Weight Annealing Heuristics for Solving Bin Packing Problems

Kok-Hua Loh
University of Maryland

Bruce Golden
University of Maryland

Edward Wasil
American University

INFORMS Annual Meeting
October 5, 2006

Outline of Presentation

- Introduction
- Concept of Weight Annealing
- One-Dimensional Bin Packing Problem
- Two-Dimensional Bin Packing Problem
- Conclusions

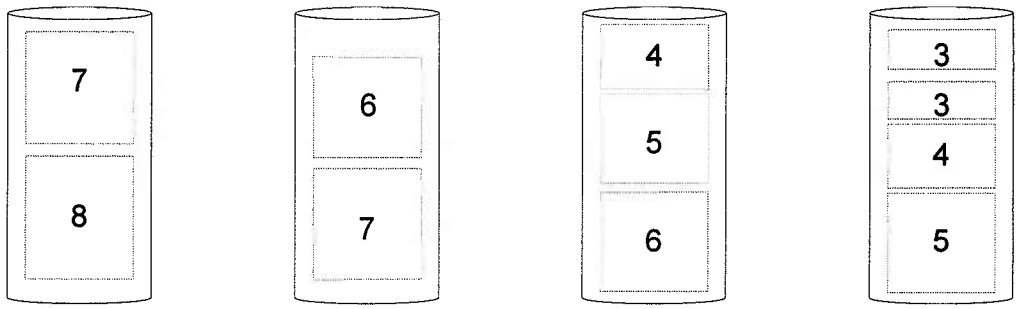
Weight Annealing Concept

- Assigning different weights to different parts of a combinatorial problem to guide computational effort to poorly solved regions.
  - Ninio and Schneider (2005)
  - Elidan et al. (2002)
- Allowing both uphill and downhill moves to escape from a poor local optimum.
- Tracking changes in the objective function value, as well as how well every region is being solved.
- Applied to the Traveling Salesman Problem. (Ninio and Schneider 2005)
  - Weight annealing led to mostly better results than simulated annealing.

One-Dimensional Bin Packing Problem

- Pack a set of $N = \{1, 2, \ldots, n\}$ items, each with size t_i, $i=1, 2, \ldots, n$, into identical bins, each with capacity C.
- Minimize the number of bins without violating the capacity constraints.
- Large literature on solving this NP-hard problem.

Item List ={8,7,7,6,6,5,4,4,3,3} Bin Capacity =15



Outline of Weight Annealing Algorithm

- Construct an initial solution using first-fit decreasing.
- Compute and assign weights to items to distort sizes according to the packing solutions of individual bins.
- Perform local search by swapping items between all pairs of bins.
- Carry out re-weighting based on the result of the previous optimization run.
- Reduce weight distortion according to a cooling schedule.

Neighborhood Search for Bin Packing Problem

- From a current solution, obtain the next solution by swapping items between bins with the following objective function (suggested by Fleszar and Hindi 2002)

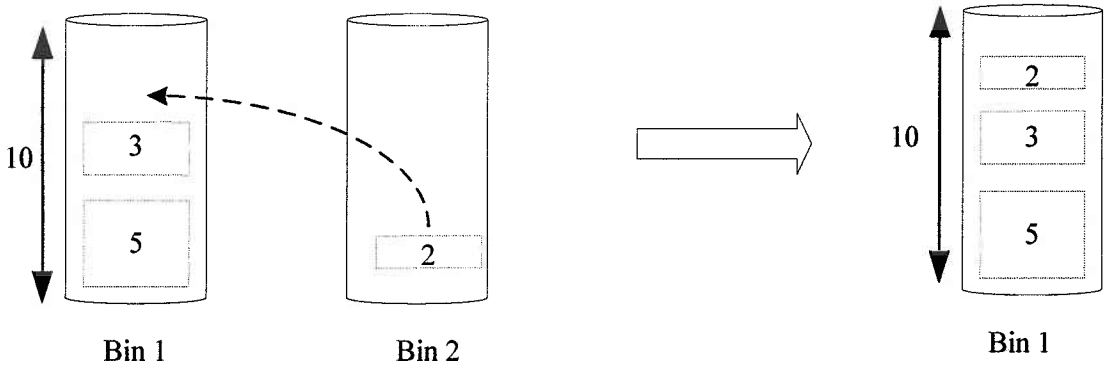
$$\text{Maximize} \quad f = \sum_{i=1}^{p} (l_i)^2$$

$$l_i = \sum_{j=1}^{q_i} t_j \quad \text{bin load } i$$

p = number of bins

q_i = number of items in bin i

t_j = size of item j



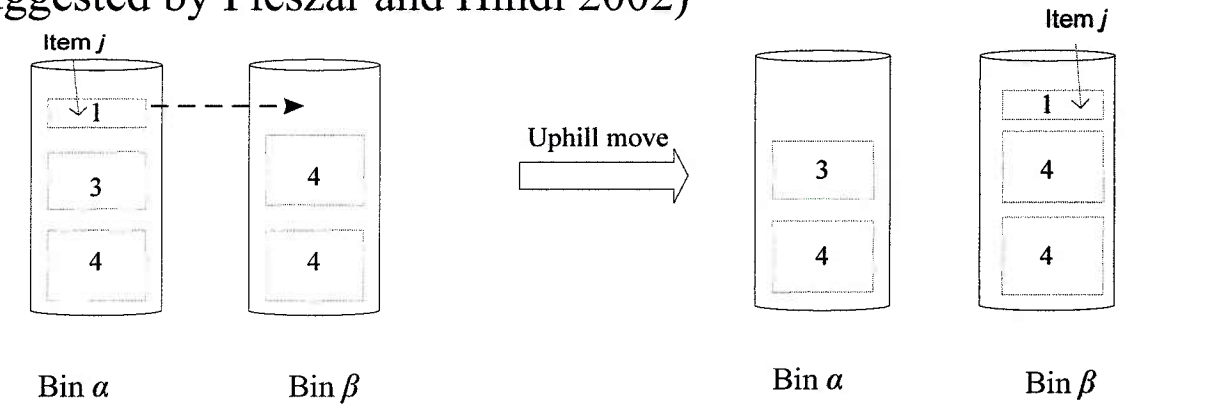
$$f = (5+3)^2 + 2^2 = 68$$

$$f_{new} = (5+3+2)^2 = 100$$

Neighborhood Search for Bin Packing Problem

- Swap schemes
  - Swap items between two bins.
  - Carry out Swap (1,0), Swap (1,1), Swap (1,2), Swap (2,2) for all pairs of bins.
  - Analogous to 2-Opt and 3-Opt.

- Swap (1,0) (suggested by Fleszar and Hindi 2002)



$$f = (1+3+4)^2 + (4+4)^2 = 128$$

$$f_{new} = (3+4)^2 + (1+4+4)^2 = 130$$

- Need to evaluate only the change in the objective function value.

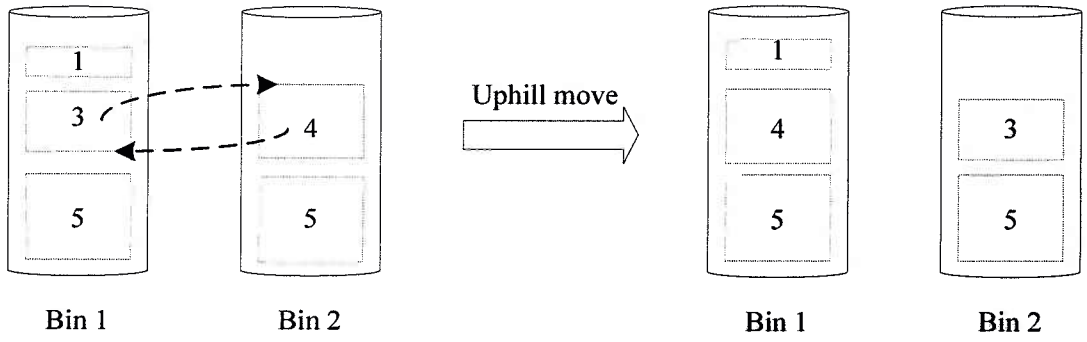
$$\Delta f = (l_\alpha - t_j)^2 + (l_\beta + t_j)^2 - l_\alpha^2 - l_\beta^2$$

l_α = total load of bin α

t_i = size of item i

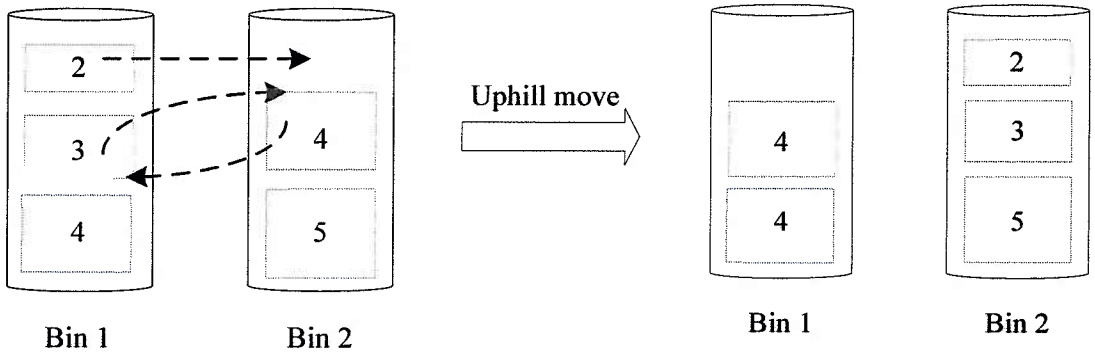
Neighborhood Search for Bin Packing Problem

- Swap (1,1)



($f = 162$) ($f_{new} = 164$)

- Swap (1,2)



($f = 162$) ($f_{new} = 164$)

Weight Annealing for Bin Packing Problem

- Weight of item i

$$w_i = 1 + K r_i$$

residual capacity $$r_i = \left(\frac{C - l_i}{C} \right)$$

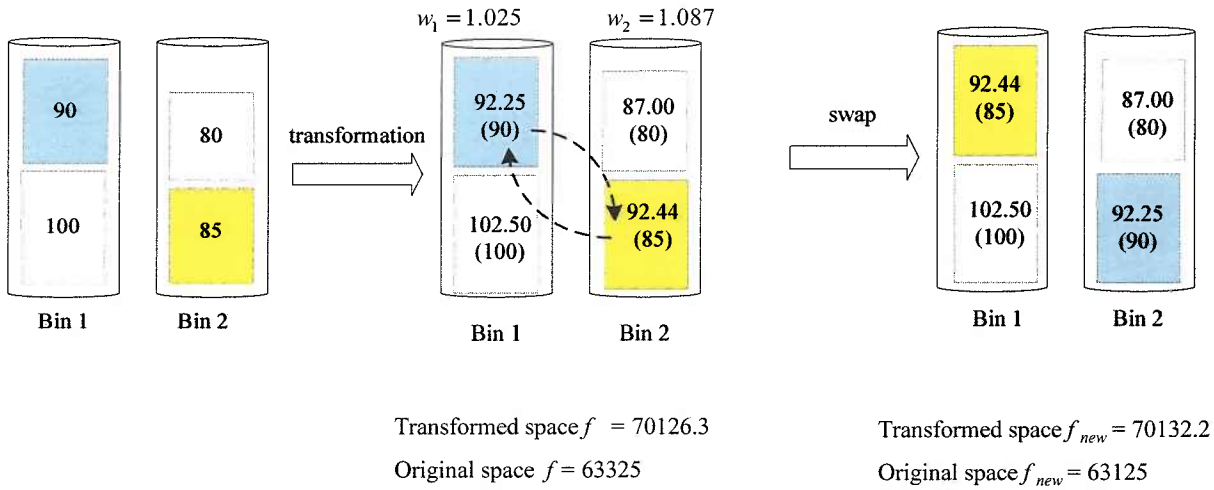
C = capacity

l_i = load of bin i

- An item in a not-so-well-packed bin, with large r_i, will have its size distorted by a large amount.
- No size distortions for items in fully packed bins.
- K controls the size distortion, given a fixed r_i .

Weight Annealing for Bin Packing Problem

- Weight annealing allows downhill moves in a maximization problem.
- Example $C = 200,\ K = 0.5,\ w_i = 1 + 0.5\left(\frac{200 - l_i}{200}\right)$



Transformed space f = 70126.3
Original space f = 63325

Transformed space f_{new} = 70132.2
Original space f_{new} = 63125

- Transformed space - uphill move
- Original space - downhill move

Solution Procedures (1BP)

- BISON (Scholl, Klein, and Jürgens 1997)
  - Hybrid method combining tabu search and branch-and-bound
  - New branch schemes
- MTPCS (Schwerin and Wäscher 1999)
  - Bounding procedures based on a cutting stock problem (CS)
  - Integrating the lower bound into Martello and Toth procedure (MTP)
- PMBS' +VNS (Fleszar and Hindi 2002)
  - Minimum bin slack heuristic
  - Variable neighborhood search
- HI_BP (Alvim, Ribeiro, Glover, and Aloise 2004)
  - Sophisticated hybrid improvement heuristic
  - Tabu search to move items between bins
- WA1BP
  - Weight annealing heuristic that creates dimension distortions to different parts of the search space during the local search.

Computational Results (1BP)

- Benchmark problems
  - Five sets of test problems
    - Uniform U120, U205, U500, U1000
    - Triplet T60, T120, T249, T501
    - Set Set1, Set2, Set3
    - Was Was1, Was2
    - Gau Gau1
  - A total of 1587 problem instances

Computational Results (1BP)

- Weight annealing performed slightly better than HI_BP.
  - Generated more optimal solutions to the Gau set (17 versus 14).
- Weight annealing performed much better than BISON, PMBS' +VNS, and MTPCS.
  - Generated more optimal solutions to Set benchmark problems.
    - Weigh annealing found optimal solutions to all 1210 instances.
    - BISON, PMBS' +VNS and MTPCS fell short (by 37, 40, and 94 instances).
  - Was faster than BISON and MTPCS (0.18s versus 31.5s - 118.2s).
- Overall Performance of the weight annealing algorithm
  - Found 1582 optimal solutions to 1587 problem instances.
  - Found three new optimal solutions to the Gau set.
  - Took 0.16s on average to solve an instance.

Two-Dimensional Bin Packing Problems

Problem statement

- Allocate, without overlapping, *n* rectangular items to identical rectangular bins.
- Pack items such that the edges of bins and items are parallel to each other.
- Minimize the total number of rectangular bins (NP-hard).

Classifications

- Guillotine Cutting
  - 2BP|O|G — Fixed Orientation (O), Guillotine Cutting (G)
  - 2BP|R|G — Allowable 90° Rotation (R), Guillotine Cutting (G)

- Free Cutting
  - 2BP|O|F — Fixed Orientation (O), Free Cutting (F)
  - 2BP|R|F — Allowable 90° Rotation (R), Free Cutting (F)

Two-Dimensional Bin Packing Problems (2BP|O|G)

Hybrid first-fit algorithm

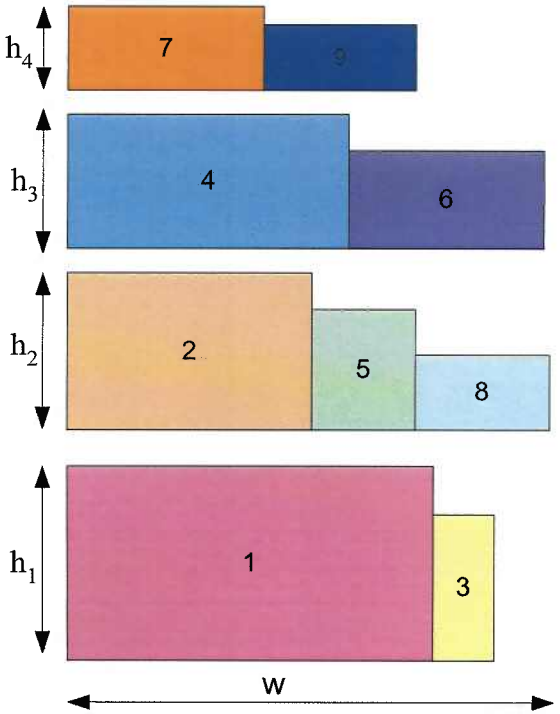
- Phase One (one-dimensional horizontal level packing)
  - Arrange the items in the order of non-increasing height.
  - Pack the items from left to right into levels, each level i with the same width W.
  - Pack an item (left justified) on the first level that can accommodate it; start a new level if no level can accommodate it.

- Phase Two (one-dimensional vertical bin packing)
  - Arrange the levels in the order of non-increasing height h_i; this is the height of the first item on the left.
  - Solve one-dimensional bin packing problems, each item i with size h_i, and bin size H.

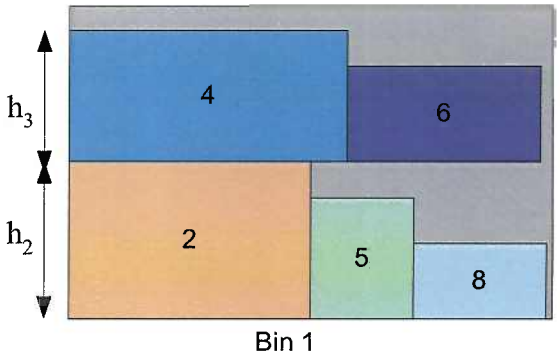
Two-Dimensional Bin Packing Problems (2BP|O|G)

An example of hybrid first-fit

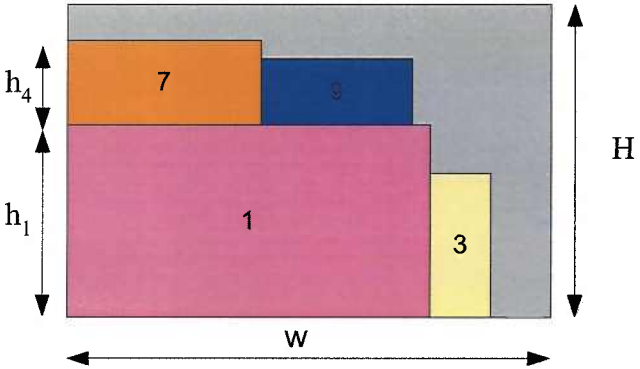
Phase 1 - One Dimensional
Horizontal Level Packing



Phase 2 - One Dimensional
Vertical Bin Packing



Bin 1



Bin 2

Two-Dimensional Bin Packing Problems (2BP|O|G)

- Weakness of hybrid first-fit