Management Science and Financial Engineering
Vol 19, No 1, May 2013, pp.25-28 http://dx.doi.org/10.7737/MSFE.2013.19.1.025
ISSN 2287-2043 | EISSN 2287-2361 |

Inverse Bin-Packing Number Problems: Polynomially Solvable Cases

Yerim Chung*
School of Business, Yonsei University

(Received: November 24, 2012 / Revised: November 30, 2012 / Accepted: December 2, 2012)

ABSTRACT

Consider the inverse bin-packing number problem. Given a set of items and a prescribed number K of bins, the inverse bin-packing number problem, IBPN for short, is concerned with determining the minimum perturbation to the item-size vector so that all the items can be packed into K bins or less. It is known that this problem is NP-hard (Chung, 2012). In this paper, we investigate some special cases of IBPN that can be solved in polynomial time. We propose an optimal algorithm for solving the IBPN instances with two distinct item sizes and the instances with large items.

Keywords: Inverse Combinatorial Optimization, Bin-Packing Problem, Computational Complexity

* Corresponding Author, E-mail: yerimchung@yonsei.ac.kr

1. INTRODUCTION

The *minimum bin-packing problem* is one of the most widely studied problem in combinatorial optimization. Given a set of items, the task is to pack all the items into a minimum number of identical bins so that the sum of the item sizes in each bin does not exceed the bin capacity. It is well known that the minimum bin-packing problem is NP-hard in the strong sense (Garey and Johnson, 1979). In this research note, we consider an interesting variant of the minimum bin-packing problem defined as follows. Given a set X of n items, each having size x_i, $i \in \{1, \cdots, n\}$ (let $x=(x_1, x_2, \cdots, x_n)$ be a vector with n item sizes), fix a positive integer bin number K. The objective is to determine a new item-size vector x' such that all the items can be packed into K bins or less, and the discrepancy from the original item-size vector x is minimum under a fixed norm. This problem can be seen as an *inverse counterpart* of the minimum bin-packing problem with the feasibility criterion, called the *inverse bin-packing number problem* (Chung, 2012).

Given an instance of an optimization problem and a prescribed value K, *inverse number problems* are concerned with determining the minimum perturbation to the original instance so that the resulting instance admits an optimal solution of value K or less. Recently, there are several papers discussing inverse number problems, see e.g., (Chung, 2012; Chung *et al*., 2008, 2013; Heuberger, 2004). In particular, the inverse bin-packing number problem, IBPN for short, has been studied in (Chung, 2012), and the status of complexity and approximability is settled for the inverse bin-packing number problem. In fact, it is shown in (Chung, 2012) that IBPN under the L_p-norm, $p \in \{1, \cdots, \infty\}$, is NP-hard and that IBPN under the L_1-norm admits an input dependent, asymptotic approximation algorithm.

The intractability, i.e., the NP-hardness of IBPN provides a motivation for detecting some particular instances of IBPN that can be polynomially solved. In this research note, we extend the work of Chung (2012) by investigating the polynomially tractable cases of IBPN under the L_1-norm. We restrict our attention to the instances with a constant number of different item sizes, and the instances with a lower bound for item sizes.

This note is organized as follows. Section 2 deals with IBPN instances where there are only two different item sizes. Section 3 is devoted to IBPN instances where item sizes are all greater than 1/3 of the bin capacity. We propose an efficient algorithm for solving each of

these instances. Section 4 concludes the paper by providing research ideas for future research.

2. IBPN WITH TWO DISTINCT ITEM SIZES

In this section, we deal with IBPN instances where there are only two distinct item sizes, either x_1 or x_2. Given K bins of capacity B and n items of size x_1 or x_2, the problem, denoted by $\text{IBPN}_{\{x_1, x_2\}}$ is concerned with decreasing some item sizes so that a feasible packing with K bins is ensured and the modification is minimum. We assume that $0 < x_1 < x_2 \le B$ and that new item sizes should have a value of 0, x_1 or x_2. Given n items, let n_1 and n_2 be the number of items of size x_1 and x_2, respectively. Let i, $1 \le i \le n_1$, be the number of items whose size is changed from x_1 to 0. We denote by an integer j, $1 \le j \le n_2$, the number of items whose size is changed from x_2 to 0 and by h, $1 \le h \le n_2$ the number of items whose size is changed from x_2 to x_1. Then, $\text{IBPN}_{\{x_1, x_2\}}$ can be written as the problem of finding a pair (l_1, l_2) of integers such that $l_1 = n_1 - i + h$ items of size x_1 and $l_2 = n_2 - j - h$ items of size x_2 can be packed into K bins or less and the total cost is minimum. The cost for modification is given by $x_1(n_1 - l_1) + x_2(n_2 - l_2)$. Let us denote by (n_1, n_2) and (l_1, l_2) the original and new instance of IBPN, respectively. A new instance (l_1, l_2) of IBPN makes sense if and only if it satisfies the following conditions: (*i*) $l_2 \le n_2$ and (*ii*) $l_1 + l_2 \le n_1 + n_2$.

Let $Q = \{(m_1, m_2) \in \mathbf{N}^2 \mid m_1 x_1 + m_2 x_2 \le B\}$ be the set of feasible one-bin packings; $(m_1, m_2) \in Q$ means that m_1 items of size x_2 and m_2 items of size x_2 can be packed in a same bin. Let P be the integer hull of the points in Q. Then, all vertices of P can be computed in $O(\log B)$ time by using Harvey's Algorithm (Harvey, 1999).

By using the following result (McCormick *et al*., 2001), we devise a polynomial-time algorithm for solving $\text{IBPN}_{\{x_1, x_2\}}$.

Corollary 1: (McCormick *et al.* (2001)) *Consider an instance of* IBPN *with l_i items of size x_i, $i \in \{1, 2\}$, and K bins of capacity B. This instance is feasible if and only if the point $(\frac{l_1}{K}, \frac{l_2}{K})$ is in P where P is the integer hull of the points in $Q = \{(m_1, m_2) \in \mathbf{N}^2 \mid m_1 x_1 + m_2 x_2 \le B\}$.*

We briefly describe how our algorithm works.

1. For any fixed $l_2 \in \{0, \cdots, n_2\}$, determine the set M of the points contained in P by using Harvey's Algorithm (Harvey, 1999): $M = \{(\frac{l_1}{K}, \frac{l_2}{K}) \in P\}$.
2. For any point $(\frac{l_1}{K}, \frac{l_2}{K}) \in M$, if $l_1 + l_2 > n_1 + n_2$ then delete it from M.
3. Compute the cost $x_1(n_1 - l_1) + x_2(n_2 - l_2)$ for each point in M.
4. Choose the point $(\frac{l_1^*}{K}, \frac{l_2^*}{K}) \in M$ which realizes the minimum modification.

Clearly, the above algorithm optimally solves $\text{IBPN}_{\{x_1, x_2\}}$ in fact, it computes exhaustively all possible solutions and returns the one with minimum value.

Proposition 1: $\text{IBPN}_{\{x_1, x_2\}}$ *can be solved in $O(n^2 + \log B)$ time, where n is the number of the items and B is the bin capacity.*

3. IBPN WITH LARGE ITEMS

In this section, we consider IBPN instances where item sizes are all greater than $B/3$ (B is the bin capacity). In such particular instances, packing three items in a same bin always exceeds the bin capacity. For a fixed constant K, if there are more than $3K$ items, then IBPN with large items is trivially solved, and the total modification is given by $\sum_{i=1}^{n} x_i - KB$. In the case where there are $3K$ items or less, a constant number of item assignments[1] are possible, and the complete enumeration solves the problem in constant time. Here, we propose an efficient algorithm for solving IBPN with large items, faster than the complete enumeration.

The algorithm we propose is implemented in four steps: *First Fit Decreasing-fitting,*[2] *Greedy adding*, *Bin grouping*, and *Item exchanging*.

1. *FFD-fitting*:
1-1. Compute by using FFD an optimal[3] bin-packing for the given instance: let λ (FFD) be the number of bins used by FFD.
1-2. If $K \ge \lambda$ (FFD), then no modification is needed.
1-3. If $K < \lambda$ (FFD), then select K bins such that the total sum of the sizes of the items packed is as large as possible. Let $Y \subset X$ be the set of items that cannot be packed in such K bins. Then, any of item in Y cannot be added to a bin without exceeding the bin capacity.
1-4. Return the K selected bins; we call this solution the *FFD-solution*.

2.*Greedy adding*:
2-1. Assign one additional item to some bins of FFD-solution (even if bins are overloaded) in such a way

1) There are $C_3^{6K-2n} \times C_2^{3n-6K}$ item assignments to K bins.
2) First Fit Decreasing, FFD for short, is an approximation algorithm for the bin-packing problem. It first sorts the items in non-increasing order with respect to their size. According to this order, FFD takes an item and put it in the first available bin of the list. If there is no such a bin, FFD opens a new bin.
3) It is known that FFD optimally solves the minimum bin-packing problem if item sizes are all greater than $B/3$ (see, Ausiello *et al*., 1999).

that the total sum of remaining spaces in K bins becomes minimum.

2-2. Return the solution obtained; we call this solution the *FFD-greedy solution.*

3-3. Compute the total excess of FFD-greedy solution.

3. *Bin grouping*:

3-1. Partition K bins of FFD-greedy solution into 4 groups, $\tilde{A}$, $\tilde{B}$, $\tilde{C}$, and $\tilde{D}$, according to their contents.

$$\tilde{A} = \{a = \lfloor a_1, a_2 \rfloor \mid a_1 + a_2 > B,\ a_1 \geq a_2 > B/3\};$$
$$\tilde{B} = \{b = \lfloor b_1, b_2 \rfloor \mid b_1 + b_2 < B, b_1 \geq b_2 > B/3\};$$
$$\tilde{C} = \{c = \lfloor c_1, c_2, c_3 \rfloor \mid c_1 + c_2 + c_3 > B, c_1 \geq c_2 > B/3, c_1 \geq c_3 > B/3\};$$
$$\tilde{D} = \{d = \lfloor d_1 \rfloor \mid d_1 > B/3\}.$$

$\tilde{A}$ is the set of bins containing two items whose sum of sizes exceeds the bin capacity B. $\tilde{B}$ is the set of bins with two items whose sum does not exceed B and $\tilde{C}$ is the one with three items. $\tilde{D}$ is the set of bins containing only one item.

4. *Item exchanging*

4-1. Apply a subroutine, Exchange (b, c).

4-2. Apply a subroutine, Exchange (a, b).

The algorithms Exchange (a, b) and Exchange (b, c) are devised to improve the use of spaces in bins of $\tilde{B}$, i.e., to minimize the empty spaces remaining in bins of $\tilde{B}$. These two algorithms stop when there is no empty space in bins of type $\tilde{B}$ or there is no pair of bins satisfying the exchange conditions.

Exchange (b, c) is concerned with item exchange between two bins, $b = \lfloor b_1, b_2 \rfloor \in \tilde{B}$ and $c = \lfloor c_1, c_2, c_3 \rfloor \in \tilde{C}$. We exchange c_1 and b_2 if $c_1 > b_2$.[4] Exchange (a, b) executes item exchange between $a = \lfloor a_1, a_2 \rfloor \in \tilde{A}$ and $b = \lfloor b_1, b_2 \rfloor \in \tilde{B}$. If $a_1 \geq b_1$ and $a_2 > b_2$, then we exchange the item a_1 with b_1. Once an item exchange is executed, return to the bin grouping step and repeat this till the termination condition is satisfied. The running time of these algorithms is bounded by the number of bins in $\tilde{B}$.

Here, we provide five lemmas which are useful for proving the optimality of our algorithm. For the conciseness of the paper, we omit the proofs.

Lemma 1: *Given two bins* $b = \lfloor b_1, b_2 \rfloor \in \tilde{B}$ *and* $c = \lfloor c_1, c_2, c_3 \rfloor \in \tilde{C}$, *exchanging the item* b_i, $i \in \{1, 2\}$ *with the item* c_j, $j \in \{1, 2, 3\}$ *improves the solution if and only if* $b_i < c_j$.

Lemma 2: *Given two bins* between $a = \lfloor a_1, a_2 \rfloor \in \tilde{A}$ *and* $b = \lfloor b_1, b_2 \rfloor \in \tilde{B}$, *exchanging the item* a_i, $i \in \{1, 2\}$ *with the item* b_j, $j \in \{1, 2\}$ *improves the solution if and only if* $a_1 \geq b_1$ and $a_2 > b_2$.

Lemma 3: *Any item exchange between two bins belonging to a same group cannot improve the solution obtained in step 3-1.*

Lemma 4: *Item exchange between* $\tilde{D}$ *and any of other groups, and item exchange between* $\tilde{A}$ *and* $\tilde{C}$ *cannot improve the solution.*

Lemma 5: *Item permutation*[5] *among three (or more) bins of* $\tilde{A}$, $\tilde{B}$ *and* $\tilde{C}$ *cannot reduce the cost as much as a direct positive item exchange does.*

Proposition 2: *When there is no item of size B/3 or less, IBPN can be solved in time* $O(n \log n + K)$.

Proof: The optimal inverse bin packing can be obtained if and only if the total bin load in K bins is maximum, i.e., the remaining empty spaces in K bins is minimum. Using the spaces in bins of $\tilde{B}$ to the maximum ensures the optimality of the related solution (see Lemma 3 and 4). In addition, Lemma 1, 2 and 5 imply that there is no item exchange other than the ones given by the procedures Exchange (a, b) and Exchange (b, c) which makes better use of spaces in bins of $\tilde{B}$. So, we conclude that our algorithm optimally solves IBPN with large items.

Let us show that the running time of the subroutines Exchange (a, b) and Exchange (b, c) is bounded by the number of bins in $\tilde{B}$. Consider first Exchange (b, c). A favorable item exchange between $b = \lfloor b_1, b_2 \rfloor \in \tilde{B}$ and $c = \lfloor c_1, c_2, c_3 \rfloor \in \tilde{C}$ can produce either two bins belonging to $\tilde{A}$, or one bin in $\tilde{B}$ and the other in $\tilde{C}$. The number of bins belonging to $\tilde{B}$ reduces by one in the first case while it remains the same in the second case. Let b_2 be an item exchanged with c_1. Then, by construction of the algorithm, b_1 will never return to any bin in $\tilde{B}$. Hence, the subroutine Exchange (b, c) is repeated at most *max* $\{|\tilde{B}|, |\tilde{C}|\}$ times.

Any favorable item exchange between $a \in \tilde{A}$ and $b \in \tilde{B}$ yields two bins belonging to $\tilde{A}$. So, whenever an item exchange between a and b is executed, the number of bins belonging to $\tilde{B}$ reduces by one. Hence, the subroutine Exchange (a, b) is repeated at most $|\tilde{B}|$ times.

Since FFD requires $O(n \log n)$ time, our algorithm runs in time $O(n \log n + K)$.

4. CONCLUDING REMARKS

In this research note, we have investigated some polynomially solvable cases of the inverse bin-packing problem which is in general NP-hard. In particular, we considered IBPN instances with two item sizes, and proposed a polynomial time algorithm for optimally

4) If there are several pairs of bins that satisfy the condition for exchange, then we consider the bins with the largest empty space among the bins in B and takes a bin with smallest b_1, and a bin containing the biggest c_1 among the ones in $\tilde{C}$.

5) Every item permutation with three items among $\tilde{A}$, $\tilde{B}$ and $\tilde{C}$ is made up with two consecutive item exchanges; for example by item exchange between $\tilde{A}$ and $\tilde{C}$, and then item exchange between $\tilde{B}$ and $\tilde{C}$.

solving such instances. However, our algorithm does not work if there are three or more different item sizes. An efficient algorithm for such instances needs to be studied. We also considered IBPN instances with large items (having size greater than $B/3$) and devised an efficient algorithm based on FFD. In our future research, we will show that IBPN restricted to items of bounded sizes is closely related to the problem of finding a minimum cost graph partition into K cliques of bounded sizes. Developing a faster algorithm for IBPN will contribute to the efficient solution for the latter problem.

ACKNOWLEDGEMENT

This work was supported by the National Research foundation of Korea Grant, funded by the Korean government (NRF-2011-330-B00076).

REFERENCES

Ausiello, G., P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccame-La, and M. Protasi, *Complexity and approximation* (*Combinatorial optimization problems and their approximability properties*), Springer, Berlin, 1999.

Chung, Y., "Inverse bin-packing number problems: Hardness and approximation results," *Management Science and Financial Engineering* 18, 2 (2012), 1-4.

Chung, Y., J. F. Culus, and M. Demange, "Inverse booking problems," *In Proceedings of the 2nd Work-shop on Algorithms and Computation*, *WALCOM Lecture Notes in Computer Science*, 4921 (2008), 180-187.

Chung, Y., S. Hong, and M. Park, "Inverse interval scheduling problems," *Manuscript* 2013.

Garey, M. R. and D. S. Johnson, *Computers and Intractability- A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, 1979.

Harvey, W., "Computing two-dimensional integer hulls," *SIAM Journal on Computing* 28, 6 (1999), 2285-2299.

Heuberger, C., "Inverse combinatorial optimization: A survey on problems, methods, and Results," *Journal of Combinatorial Optimization* 8, 3 (2004), 329-361.

McCormick, S. T., S. R. Smallwood, and F. C. R. Spieksma, "A polynomial algorithm for multiprocessor scheduling with two job lengths," *Mathematics of Operations Research* 26, 1 (2001), 31-49.