Applied Mathematics & Information Sciences
*An International Journal*

http://dx.doi.org/10.12785/amis/080517

# Optimization of a Dish Cart Storage using the Two-Dimensional Single Bin Packing Problem

*Kang-Hee Lee*[1] *and A-Ra Khil*[2,*]

[1] Global School of Media, Soongsil University, Republic of Korea
[2] School of Computer Science and Engineering, Soongsil University, Republic of Korea

Received: 20 Aug. 2013, Revised: 17 Nov. 2013, Accepted: 18 Nov. 2013
Published online: 1 Sep. 2014

**Abstract:** Given a set of rectangular items such as dishes, cups, saucers, or forks and a rectangular shelf of a dish cart, the two-dimensional single bin packing problem (2D-BPP) involves orthogonally packing a subset of the items within the shelf such that the sum of the values of the packed items is maximized. If the value of an item is given by its area, the objective is to maximize the covered area of the shelf. Thus, this paper aims to optimize the transport capacity of a serving robot carrying a dish cart. This experiment, the first of its type, proves the feasibility of this endeavor efficiently.

**Keywords:** Two-dimensional single bin packing problem, two-dimensional knapsack problem, serving robot, dish cart storage, rectangular pieces

## 1 Introduction

When we have dinner at a corner table in a crowded restaurant, we want to be served by a kind and experienced waiter/waitress. This person should also be strong, as he/she serves dinner on a heavy dish cart or a tray filled with the ordered food, which can reduce the waiting time as a result. This mechanism can also be applied to a serving robot. If a stronger serving robot carries twice the number of dishes compared to other normal serving robots, it will be able to save time when multiple dishes of food are served. The longer the time taken and the greater the distance covered, the more important storage efficiency becomes when a robot serves in this manner. Thus, this paper formulates this problem as a two-dimensional finite bin packing problem (2D-BPP) or a two-dimensional knapsack problem (2D-KP) and aims to optimize the amount of dish cart storage which the serving robot can carry at one time.

The rest of the paper is composed as follows: Section 2 introduces the existing works related to the contribution of this paper: serving robots and 2D-BPP. In Section 3, we formulate the optimization problem of a dish cart storage using the 2D-BPP. Section 4 describes the structure of the branch-and-bound algorithm [1][2] for finding possible positions for placing items. The experiments in Section 5 are performed to demonstrate the feasibility of the proposed scheme in Sections 3 and 4. The empirical and theoretical analyses follow to investigate its characteristics. Concluding remarks follow in Section 6.

## 2 Related Works

While state-of-the art serving robots focus only on mobility, manipulability, and types of foods to be served, we aim to increase the transport or storage capacity of the dish cart carried by a serving robot. Hence, it is formulated as a two-dimensional finite bin packing problem (2D-BPP) in order to optimize the amount of dish cart storage which the serving robot can carry at one time.

### *2.1 Serving Robots*

The worlds first serving robot is the fascinating Karakuri, which serves tea. It was designed nearly four centuries ago and today remains a remarkable example of Japans keen sense of robotics [3].

* Corresponding author e-mail: ara@ssu.ac.kr

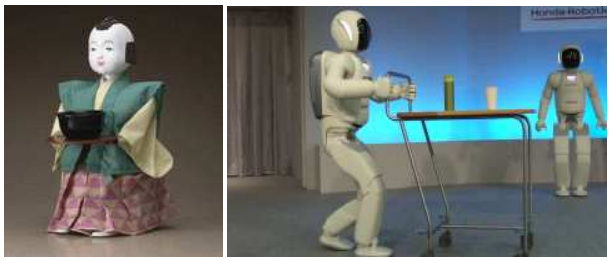


**Fig. 1:** Worlds first Tea serving robot: Karakuri (left) and worlds best universal serving robot: ASIMO (right)

Currently, the all-new ASIMO, developed by Honda, can perform serving tasks, such as carrying a dish cart, picking up a glass bottle and twisting off the cap, or gently holding a soft paper cup while pouring a liquid into it, with great dexterity [4] (Refer to Figure 1). JUSTIN developed by DLR, can reach for a can and grasp it using its Pick and Place operation. It can also handle simple manipulation tasks such as unscrewing a lid [5]. The dON, a wine-serving robot, is specialized for serving many glasses of wines in the sense of mass production [6]. A universal smart serving robot, FURO, can substitute real persons job at restaurant such that can serve coffee, beverage, tea, juice, bakery, etc for restaurant service [7]. A sweet-serving robot, T82, is fully capable of scooping sweets up and placing them into a plastic bag. It can also open a drinks bottle, which may seem easy, but for a robot to grip the plastic bottle and twist the cap off [8].

### *2.2 Two-dimensional Bin Packing Problem*

The Two-Dimensional Bin Packing Problem (2D-BPP) is a generalization of One-Dimensional Bin Packing Problem (1D-BPP). 2D-BPP is NP-hard in the strong sense [9]. We are given a set of rectangular pieces which must be cut from a large set of standardized stock pieces, also called (finite) bins, having height $H$ and width $W$. Each piece $j \in J = \{1, \cdots, n\}$ is characterized by its height $h_j \leq H$ and width $w_j \leq W$, and may not be rotated. A piece will alternatively be denoted by $w_j \times h_j$ (or $H \times W$). The Two-Dimensional Finite Bin Packing Problem (2D-BPP) calls for the determination of orthogonal cutting patterns which provide all the pieces and such that the total number of required stock pieces is minimized [10]. Most of the off-line bin-packing algorithms are of greedy type, and can be classified into two families: one-phase algorithms and two-phase algorithm. The one is to pack the items into the finite bins directly such as Finite Next-Fit (FNF), Finite First-Fit (FFF), Finite Bottom-Left (FBL), Next Bottom-Left (NBL) [11], and Alternate Directions (AD) [12]. The other starts by packing the items into a single strip, i.e., a bin having width and infinite height. In the second phase, the strip solution is used to construct a packing into finite bins. For example, there are Hybrid First-Fit (HFF) [13], Hybrid Next-Fit (HNF) [14], Hybrid Best-Fit (HBF) [1], and Floor-Ceiling (FC) algorithms [12].

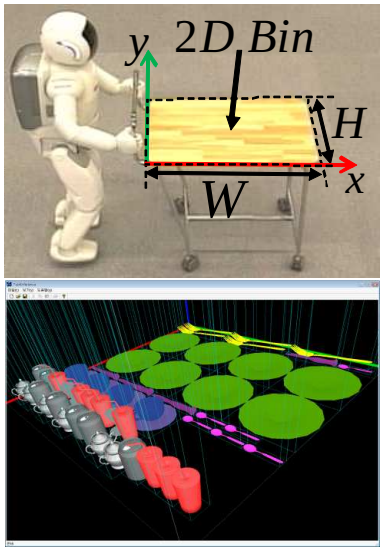



**Fig. 2:** A serving robot carrying a dish cart (top view) and the optimized storage concept (bottom view)

## 3 Problem Formulation

In this section, we model a dish cart storage filling problem (DCFP) with a two-dimensional single bin packing problem (2D-BPP).

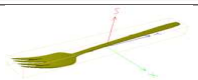
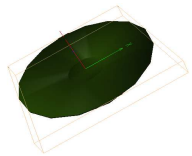
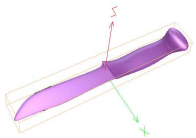
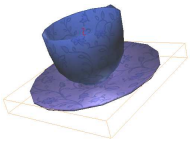
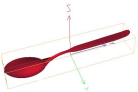
The objective is to find a feasible arrangement of a subset of the items on the shelf that maximizes the total value of the packed items. If the value of an item is given by its area, the aim is to maximize the covered area of the shelf. An arrangement of items, known as a setting plan, is regarded as feasible if each item is placed orthogonally (i.e., parallel to the shelf edges), if each is completely inside the shelf, and if no two placed items overlap [9].

### *3.1 Modeling of a DCFP with a 2D-BPP*

This paper addresses a two-dimensional bin packing problem (2D-BPP) with a set of $n$ small rectangular items and a larger rectangle (bin), which in this study is the shelf of a dish cart having height $H$ and width $W$ (Refer to Figure 2).

As listed in Table 1, an item corresponds to a rectangular piece. Each item $j \in J = \{1, \cdots, n\}$ is characterized by its height $h_j \leq H$ and width $w_j \leq W$, and may not be rotated. A piece (item) will alternatively be denoted by $w_j \times h_j$.

**Table 1:** Item list

| $j$ | 3D shape | $w_j$ | $h_j$ | Description |
|---|---|---|---|---|
| 1 |  | 5 | 32 | FORK |
| 2 |  | 25 | 25 | PLATE |
| 3 |  | 4 | 22 | KNIFE |
| 4 |  | 16 | 16 | SAUCERCUP |
| 5 |  | 3 | 12 | TEASPOON |
| 6 |  | 10 | 7 | BOWL |
| 7 |  | 7 | 7 | CAN |
| 8 |  | 6 | 6 | COCKTAILGLASS |
| 9 |  | 8 | 8 | COLACUP |

The items such as dishes, cups, forks, and knifes are different from shapes, styles, and materials. Generally, however, they are considered as rectangular items because the objective is to maximize the covered area of the shelf of the dish cart or of a large tray carried by a serving robot to a target table.

### *3.2 Parameters and Lower Bounds*

In this study, we use an enumerative algorithm for the exact solution of 2D-BPP that Martello and Vigo [2] proposed. Therefore, the parameters including lower bounds, $L_0, \cdots, L_4$ in Table 2 [2] can be followed directly.

**Table 2:** Parameters and Lower Bounds for 2D-BPP

| Index | Par. | Description |
|---|---|---|
| 1 | $I$ | Instance defined by item j and all the pieces currently assigned to bin i |
| 2 | $i$ | Bin index |
| 3 | $n$ | Number of items or rectangular pieces |
| 4 | $J$ | $\{1, \cdots, n\}$ A set of items |
| 5 | $j$ | Item index or item name |
| 6 | $W$ | Bin (shelf) width |
| 7 | $W$ | Bin (shelf) height |
| 8 | $w_j$ | Width of item j |
| 9 | $h_j$ | Height of item j |
| 10 | $I'$ | Instance defined by the unassigned items and all the pieces currently assigned to active bins |
| 11 | $c$ | Number of currently closed bins |
| 12 | $L_0$ | $L_0 = \lceil \frac{\sum_{j=1}^{n} h_j W_j}{HW} \rceil$ |
| 13 | $L_1$ | $max\{L_1^W, L_1^H\}$ |
| 14 | $L_2$ | $max\{L_2^W, L_2^H\}$ |
| 15 | $L_3$ | $max_{1 \le p \le (1/2)H, 1 \le q \le (1/2)W} \{L_3(p,q)\}$ |
| 16 | $L_4$ | $max\{L_2, L_3\}$ |
| 17 | $z^*$ | Best incumbent solution value |

The detailed theorems and proofs can be referred in [2]. Especially, the worst-case performance ratio of continuous lower bound $L_0$ is 1/4 if rotation of the pieces (by any angles) is allowed. It is intuitively easy to understand and $L_4$ dominates $L_3$ and $L_4$.

## 4 Branch-and-bound Algorithm: Finding Possible Positions for Placing an Item

The branch-and-bound algorithm [1] on which this paper is based for finding possible positions for placing each item on a shelf is presented. This algorithm repeatedly solves associated sub-problems in which all the items of a given subset $J$ have to be packed, if possible, into a single bin. This problem too is NP-hard in the strong sense.

The items are initially sorted in non-increasing order of their area, and a reduction procedure tries to determine the optimal packing of some bins [2]. The lower bounds [2] of the previous sections have been embedded into a branch-and-bound algorithm for the exact solution of 2D-BPP. The algorithm adopts a nested branching scheme. At each decision-node of *an outer branch-decision tree*, a piece $j$ is assigned to a bin. If a feasible packing of $j$ and all the pieces currently assigned to such a bin can be determined heuristically, the search continues. Otherwise, the problem of establishing whether such a feasible packing exists is solved through *an inner branch-decision tree*: if the answer is yes, the search (in the outer tree) continues; otherwise, a backtracking occurs. Let $z^*$ denote the best incumbent

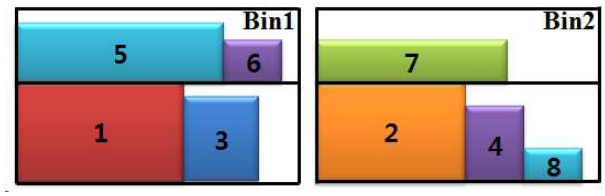



**Fig. 3:** Finite First Fit

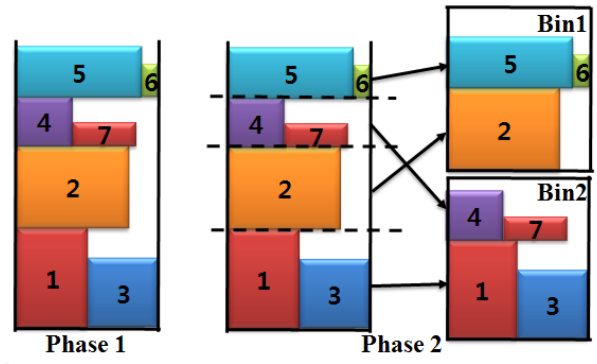



**Fig. 4:** Finite Best Fit

solution value. The algorithm starts by initializing $z^*$ through heuristic algorithms FFF, FBS, and FBS'.

## 4.1 Heuristics

**Finite First-Fit (FFF)**: An item is packed on the lowest level of the first bin where it fits; if no level can accommodate it, a new level is created in the first bin having sufficient vertical space, otherwise, the new level is created in a new bin. The example is shown in Figure 3 and the sequence of packed items is numbered. Figure 3 shows the example and here the number means the sequence of packed items. The worst-case time complexity of FFF is $o(n\log n)$.

**Finite Best-Fit (FBS)** is a two-phase algorithm in which the pieces are packed into an infinite height-strip (phase 1) using a best-fit algorithm to select the level for each item and the strip is divided into blocks, which are then packed into finite bins (phase 2) using a best-fit heuristic (Scan all bins and place item in bin with tightest fit. If no bin is large enough, a new bin is created for the BPP [1]. The worst-case time complexity of FBS is $o(n\log n)$.

**Finite Best-Fit′ (FBS′)** is a variant of algorithm FBS, in which the packing of the second phase is obtained for the exact solution of BPP, by limiting to 5000 the number of backtrackings allowed [9]. Within this limit indeed, most BPP instances are exactly solved and this variant often determines better solutions with respect to FFF and FBS; however, the best choice was to use FBS′ only at the root node of the branch-decision tree because of the higher computing time.

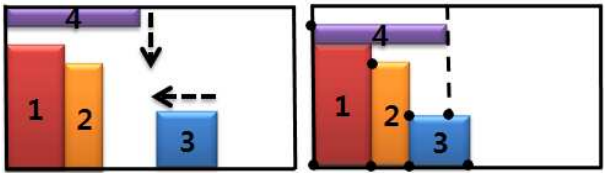



**Fig. 5:** Left-most downward strategy

**Left-most downward strategy [11]**: the coordinates considered for an item $j$ are those in which $j$ has its left edge adjacent either to the right edge of a previously packed item or to the left edge of the bin, and its bottom edge adjacent either to a packed item or to the bottom edge of the bin. For example, as shown in Figure 5, the left configuration is dominated whereas the right configuration is non-dominated. The right packing can be obtained in the following sequence: $1 \to 4 \to 2 \to 3$.

## 4.2 Outer Tree

The outer branch-decision tree is based on a depth-first branching scheme. The pieces are initially sorted in non-increasing order of their area and renumbered accordingly. The bins are numbered according to the order in which they are initialized. Whenever it is possible to establish that no more unassigned pieces can be assigned to a given initialized bin, such a bin is *closed*: an initialized and not closed bin is called *active*. At each level $j$ $(j = 1,\cdots,n)$, piece $j$ is considered and assigned, in turn, to all active bins and, possibly, to a new bin.

Let us consider a decision node in which piece $j$ is assigned to an active bin $i$: let $I$ denote the instance defined by piece $j$ and all the pieces currently assigned to bin $i$. The node is explored as follows:

**Step 1:** The feasibility of the packing is first heuristically checked in the following way. Lower bound $L_4$ is computed for $I$: if $L_4 > 1$, then the insertion is not possible, so a backtracking is performed. Otherwise, the heuristic algorithms FFF and FBS are applied to $I$: if a feasible packing of the pieces of into a single bin is not found, the inner branching scheme determines whether such a packing exists. If the answer is "no," a backtracking is executed; otherwise.

**Step 2:** The possibility of closing bin $i$ is checked by computing lower bound $L_4$ for the instances defined by $I \cup \{h\}$ $(h = j+1,\cdots,n)$ : if all these bounds are greater than 1 we conclude that no more pieces can be added to the bin, hence we close it. If the bin is not closed, the node exploration ends; otherwise.

**Step 3:** Lower bound $L_4$ is computed for the instance $I'$ defined by the unassigned pieces and all the pieces

currently assigned to active bins: if $L_4 + c \geq z^*$ (where $c$ denotes the number of currently closed bins) we backtrack. The node exploration is concluded in this case by executing the heuristics FFF and FBS, for instance $F'$, in order to possibly improve the incumbent solution.

When, at the current level $j$, the total number of active and closed bins is less than $z^*$-1, an additional decision node is generated by assigning piece $j$ to a new bin. Since we know that the packing is always feasible, the exploration of this type of nodes is simpler so that only Steps 2 and (possibly) 3 are executed.

### *4.3 Inner Tree*

Whenever the heuristics can not find any feasible packing of the pieces of set $I$ at each node of the outer tree, the inner branching procedure is executed. It generates all the possible patterns to pack $I$ into a bin through the "left-most downward" strategy [11] for the two-dimensional knapsack problem. The pieces are packed according to non-increasing area: at each level of the branch-decision tree the next piece is packed. If no feasible position exists for the current piece (i.e., no further vertical-downwards or horizontal leftwards movement of next piece $j$ is possible), a backtracking is performed. As soon as a feasible packing is found for the last piece, the procedure terminates.

## 5 Experiments

The experiments are performed to demonstrate the feasibility of the proposed scheme. The results obtained by applying the algorithms to various test problems are discussed. In these test problems, the sizes of the pieces to be packed were varied and the bin sizes (shelf) were always same as $W \times H = 1 \times 1 = 1$. We analyze the packing results by relating them to known bin-packing performance measures. The experiment was implemented in visual C++ 2008, OpenGL 2.0, and WinXP. It works well on an Intel PC with core i7 2.4GHZ CPU and 2GB RAM. Problems EXP1-EXP, EXP10-11, and EXP12-13 are the instances of Section 5.1, Section 5.2, and Section 5.3, respectively.

### *5.1 Packing for each item : EXP1-EXP9*

The objective of this section is to find a feasible arrangement of a subset of the items (totally 9 items) on the shelf that maximizes the total value of the packed items. In other words, the goal is to maximize the covered area of the shelf.

As shown in Figure 6 and listed in Table 3, FORK ($j = 1$) in maximum packing is to maximize the covered area of the shelf.

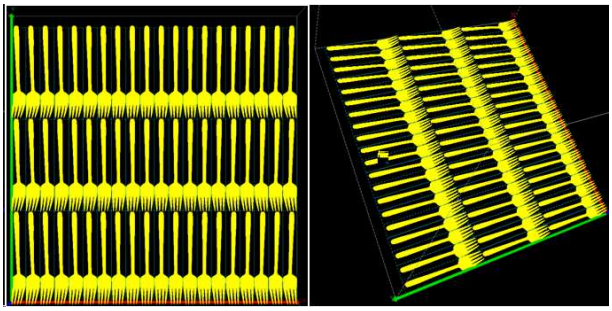


**Fig. 6:** The views of maximum packing for FORK (left: top view, right: 3D view)

**Table 3:** Computational results on problem instances

| EXP No. | Name | $L_0 z^*$ | $UB$ | Opt. No. | $z^*$ | Time |
|---|---|---|---|---|---|---|
| 1 | FORK | 2 | 2 | 10 | 68 | 0.13 |
| 2 | PLATE | 1 | 1 | 10 | 16 | 0.001 |
| 3 | KNIFE | 1 | 1 | 10 | 100 | 0.02 |
| 4 | SAUCERCUP | 1 | 1 | 10 | 36 | 0.001 |
| 5 | TEASPOON | 1 | 1 | 10 | 264 | 0.02 |
| 6 | BOWL | 1 | 1 | 10 | 140 | 0.01 |
| 7 | CAN | 1 | 1 | 10 | 196 | 0.01 |
| 8 | COCKTAIL GLASS | 1 | 1 | 10 | 256 | 0.02 |
| 9 | COLACUP | 1 | 1 | 10 | 144 | 0.01 |
| 10 | STEAK SET | 1 | 1 | 10 | 30 | 0.05 |
| 11 | COFFEE SET | 1 | 1 | 10 | 58 | 0.11 |
| 12 | RANDOM1 | 1 | 1 | 10 | 40 | 0.05 |
| 13 | RANDOM2 | 1 | 1 | 10 | 48 | 0.06 |

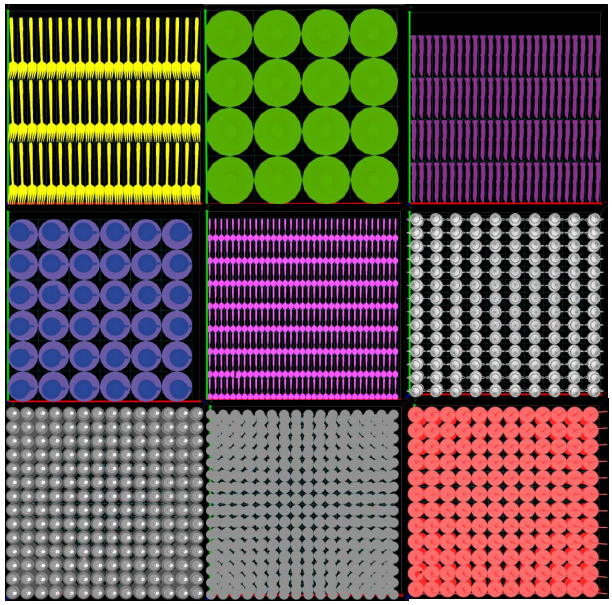


**Fig. 7:** Maximum packing for each item

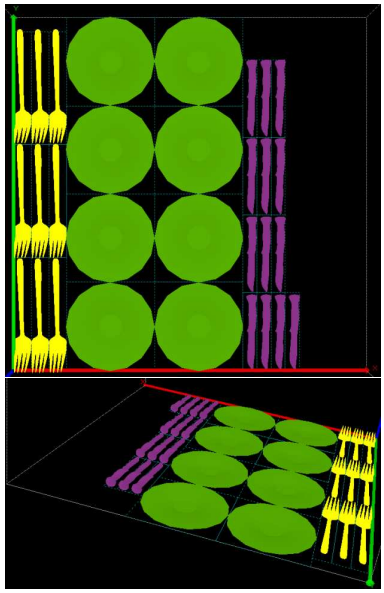


**Fig. 8:** Steak set = FORK + KNIFE + PLATE

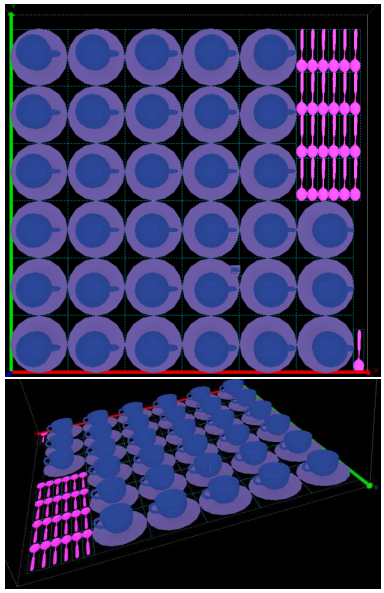


**Fig. 9:** Coffee set = SAUCERCUP + TEASPOON

Figure 7 shows that from left to right, as single item FORK, PLATE, KNIFE, SAUCERCUP, TEASPOON, BOWL, CAN, COCKTAILGLASS, and COLACUP are packed in maximum. 68 FORKs, 16 PLATEs, 100 KNIFEs, 36 SAUCERCUPs, 264 TEASPOONs, 140 BOWLs, 196 CANs, 256 COCKTAIL GLASSes, and 144 COLACUPs are maximized at each finite bin. As like taken 0.13sec, 0.001sec, $\cdots$, 0.06sec for each item, it is feasible for a serving robot to plan the real-time task in various situations. Figures 6, 7 and Table 3 show the experimental results, which prove the feasibility of the

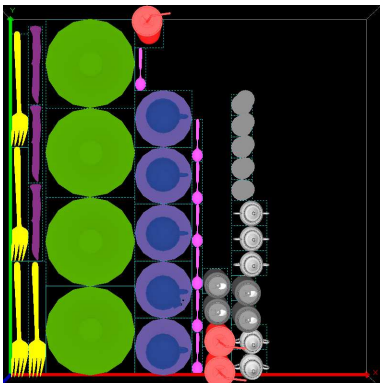


**Fig. 10:** The view of packing for 40 items

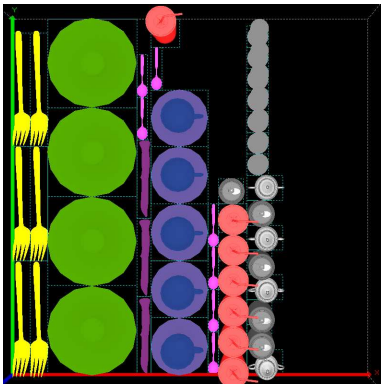


**Fig. 11:** The view of packing for 48 items

transport capacity optimization scheme of the serving robot in real time.

## *5.2 Packing for each set : EXP10-EXP11*

In this section, we show the experimental results of packing for each set. As a serving robot serves a food, there must be a case that the correlated items are served as a set. For example, Figure 8 shows the steak set consisting of 9 FORKs, 8 PLATEs, and 13 KNIFEs, which are randomly generated. Figure 9 shows the coffee set consisting of 33 SAUCERCUPs and 25 TEASPOONs.

Figures 8 and 9 show two important points. First, in case that the areas of items are smaller than a bin size the items are packed for the minimized areas (Figure 8). Second, even though it is for an item set packing problem, the feasibility to maximize the covered areas of the shelf is assured.

## *5.3 Packing for Random items : EXP12-EXP13*

In this section, finally, we examine general cases of randomly generated problem instances having different combinations of items. The numbers of randomly

generated items are 40 and 48 in Figure 10 and Figure 11, respectively.

As shown in Figures 10 and 11, even though a bin size is enough for the number of items, each item can be packed one by one. Hence, even though any item is assigned additionally, it is feasible for optimal or suboptimal item packing. Thus, it is easy to apply to the case of packing additional other items by a serving robot. In particular, Figures 10 and 11 show the tendency of left-most downward strategy which showed in Figure 5. It needs to improve the algorithm for diminishing the elapsed time when an end-effector of a serving robot manipulates for serving same kind of items.

## 6 Conclusion

In this paper, we present the worlds first two-dimensional bin packing problem applied to the situation of the maximization of the storage capacity of a dish cart. For future studies, three-dimensional knapsack problems or multi-objective problems can reduce the elapsed time and the energy cost such that more effective service can be offered by a serving robot. Furthermore, if robotic arm control scheme or ubiquitous environmental service [15] scheme is applied to the constrained 3D-BPP, more feasible and realistic solution can be expected.

## Acknowledgement

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology (grant number 2011-0015064).

## References

[1] J. 0. Berkey and P. Y. Wang, "Two Dimensional Finite Bin Packing Algorithms," Journal of Operational Research Society, **38**, 423-429 (1987).
[2] S. Martello and D. Vigo, "Exact Solution of the Two-Dimensional Finite Bin Packing Problem," Management Science, **44**, 388-399 (1998).
[3] 400 year old tea-serving robot kit (2005), http://www.boingboing.net/2005/10/29/400-year-old-teaserv.htm
[4] ASIMO, http://world.honda.com/ASIMO
[5] C. Borst, and Coauthors, "Rollin' Justin - Mobile Platform with Variable Base," IEEE International Conference on Robotics and Automation, 1597-1598 (2009).
[6] S. Yang, and C. Seo, "A Research on the Development of a Wine-Serving Robot: dON," Journal of Korean Society of Design Science, **21**, 65 (2011).
[7] Food-Serving Robot: Furo, http://www.futurerobot.com
[8] Sweet serving robot, http://zedomax.com/blog/2011/01/13/sweet-serving-robot
[9] S. Martello and D. Vigo, "Two-Dimensional Finite Bin Packing Problem," Management Science, **44**, 395 (1998).
[10] N. Christofides and C. Whitlock, "An Algorithm for Two-dimensional Cutting Problems," Operations Research, **25**, 30-44 (1977).
[11] E. Hadjiconstantinou and N. Christofides, "An Exact Algorithm for General, Orthogonal, Two-dimensional Knapsack Problems," European Journal of Operational Research, **83**, 39-56 (1995).
[12] A. Lodi, S. Martello, and D. Vigo, "Heuristic and metaheuristic apporaches for a class of two-dimensional bin packing problems," INFORMS Journal on Computing, **11**, 345-357 (1999).
[13] F. K. R. Chung, M. R. Garey, and D. S. Johnson, "On Packing two-dimensional bins," SIAM J. Algegraic Discrete Method, **3**, 66-76 (1982).
[14] J. B. Frenk and G. G. Galambos, "Hybrid next-fit algorithm for the two-dimensional rectangle bin-packing problem," Computing, **39**, 201-217 (1987).
[15] K. H. Lee, "Ubiquitous Robotic Game incorporating Hardware and Software," The Journal of Future Game Technology, **1**, 35-42 (2011).

**Kang-Hee Lee** received the B.S., M.S., and Ph.D degrees in electrical engineering and computer science from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 1999, 2001, and 2006, respectively. Since 2006, he has been a Senior Engineer in Digital Media & Communication Research Center, Samsung Electronics Company, Ltd., Korea. He has been a dispatched researcher in the Robotics Institute, Carnegie Mellon University in 2008. Since moving to Soongsil University in 2009, he is with Global School of Media, Soongsil University, Seoul, Korea. His current research interests include the areas of ubiquitous robotics, evolutionary robotics, emotional robotics, media robotics, cognitive task planning system, and knowledge-based reasoning system.

**A-Ra Khil** received the B.S. degree in computer science from Ewha Womens University in 1987, the M.S. and Ph.D degrees in computer science from Korea Advanced Institute of Science and Technology(KAIST), Daejeon, Korea, in 1990 and 1997, respectively. Since 1995, she has been a senior engineer in Serome Ltd., Korea. Since 2003, she has been a member of board of directors in Dialpad Ltd., USA. Since 1997, she is with School of Computer Science and Engineering, Soongsil University, Seoul, Korea. Her current research interests include the real-time Systems, emotional robotics, ubiquitous communications, wireless sensor networks, and embedded Operating system.