Solving the Maximum Cardinality Bin Packing Problem with a Weight Annealing-Based Algorithm

Kok-Hua Loh
University of Maryland

Bruce Golden
University of Maryland

Edward Wasil
American University

10th ICS Conference
January 2007

Outline of Presentation

- Introduction
- Concept of Weight Annealing
- Maximum Cardinality Bin Packing Problem
- Conclusions

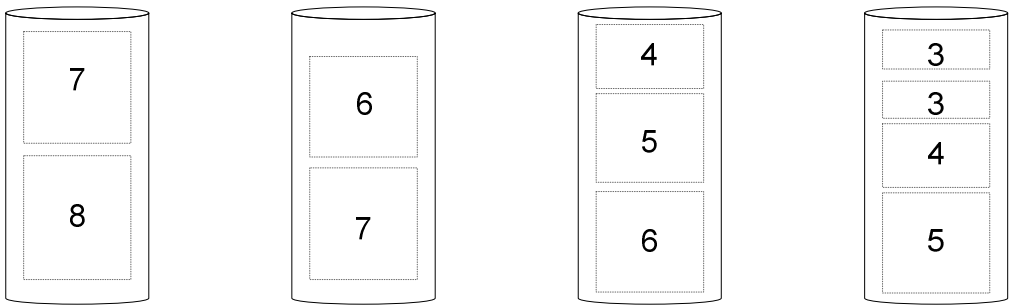
Weight Annealing Concept

- 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 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 (1BP)

- 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).

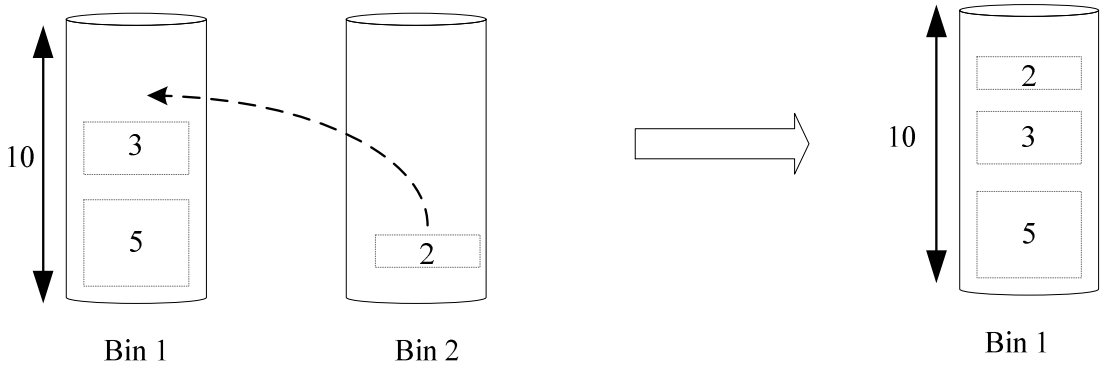
$$\text{Maximize} \quad f = \sum_{i=1}^{p} (l_i)^2$$

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

p = number of bins

q_i = number of items in bin i

t_j = size of item j



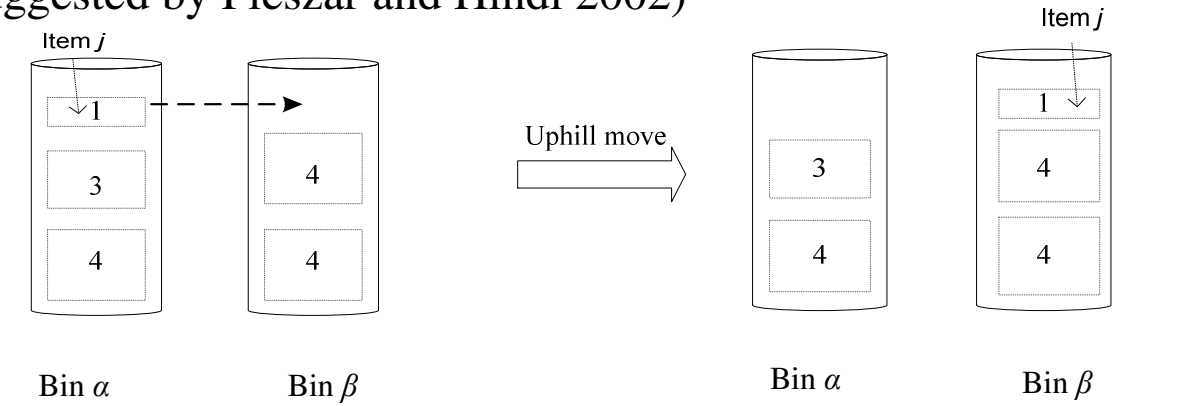
$$f = (5+3)^2 + 2^2 = 68$$

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

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.

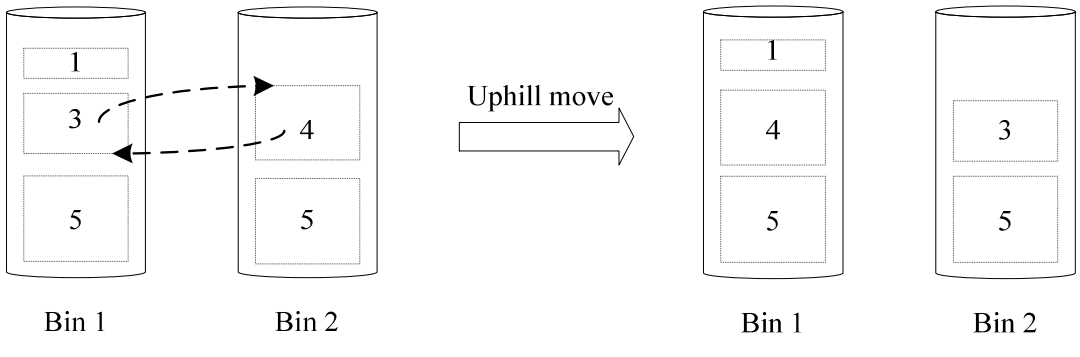
$$\Delta f = (l_\alpha - t_j)^2 + (l_\beta + t_j)^2 - l_\alpha^2 - l_\beta^2$$

l_α = total load of bin α

t_i = size of item i

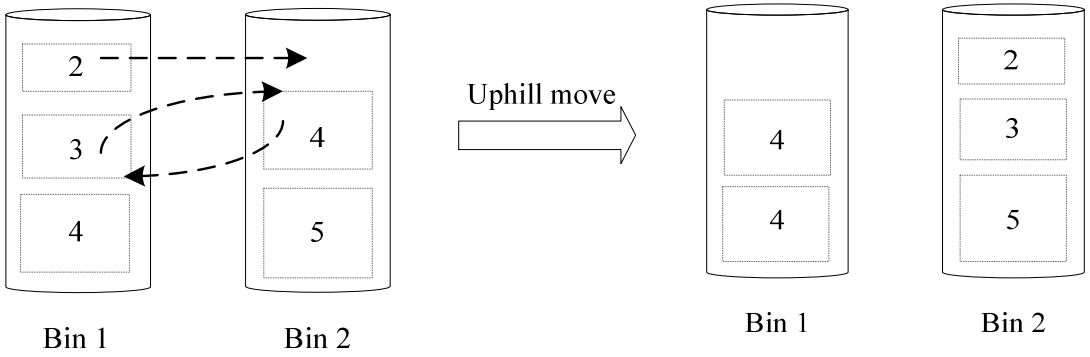
Neighborhood Search for Bin Packing Problem

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

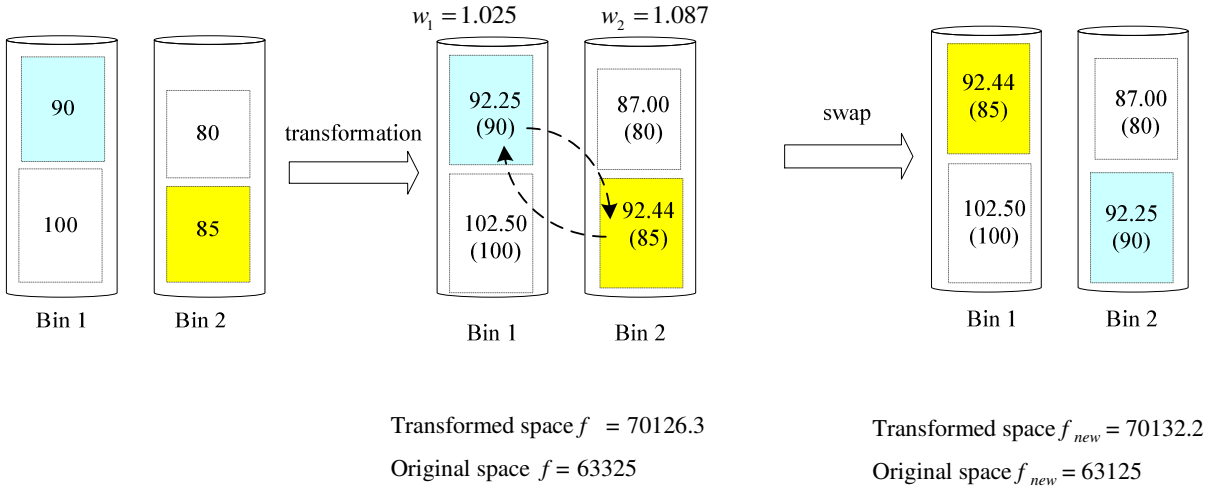
C = capacity

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

Maximum Cardinality Bin Packing Problem (MCBP)

- Problem statement
  - Assign a subset of n items with sizes t_i to a fixed number of m bins of identical capacity c.
  - Maximize the number of items assigned.

- Formulation

$$\text{maximize} \quad \sum_{i=1}^{n} \sum_{j=1}^{m} x_{ij}$$

subject to

$$\sum_{i=1}^{n} t_i x_{ij} \le c \qquad j \in \{1,\ldots,m\}$$

$$\sum_{j=1}^{m} x_{ij} \le 1 \qquad i \in \{1,\ldots,n\}$$

$$x_{ij} = 0 \text{ or } 1 \qquad i \in \{1,\ldots,n\}, \quad j \in \{1,\ldots,m\}$$

where $x_{ij}=1$ if item i is assigned to bin i and $x_{ij}=0$ otherwise.

Maximal Cardinality Bin Packing Problem

- Practical applications
  - Computing.
    - Assign variable-length records to a fixed amount of storage.
    - Maximize the number of records stored in fast memory so as to ensure a minimum access time to the records.
  - Management of real time multi-processors.
    - Maximize the number of completed tasks with varying job durations before a given deadline.
  - Computer design.
    - Designing processors for mainframe computers.
    - Designing the layout of electronic circuits.

Bounds for Maximal Cardinality Bin Packing Problem

- We use the three upper bounds on the optimal number of items developed by Labbé, Laporte, and Martello (2003): $\overline{U}_0$, $\overline{U}_1$, and $\overline{U}_2$.

- We use the two lower bounds on the minimal number of bins developed by Martello and Toth (1990): L_2 and L_3.

Outline of Weight Annealing Algorithm (WAMC)

- Arrange items in the order of non-decreasing size.
- Compute a priori upper bound on the optimal number of items (U^*).
  - $U^* = \min\{\overline{U}_0, \overline{U}_1, \overline{U}_2\}$.
  - Update ordered list by removing item i with size t_i for which $i > U^*$.

  (The optimal solution z^* is obtained by selecting the first z^* smallest items.)
- Improve the upper bound U^*.
  - Find lower bound on the minimum number of bins required (L_3).
  - If $L_3 > m$, reduce U^* by 1.
  - Update ordered list by removing item i with size t_i for which $i > U^*$.
  - Iterate until $L_3 = m$.
- Find feasible packing solution for the ordered list with the weight annealing algorithm for 1BP.
- Output results.

Solution Procedures for MCBP

- Enumeration algorithm (LA) by Labbé, Laporte, and Martello (2003)
  - Compute a priori upper bounds.
  - Embed the upper bounds into an enumeration algorithm.
- Branch-and-price algorithm (BP) by Peeters and Degraeve (2006)
  - Compute a priori and LP upper bounds.
  - Solve the problem with heuristics in a branch-and-price framework.

Test Problems

- Labbé, Laporte, and Martello (2003)
  - 180 combinations of three parameters
    - number of bins m = 2, 3, 5, 10, 15, 20
    - capacity c = 100, 120, 150, 200, 300, 400, 500, 600, 700, 800
    - size interval [t_{min} , 99]; t_{min} = 1, 20, 50
  - For each combination (m, c, t_{min}), create 10 instances by generating item size t_i in the given size interval until $\sum_i t_i > mc$.
- Peeters and Degraeve (2006)
  - 270 combinations of three parameters
    - capacity c=1000, 1200, 1500, 2000, 3000, 4000, 5000, 6000, 7000, 8000
    - size interval [t_{min} , 999]; t_{min} = 1, 20, 50
    - desired number of items $\bar{n} = 100, 150, 200, 250, 300, 350, 400, 450, 500$

      $m = \bar{n}(t_{\min} + 999)/2c$
  - For each combination (m, c, t_{min}), create 10 instances by generating item size t_i in the given size interval until $\sum_i t_i > mc$.

Computational Results

- Results on the test problems of Labbé, Laporte, and Martello (2003)
  - Generated 1800 problems for testing on WAMC .
  - LA and BP used a different set of 1800 problems.
  - Number of instances solved to optimality
    - BP 1800
    - WAMC 1793
    - LA 1759
  - Average running times
    - BP < 0.01 sec (500 MHz Intel Pentium III)
    - WAMC 0.03 sec (3 GHz Intel Pentium IV)
    - LA 3.16 sec (Digital VaxStation 3100)

Computational Results

- Generated 2700 problems for testing on WAMC; BP used a different set of 2700 problems.
- Computational Results

Solution Procedures	Number of Instances Solved to Optimality	Average Running Time (sec)
WAMC	2665	0.20
BP	2519	2.85

- WAMC outperforms BP.
  - BP had difficulties solving instances with
    - Large bin capacities (5000-8000)
    - Large number of items (350-500).
  - WAMC solved all instances with bin capacities ≥ 2000
  - WAMC was faster.

Conclusions

- WAMC is easy to understand and simple to code.
  - Weight annealing has wide applicability(1BP, 2BP).
- WAMC produced high-quality solutions to the maximum cardinality bin packing problem.
- WAMC solved 99% (4458/4500) of the test instances to optimality with an average time of a few tenths of a second.