Flexible Demand Assignment Problem

*@**Fan Wang** and ***Andrew Lim** and +**Hong Chen** [1]

Abstract. This paper proposed a state-of-the-art local search for solving Flexible Demand Assignment problem (FDA) which considers the balance between revenue and cost in demand assignment. Different than the published studies, our research splits the FDA problem into three core subproblems as operators for neighborhood construction. The three specified subproblems One Bin Repack, Two Bins Repack and Unpack are proposed completely based on mathematical modelling, computational complexity, executive conditions and greedy solving methods. Benchmark experimental results have shown that the proposed local search improved to the best published heuristics by 2.34%.

1 Introduction

In business operations and manufacturing societies, the demand assignment model is quite popular and widely applied. Considering the various configurations of products and raw materials, the operations planners have to seek the optimal assignment results to minimize the cost. For instance, the well-know bin packing problem [3] studies how to use the minimal number of bins to cover all required items satisfying the capacity constraint that the total size of the items in each bin cannot exceed the corresponding fixed capacity of that bin.

However, in most real applications in manufacturing and supply chain management, the planner must incorporate the revenue versus cost trade-off in the production planning decision, namely *Flexible Demand Assignment (FDA)*. The task of FDA is to exploit demand flexibility during production planning to optimize both revenues and costs. Compared with the classical demand assignment model, such as the bin packing problem, each item in the FDA has a flexible demand rather than the fixed demand. Under the capacity constraint that the total size of the items in each bin cannot exceed the corresponding fixed capacity of that bin, the goal of FDA is to find the optimal net contribution to balance the revenues and costs, rather than to minimize costs only. The FDA model generalizes both the well-know capacitated plant location problem with single-source constraints [2] [6] and the bin packing problem [8] [7]. It is therefore NP-hard. The standard bin packing and capacitated plant location problem assume fixed item size or fixed demand, whereas the FDA model accounts for expandable items or flexible demand.

The FDA problem was first studied by Barlakrishnan and Geunes. The FDA was formulated as a profit-maximizing mixed-integer programming model with an embedded packing subproblem using "expandable" items [1]. A lot of techniques in operation research have been proposed to solve the FDA, which include a Lagrangian relaxation scheme that decomposes the FDA into a set of knapsack subproblems, Lagrangian based heuristic, bin packing based heuristic and linear programming routing heuristic.

This paper proposes a new local search under a specified neighborhood to solve the FDA problem efficiently. Compared with the published methods, the new local search heuristic is simple, more accuracy and has a shorter running time. In our research, we first studied a key component subproblem of the FDA - Single Bin FDA (SBFDA). Then, three specified operators for neighborhood construction were discussed in terms of modelling, executive conditions and solving methods. A state-of-the-art local search heuristic was then developed under these three operators. The experimental results on benchmark data have shown that the proposed simple local search outperforms all the published methods significantly.

2 Problem Statement

We state the FDA problem formally by the following notations. Let I be the set of items and J be the set of bins. For each item $i (i \in I)$, R_i represents the constant unit revenue for i. l_i, u_i denote the minimum (lower) and maximum (upper) size of item i. For each bin $j (j \in J)$, C_j denotes the fixed cost while W_j means its capacity. We also denote r as the unit recycling cost if the capacity of a bin is not fully occupied. Given the above parameters of item set I and bin set J, the task of the FDA is to decide which bin to use, which items to assign to each selected bin, and for each item, what item size to produce within its specified size range. The FDA maximizes the net contribution to profit, defined as the total revenue from items less the cost and the recycling costs of the used bins, whose objective function is formulated as follows:

(FDA) Maximize

$$\sum_{i\in I}\sum_{j\in J} R_i s_{ij} - \sum_{j\in J} C_j z_j - r\sum_{j\in J}(W_j z_j - \sum_{i\in I} s_{ij})$$

subject to

(1) Assignment:

$\sum_{j\in J} x_{ij} = 1, \forall i \in I$

(2) Item size range:

$l_i x_{ij} \le s_{ij} \le u_i x_{ij}, \forall i \in I, j \in J$

(3) Bin capacity:

$\sum_{i\in I} s_{ij} \le W_j z_j, \forall j \in J$, and

(4) Nonnegativity and integrality

$s_{ij} \ge 0, x_{ij} \in \{0,1\}, z_j \in \{0,1\}, \forall i \in I, j \in J$

[1] *Department of Industrial Engineering and Engineering Management, Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong. + Department of Computer Science and Technology, Tsinghua University, Beijing, China. @ Corresponding Author, email: fanwang@ust.hk

where there are three sets of decision variables:

(1) Bin selection: $z_i = 1$ if bin j is used, 0 otherwise;

(2) Item-to-bin assignment: $x_{ij} = 1$ if item i is assigned to bin j, 0 otherwise;

(3) Item sizing: s_{ij} = size of item i assigned to bin j.

If the minimum size is equal to the maximum size for each item, e.g. $l_i = u_i, \forall i \in I$, the FDA problem is reduced to the capacitated facility location problem with the restriction that each customer must be assigned to a single facility. Furthermore, the FDA can be reduced to a one dimension bin packing problem when all fixed-sizing items have the same unit revenue and all bins are identical in cost and capacity. Hence, FDA is an NP-hard problem because of the NP-hardness of the bin packing problem [4].

3 Single Bin Flexible Demand Assignment

In this section, we will present a special case of FDA, FDA for single bin, and prove its optimal solution.

If there is only one bin, the question of FDA becomes how to pack all the items into a single bin to maximize the net contribution. We call this problem Single Bin Flexible Demand Assignment (SBFDA) and it is formulated as follows:

(SBFDA) Maximize
$\sum_{i \in I} R_i s_i - C - r(W - \sum_{i \in I} s_i)$
subject to
Item size range:
$l_i \leq s_i \leq u_i, \forall i \in I$, and
Bin capacity:
$\sum_{i \in I} s_i \leq W$

We present a greedy algorithm to obtain the optimal solution for SBFDA in Algorithm 1. The input instances are split into three cases: (1) if the capacity of the bin is bigger than the total upper size of all the items, we expand all the items to their upper sizes to obtain the revenue in maximum; (2) if the capacity of the bin is smaller than the total lower size of all items, we return the output "Impossible", which means it is impossible to pack all the items into the bin; (3) in the third case, we first pack all the items into the bin by their lower size. Then the size of each item is expended one by one from its lower size to its upper size, in the order of their revenue, until the bin is fully packed.

PROPOSITION 1. The solution obtained by algorithm Greedy-SBFDA is the optimal one for SBFDA.

Proof: Figure 1 illustrates the proof. We proof it by contradiction as follows. For an instance $\{\{R_i\}, C, W, r, \{l_i, u_i\}\}$, the optimal solution is denoted as $\{s^*\}$. Since $\{s^*\}$ is not obtained by the method of algorithm Greedy-SBFDA where items are resized in order of their revenues, there should be two items i and j satisfying $R_i > R_j$ and $s_i^* - l_i < s_j^* - l_j$. Then, we can modify s_i^* and s_j^* to obtain a better solution than $\{s^*\}$, i.e. $s_i' = min\{u_i, s_i^* + (s_j^* - l_j)\}$ and $s_j' = s_i^* + s_j^* - s_j'$. Therefore, we have $R_i s_i' + R_j s_j' > R_i s_i^* + R_j s_j^*$. In the above modification, since the total size of all the items keeps unchange, we get a better net contribution after resizing items i and j. Hence, $\{s^*\}$ is not the optimal solution for SBFDA.

Algorithm 1 Greedy-SBFDA

```
if $\sum_{i \in I} u_i \leq W$ then
  return $\{s_i = u_i\}, \forall i \in I$;
end if
if $\sum_{i \in I} l_i \geq W$ then
  return "Impossible";
end if
if $\sum_{i \in I} l_i \leq W < \sum_{i \in I} u_i$ then
  set $s_i = l_i, \forall i \in I$;
  $I' = sort(I, R_i)$;
  $SumSize = \sum_{i \in I} s_i$;
  while $SumSize < W$ do
    $i = Pop(I')$;
    if $SumSize + (u_i - l_i) \leq W$ then
      $s_i = u_i, SumSize = SumSize + (u_i - l_i)$;
    else
      $s_i = s_i + (W - SumSize), SumSize = W$;
    end if
  end while
  return $\{s_i\}, \forall i \in I$;
end if
```

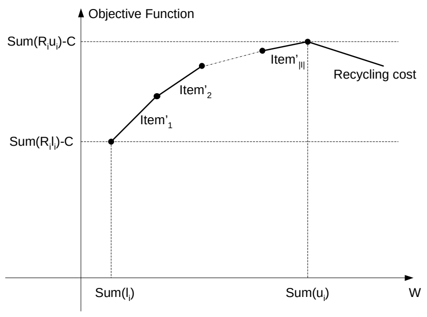

Figure 1. Proof of PROPOSITION 1

4 Neighborhood

In this section, three efficient operators are created for neighborhood construction of search. In fact, they are the three core subproblems of FDA. Their mathematical modelling, computational complexity, executive conditions and solving methods will be discussed.

4.1 One Bin Repack

One Bin Repack (1-Repack) resizes all the items within one bin (denoted as "Bin 1") to obtain the maximum net contribution, illustrated in Figure 2. The goal of 1-Repack is to modify the size of each item within its specified size range to increase the total revenue while the recycling cost is decreased. It is clear to see that 1-Repack is the same as SBFDA which can be solved by Algorithm 1 optimally in $O(|J|)$ time. The executive condition of 1-Repack is $\sum_{x_i 1} u_i > W_1$ in the current solution. Otherwise, the repack gives no added-value to the net contribution if all the items have been expanded to their upper size at the beginning.

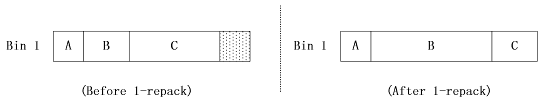

Figure 2. 1-Repack

4.2 Two Bins Repack

Illustrated by Figure 3, the Two Bins Repack (2-Repack) operator includes two steps: first, we unpack all the original items from two bins (denoted as "Bin 1" and "Bin 2"); second we repack all the unpacked items into these two empty bins to obtain the maximum net contribution. The executive condition of 2-Repack is either $\sum_{x_{i1}=1} u_i > W_1$ or $\sum_{x_{i2}=1} u_i > W_2$. Otherwise, such a repack gives no added-value to the net contribution if all the items have been resized to their upper sizes in the beginning.

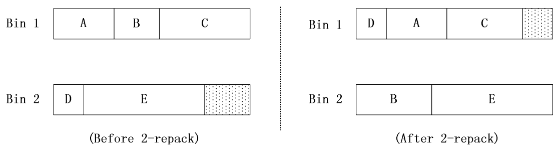

Figure 3. 2-Repack

The goal of the 2-Repack operator can be modelled by the following mix-integer programming model.

(2-Repack) Maximize

$$\sum_{i\in I}\sum_{j=1}^{2} R_i s_{ij} - \sum_{j=1}^{2} C_j z_j - r\sum_{j=1}^{2}(W_j - \sum_{i\in I} s_{ij})$$

subject to

Assignment:

$x_{i1} + x_{i2} = 1, \forall i \in I$

Item size range:

$l_i x_{ij} \le s_{ij} \le u_i x_{ij}, \forall i \in I, j = 1, 2$

Bin Capacity:

$\sum_{i\in I} s_{ij} \le W_j, \forall j = 1, 2$, and

Nonnegativity and integrality

$s_{ij} \ge 0, x_{ij} \in \{0, 1\}, z_j \in \{0, 1\} \forall i \in I, j \in J$

PROPOSITION 2: The problem 2-Repack is NP-hard.

Proof: We set a special instance to reduced the 2-Repack problem to a Partition-2 problem [4]. Assume it has fixed item size, i.e., $l_i = u_i = k_i (\forall i \in I)$ and unique bin capacity $W_1 = W_2 = \frac{\sum_{i\in I} k_i}{2}$, the problem becomes that how to assign these sizes ($\{k_i\}$) into two parts (W_1 and W_2) such that $\sum_{i\in Bin1} k_i = \sum_{i\in Bin2} k_i$ exactly. Because of the NP-hardness of the Partition-2 problem, the 2-Repack problem is also NP-hard.

We will develop a two-stage algorithm to solve the 2-Repack problem: 2-Partition + 1-Repack. The high level part 2-Partition focuses on partitioning items into two parts. Once the partition is determined, 1-Repack is applied to obtain the best resizing solution for the items in each bin. Here, we have two strategies for the high level 2-Partition stage:

1. Exact Algorithm: using DFS with pruning to enumerate all possible partition solutions;
2. Local Search Heuristic: a local search heuristic with "opt-2" neighborhood where two items in separate bins are exchanged to produce a new solution in each iteration.

4.3 Unpack

Suppose there are N bins originally ($N = |J|$). The Unpack operator first unpacks all the items in one bin (denoted as "Bin 1"). Then it repacks all the above unpacked items (denoted as $I^* = \{i | i \in Bin1\}$) to the $N - 1$ remaining bins (denoted as $J' = J - \{1\}$ including "Bin 2", "Bin 3", ..., "Bin N") and resizes both the adding items I^* and the original remaining items $I - I^*$ for each bin. Different from operators 1-Repack and 2-Repack, the Unpack operator can reduce the number of selected bins. Figure 4 illustrates the Unpack operator.

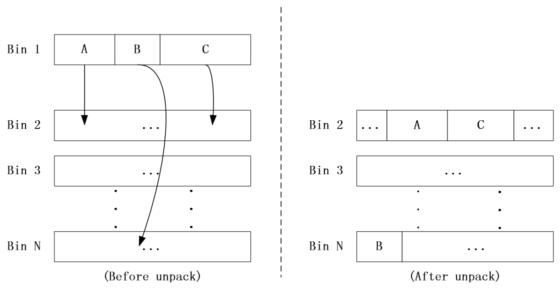

Figure 4. Unpack

The goal of Unpack is to determine which remaining bin should receive each unpacked item and how to resize all the items to get the maximum net contribution. The following mix-integer programming model formulates the problem of Unpack.

(Unpack) Maximize

$$\sum_{i\in I}\sum_{j\in J'} R_i s_{ij} - \sum_{j\in J} C_j - r\sum_{j\in J'}(W_j - \sum_{i\in I} s_{ij})$$

subject to

Assignment:

$\sum_{j\in J'} x_{ij} = 1, \forall i \in I^*$

Item size range:

$l_i x_{ij} \le s_{ij} \le u_i x_{ij}, \forall i \in I, j \in J'$

Bin Capacity:

$\sum_{i\in I} s_{ij} \le W_j, \forall j \in J'$, and

Nonnegativity and integrality
$s_{ij} \geq 0, \forall i \in I, j \in J'$ and $x_{ij} \in \{0,1\}, \forall i \in I^*$

PROPOSITION 3. The problem Unpack is NP-hard.

Proof: We set a special instance that all items are located in "Bin 1" and the remaining $N-1$ bins are empty originally, i.e., $x_{i1} = 1, x_{ij} = 0, \forall i \in I, j \in J'$. Hence, the problem Unpack is reduced to our original FDA problem whose NP-hardness has been proved. Therefore, problem Unpack is NP-hard.

A greedy heuristic is proposed to solve the Unpack problem, described in the following four steps, where $W'_j = W_j - \sum_{i \in I} x_{ij} l_{ij}, \forall j \in J'$ is defined as the potential left space for each remaining used bin.

[Step 1]: fix all items to their lower sizes, i.e., $s_{ij} = l_i x_{ij}, \forall i \in I$ and $j \in J$;

[Step 2]: sort the remaining bins of J' in the decreasing order of their potential left size $W'_j, \forall j \in J$;

[Step 3]: use the Best-Fit Decreasing scheme from solving the one dimension bin packing problem to add unpacked items of I^* one by one into the remaining bins J';

[Step 4]: resize all items in each bin by using operator 1-Repack.

The above heuristic consists two parts. First, the sizes of all the items are fixed to their lower sizes and the Best-Fit scheme is applied to the assignment of the unpacked items greedily. Then, for each bin, the size of their own items are resized optimally by SBDFA.

Here, we will also discuss a special case of Unpack where a new constraint is added so that each bin can be inserted into at most one item only. We call it Single Unpack which is formulated by the above mix-integer programming model plus the following Single Assignment constraint:

Single Assignment:
$\sum_{i \in I*} x_{ij} \leq 1, \forall j \in J'$

PROPOSITION 4. Problem Single Unpack can be solved optimally in polynomial time if the number of remaining bins is not smaller than the number of unpacked items, i.e., $|J'| \geq |I^*|$.

Proof: We model the problem Single Unpack as a Maximum Weighted Bipartite Matching [5]. An undirected weighted graph $G = (V, E)$ is built as shown Figure 5. The vertices set is the union of two disjoint sets - $V = I^* \cup J'$ where I^* represents the set of unpacked items and J' means the set of remaining bins. Moreover, the weight of the edge e_{ij} between the vertex $i (i \in I^*)$ and the vertex $j (j \in J')$ is assigned by the added-value of net contribution if item i is inserted into bin j. It can be calculated by SBFDA directly. Numerous polynomial-time algorithms have been published to find the optimal solution for Maximum Weighted Bipartite Matching in polynomial time, such as the classical Hungarian Algorithm. Hence, the problem Single Unpack can be solved optimally in polynomial time.

5 Local Search

In this section, we will describe the local search heuristic algorithm for solving FDA.

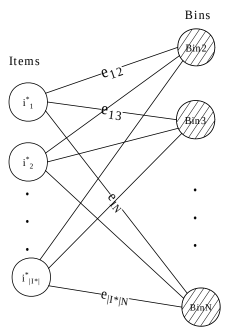

Figure 5. Single Unpack modelled by Weighted Bipartite Matching Graph

5.1 Initial Solution

In the part of initial feasible solution generation, there are two tasks - to determine the initial fixed size of each item and to which bin it should be assigned. We have two ways to determine the initial size of each item:

1. Random Distribution (RD): assign a random size to each item between its specified upper size and lower size;
2. Uniform Distribution (UD): generate several initial instances of item size on uniform distribution, i.e., $s_{ij} = q\frac{(u_i - l_i)}{p} + l_i, q = 0, 1, \cdots, p$, where p is the number of instances.

Moreover, three methods are applied to assign the items with fixed size to the bins:

1. Best-Fit Decreasing (BFD): sort the items in decreasing order of size, and sequentially assign them (in the sorted order) to the open bin that has the least capacity remaining after packing the item. If the current item does not fit in any open bin, a new bin is opened.

2. Worst-Fit Decreasing (WFD): sort the items in decreasing order of size, and sequentially assign them (in the sorted order) to the open bin that has the most capacity remaining after packing the item. If the current item does not fit in any open bin, a new bin is opened.

3. Random-Fit Decreasing (RFD): sort the items in decreasing order of size, and sequentially assign them (in the sorted order) to any random open bin that has enough capacity to pack the item. If none of the open bins have enough capacity to pack the current item, a new bin is opened.

5.2 Algorithm

The local search algorithm is described in Algorithm 2. For each initial solution, the local search is iterated many times. In each iteration, first, the 1-Repack and 2-Repack operators are run alternately to improve the the solution. If no improvement could be found for the current solution by these two operators, the Unpack operator is then applied.

6 Computational Results

In this section, we will show the assessment of effectiveness of our local search for solving FDA, by comparing it to the

Algorithm 2 Local search for solving FDA

```
for $m = 1$ to $M$ do
  {$M$ is the number of initial solutions}
  call initial solution generation($m$);
  repeat
    repeat
      call 1-Repack operator;
      call 2-Repack operator;
    until No improvement found;
    call Unpack operator;
  until No improvement found;
end for
return CurrentBestSolution;
```

published methods under the benchmark data (Balakrishnan 2003).

The benchmark data was set up to evaluate the performance of the heuristics for a wider range of parameter variation. There are five sets of data in total to consider the following five impacts of different parameters on the performance of various components in the solving methods:

Set 1: Nonidentical item sizes with one FDA instance;

Set 2: Large item size range with fixed bin capacity;

Set 3: Large variation of item size range with fixed median of size and bin capacity;

Set 4: Variation of bin capacity with fixed distribution of item size;

Set 5: Variation of unit revenue

Each set of data has 15 random instances - 5 instances each of small (20 items), medium (40 items), and large (60 items) problems. While, the number of available bins is from 5 to 24.

Based on the computational results supported by Geunes on the same benchmark data (Balakrishnan 2003), Table 1 presents the average percentage that our local search heuristic outperforms the published heuristics - CPLEX (a widely used standard branch-and-bound solving software), LP-R (Linear Programming Rounding heuristic), BPEI (Bin-Packing heuristic with Expandable Items) , LR (Lagrangian Relaxation) and BestP (the best result among LP-R, BPEI and LR). The last row shows the overall improvement gap for the whole benchmark data. Here, the results of CPLEX was obtained based on the FDA mix-integer model within 4-hour running time limit.

From the performance results of the four published heuristics, in summary, the solution found by CPLEX is better than the other three published heuristics. However, the accuracy of the solutions found by CPLEX decreases as instance size increases. It is due to the 4-hour running time limit. When instance size increases, the branch-and-bound method cannot generate enough nodes in the search tree. From the heuristic of the bin packing problem, BPEI has the worst performance in Set 1 where there are different bin capacities in the instances. Different from BPEI, the performance of LR remains constant even as we vary the parameters. From the experimental results, our local search outperforms the above four methods, improving CPLEX by 0.72%, LP-R by 7.32%, BPEI by 4.28%, LR by 2.34% and BestP by 1.57% on average. In addition, the average running time for the local search is within minutes. Moreover, the results produced by the proposed local search are close extremely to the upper bound results.

We also compare the performance of the various methods of initial solution generation for our local search. In terms of the initial assignment of items, RFD is better than both the BFD and the WFD heuristics from bin packing. We can clearly understand the difference between bin packing and FDA in this point of view that a more diverse initial solution will benefit the search of FDA. Similarly, RFD-RD outperforms RFD since much more diversity will be added to the initial solution.

Table 1. Improvement Gap (%)

	Improve to				
Data	CPLEX	LP-R	BPEI	LR	BestP
Set 1	0.52	10.56	7.93	2.70	2.70
Set 2	0.14	7.45	6.33	2.08	1.65
Set 3	0.38	6.36	2.35	0.76	0.62
Set 4	1.46	7.59	0.96	4.25	0.90
Set 5	1.19	4.49	2.79	1.96	1.66
Overall	0.72	7.32	4.28	2.34	1.57

7 Conclusions

FDA problem is an interesting and useful extension of the bin packing problem. It considers the balance between revenue and cost in demand assignment. Different than the published studies, our research focus on the state-of-the-art approximation algorithm by splitting the problem into many core subproblems as operators for neighborhood construction. The three specified subproblems One Bin Repack, Two Bins Repack and Unpack are proposed completely based on mathematical modelling, computational complexity, executive conditions and greedy solving methods. Benchmark experimental results show that the above neighborhood helps the local search to find an accurate solution effectively and significantly.

REFERENCES

[1] A. Balakrishnan and J. Geunes, 'Production planning with flexible product specifications: an application to specialty steel manufacturing', *Oper. Res.*, **51**(1), 94–112, (2003).

[2] J. Barcelo and J. Cansanovas, 'A heuristic lagrangian algorithm for the capacitated plant location problem', *Eur. J. Oper. Res.*, **15**, 212–226, (1984).

[3] M.R. Garey E.G. Coffman and D.S. Johnson, *Approximation algorithms for bin-packing - An updated survey*, 49–106, Springer-Verlag, New York, 1984.

[4] M.R. Garey and D.S. Johnson, W.H. Freeman and Co., New York, 1979.

[5] D.S. Johnson and C.C. McGeoch, *Network flows and matching: First DIMACS implementation challenge.*

[6] J.G. Klincewicz and H. Luss, 'A lagrangian relaxation heuristic for capacitated facility location with single-source constraints', *J. Oper. Res. Soc.*, **37**(5), 495–500, (1986).

[7] R. Korf, 'An improved algorithm for optimal bin packing', in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-03)*, pp. 1252–1258, Acapulco, Mexico, (August, 2003).

[8] R. Korf, 'A new algorithm for optimal bin packing', in *Proceedings of the National Conference on Artificial Intelligence, AAAI-02*, pp. 731–736, Edmonton, Alberta, Canada, (July, 2002).