Bin packing problem

Team Parallelites

By - Akshay Jain
Appurv Jain
Sai Manohar Nethi
Agenda

- Topic overview
- Analysis of Research Paper 1
- Analysis of Research Paper 2
- Analysis of Research Paper 3
- Progress
- Pseudo code
Overview of topic
We consider a 1-d basic bin packing problem – packing of varying weighted items into minimum no. of equal capacity bins.

Take ‘n’ objects of weight $w_i$

Each object requires $b_i$ bin capacity.

Determine minimum no. of bins to accommodate all objects.

The total bin’s capacity is not exceeded.
Overview of topic

- One of the oldest and highly researched problems.
- NP-Complete problem – if formulated as a decision problem.
- NP-Hard – if optimization involved.
- General idea – to pack certain items with some weights in to bins/boxes.
Using genetic algorithms to solve quality-related bin packing problem

Authors: Felix T.S. Chan, K.C. Au, L.Y. Chan, T.L. Lau

Idea: Talks about generating a bin packing genetic algorithm (BPGA)

Publication: Science Direct

Date: 4 November 2005

Pages: 71–81
Using genetic algorithms to solve quality-related bin packing problem

This paper talks about generating a bin packing genetic algorithm (BPGA) which takes into account many factors apart from just the weight of the items.

The industrial application is about the manufacturing of Ion Planting (IP) cell. Not only does the paper talk about measures to reduce the overall cost of production, it also lays emphasis on the quality of product and its service.

Here, the algorithm is applied to the application of Ion Plating (IP) cells. First, the paper gives a brief overview of the types of bin packing problem in industry (production line scheduling, manufacturing cost reduction) and then goes over to describe the IP manufacturing and quality assurance process.

Further, the paper reiterates the concept of Genetic Algorithms and proposes the variant of it that they are employing in their operation (Grouped Genetic Algorithm).

They are basically search algorithms, which work over the entire solution space and iterate with all the attributes in consideration. But, the distinguishing factor is GAs can be applied to various domains and incorporate multiple variables.

Their ability to obtain near-optimal results, and capability to handle large amount of data, make them suitable for NP hard type problems.
Using genetic algorithms to solve quality-related bin packing problem

So, what we intend to use/apply from this paper is - The logic about Grouped Genetic Algorithm. While GAs are capable to handle varied type of data, there is still a possibility of redundancy.

For example, there may be many possible assignments that are redundant or infeasible. So, as per GGA, we not only find the min. no. of bins for the problem, we also talk about meta-heuristics, which in our case, would be a likelihood/quality of assignment.

It means, while assigning bins to the problem, we not only try to utilize the minimum no. of bins, but we also try to reduce the overhead cost of possible re-assignment. This ensures, that redundant data is not generated/computed.
Worst-Case Performance Bounds for Simple One-Dimensional Packing Algorithms


Idea: Discusses about the worst case scenario and bound check for the algorithm. Important for implementation.

Publication: SIAM journal on computing, Vol.3, issue 4

Date: 13 July 2006

Pages: 299–325

URL: http://dx.doi.org/10.1137/0203025
The paper introduces different performance measures like First Fit (FF), Best Fit (BF), First Fit Decreasing (FFD) and Best Fit Decreasing (BFD).

Though we know that sometimes a solution for a hard problem might exist, we do not know the time it might take to compute. So, in the best guess to solve the problem in minimum amount of time, should find the right solution be the criteria or finding a good solution.

An optimal solution does exist, but in exponential space.

In other words, to know an optimal solution for such NP hard problems, like bin-packing, it would exponential time to compute.

More often, we do not want/need to waste the resources to find that optimal solution.

Hence, the following paper, presents a benchmark of the above mentioned algorithms. It denotes the worst case performance of each, and relays the fact, that the user take these bounds into consideration while designing/working out a solution.
Worst-Case Performance Bounds for Simple One-Dimensional Packing Algorithms

- Heuristics are important in optimal solution problems, because they give a boundary/limit to which one should extend computation.

- For example, the paper provides a limiting bound for first-fit and best-fit algorithm which is $1.7L^* + 2$ bins (where $L^*$ is the optimal no. of bins).

- What we intend to take from this paper - The most important and necessary factor that will help us in working out the computation for the problem, would the bounds (limits) of the algorithms that we are investigating.

- As, the paper has given a generic formula for the no. of bins for a problem (given we know the optimal solution), it helped us understand other alternatives for a good solution but yet not an optimal solution.
Heuristics for Vector bin packing

Authors: Rina Panigrahy, Kunal Talwar, Lincoln Uyeda and Udi Wieder

Idea: Discusses variants of FFD based heuristics and benchmark performance

Publication: Microsoft research

Year: 2011

Heuristics for Vector bin packing

It discusses a number of algorithms like –

- FFD bin centric, Bubble search, Norm based greedy approach.

Bin centric FFD –
- This considers only one bin at a time and tries to fit largest of the remaining items in the bin.
- If no more items can be placed in the bin, the bin is shut and a new bin is opened and a shut bin is never revisited.

Bubble Search –
- Of all the possible items that can be fit in a given bin, it picks the kth largest with a probability $1-p^k$ where p is constant.

The paper talks about multi-dimensional bin packing algorithms which are not related to this context.
Progress

- Researched about two other papers
- Idea and pseudo code for sequential algorithm using an iterative approach rather than a recursive approach.
Pseudo Code

110 -> 111->112->113->121->122->123........->333

for 3 weights

```python
def getMin(weights):
    nWeights = len(weights)
    minBins = nWeights
    currPerm = [0] * nWeights
    bestPerm = currPerm
    make all values in currPerm array '1' except at index '0', which should be made '0'
    for i in 1..nWeights
        if (++currPerm[0] > nWeights)
            adjust(currPerm)
            if (check(currPerm) and reqBins(currPerm) < minBins)
                bestPerm = currPerm
                minBins = reqBins(currPerm)
        end
    end
    return bestPerm
```
Pseudo Code

```plaintext
check(perm[])
    bin[] = int[len(perm)]
    set all bin values to zero
    for i in 1..len(perm)
        if((bin[perm[i]-1]+= weight[i]) > 1)
            return false
    end

end
return true
end
```
adjust(perm[])  
  i = 0  
  while (i < len(perm) && perm[i] > nWeights)  
    perm[i] = 1  
    ++perm[i+1]  
    ++i  
  end  
end
reqBins(perm[])  
    binCount = 0  
    boolean binUsed = boolean[len(perm)]  
    make all values in binUsed false  
    for each val in perm  
        if not binUsed[val]  
            binUsed[val] = true;  
            ++binCount  
        end  
    end  
end  
return binCount
Thank you!

Any questions?