and Wang Class I, Size 60, *pcf* BL and ATS-BP Results.

Berky and Wang Class 1 Size 80				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	pcf-BL/ILB	ATS-BP/ILB	Time	Best
Cl1 80-01	23	24	24	1.04	1.04	0.33	0.33
Cl1 80-02	25	26	25	1.04	1.00	0.38	2.39
Cl1 80-03	24	27	27	1.13	1.13	0.42	0.42
Cl1 80-04	25	26	26	1.04	1.04	0.38	0.38
Cl1 80-05	24	26	26	1.08	1.08	0.59	0.59
Cl1 80-06	26	28	28	1.08	1.08	0.38	0.38
Cl1 80-07	28	31	31	1.11	1.11	0.49	0.49
Cl1 80-08	27	29	29	1.07	1.07	0.39	0.39
Cl1 80-09	28	30	29	1.07	1.04	0.34	3.95
Cl1 80-10	25	25	25	1.00	1.00	0.38	0.38
Averages	**25.5**	**27.2**	**27.0**	**1.07**	**1.06**	**0.41**	**0.97**

Table A.1.4 Berkey and Wang Class I, Size 80, *pcf* BL and ATS-BP Results.

Berky and Wang Class 1 Size 100				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	pcf-BL/ILB	ATS-BP/ILB	Time	Best
Cl1 100-01	28	28	28	1.00	1.00	0.41	0.41
Cl1 100-02	31	32	31	1.03	1.00	0.33	2.80
Cl1 100-03	27	28	28	1.04	1.04	0.42	0.42
Cl1 100-04	29	31	30	1.07	1.03	0.44	1.42
Cl1 100-05	31	31	31	1.00	1.00	0.41	0.41
Cl1 100-06	34	36	36	1.06	1.06	0.50	0.50
Cl1 100-07	28	29	28	1.04	1.00	0.39	4.28
Cl1 100-08	32	33	33	1.03	1.03	0.42	0.42
Cl1 100-09	30	31	31	1.03	1.03	0.64	0.64
Cl1 100-10	36	38	38	1.06	1.06	0.70	0.70
Averages	**30.6**	**31.7**	**31.4**	**1.04**	**1.03**	**0.47**	**1.20**

Table A.1.5 Berkey and Wang Class I, Size 100, *pcf* BL and ATS-BP Results.

A.2 CLASS II

Class II had items h_i and w_i uniformly random [1, 10] with bins W=H=30.

Berky and Wang Class 2 Size 20				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	pcf-BL/ILB	ATS-BP/ILB	Time	Best
CI2 20-01	1	1	1	1.00	1.00	0.13	0.13
CI2 20-02	1	1	1	1.00	1.00	0.16	0.16
CI2 20-03	1	1	1	1.00	1.00	0.11	0.11
CI2 20-04	1	1	1	1.00	1.00	0.13	0.13
CI2 20-05	1	1	1	1.00	1.00	0.13	0.13
CI2 20-06	1	1	1	1.00	1.00	0.13	0.13
CI2 20-07	1	1	1	1.00	1.00	0.14	0.14
CI2 20-08	1	1	1	1.00	1.00	0.14	0.14
CI2 20-09	1	1	1	1.00	1.00	0.14	0.14
CI2 20-10	1	1	1	1.00	1.00	0.13	0.13
Averages	**1.0**	**1.0**	**1.0**	**1.00**	**1.00**	**0.13**	**0.13**

Table A.2.1 Berkey and Wang Class II, Size 20, *pcf* BL and ATS-BP Results.

Berky and Wang Class 2 Size 40				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	pcf-BL/ILB	ATS-BP/ILB	Time	Best
CI2 40-01	1	2	1	2.00	1.00	0.20	12.72
CI2 40-02	2	2	2	1.00	1.00	0.19	0.19
CI2 40-03	2	2	2	1.00	1.00	0.19	0.19
CI2 40-04	2	2	2	1.00	1.00	0.19	0.19
CI2 40-05	2	2	2	1.00	1.00	0.20	0.20
CI2 40-06	2	2	2	1.00	1.00	0.22	0.22
CI2 40-07	2	2	2	1.00	1.00	0.19	0.19
CI2 40-08	2	2	2	1.00	1.00	0.20	0.20
CI2 40-09	2	2	2	1.00	1.00	0.24	0.24
CI2 40-10	2	2	2	1.00	1.00	0.27	0.27
Averages	**1.9**	**2.0**	**1.9**	**1.10**	**1.00**	**0.21**	**1.46**

Table A.2.2 Berkey and Wang Class II, Size 40, *pcf* BL and ATS-BP Results.

Berky and Wang Class 2 Size 60				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	pcf-BL/ILB	ATS-BP/ILB	Time	Best
CI2 60-01	3	3	3	1.00	1.00	0.24	0.24
CI2 60-02	2	3	2	1.50	1.00	0.30	8.84
CI2 60-03	3	3	3	1.00	1.00	0.31	0.31
CI2 60-04	3	3	3	1.00	1.00	0.30	0.30
CI2 60-05	2	2	2	1.00	1.00	0.22	0.22
CI2 60-06	2	2	2	1.00	1.00	0.30	0.30
CI2 60-07	2	2	2	1.00	1.00	0.31	0.31
CI2 60-08	3	3	3	1.00	1.00	0.34	0.34
CI2 60-09	2	2	2	1.00	1.00	0.20	0.20
CI2 60-10	3	3	3	1.00	1.00	0.30	0.30
Averages	**2.5**	**2.6**	**2.5**	**1.05**	**1.00**	**0.28**	**1.14**

Table A.2.3 Berkey and Wang Class II, Size 60, *pcf* BL and ATS-BP Results.

Berky and Wang Class 2 Size 80				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	pcf-BL/ILB	ATS-BP/ILB	Time	Best
CI2 80-01	3	3	3	1.00	1.00	0.33	0.33
CI2 80-02	3	3	3	1.00	1.00	0.34	0.34
CI2 80-03	3	3	3	1.00	1.00	0.41	0.41
CI2 80-04	3	3	3	1.00	1.00	0.38	0.38
CI2 80-05	3	3	3	1.00	1.00	0.44	0.44
CI2 80-06	3	3	3	1.00	1.00	0.30	0.30
CI2 80-07	3	4	3	1.33	1.00	0.36	198.89
CI2 80-08	3	4	3	1.33	1.00	0.34	24.30
CI2 80-09	4	4	4	1.00	1.00	0.33	0.33
CI2 80-10	**3**	**3**	**3**	1.00	1.00	0.39	0.39
Averages	**3.1**	**3.3**	**3.1**	**1.07**	**1.00**	**0.36**	**22.61**

Table A.2.4 Berkey and Wang Class II, Size 80, *pcf* BL and ATS-BP Results.

Berky and Wang Class 2 Size 100				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	pcf-BL/ILB	ATS-BP/ILB	Time	Best
CI2 100-01	4	4	4	1.00	1.00	0.50	0.64
CI2 100-02	4	4	4	1.00	1.00	0.47	0.56
CI2 100-03	3	4	3	1.33	1.00	0.41	36.95
CI2 100-04	4	4	4	1.00	1.00	0.45	0.66
CI2 100-05	4	4	4	1.00	1.00	0.41	0.61
CI2 100-06	4	4	4	1.00	1.00	0.44	0.56
CI2 100-07	4	4	4	1.00	1.00	0.42	0.64
CI2 100-08	4	4	4	1.00	1.00	0.44	0.67
CI2 100-09	4	4	4	1.00	1.00	0.41	0.64
CI2 100-10	4	4	4	1.00	1.00	0.47	0.53
Averages	**3.9**	**4.0**	**3.9**	**1.03**	**1.00**	**0.44**	**4.25**

Table A.2.5 Berkey and Wang Class II, Size 100, *pcf* BL and ATS-BP Results.

A.3 CLASS III

Class III had items h_i and w_i uniformly random [1, 35] with bins W=H=35.

Berky and Wang Class 3 Size 20				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	pcf-BL/ILB	ATS-BP/ILB	Time	Best
CI3 20-01	5	6	5	1.20	1.00	0.19	18.95
CI3 20-02	3	3	3	1.00	1.00	0.17	0.17
CI3 20-03	5	6	5	1.20	1.00	0.19	93.17
CI3 20-04	4	4	4	1.00	1.00	0.22	0.22
CI3 20-05	4	4	4	1.00	1.00	0.16	0.16
CI3 20-06	6	6	6	1.00	1.00	0.17	0.17
CI3 20-07	4	4	4	1.00	1.00	0.16	0.16
CI3 20-08	4	4	4	1.00	1.00	0.16	0.16
CI3 20-09	5	5	5	1.00	1.00	0.16	0.16
CI3 20-10	6	7	7	1.17	1.17	0.16	0.16
Averages	**4.6**	**4.9**	**4.7**	**1.06**	**1.02**	**0.17**	**11.35**

Table A.3.1 Berkey and Wang Class III, Size 20, *pcf* BL and ATS-BP Results.

Berky and Wang Class 3 Size 40				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	pcf-BL/ILB	ATS-BP/ILB	Time	Best
CI3 40-01	6	7	6	1.17	1.00	0.25	1.31
CI3 40-02	8	8	8	1.00	1.00	0.27	0.27
CI3 40-03	10	11	11	1.10	1.10	0.27	0.27
CI3 40-04	9	10	10	1.11	1.11	0.30	0.30
CI3 40-05	10	12	12	1.20	1.20	0.28	0.28
CI3 40-06	9	10	10	1.11	1.11	0.27	0.27
CI3 40-07	8	8	8	1.00	1.00	0.27	0.27
CI3 40-08	12	13	13	1.08	1.08	0.27	0.27
CI3 40-09	7	8	8	1.14	1.14	0.30	0.30
CI3 40-10	7	8	8	1.14	1.14	0.30	0.30
Averages	**8.6**	**9.5**	**9.4**	**1.11**	**1.09**	**0.28**	**0.38**

Table A.3.2 Berkey and Wang Class III, Size 40, *pcf* BL and ATS-BP Results.

Berky and Wang Class 3 Size 60				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	pcf-BL/ILB	ATS-BP/ILB	Time	Best
CI3 60-01	14	16	16	1.14	1.14	0.53	0.53
CI3 60-02	12	13	13	1.08	1.08	0.33	0.33
CI3 60-03	13	14	13	1.08	1.00	0.33	595.83
CI3 60-04	13	14	14	1.08	1.08	0.47	0.47
CI3 60-05	11	12	12	1.09	1.09	0.44	0.44
CI3 60-06	12	13	12	1.08	1.00	0.39	49.31
CI3 60-07	10	11	11	1.10	1.10	0.36	0.36
CI3 60-08	13	15	15	1.15	1.15	0.42	0.42
CI3 60-09	12	13	13	1.08	1.08	0.34	0.34
CI3 60-10	16	17	17	1.06	1.06	0.42	0.42
Averages	**12.6**	**13.8**	**13.6**	**1.10**	**1.08**	**0.40**	**64.85**

Table A.3.3 Berkey and Wang Class III, Size 60, *pcf* BL and ATS-BP Results.

Berky and Wang Class 3 Size 80				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	pcf-BL/ILB	ATS-BP/ILB	Time	Best
CI3 80-01	16	17	17	1.06	1.06	0.44	0.44
CI3 80-02	16	17	17	1.06	1.06	0.45	0.45
CI3 80-03	16	18	17	1.13	1.06	0.42	9.06
CI3 80-04	17	18	18	1.06	1.06	0.55	0.55
CI3 80-05	16	17	17	1.06	1.06	0.44	0.44
CI3 80-06	19	20	20	1.05	1.05	0.52	0.52
CI3 80-07	19	20	20	1.05	1.05	0.58	0.58
CI3 80-08	19	21	21	1.11	1.11	0.50	0.50
CI3 80-09	19	22	21	1.16	1.11	0.52	86.36
CI3 80-10	17	18	18	1.06	1.06	0.48	0.48
Averages	**17.4**	**18.8**	**18.6**	**1.08**	**1.07**	**0.49**	**9.94**

Table A.3.4 Berkey and Wang Class III, Size 80, *pcf* BL and ATS-BP Results.

Berky and Wang Class 3 Size 100				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	pcf-BL/ILB	ATS-BP/ILB	Time	Best
CI3 100-01	18	20	19	1.11	1.06	0.50	633.58
CI3 100-02	21	23	23	1.10	1.10	0.47	0.47
CI3 100-03	18	19	19	1.06	1.06	0.64	0.64
CI3 100-04	20	21	20	1.05	1.00	0.63	3.73
CI3 100-05	21	22	22	1.05	1.05	0.72	0.72
CI3 100-06	25	26	26	1.04	1.04	0.59	0.59
CI3 100-07	19	20	20	1.05	1.05	0.59	0.59
CI3 100-08	21	23	23	1.10	1.10	0.73	0.73
CI3 100-09	20	23	22	1.15	1.10	0.63	38.17
CI3 100-10	27	29	29	1.07	1.07	1.23	1.23
Averages	**21**	**22.6**	**22.3**	**1.08**	**1.06**	**0.67**	**68.05**

Table A.3.5 Berkey and Wang Class III, Size 100, *pcf* BL and ATS-BP Results.

A.4 CLASS IV

Class IV had items h_i and w_i uniformly random [1, 35] with bins W=H=100.

Berky and Wang Class 4 Size 20				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	pcf-BL/ILB	ATS-BP/ILB	Time	Best
Cl4 20-01	1	1	1	1.00	1.00	0.16	0.16
Cl4 20-02	1	1	1	1.00	1.00	0.14	0.14
Cl4 20-03	1	1	1	1.00	1.00	0.14	0.14
Cl4 20-04	1	1	1	1.00	1.00	0.13	0.13
Cl4 20-05	1	1	1	1.00	1.00	0.14	0.14
Cl4 20-06	1	1	1	1.00	1.00	0.14	0.14
Cl4 20-07	1	1	1	1.00	1.00	0.16	0.16
Cl4 20-08	1	1	1	1.00	1.00	0.13	0.13
Cl4 20-09	1	1	1	1.00	1.00	0.14	0.14
Cl4 20-10	1	1	1	1.00	1.00	0.13	0.13
Averages	**1.0**	**1.0**	**1.0**	**1.00**	**1.00**	**0.14**	**0.14**

Table A.4.1 Berkey and Wang Class IV, Size 20, *pcf* BL and ATS-BP Results.

Berky and Wang Class 4 Size 40				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	pcf-BL/ILB	ATS-BP/ILB	Time	Best
Cl4 40-01	1	1	1	1.00	1.00	0.19	0.19
Cl4 40-02	2	2	2	1.00	1.00	0.23	0.23
Cl4 40-03	2	2	2	1.00	1.00	0.27	0.27
Cl4 40-04	2	2	2	1.00	1.00	0.28	0.28
Cl4 40-05	2	2	2	1.00	1.00	0.28	0.28
Cl4 40-06	2	2	2	1.00	1.00	0.20	0.20
Cl4 40-07	2	2	2	1.00	1.00	0.25	0.25
Cl4 40-08	2	2	2	1.00	1.00	0.24	0.24
Cl4 40-09	2	2	2	1.00	1.00	0.28	0.28
Cl4 40-10	2	2	2	1.00	1.00	,204	,204
Averages	**1.9**	**1.9**	**1.9**	**1.00**	**1.00**	**0.25**	**0.25**

Table A.4.2 Berkey and Wang Class IV, Size 40, *pcf* BL and ATS-BP Results.

Berky and Wang Class 4 Size 60				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	pcf-BL/ILB	ATS-BP/ILB	Time	Best
CI4 60-01	3	3	3	1.00	1.00	0.31	0.31
CI4 60-02	2	2	2	1.00	1.00	0.31	0.31
CI4 60-03	2	3	3	1.50	1.50	0.30	0.30
CI4 60-04	2	3	2	1.50	1.00	0.34	285.25
CI4 60-05	2	2	2	1.00	1.00	0.34	0.34
CI4 60-06	2	2	2	1.00	1.00	0.28	0.28
CI4 60-07	2	2	2	1.00	1.00	0.31	0.31
CI4 60-08	3	3	3	1.00	1.00	0.33	0.33
CI4 60-09	2	2	2	1.00	1.00	0.30	0.30
CI4 60-10	3	3	3	1.00	1.00	0.28	0.28
Averages	**2.3**	**2.5**	**2.4**	**1.10**	**1.05**	**0.31**	**28.80**

Table A.4.3 Berkey and Wang Class IV, Size 60, *pcf* BL and ATS-BP Results.

Berky and Wang Class 4 Size 80				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	pcf-BL/ILB	ATS-BP/ILB	Time	Best
CI4 80-01	3	3	3	1.00	1.00	0.42	0.42
CI4 80-02	3	3	3	1.00	1.00	0.45	0.45
CI4 80-03	3	3	3	1.00	1.00	0.38	0.38
CI4 80-04	3	3	3	1.00	1.00	0.38	0.38
CI4 80-05	3	3	3	1.00	1.00	0.41	0.41
CI4 80-06	3	3	3	1.00	1.00	0.34	0.34
CI4 80-07	3	4	4	1.33	1.33	0.34	0.34
CI4 80-08	3	4	3	1.33	1.00	0.45	357.77
CI4 80-09	3	4	4	1.33	1.33	0.56	0.56
CI4 80-10	3	3	3	1.00	1.00	0.53	0.53
Averages	**3.0**	**3.3**		**1.10**	**1.07**	**0.43**	**36.16**

Table A.4.4 Berkey and Wang Class IV, Size 80, *pcf* BL and ATS-BP Results.

Berky and Wang Class 4 Size 100				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	pcf-BL/ILB	ATS-BP/ILB	Time	Best
CI4 100-01	3	4	3	1.33	1.00	0.53	86.83
CI4 100-02	4	4	4	1.00	1.00	0.55	0.55
CI4 100-03	3	3	3	1.00	1.00	0.59	0.59
CI4 100-04	4	4	4	1.00	1.00	0.64	0.64
CI4 100-05	4	4	4	1.00	1.00	0.70	0.70
CI4 100-06	4	4	4	1.00	1.00	0.63	0.63
CI4 100-07	3	4	4	1.33	1.33	0.47	0.47
CI4 100-08	4	4	4	1.00	1.00	0.55	0.55
CI4 100-09	4	4	4	1.00	1.00	0.52	0.52
CI4 100-10	4	4	4	1.00	1.00	0.47	0.47
Averages	**3.7**	**3.9**	**3.8**	**1.07**	**1.03**	**0.56**	**9.19**

Table A.4.5 Berkey and Wang Class IV, Size 100, *pcf* BL and ATS-BP Results.

A.5 CLASS V

Class V had items h_i and w_i uniformly random [1, 100] with bins W=H=100.

Berky and Wang Class 5 Size 20				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	pcf-BL/ILB	ATS-BP/ILB	Time	Best
CI5 20-01	7	7	7	1.00	1.00	0.17	0.17
CI5 20-02	4	4	4	1.00	1.00	0.14	0.14
CI5 20-03	6	7	7	1.17	1.17	0.17	0.17
CI5 20-04	4	5	5	1.25	1.25	0.16	0.16
CI5 20-05	5	5	5	1.00	1.00	0.22	0.22
CI5 20-06	8	8	8	1.00	1.00	0.20	0.20
CI5 20-07	5	5	5	1.00	1.00	0.17	0.17
CI5 20-08	5	5	5	1.00	1.00	0.22	0.22
CI5 20-09	6	7	6	1.17	1.00	0.19	11.53
CI5 20-10	7	7	7	1.00	1.00	0.16	0.16
Averages	**5.7**	**6.0**	**5.9**	**1.06**	**1.04**	**0.18**	**1.31**

Table A.5.1 Berkey and Wang Class V, Size 20, *pcf* BL and ATS-BP Results.

Berky and Wang Class 5 Size 40				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	pcf-BL/ILB	ATS-BP/ILB	Time	Best
CI5 40-01	7	8	8	1.14	1.14	0.25	0.25
CI5 40-02	9	10	10	1.11	1.11	0.30	0.30
CI5 40-03	14	14	14	1.00	1.00	0.27	0.27
CI5 40-04	11	13	12	1.18	1.09	0.30	225.16
CI5 40-05	12	13	13	1.08	1.08	0.25	0.25
CI5 40-06	12	12	12	1.00	1.00	0.25	0.25
CI5 40-07	10	10	10	1.00	1.00	0.25	0.25
CI5 40-08	15	16	16	1.07	1.07	0.25	0.25
CI5 40-09	9	10	9	1.11	1.00	0.30	184.80
CI5 40-10	9	10	10	1.11	1.11	0.30	0.30
Averages	**10.8**	**11.6**	**11.4**	**1.08**	**1.06**	**0.27**	**41.21**

Table A.5.2 Berkey and Wang Class V, Size 40, *pcf* BL and ATS-BP Results.

Berky and Wang Class 5 Size 60				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	pcf-BL/ILB	ATS-BP/ILB	Time	Best
CI5 60-01	18	21	20	1.17	1.11	0.42	56.16
CI5 60-02	16	17	16	1.06	1.00	0.42	1.92
CI5 60-03	16	19	18	1.19	1.13	0.41	2.70
CI5 60-04	18	19	19	1.06	1.06	0.45	0.45
CI5 60-05	14	16	15	1.14	1.07	0.39	66.78
CI5 60-06	14	16	15	1.14	1.07	0.41	611.57
CI5 60-07	13	14	14	1.08	1.08	0.34	0.34
CI5 60-08	18	20	19	1.11	1.06	0.39	1.52
CI5 60-09	15	17	16	1.13	1.07	0.41	3.97
CI5 60-10	23	23	23	1.00	1.00	0.52	0.52
Averages	**16.5**	**18.2**	**17.5**	**1.11**	**1.06**	**0.42**	**74.59**

Table A.5.3 Berkey and Wang Class V, Size 60, *pcf* BL and ATS-BP Results.

Berky and Wang Class 5 Size 80				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	pcf-BL/ILB	ATS-BP/ILB	Time	Best
CI5 80-01	22	23	22	1.05	1.00	0.53	4.08
CI5 80-02	21	23	22	1.10	1.05	0.52	4.11
CI5 80-03	21	24	24	1.14	1.14	0.59	0.59
CI5 80-04	22	23	23	1.05	1.05	0.50	0.50
CI5 80-05	21	22	22	1.05	1.05	0.50	0.50
CI5 80-06	24	26	25	1.08	1.04	0.56	6.03
CI5 80-07	25	27	26	1.08	1.04	0.55	75.08
CI5 80-08	24	26	26	1.08	1.08	0.53	0.53
CI5 80-09	24	27	27	1.13	1.13	0.53	0.53
CI5 80-10	21	23	22	1.10	1.05	0.53	1338.17
Averages	**22.5**	**24.4**	**23.9**	**1.08**	**1.06**	**0.53**	**143.01**

Table A.5.4 Berkey and Wang Class V, Size 80, *pcf* BL and ATS-BP Results.

Berky and Wang Class 5 Size 100				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	pcf-BL/ILB	ATS-BP/ILB	Time	Best
CI5 100-01	23	25	24	1.09	1.04	0.59	175.25
CI5 100-02	27	29	28	1.07	1.04	0.75	883.18
CI5 100-03	23	24	24	1.04	1.04	0.75	0.75
CI5 100-04	25	26	26	1.04	1.04	0.84	0.84
CI5 100-05	26	29	28	1.12	1.08	1.17	198.94
CI5 100-06	32	33	33	1.03	1.03	1.72	1.72
CI5 100-07	24	26	25	1.08	1.04	0.77	692.41
CI5 100-08	27	30	30	1.11	1.11	1.80	1.80
CI5 100-09	25	28	27	1.12	1.08	0.56	5.38
CI5 100-10	34	35	35	1.03	1.03	2.00	2.00
Averages	**26.6**	**28.5**	**28.0**	**1.07**	**1.05**	**1.10**	**196.23**

Table A.5.5 Berkey and Wang Class V, Size 100, *pcf* BL and ATS-BP Results.

A.6 CLASS VI

Class VI had items h_i and w_i uniformly random [1, 100] with bins W=H=300.

Berky and Wang Class 6 Size 20				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	pcf-BL/ILB	ATS-BP/ILB	Time	Best
CI6 20-01	1	1	1	1.00	1.00	0.13	0.13
CI6 20-02	1	1	1	1.00	1.00	0.22	0.22
CI6 20-03	1	1	1	1.00	1.00	0.17	0.17
CI6 20-04	1	1	1	1.00	1.00	0.20	0.20
CI6 20-05	1	1	1	1.00	1.00	0.19	0.19
CI6 20-06	1	1	1	1.00	1.00	0.19	0.19
CI6 20-07	1	1	1	1.00	1.00	0.14	0.14
CI6 20-08	1	1	1	1.00	1.00	0.14	0.14
CI6 20-09	1	1	1	1.00	1.00	0.19	0.19
CI6 20-10	1	1	1	1.00	1.00	0.14	0.14
Averages	**1.0**	**1.0**	**1.0**	**1.00**	**1.00**	**0.17**	**0.17**

Table A.6.1 Berkey and Wang Class VI, Size 20, *pcf* BL and ATS-BP Results.

Berky and Wang Class 6 Size 40				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	pcf-BL/ILB	ATS-BP/ILB	Time	Best
CI6 40-01	1	1	1	1.00	1.00	0.30	0.30
CI6 40-02	1	2	2	2.00	2.00	0.30	0.30
CI6 40-03	2	2	2	1.00	1.00	0.25	0.25
CI6 40-04	2	2	2	1.00	1.00	0.22	0.22
CI6 40-05	2	2	2	1.00	1.00	0.36	0.36
CI6 40-06	1	2	1	2.00	1.00	0.20	25.32
CI6 40-07	2	2	2	1.00	1.00	0.27	0.27
CI6 40-08	2	2	2	1.00	1.00	0.33	0.33
CI6 40-09	1	2	1	2.00	1.00	0.27	13.48
CI6 40-10	1	2	1	2.00	1.00	0.19	18.84
Averages	**1.5**	**1.9**	**1.6**	**1.40**	**1.10**	**0.27**	**5.97**

Table A.6.2 Berkey and Wang Class VI, Size 40, *pcf* BL and ATS-BP Results.

Berky and Wang Class 6 Size 60				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	pcf-BL/ILB	ATS-BP/ILB	Time	Best
CI6 60-01	2	3	2	1.50	1.00	0.31	256.22
CI6 60-02	2	2	2	1.00	1.00	0.25	0.25
CI6 60-03	2	2	2	1.00	1.00	0.33	0.33
CI6 60-04	2	2	2	1.00	1.00	0.27	0.27
CI6 60-05	2	2	2	1.00	1.00	0.31	0.31
CI6 60-06	2	2	2	1.00	1.00	0.30	0.30
CI6 60-07	2	2	2	1.00	1.00	0.33	0.33
CI6 60-08	2	2	2	1.00	1.00	0.38	0.38
CI6 60-09	2	2	2	1.00	1.00	0.39	0.39
CI6 60-10	3	3	3	1.00	1.00	0.41	0.41
Averages	**2.1**	**2.2**	**2.1**	**1.05**	**1.00**	**0.33**	**25.92**

Table A.6.3 Berkey and Wang Class VI, Size 60, *pcf* BL and ATS-BP Results.

Berky and Wang Class 6 Size 80				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	pcf-BL/ILB	ATS-BP/ILB	Time	Best
CI6 80-01	3	3	3	1.00	1.00	0.44	0.44
CI6 80-02	3	3	3	1.00	1.00	0.49	0.49
CI6 80-03	3	3	3	1.00	1.00	0.41	0.41
CI6 80-04	3	3	3	1.00	1.00	0.41	0.41
CI6 80-05	3	3	3	1.00	1.00	0.39	0.39
CI6 80-06	3	3	3	1.00	1.00	0.34	0.34
CI6 80-07	3	3	3	1.00	1.00	0.38	0.38
CI6 80-08	3	3	3	1.00	1.00	0.44	0.44
CI6 80-09	3	3	3	1.00	1.00	0.44	0.44
CI6 80-10	3	3	3	1.00	1.00	0.45	0.45
Averages	**3.0**	**3.0**	**3.0**	**1.00**	**1.00**	**0.42**	**0.42**

Table A.6.4 Berkey and Wang Class VI, Size 80, *pcf* BL and ATS-BP Results.

Berky and Wang Class 6 Size 100				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	pcf-BL/ILB	ATS-BP/ILB	Time	Best
CI6 100-01	3	3	3	1.00	1.00	0.47	0.47
CI6 100-02	3	4	4	1.33	1.33	0.52	0.52
CI6 100-03	3	3	3	1.00	1.00	0.52	0.52
CI6 100-04	3	3	3	1.00	1.00	0.45	0.45
CI6 100-05	3	4	3	1.33	1.00	0.53	72.11
CI6 100-06	4	4	4	1.00	1.00	0.50	0.50
CI6 100-07	3	3	3	1.00	1.00	0.48	0.48
CI6 100-08	3	4	4	1.33	1.33	0.45	0.45
CI6 100-09	3	3	3	1.00	1.00	0.42	0.42
CI6 100-10	4	4	4	1.00	1.00	0.48	0.48
Averages	**3.2**	**3.5**	**3.4**	**1.10**	**1.07**	**0.48**	**7.64**

Table A.6.5 Berkey and Wang Class VI, Size 100, *pcf* BL and ATS-BP Results.

A.7 CLASS VII

Type 1: w_j uniformly random in [2/3W, W] and h_j uniformly random in [1,1/2H]

Type 2: w_j uniformly random in [1, 1/2W] and h_j uniformly random in [2/3H,H]

Type 3: w_j uniformly random in [1/2W, W] and h_j uniformly random in [1/2H,H]

Type 4 w_j uniformly random in [1, 1/2W] and h_j uniformly random in [1,1/2H]

Class VII: *Type 1* with probability 70%, *Type* 2, 3, 4 with probability 10% each.

Martello and Vigo Class 7 Size 20				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	pcf-BL/ILB	ATS-BP/ILB	Time	Best
CI7 20-01	5	6	5	1.20	1.00	0.16	5.47
CI7 20-02	4	5	5	1.25	1.25	0.17	0.17
CI7 20-03	4	5	5	1.25	1.25	0.13	0.13
CI7 20-04	6	6	6	1.00	1.00	0.14	0.14
CI7 20-05	5	6	6	1.20	1.20	0.14	0.14
CI7 20-06	5	6	5	1.20	1.00	0.16	1.06
CI7 20-07	4	4	4	1.00	1.00	0.14	0.14
CI7 20-08	5	6	6	1.20	1.20	0.16	0.16
CI7 20-09	5	6	6	1.20	1.20	0.14	0.14
CI7 20-10	4	4	4	1.00	1.00	0.17	0.17
Averages	**4.7**	**5.4**	**5.2**	**1.15**	**1.11**	**0.15**	**0.77**

Table A.7.1 Martello And Vigo Class VII Size 20 *pcf* BL and ATS-BP results

Martello and Vigo Class 7 Size 40				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	pcf-BL/ILB	ATS-BP/ILB	Time	Best
CI7 40-01	9	10	9	1.11	1.00	0.30	229.17
CI7 40-02	11	12	12	1.09	1.09	0.36	0.24
CI7 40-03	9	10	9	1.11	1.00	0.28	122.11
CI7 40-04	12	13	13	1.08	1.08	0.25	0.25
CI7 40-05	8	9	9	1.13	1.13	0.24	0.24
CI7 40-06	10	11	11	1.10	1.10	0.25	0.25
CI7 40-07	10	11	10	1.10	1.00	0.25	623.72
CI7 40-08	10	12	11	1.20	1.10	0.25	16.27
CI7 40-09	7	8	8	1.14	1.14	0.30	0.30
CI7 40-10	11	12	12	1.09	1.09	0.25	0.25
Averages	**9.7**	**10.8**	**10.4**	**1.12**	**1.07**	**0.27**	**99.28**

Table A.7.2 Martello And Vigo Class VII Size 40 *pcf* BL and ATS-BP results

Martello and Vigo Class 7 Size 60				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	pcf-BL/ILB	ATS-BP/ILB	Time	Best
CI7 60-01	15	17	16	1.13	1.07	0.56	711.05
CI7 60-02	13	14	13	1.08	1.00	0.30	790.08
CI7 60-03	15	16	15	1.07	1.00	0.44	3317.80
CI7 60-04	13	15	14	1.15	1.08	0.31	2.75
CI7 60-05	13	14	13	1.08	1.00	0.41	113.44
CI7 60-06	13	14	13	1.08	1.00	0.30	269.40
CI7 60-07	13	15	14	1.15	1.08	0.45	0.86
CI7 60-08	15	17	16	1.13	1.07	0.34	60.92
CI7 60-09	13	14	14	1.08	1.08	0.36	0.36
CI7 60-10	17	19	18	1.12	1.06	0.33	871.70
Averages	**14.0**	**15.5**	**14.6**	**1.11**	**1.04**	**0.38**	**613.83**

Table A.7.3 Martello And Vigo Class VII Size 60 *pcf* BL and ATS-BP results

Martello and Vigo Class 7 Size 80				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	pcf-BL/ILB	ATS-BP/ILB	Time	Best
CI7 80-01	18	20	20	1.11	1.11	0.56	0.56
CI7 80-02	21	24	23	1.14	1.10	0.61	586.84
CI7 80-03	18	20	19	1.11	1.06	0.59	142.73
CI7 80-04	19	21	21	1.11	1.11	0.50	0.50
CI7 80-05	20	22	21	1.10	1.05	0.41	1021.66
CI7 80-06	20	22	21	1.10	1.05	0.97	173.67
CI7 80-07	21	24	23	1.14	1.10	0.59	220.38
CI7 80-08	19	21	21	1.11	1.11	0.55	0.55
CI7 80-09	20	22	22	1.10	1.10	0.55	0.55
CI7 80-10	21	23	22	1.10	1.05	0.63	450.34
Averages	**19.7**	**21.9**	**21.3**	**1.11**	**1.08**	**0.60**	**259.78**

Table A.7.4 Martello And Vigo Class VII Size 80 *pcf* BL and ATS-BP results

Martello and Vigo Class 7 Size 100				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	pcf-BL/ILB	ATS-BP/ILB	Time	Best
CI7 100-01	24	26	26	1.08	1.08	0.48	0.48
CI7 100-02	24	26	25	1.08	1.04	0.61	148.58
CI7 100-03	21	22	22	1.05	1.05	0.55	0.55
CI7 100-04	23	25	24	1.09	1.04	0.56	526.70
CI7 100-05	22	24	23	1.09	1.05	0.63	117.80
CI7 100-06	25	29	28	1.16	1.12	0.55	158.31
CI7 100-07	23	25	25	1.09	1.09	0.61	0.61
CI7 100-08	25	28	27	1.12	1.08	0.58	30.45
CI7 100-09	23	25	24	1.09	1.04	0.63	1332.47
CI7 100-10	28	31	31	1.11	1.11	0.64	0.64
Averages	**23.8**	**26.1**	**25.5**	**1.10**	**1.07**	**0.58**	**231.66**

Table A.7.5 Martello And Vigo Class VII, Size 100, *pcf* BL and ATS-BP results

A.8 CLASS VIII

Class VIII: *Type 2* with probability 70%, *Type* 1, 3, 4 with probability 10% each.

Martello and Vigo Class 8 Size 20				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	pcf-BL/ILB	ATS-BP/ILB	Time	Best
CI8 20-01	5	6	6	1.20	1.20	0.17	0.17
CI8 20-02	5	6	6	1.20	1.20	0.16	0.16
CI8 20-03	5	6	5	1.20	1.00	0.17	11.78
CI8 20-04	5	6	6	1.20	1.20	0.20	0.20
CI8 20-05	5	5	5	1.00	1.00	0.16	0.16
CI8 20-06	5	6	6	1.20	1.20	0.19	0.19
CI8 20-07	4	4	4	1.00	1.00	0.19	0.19
CI8 20-08	5	5	5	1.00	1.00	0.17	0.17
CI8 20-09	5	6	6	1.20	1.20	0.22	0.22
CI8 20-10	4	4	4	1.00	1.00	0.17	0.17
Averages	**4.8**	**5.4**	**5.3**	**1.12**	**1.10**	**0.18**	**1.34**

Table A.8.1 Martello And Vigo Class VIII Size 20 *pcf* BL and ATS-BP results

Martello and Vigo Class 8 Size 40				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	pcf-BL/ILB	ATS-BP/ILB	Time	Best
CI8 40-01	10	11	11	1.10	1.10	0.25	0.25
CI8 40-02	11	12	12	1.09	1.09	0.30	0.30
CI8 40-03	9	10	10	1.11	1.11	0.30	0.30
CI8 40-04	10	11	11	1.10	1.10	0.28	0.28
CI8 40-05	8	9	8	1.13	1.00	0.30	13.22
CI8 40-06	10	11	11	1.10	1.10	0.28	0.28
CI8 40-07	9	11	10	1.22	1.11	0.27	11.61
CI8 40-08	10	11	10	1.10	1.00	0.31	752.36
CI8 40-09	8	9	9	1.13	1.13	0.28	0.28
CI8 40-10	11	12	12	1.09	1.09	0.27	0.27
Averages	**9.6**	**10.7**	**10.4**	**1.12**	**1.08**	**0.28**	**77.91**

Table A.8.2 Martello And Vigo Class VIII Size 40 *pcf* BL and ATS-BP results

Martello and Vigo Class 8 Size 60				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	pcf-BL/ILB	ATS-BP/ILB	Time	Best
CI8 60-01	15	17	16	1.13	1.07	0.44	105.86
CI8 60-02	15	16	16	1.07	1.07	0.34	0.34
CI8 60-03	15	16	16	1.07	1.07	0.52	0.52
CI8 60-04	13	14	14	1.08	1.08	0.39	0.39
CI8 60-05	13	14	13	1.08	1.00	0.44	85.19
CI8 60-06	13	15	14	1.15	1.08	0.47	203.53
CI8 60-07	13	14	14	1.08	1.08	0.55	0.55
CI8 60-08	15	17	16	1.13	1.07	0.45	120.03
CI8 60-09	14	15	15	1.07	1.07	0.47	0.47
CI8 60-10	15	17	17	1.13	1.13	0.56	0.56
Averages	**14.1**	**15.5**	**15.1**	**1.10**	**1.07**	**0.46**	**51.74**

Table A.8.3 Martello And Vigo Class VIII Size 60 *pcf* BL and ATS-BP results

Martello and Vigo Class 8 Size 80				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	pcf-BL/ILB	ATS-BP/ILB	Time	Best
CI8 80-01	19	21	20	1.11	1.05	0.63	6.56
CI8 80-02	20	21	21	1.05	1.05	0.50	0.50
CI8 80-03	18	19	19	1.06	1.06	0.52	0.52
CI8 80-04	18	19	19	1.06	1.06	0.42	0.42
CI8 80-05	22	25	25	1.14	1.14	0.72	0.72
CI8 80-06	19	21	20	1.11	1.05	0.59	982.67
CI8 80-07	19	21	20	1.11	1.05	0.48	2428.94
CI8 80-08	20	22	21	1.10	1.05	0.52	12.13
CI8 80-09	19	21	20	1.11	1.05	0.47	1777.06
CI8 80-10	21	24	23	1.14	1.10	0.49	84.47
Averages	**19.5**	**21.4**	**20.8**	**1.10**	**1.07**	**0.53**	**529.40**

Table A.8.4 Martello And Vigo Class VIII Size 80 *pcf* BL and ATS-BP results

Martello and Vigo Class 8 Size 100				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	pcf-BL/ILB	ATS-BP/ILB	Time	Best
CI8 100-01	23	25	24	1.09	1.04	0.69	2229.53
CI8 100-02	23	25	25	1.09	1.09	0.56	0.56
CI8 100-03	21	23	22	1.10	1.05	0.64	80.08
CI8 100-04	26	29	29	1.12	1.12	0.64	0.64
CI8 100-05	25	27	27	1.08	1.08	0.67	0.67
CI8 100-06	23	25	25	1.09	1.09	0.59	0.59
CI8 100-07	23	25	25	1.09	1.09	0.56	0.56
CI8 100-08	25	28	27	1.12	1.08	0.67	1.58
CI8 100-09	24	26	26	1.08	1.08	0.72	0.72
CI8 100-10	28	31	30	1.11	1.07	0.59	13.88
Averages	**24.1**	**26.4**	**26.0**	**1.09**	**1.08**	**0.63**	**232.88**

Table A.8.5 Martello And Vigo Class VIII, Size 100, *pcf* BL and ATS-BP results

A.9 CLASS IX

Class IX: *Type 3* with probability 70%, *Type* 1, 2, 4 with probability 10% each.

Martello and Vigo Class 9 Size 20				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	pcf-BL/ILB	ATS-BP/ILB	Time	Best
CI9 20-01	19	19	19	1.00	1.00	0.16	0.16
CI9 20-02	13	13	13	1.00	1.00	0.23	0.23
CI9 20-03	14	14	14	1.00	1.00	0.22	0.22
CI9 20-04	16	16	16	1.00	1.00	0.24	0.24
CI9 20-05	16	16	16	1.00	1.00	0.19	0.19
CI9 20-06	14	15	14	1.07	1.00	0.17	0.89
CI9 20-07	9	9	9	1.00	1.00	0.17	0.17
CI9 20-08	14	14	14	1.00	1.00	0.17	0.17
CI9 20-09	15	15	15	1.00	1.00	0.14	0.14
CI9 20-10	13	13	13	1.00	1.00	0.22	0.22
Averages	**14.3**	**14.4**	**14.3**	**1.01**	**1.00**	**0.19**	**0.26**

Table A.9.1 Martello And Vigo Class IX Size 20 *pcf* BL and ATS-BP results

Martello and Vigo Class 9 Size 40				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	pcf-BL/ILB	ATS-BP/ILB	Time	Best
CI9 40-01	24	25	24	1.04	1.00	0.30	3.17
CI9 40-02	32	32	32	1.00	1.00	0.30	0.30
CI9 40-03	28	29	28	1.04	1.00	0.28	6.88
CI9 40-04	30	31	31	1.03	1.03	0.28	0.28
CI9 40-05	27	27	27	1.00	1.00	0.33	0.33
CI9 40-06	28	30	30	1.07	1.07	0.27	0.27
CI9 40-07	24	24	24	1.00	1.00	0.30	0.30
CI9 40-08	25	26	26	1.04	1.04	0.31	0.31
CI9 40-09	21	21	21	1.00	1.00	0.33	0.33
CI9 40-10	33	33	33	1.00	1.00	0.33	0.33
Averages	**27.2**	**27.8**	**27.6**	**1.02**	**1.01**	**0.30**	**1.25**

Table A.9.2 Martello And Vigo Class IX Size 40 *pcf* BL and ATS-BP results

Martello and Vigo Class 9 Size 60				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	pcf-BL/ILB	ATS-BP/ILB	Time	Best
CI9 60-01	45	46	46	1.02	1.02	0.39	0.39
CI9 60-02	43	45	44	1.05	1.02	0.53	20.75
CI9 60-03	46	46	46	1.00	1.00	0.47	0.47
CI9 60-04	43	44	44	1.02	1.02	0.56	0.56
CI9 60-05	41	41	41	1.00	1.00	0.63	0.63
CI9 60-06	37	37	37	1.00	1.00	0.45	0.45
CI9 60-07	40	40	40	1.00	1.00	0.50	0.50
CI9 60-08	47	47	47	1.00	1.00	0.61	0.61
CI9 60-09	45	45	45	1.00	1.00	0.61	0.61
CI9 60-10	45	45	45	1.00	1.00	0.44	0.44
Averages	**43.2**	**43.6**	**43.5**	**1.01**	**1.01**	**0.52**	**2.54**

Table A.9.3 Martello And Vigo Class IX Size 60 *pcf* BL and ATS-BP results

Martello and Vigo Class 9 Size 80				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	pcf-BL/ILB	ATS-BP/ILB	Time	Best
CI9 80-01	58	58	58	1.00	1.00	0.67	0.67
CI9 80-02	56	57	57	1.02	1.02	0.73	0.73
CI9 80-03	56	58	57	1.04	1.02	0.77	10.69
CI9 80-04	52	52	52	1.00	1.00	0.59	0.59
CI9 80-05	61	61	61	1.00	1.00	0.56	0.56
CI9 80-06	62	62	62	1.00	1.00	0.78	0.78
CI9 80-07	59	59	59	1.00	1.00	0.58	0.58
CI9 80-08	57	58	58	1.02	1.02	0.74	0.74
CI9 80-09	48	49	49	1.02	1.02	0.64	0.64
CI9 80-10	59	60	60	1.02	1.02	0.47	0.47
Averages	**56.8**	**57.4**	**57.3**	**1.01**	**1.01**	**0.65**	**1.65**

Table A.9.4 Martello And Vigo Class IX Size 80 *pcf* BL and ATS-BP results

Martello and Vigo Class 9 Size 100				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	pcf-BL/ILB	ATS-BP/ILB	Time	Best
CI9 100-01	71	71	71	1.00	1.00	1.77	1.77
CI9 100-02	63	64	64	1.02	1.02	1.55	1.55
CI9 100-03	68	68	68	1.00	1.00	1.20	1.20
CI9 100-04	77	78	78	1.01	1.01	1.34	1.34
CI9 100-05	64	65	65	1.02	1.02	1.20	1.20
CI9 100-06	71	71	71	1.00	1.00	1.70	1.70
CI9 100-07	66	66	66	1.00	1.00	1.30	1.30
CI9 100-08	72	73	73	1.01	1.01	1.23	1.23
CI9 100-09	65	66	65	1.02	1.00	1.34	514.34
CI9 100-10	71	72	72	1.01	1.01	1.28	1.28
Averages	**68.8**	**69.4**	**69.3**	**1.01**	**1.01**	**1.39**	**52.69**

Table A.9.5 Martello And Vigo Class IX, Size 100, *pcf* BL and ATS-BP results

A.10 CLASS X

Class X: *Type 4* with probability 70%, *Type* 1, 2, 3 with probability 10% each.

Martello and Vigo Class 10 Size 20				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	pcf-BL/ILB	ATS-BP/ILB	Time	Best
CI10 20-01	6	6	6	1.00	1.00	0.17	0.17
CI10 20-02	3	3	3	1.00	1.00	0.17	0.17
CI10 20-03	4	5	4	1.25	1.00	0.19	1.97
CI10 20-04	4	4	4	1.00	1.00	0.17	0.17
CI10 20-05	4	4	4	1.00	1.00	0.16	0.16
CI10 20-06	4	4	4	1.00	1.00	0.19	0.19
CI10 20-07	4	5	5	1.25	1.25	0.23	0.23
CI10 20-08	2	3	3	1.50	1.50	0.22	0.22
CI10 20-09	5	5	5	1.00	1.00	0.19	0.19
CI10 20-10	2	3	3	1.50	1.50	0.14	0.14
Averages	**3.8**	**4.2**	**4.1**	**1.15**	**1.12**	**0.18**	**0.36**

Table A.10.1 Martello And Vigo Class X Size 20 *pcf* BL and ATS-BP results

Martello and Vigo Class 10 Size 40				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	IBL/ILB	ATS-BP/ILB	Time	Best
CI10 40-01	8	8	8	1.00	1.00	0.28	0.28
CI10 40-02	7	8	8	1.14	1.14	0.27	0.27
CI10 40-03	8	9	9	1.13	1.13	0.31	0.31
CI10 40-04	6	7	6	1.17	1.00	0.30	36.83
CI10 40-05	6	6	6	1.00	1.00	0.28	0.28
CI10 40-06	5	6	6	1.20	1.20	0.22	0.22
CI10 40-07	7	7	7	1.00	1.00	0.25	0.25
CI10 40-08	7	7	7	1.00	1.00	0.30	0.30
CI10 40-09	7	8	8	1.14	1.14	0.31	0.31
CI10 40-10	8	9	8	1.13	1.00	0.25	0.25
Averages	**6.9**	**7.5**	**7.3**	**1.09**	**1.06**	**0.28**	**3.93**

Table A.10.2 Martello And Vigo Class X Size 40 *pcf* BL and ATS-BP results

Martello and Vigo Class 10 Size 60				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	IBL/ILB	ATS-BP/ILB	Time	Best
CI10 60-01	11	12	11	1.09	1.00	0.44	1517.48
CI10 60-02	11	13	12	1.18	1.09	0.36	5.00
CI10 60-03	11	12	11	1.09	1.00	0.38	824.42
CI10 60-04	7	8	8	1.14	1.14	0.34	0.34
CI10 60-05	8	8	8	1.00	1.00	0.45	0.45
CI10 60-06	12	12	12	1.00	1.00	0.41	0.41
CI10 60-07	9	10	10	1.11	1.11	0.56	0.56
CI10 60-08	9	10	10	1.11	1.11	0.38	0.38
CI10 60-09	8	9	9	1.13	1.13	0.36	0.36
CI10 60-10	8	9	8	1.13	1.00	0.38	2.31
Averages	**9.4**	**10.3**	**9.9**	**1.10**	**1.06**	**0.40**	**235.17**

Table A.10.3 Martello And Vigo Class X Size 60 *pcf* BL and ATS-BP results

Martello and Vigo Class 10 Size 80				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	IBL/ILB	ATS-BP/ILB	Time	Best
CI10 80-01	12	13	13	1.08	1.08	0.59	0.59
CI10 80-02	10	11	11	1.10	1.10	0.63	0.63
CI10 80-03	11	11	11	1.00	1.00	0.53	0.53
CI10 80-04	13	14	14	1.08	1.08	0.56	0.56
CI10 80-05	13	14	14	1.08	1.08	0.58	0.58
CI10 80-06	13	13	13	1.00	1.00	0.66	0.66
CI10 80-07	14	15	14	1.07	1.00	0.59	777.45
CI10 80-08	10	10	10	1.00	1.00	0.49	0.49
CI10 80-09	12	13	13	1.08	1.08	0.56	0.56
CI10 80-10	14	15	15	1.07	1.07	0.63	0.63
Averages	**12.2**	**12.9**	**12.8**	**1.06**	**1.05**	**0.58**	**78.27**

Table A.10.4 Martello And Vigo Class X Size 80 *pcf* BL and ATS-BP results

Martello and Vigo Class 10 Size 100				Ratio	Ratio	Initial	Time to
Instance	ILB	pcf-BL	ATS-BP	IBL/ILB	ATS-BP/ILB	Time	Best
CI10 100-01	14	15	15	1.07	1.07	0.55	0.55
CI10 100-02	15	16	16	1.07	1.07	0.52	0.52
CI10 100-03	15	16	16	1.07	1.07	0.72	0.72
CI10 100-04	17	18	18	1.06	1.06	0.72	0.72
CI10 100-05	17	18	18	1.06	1.06	0.69	0.69
CI10 100-06	13	14	13	1.08	1.00	0.66	100.42
CI10 100-07	13	14	14	1.08	1.08	0.55	0.55
CI10 100-08	18	19	18	1.06	1.00	0.69	2574.97
CI10 100-09	16	17	16	1.06	1.00	0.63	11.49
CI10 100-10	15	16	15	1.07	1.00	0.58	6.14
Averages	**15.3**	**16.3**	**15.9**	**1.07**	**1.04**	**0.63**	**269.68**

Table A.10.5 Martello And Vigo Class X, Size 100, *pcf* BL and ATS-BP results

Bibliography

Alexandrov, V. N, and G. M. Megson. *Parallel Algorithms for Knapsack Type Problems,* World Scientific, London, 1997.

Alvim, Adriana C.F., Fred S. Glover, Celso C. Ribeiro, and Dario J. Aloise "Local search for the bin packing problem" Catholic University of Rio de Janeiro, Technical Report, 1999.

Baltacioglu, Erhan, James T. Moore, and Raymond R. Hill Jr. "The distributors three-dimensional pallet-packing problem: a human intelligence-based heuristic" Air Force Institute of Technology (AU), Wright Paterson AFB OH, 2001 (TBP).

Baker, B.S., E.G. Coffman Jr., R.L. Rivest, "Orthogonal packing in two dimensions", *Siam Journal on Computing* 9, 1980, p846-855.

Barnes, J.Wesley. AND John .B. Chambers. "Solving the job shop scheduling problem using tabu search". *IIETransactions,* 27, 1995, p257-263.

Barnes, J. Wesley and Bruce Colletti "Linearity in the Traveling Salesman Problem," *Applied Mathematics Letters*, v13, No 3, April 2000, p27-32.

Barnes, J. Wesley, Metaheuristics*, Graduate Course OR391Q*, The Graduate Program in Operations Research and Industrial Engineering, The University of Texas at Austin, 2001.

Barnes, J. Wesley and Bruce Colletti "Local Search Structure in the Symmetric Traveling Salesman Problem under a General Class of Rearrangement Neighborhoods," *Applied Mathematics Letters*, v14, 2001, p105-108.

Barnes, J. Wesley, Bruce Colletti and David Neuway "Using Group Theory and Transition Matrices to Study a Class of Metaheuristic Neighborhoods", *European Journal of Operational Research*, 138, May 2002, p531-544.

Barnes, J . Wesley, with S. Dokov, E. Acevedo, E. and A. Solomon, "A Note on Distance Matrices Yielding Elementary Landscapes for the TSP", *Journal of Mathematical Chemistry,* 31, 2002, p.233-235.

Barnes, J . Wesley, with S. Dokov, E. Acevedo, E. and A. Solomon, " Weakly Symmetric Distance Matrices and the TSP *Applied Mathematics Letters,* 16. 2003, p.401-407.

Barnes, J. Wesley, B. Dimova, S. Dokov and Andrew Solomon "The Theory of Landscapes *Applied Mathematics Letters,* 16. 2003, p.337-343.

Barnes, J. Wesley, James T. Moore, David M. Ryer, and Victor D. Wiley "Solving the aerial fleet refueling problem using group theoretic tabu search". (to appear *Mathematical and Computer Modeling on Defense Transportation Issues*) 2002.

Battiti, Roberto "Reactive Search: Toward Self-Tuning Heuristics" in *Modern Heuristic Search Methods* edited by V.J. Rayward-Smith, I. H. Osman, C.R. Reeves, and G.D. Smith, 1996, p.61-89.

Battiti, Roberto and Giampietro Tecchiolli "The Reactive Tabu Search" *ORSA Journal on Computing*, Vol. 6, No. 2, Spring 1994, p.126-141.

Baumslag, Benjamin and Bruce Chandler, *Group Theory,* Schaum's Outline Series, McGraw-Hill, 1968.

Bennell, Julia A. and Kathryn A. Dowsland "A tabu thresholding implementation for the irregular stock cutting problem" *International Journal of Production Research* 18, 1999, p.4259-4275.

Bennell, Julia A, Kathryn A Dowsland, William B. Dowsland W., "The Irregular Cutting-Stock Problem - a New Procedure for Deriving the No-Fit Polygon", *Computers and Operations Research*, 28, 2001, p.271-287.

Berkey, J.O. and P.Y. Wang, "Two Dimensional Finite Bin Packing Algorithms" Journal of the Operational Research Society 38, 1987, p.423-429.

Bischoff, E. E., F. Janetz and M. S. W. Ratcliff "Loading pallets with non-identical items" *European Journal of Operational Research* 84, 1995, p.681-692.

Bogart, Kenneth P., *Introductory Combinatorics,* Harcourt Brace Jovanovich Publishers, 1990.

Bortfeldt, Andreas and Hermann Gehring "A hybrid genetic algorithm for the container loading problem" *European Journal of Operational Research* 131, 2001, p.143-161.

Bortfeldt, Andreas and Hermann Gehring "A genetic algorithm for the container loading problem" *International Transactions of Operational Research* 4(5/6), 1997, p.401-418.

Bortfeldt, Andreas and Hermann Gehring "Ein tabu search-verfahren fur Containerbeladeprobleme mit schwach heterogenem" *Operation Research Spektrum* 20, 1998, p.237-250.

Bortfeldt, Andreas and Hermann Gehring "Applying tabu search to container loading problems" Fern Universitat Hagen Technical Report 1998.

Burke, Edmund and Kendall Graham; "Comparison of meta-heuristic algorithms for clustering rectangles" *Computers & Industrial Engineering,* 1999, p.383-386.

Carlton, William B. *A tabu search approach to the general vehicle routing problem.* Ph.D. dissertation, The University of Texas at Austin, 1995.

Carlton, William B. and J. Wesley Barnes, "Solving the traveling salesman problem with time windows using tabu search." *IIE Transactions* 28:8, 1996. p.617-629.

Chazelle, Bernard, "The Bottom-Left Bin-Packing Heuristic: An Efficient Implementation" *IEEE Transactions on Computers*. c-32: 8, 1983 p.697-707.

Chocalaad, Christopher A. *Solving Geometric Knapsack Problems using Tabu Search Heuristics* Air Forces Institute of Technology MS Thesis, 1998.

Colletti, Bruce W. *Group Theory and Metaheuristics,* Ph.D. Dissertation, The University of Texas at Austin, 1999.

Colletti, B. W. and J. Wesley Barnes "Group theory and metaheuristic search neighborhoods", The University of Texas at Austin, Graduate Program in Operations Research. Technical Report 1999-02.

Colletti, Bruce. W. and J. Wesley Barnes "Linearity in the traveling salesman problem", *Applied Mathematics Letters* 13(3) 2000, p.27--32.

Colletti, Bruce. W., , J. Wesley Barnes. and S. Dokov, "A note on characterizing the k-OPT neighborhood via group theory", *Journal of Heuristics* 5 1999., p.47--51.

Conway, John H. and Charles Radin "Quaquaversal Tilings and rotations" *Inventiones mathematicae* No. 132, 1998 p.179-188.

Chu, P.C. and J.E. Beasley "A Genetic Algorithm for the Multidimensional Knapsack Problem", *Journal of Heuristics*, 1998, p.63-86.

Chien C.F., and Wu, W.T "A framework of modularized heuristics for determining the container loading patterns" *Computers & Industrial Engineering,*1999, p.339-342.

Crino, John Ph.D., *A group theoretic tabu search methodology for solving the theater distribution vehicle routing and scheduling problem*, Ph.D dissertation, The Air Force Institute of Technology, 2002.

Crino, J., J. Moore, J. W. Barnes, and W. Nanry, "Solving The Military Theater Distribution Vehicle Routing and Scheduling Problem Using Group Theoretic Tabu Search", to appear Mathematical and Computer Modeling on Defense Transportation Issues

Daniels, Karen, and Victor J. Milenkovic "Translational Polygon Containment and Minimal Enclosure using Mathematical Programming", *Transactions in Operational Research* 6, 1999 p.525-554.

Davies, A. Paul and Eberhard E. Bischoff "Weight distribution considerations in container loading", *European Journal of Operational Research* 114, 1999, p.509-527.

Dell'Amico, Mauro, , Silvano Martello, and Daniele Vigo "A lower bound for the non-oriented two-dimensional bin packing problem", *Discrete Applied Mathmatics* 118, 2002 p.13-24.

Dowsland, Kathryn A. "Simple Tabu Thresholding and the Pallet Loading Problem" *Meta-Heuristics Theory and Application* edited by Ibrahim H. Osman and James P. Kelly, 1996, p.381-405.

Dowsland, Kathryn A., Subodh Vaid, and William Dowsland "An algorithm for polygon placement using a bottom-left strategy" *European Journal of Operational Research,* 141 2002 p.371-381

Dorndorf, Ulrich and Erwin. Pesch, "Fast Clustering Algorithms", *ORSA Journal on Computing* 6, 1994, p.141-153.

Dyckhoff, Harald., Guntram. Scheithauer, and Johannes. Terno, "Cutting and Packing" in *Annotated Bibliographies in Combinatorial Optimization* edited by M. Dell'Amico,F. Maffioli, and S.Martello Chichester: Wiley Interscience, 1997.

Dyckhoff, Harald. and Ute. Finke *Cutting and Packing in Production and Distribution: A Typology and Bibliography* Physica-Verlag Heidelberg 1992.

Faina, Loris **"**A global optimization algorithm for the three-dimensional packing problem" *European Journal of Operational Research* 126, 2000, p.340-354

Faroe, Oluf, David Pisinger and Martin Zachariasen "Guided local search for the three-dimensional bin packing problem" University of Copenhagen, Denmark Tech Rep 99/13, 1999.

Fekete, Sándor. P., and Jörg Schepers "On more-dimensional packing I: Modeling", Technical paper ZPR97-288, Mathematishes Institut, Universität zu Köln, 1997.

Fekete, Sándor. P., and Jörg Schepers "On more-dimensional packing I: Bounds", Technical paper ZPR97-289, Mathematishes Institut, Universität zu Köln, 1997.

Fekete, Sándor. P., and Jörg Schepers "On more-dimensional packing I: Exact Algorithms", Technical paper ZPR97-290, Mathematishes Institut, Universität zu Köln, 1997.

Feng, Enmin, Xilu Wang, Xiumei Wang and Hongfei Teng "An algorithm of global optimization for solving layout problems", *European Journal of Operational Research* 114, 1999, p.430-436

Fraleigh, John B. A *First Course in Abstract Algebra*, Addison-Wesley, Reading MA, 1976.

Freville, Hanafi A., and A.El Abdellaoui "Comparison of Heuristics for the 0-1 Multidimensional Knapsack Problem" *Meta-Heuristics Theory and Application* edited by Ibrahim H. Osman and James P. Kelly, 407-427, Kluwer Academic Publishers, 1996.

Gilmore, P.C. and Gomory R.E. "Multistage cutting problems in two or more dimensions" *Operations Research* 12, 1965, p.94-119.

F. Glover. “New ejection chain and alternating path methods for traveling salesman problems”. In O. Balci, R. Sharda, and S. Zenios, editors, *Computer Science and Operations Research: New Developments in Their Interfaces*,. Pergamon, Oxford, 1992. p.491-507.

Glover, Fred and Gary A. Kochenberger “Critical event tabu search for multidimensional knapsack problems” *Meta-Heuristics Theory and Application* edited by Ibrahim H. Osman and James P. Kelly, 407-427, Kluwer Academic Publishers, 1996.

Glover, Fred and Manuel Laguna, *Tabu Search,* Kluwer Academic Publishers, 1997.

Glover, Fred "Scatter Search and Path Relinking," in *New Ideas in Optimization*, D. Corne, M. Dorigo and F. Glover, Eds., McGraw Hill, 1999, p.297-316.

Glover, Fred "Multi-Start and Strategic Oscillation Methods - Principles to Exploit Adaptive Memory," *Computing Tools for Modeling, Optimization and Simulation: Interfaces in Computer Science and Operations Research*, M. Laguna and J.L. Gozales Velarde, Eds., Kluwer Academic Publishers, 2000, p.124.

Glover, Fred, Manuel Laguna, and Rafael Marti "Fundamentals of Scatter Search and Path Relinking*," Control and Cybernetics* 29:3, 2000, p.653-684.

Glover, Fred, Manuel Laguna, and Rafael Marti, “Scatter Search and Path Relinking: Foundations and Advance Designs” Leeds School of Business, University of Colorado, Boulder Colorado, 2002.

Glover, Fred, Manuel Laguna, and Rafael Marti, “Scatter Search and Path Relinking: Advances and Applications” Leeds School of Business, University of Colorado, Boulder Colorado, 2002.

Gomes, A. Miguel, and José F. Oliveira, “A 2-exchang heuristic for nesting problems”, *European Journal of Operational Research* 141, 2002, p.359-370.

González-Velarde, José Luis and Manuel Laguna ‘Tabu Search with Simple Ejection Chains for Coloring Graphs’ Technical Report Leeds School of Business, University of Colorado, Boulder, CO, 2002.

Grossman, Israel and Wilhelm Magnus *Groups and Their Graphs,* Random House, 1964.

Hays, Jon, http://members.fortunecity.com/jonhays/group.htm, 2000.

Herstein, I.N. *Topics in Algebra,* 2nd Edition, 1975.

Hopper, E. "Review of the Application of Meta-Heuristic Algorithms to 2D Strip Packing Problems" *Artificial Intelligence Review* 16:4, 2001, p.257-300.

Hopper, E. and B. C. H. Turton "An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem", *European Journal of Operational Research* 128, 2001, p.34-57.

Hu, T.C. and M.T. Shing *Combinatorial Algorithms: Enlarged Second Edition* Dover Publications Inc. 2002.

Jakobs, Stefan "On genetic algorithms for the packing of polygons", *European Journal of Operational Research* 88, 1996, p.165-181.

Kelly, James. P. and J. Xu. "Tabu Search and Vocabulary Building for Routing Problems". Technical report, Graduate School of Business Administration, University of Colorado at Boulder, 1995.

Kreher, Donald L. and Douglas R. Stinson *Combinatorial Algorithms: Generation, Enumeration, and Search,* CRC Press, 1999.

Kreher, Donald L. with Stanislaw P Radziszowski, "Search algorithm for Ramsey graphs by union of group orbits." *Journal of Graph Theory* 12, 1988.

Kröger, Berthold "Guillotinable bin packing: A genetic approach"**,** *European Journal of Operational Research* 84, 1995, p.645-661.

Li, Zhenyu and Victor Milenkovic "Compaction and Separation Algorithms for Non-Convex Polygons and Their Applications" *European Journal of Operational Research* 84 ,1995 , p.539-561.

Lins, Lauro, Sóstenes Lins, and Reinaldo Morabito "An n-tet approach for non-guillotine packings of n-dimensional boxes into an n-container"**,** *European Journal of Operational Research* 141, 2002, p.421-439.

Liu, Dequan, personal email 2003.

Liu, Dequan and Hongfei Teng "An improved BL-algorithm for genetic algorithm of the orthogonal packing of rectangles", *European Journal of*

Operational Research 112, 1999, p.413-420.

Liu, Wei-Ping and Jeffrey B. Sidney "Bin packing using semi-ordinal data", *Operations Research Letters* 19:3, 1996, p.101-104.

Liu, Yanxi and Popplestone, R. ``A Group Theoretic Formalization of Surface Contact'', *International Journal of Robotics Research*, 13:2, April 1994, p148-161.

Liu, Yanxi *Symmetry Groups in Robotic Assembly Task Planning and Specification*, the Mathematical Methods in Technology series, Marcel Dekker, Inc. (draft chapter) 2001.

Lodi, Andrea, personal email, 2003.

Lodi, Andrea, Silvano Martello, And Daniele Vigo "Neighborhood Search Algorithm for the Guillotine Non-Oriented Two-dimensional Bin Packing Problem", in *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, S. Voss, S. Martello, I.H. Osman, and C. Roucairol, (eds.) Kluwer Academic Publishers, Boston, 1998, p.125-139.

Lodi, Andrea, Silvano Martello, And Daniele Vigo "Heuristic And Metaheuristic Approaches For A Class Of Two-Dimensional Bin Packing Problems", *INFORMS Journal On Computing* 11, 1999, p.345-357.

Lodi, Andrea, Silvano Martello and Daniele Vigo "Approximation algorithms for the oriented two-dimensional bin packing problem', *European Journal of Operational Research* 112, 1999, p.158-166.

Lodi, Andrea, Silvano Martello and Daniele Vigo "Recent advance in two-dimensional bin packing problems", *Discrete Applied Mathematics* (article in press 2001).

Lodi, Andrea, Silvano Martello and Daniele Vigo "Heuristic algorithms for the three-dimensional bin packing problem", *European Journal of Operational Research* 141, 2002, p.410-420.

Lodi, Andrea, Silvano Martello and Daniele Vigo "Two-dimensional packing problems: A Survey", *European Journal of Operational Research* 141, 2002, p.241-252

Lodi, Andrea, Silvano Martello and Daniele Vigo “TSpack: A Unified Search Code for the Multi-Dimensional Bin Packing Problem”, University of Bologna, 2002.

Lyndon, Roger C. *Groups and Geometry* Cambridge University Press, 1985.

Martello, Silvano and Pablo Toth *Knapsack Problems: Algorithms and Computer Implementations* John Wiley & Sons, 1990.

Martello, Silvano, David Pisinger, and Daniel Vigo , “The three-dimensional bin packing problem”, *Operations Research* 48:2, 2000, p.256-267.

Martello, Silvano and Daniele Vigo “Exact Solution of the Two-dimensional Finite Bin Packing Problem”, *Management Science* 44, 1998 p.388-399.

Milenkovic, V.J. and Daniels, K.M., “Translational Polygon Containment and Minimal Enclosure using Mathematical Programming”, *In International Transactions in Operational Research* special issue with papers from IFORS '96, vol. 6, 1999, p.525-554.

Neumann, Peter M., Gabrielle A. Stoy and Edward C. Thompson *Groups and Geometry*, Oxford University Press, 1994.

Ouyang, M., M. Toulouse, K. Thulasiraman, Glover F., and J.S. Deogun. "Multi-Level Cooperative Search: Application to the Circuit/Hypergraph Partitioning Problem", *Proceedings of the International Symposium on Physical Design*, ACM Press, 2000 p.192-198.

Passman, Donald *Permutation Groups* W.A. Benjamin Inc., 1968.

Papadimitriou, Christos H. and Kenneth Steiglitz *Combinatorial Optimization: Algorithms and Complexity* Dover Publications, 1998.

Pesch, Erwin and Fred Glover, “TSP Ejection Chains”, *Discrete Applied Mathematics* 76, 1997, p.165-181.

Pinsinger, David “Heuristics for the container loading problem”, *European Journal of Operational Research* 141, 2002, p.382-392.

Popplestone, R.J., Liu, Y. and Weiss, R. ``A Group Theoretical Approach to Assembly Planning", *Artificial Intelligence Magazine,* Spring 1990, p.82-97.

Rego, Cesar “A Subpath Ejection Method for the Vehicle Routing Problem”, *Management Science* 44:10, 1998, p.1447-1459.

Rego, Cesar, L. Cavique and I. Themido “Subgraph Ejection Chains and Tabu Search for the Crew Scheduling Problem”, *Journal of the Operational Research Society*50, 1999, p.608-616.

Rego, Cesar “Node Ejection Chains for the Vehicle Routing Problem: Sequential and Parallel Algorithms”, *Parallel Computing*, 27, 2000, p.201-222.

Rochat, Y and E. Taillard. “Probabilistic Diversification and Intensification in Local Search for Vehicle Routing*”. Journal of Heuristics*, 1(1): 1995, p.147-167

Romaine, Jonathan M. *Solving the Multidimensional Multiple Knapsack Problem using Tabu Search Heuristics* Air Forces Institute of Technology MS Thesis, 1999.

Srivastava, Anuj, Michael I. Miller and Ulf Grenander “Ergodic Algorithms on Special Euclidean Groups for ATR”, *Systems And Control In The Twenty-first Century* 1997.

Yu. G. Stoyan and A. V. Pankratov “Regular packing of congruent polygons on the rectangular sheet”, *European Journal of Operational Research* 113, 1999, p.653-675.

Terno, Johannes, Guntram Scheithauer, Uta Sommerweiß and Jan Riehme “An efficient approach for the multi-pallet loading problem” *European Journal of Operational Research* 123, 2000, p.372-381.

Theodoracatos, Vassilios E. and James L. Grimsley “The optimal packing of arbitrarily shaped polygons using simulated annealing an polynomial time cooling schedules”, *Computational Methods of Applied Mechanical Engineering* 125, 1995, p.53-70.

Wiley, Victor D. *The Aerial Fleet Refueling Problem*, Ph.D Dissertation, The University of Texas, 2001.

Zefran, Milos and Vijay Kumar “Interpolation schemes for rigid body motions”*, Computer-Aided Design* 30:3, 1998, p.179-189.

Vita

John Michael Harwig was born in Massillon Ohio on February 1st, 1964, the son of Dennis Duane Harwig and Mary Ann Harwig. A 1982 graduate of Tuslaw High School, Massillon, Ohio he entered the United States Military Academy in West Point, New York. He received a Bachelor of Science degree in Mechanical Engineering–Aerospace Systems from the United States Military Academy in May 1986 and was commissioned as an officer in the United States Army. His military service has been continuous to the present day, and he has held positions as an Aeroscout Platoon Leader, Squadron Logistics Officer (S4), Attack Helicopter Battalion Assistant Operations Officer (Asst. S3), Attack Helicopter Troop Commander, Air Cavalry Brigade War Plans Officers (Asst S3 Plans), and Combat Operations Research Analyst. He currently holds the rank of Major. He received the degree of Masters of Science in Engineering from the University of Texas, Austin, Texas in May 1997. In August 2000 he began doctoral studies in the Operations Research, Industrial Engineering Program at the University of Texas, Austin, Texas.

Permanent address: 16602 Pocono Drive Austin TX, 78717

This dissertation was typed by the author.