

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.

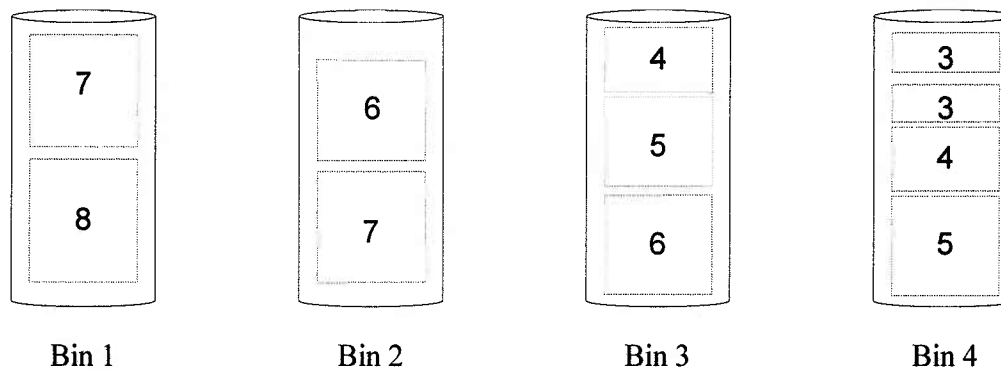
- Tracking changes in the objective function value, as well as how well every region is being solved.

- 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 $N = \{1, 2, \dots, n\}$ items, each with size t_i , $i=1, 2, \dots, n$, 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)

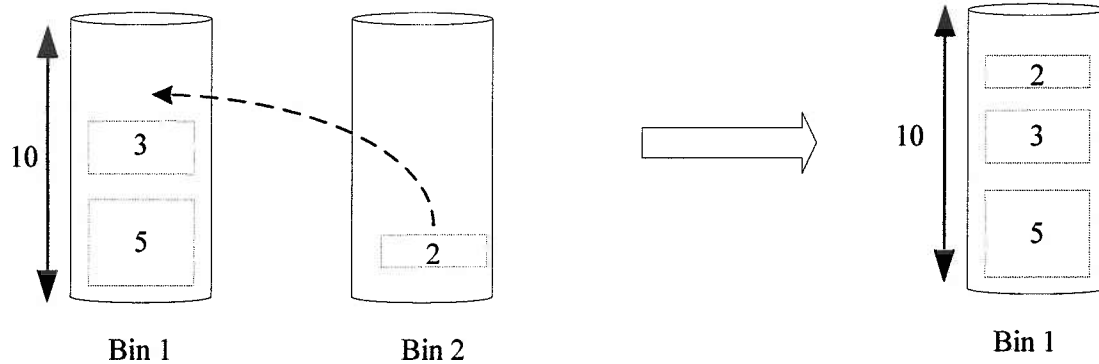
$$\text{Maximize } f = \sum_{i=1}^p (l_i)^2$$

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

p = number of bins

q_i = number of items in bin i

t_j = size of item j



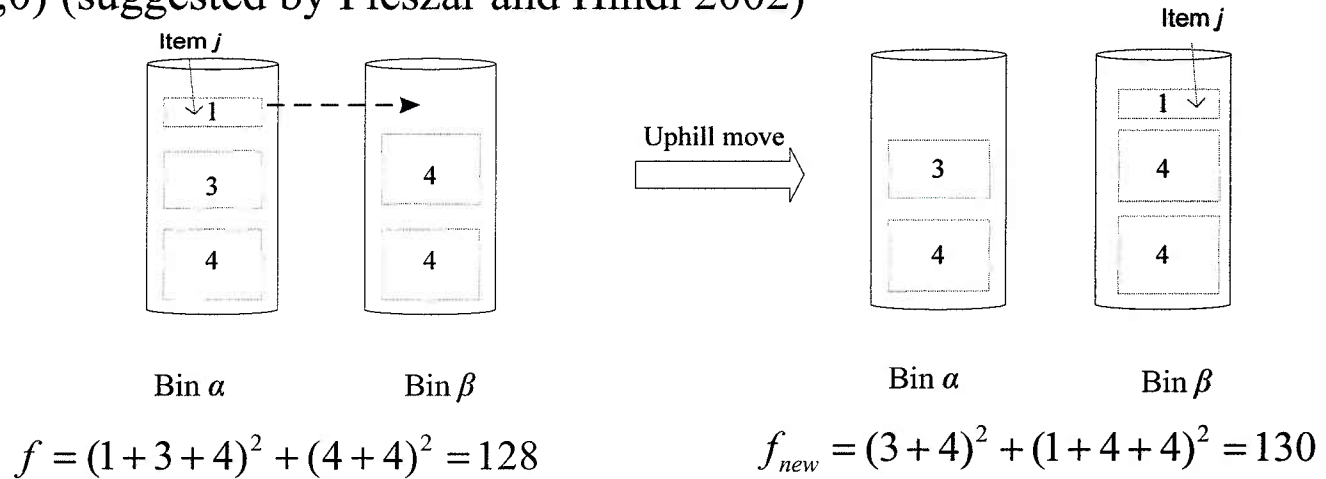
$$f = (5 + 3)^2 + 2^2 = 68$$

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

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)



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

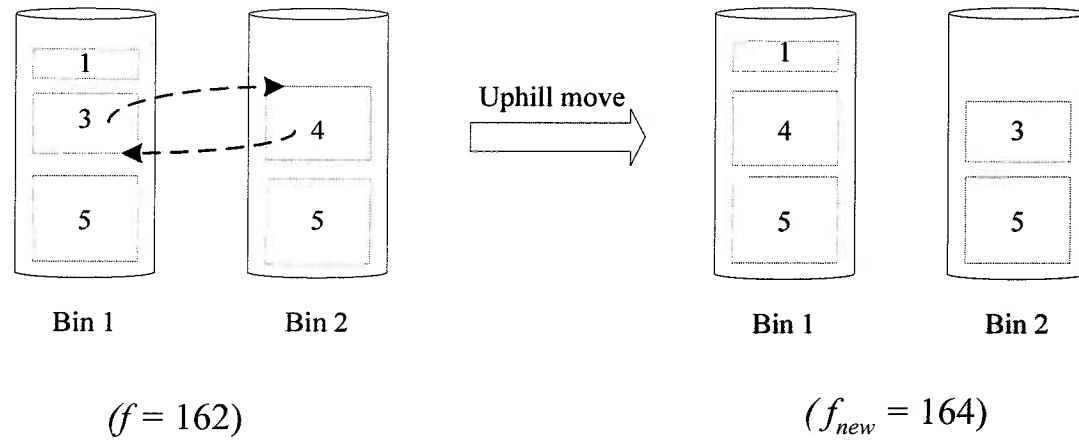
$$\Delta f = (l_\alpha - t_j)^2 + (l_\beta + t_j)^2 - l_\alpha^2 - l_\beta^2$$

l_α = total load of bin α

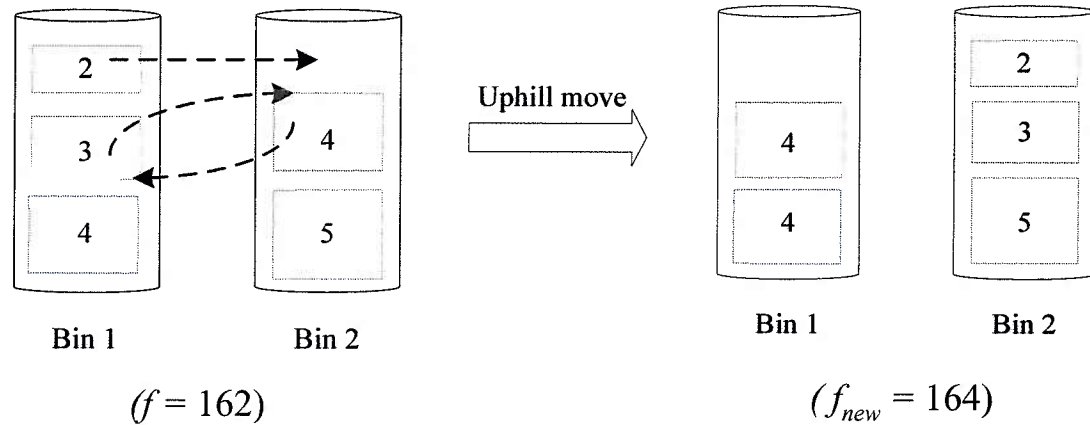
t_i = size of item i

Neighborhood Search for Bin Packing Problem

■ Swap (1,1)



■ Swap (1,2)



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 .

