A Bi-level Optimization Model for Grouping Constrained Storage Location Assignment Problems

Jing Xie, Yi Mei, *Member, IEEE,* Andreas T. Ernst, Xiaodong Li, *Senior Member, IEEE,* and Andy Song

***Abstract*—In this paper, a novel bi-level grouping optimization model is proposed for solving Storage Location Assignment Problem with Grouping Constraint (SLAP-GC). A major challenge of this problem is the grouping constraint which restricts the number of groups each product can have and the locations of items in the same group. In SLAP-GC, the problem consists of two sub-problems, one is how to group the items, and the other one is how to assign the groups to locations. It is an arduous task to solve the two sub-problems simultaneously. To overcome this difficulty, we propose a Bi-level Grouping Optimization model (BIGO). BIGO optimizes item grouping in the upper level, and uses the lower-level optimization to evaluate each item grouping. Sophisticated fitness evaluation and search operators are designed for both the upper and lower level optimization so that the feasibility of solutions can be guaranteed, and the search can focus on promising areas in the search space. Based on the BIGO model, a multi-start random search (MSRS) method and a tabu search algorithm are proposed. The experimental results on the real-world dataset validate the efficacy of the BIGO model and the advantage of the tabu search method over the random search method.**

***Index Terms*—Optimization, Storage Location Assignment Problem, Grouping Constraint, Tabu Search, Heuristics**

I. INTRODUCTION

IMPROVING productivity by optimizing the resource allocation and process is one of core business activities in warehouse industry [1]. In this paper we aim to establish a novel methodology to address a real-world optimization problem for warehouse operations.

Typical operations in a warehouse can be categorized into four types: (1) (*receiving*) receiving shipments from suppliers; (2) (*storage*) allocating stocks inside a warehouse; (3) (*picking*) picking items according to customers' orders; and (4) (*delivering*) sending picked items to customers. Studies show that among these operations, *picking* is the most costly process, consuming about 50% to 60% of the total labor works [2]. Improving the efficiency of picking is one of the most promising ways of improving productivity. There are a wide range of factors that can affect the picking efficiency. For example, an unreasonable arrangement of items inside the warehouse may significantly increase the item-picking time. The routing strategy and the order batching strategy can also affect efficiency. In this paper, we will investigate a prominent problem associated with these factors, which is called the Storage Location Assignment Problem (SLAP) [3].

J. Xie, X. Li, and A. Song are with the Department of Computer Science and IT, School of Science, Melbourne VIC, RMIT University (email: jing.xie@rmit.edu.au; xiaodong.li@rmit.edu.au; andy.song@rmit.edu.au).

Y. Mei is with the School of Engineering and Computer Science, Victoria University of Wellington, Wellington, New Zealand (email:yi.mei@ecs.vuw.ac.nz).

AT. Ernst is with the the School of Mathematical Sciences, Monash University, Clayton VIC, Australia (email:andreas.ernst@monash.edu).

The overall goal of SLAP is to minimize the warehouse operational cost, subject to a set of constraints. In literature, several operational cost measures have been reported, such as the space utilization efficiency [4], the maximum load [5] and the relocation cost [6]. There is no universally agreed measurement for SLAP. In this paper, we consider the *total picking distance* [7], which is one of the most widely used measures in literature. Intuitively, highly demanded or frequently picked products should be assigned close to the loading dock. However, simple sorting items based on their demands is not suitable for SLAP in practice because of problem-specific constraints. For example, the correlations between products are typically considered so that products that are usually listed in the same order from customers are assigned to nearby locations. In production warehouse, the correlations between products are normally estimated using the Bill-of-Material (BOM) information [8] [9]. For other warehouses where BOM information is not accessible, different approaches such as data mining techniques [10] have been used to address the issue.

In this paper, we particularly focus on warehouses storing garments, which is a special scenario not considered before. To be more specific, different garment products may have significantly different number of items. For example, T-shirt or sportswear may contain many sizes (e.g., XS, S, M, L, XL, etc.), which have different demands. In contrast, hats and gloves usually have only one size. In previously studied SLAP, a product is usually atomic, and each location is capable of holding one product. However, this assumption is not reasonable for warehouses storing garments. Otherwise, there might be some locations holding more than one hundred items while some others may have only one item. In this kind of warehouses, it is more reasonable to take each item as an inseparable storing unit, i.e. each location can hold one item rather than one product. Consequently, the correlations between items of the same product need to be considered. More specifically, one needs to consider these correlations based on the following facts: (1) items of the same product are replenished simultaneously from the same supplier; (2) the management cost will tremendously increase if items of the same product are scattered around the warehouse; and (3) items of the same product appear frequently on the same customer order, especially on orders of workwear and sportswear from companies and organizations. The *Grouping Constraint* (GC) is proposed for this problem-specific scenario. Under this constraint, each product cannot be divided into more than two groups of items. All the items of the same group have to be placed in *adjacent locations*. Fig. 1 depicts an example of a simplified warehouse layout. In this example, locations 1 to 3 are adjacent locations while locations 6 to 8 are not as they are located on different shelves.

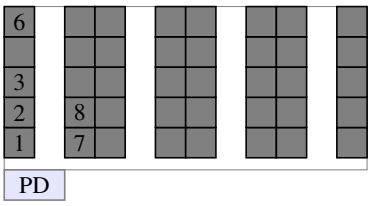


Fig. 1. An example layout of a warehouse with 8 shelves, each containing 5 bins. White areas denote corridors while gray columns denote (vertical) shelves. Area PD denotes the Pick-up/Drop-off location.

GC raises a feasibility issue that does not exist in other SLAP variants. When the same group of items are restricted to adjacent locations, it may be impossible to find a feasible solution for some of the groupings such that the number of items placed on a shelf does not exceed its capacity. This issue makes it difficult for the existing scheduling methods to be directly applied to SLAP-GC. Problem-specific algorithms need to be designed to tackle it. Hence, the goal of this paper is to achieve the following objectives:

1) To propose a Bi-level Grouping Optimization (BIGO) model to transform SLAP-GC into a simpler and more structured form;
2) To solve the proposed BIGO model by a Multi-Start Random Search (MSRS) method and a tabu search method;
3) To develop problem-specific search operators to search in the complex and constrained search space more efficiently;
4) To verify the efficacy of the proposed BIGO model and the developed algorithms using a real-world dataset.

The rest of this paper is organized as follows. Section II gives the problem description and mathematical formulations of SLAP-GC. Then, Section III gives an overview on the existing and promising approaches for SLAP and SLAP-GC. Section IV describes the proposed BIGO model for SLAP-GC, which tackles the item grouping and location assignment subproblems in different levels. Section V describes the developed MSRS and tabu search algorithms for solving the proposed BIGO model. The efficacy of the proposed BIGO model and developed algorithms are analyzed in Section VI. Section VII concludes our study and discusses several possible future research directions.

II. PROBLEM DESCRIPTION

We present a formal definition of SLAP-GC in this section. As shown in Fig. 1, the warehouse is in a rectangular shape with one Pick-up/Drop-off area. The locations are indexed shelf by shelf, i.e. the a^{th} location in the b^{th} shelf is given an index of $(b-1) \times L + a$, where L is the length of the shelf. The problem is to assign a set of products to a set of locations without splitting each product into more than two groups. Table I lists the notations used for the problem formulation.

We have two decision variables for the problem: x_{il} equals 1 if the i^{th} item is assigned to the l^{th} location; y_{pl} equals 1 if the l^{th} location is a starting point of the p^{th} product. The integer linear programming (ILP) model of the problem is presented in Eqs. (1)–(9).

TABLE I
THE NOTATIONS USED FOR THE PROBLEM DESCRIPTION OF SLAP-GC.

Notation	Description
N	total number of items/locations
M	number of the shelves
L	capacity (length) of each shelf
P	number of the products
l	index for locations, $l = 1, \dots N$
i, j	index for items, $i, j = 1, \dots, N$
$f(i)$	picking frequency of the i^{th} item
p	index for product, $p = 1, \dots, P$
v_l, h_l	the vertical/horizontal distance from location l to the P/D point
A_{ip}	equals to 1 if the i^{th} item belongs to the p^{th} product
B_{ip}	equals to 1 if the i^{th} item is the most popular in the p^{th} product

$$\min \varsigma(x) = \sum_{i=1}^{N} \sum_{l=1}^{N} f_i \left(v_l + h_l\right) x_{il} \tag{1}$$

$$s.t. \sum_{i=1}^{N} x_{il} = 1,\ l = 1, ..., N \tag{2}$$

$$\sum_{l=1}^{N} x_{il} = 1,\ i = 1, ..., N \tag{3}$$

$$\sum_{l=1}^{N} y_{pl} \leq 2,\ p = 1, ..., P \tag{4}$$

$$x_{il} \leq \sum_{p=1}^{P} A_{ip} \left(y_{pl} + \sum_{j=1}^{N} A_{jp} x_{j,l-1} \right),\ i, l = 1, ..., N \tag{5}$$

$$\sum_{p=1}^{P} y_{pl} = 1,\ \forall\ l\ mod\ L = 1 \tag{6}$$

$$\sum_{p=1}^{P} B_{ip} x_{il} \leq \sum_{p=1}^{S} B_{ip} \left(\sum_{p=1}^{P} A_{ip} y_{pl} \right),\ i = 1, ..., N \tag{7}$$

$$\sum_{i=1}^{N} A_{ip} x_{il} \geq y_{pl},\ p = 1, ..., P,\ l = 1, ..., N \tag{8}$$

$$x_{il}, y_{pl} \in \{0, 1\}. \tag{9}$$

The objective in Eq. (1) is to minimize the total picking frequency weighted distance by determining the one-to-one assignment of items to locations (Eqs. (2)–(3)). The rest of the equations formulate GC. Eq. (4) implies that each product has at most two starting points, i.e., each product has no more than two groups. Then, if a location is a starting point for a product, it is either the first location of a shelf or preceded by an item of another product (Eqs. (5)–(6)). Eq. (7) states that the most popular item of a product is always assigned to a starting point of that product. Eq. (8) makes sure that each one-item product only has one starting point.

III. RELATED STUDIES

The general SLAP has been extensively investigated in literature. Comprehensive surveys of the studies earlier than 2010 can be found in [3] [11] [12]. The research of different SLAP

variants remains active since then. For example, [13] considers the costs of re-handling containers on the top of the requested container in a three-dimensional warehouse; [14] and [15] consider the availability of the information throughout the planning horizon; in [16], the congestion/waiting time is taken into consideration when there are multiple pickers working concurrently in the same region; the study presented in [17] considers a pick-and-pass system, in which the warehouse is divided into several picking zones, and each pick only collects items in a particular zone.

For methodologies, both exact and heuristic approaches have been applied to different SLAP variants. For example, a branch-and-bound algorithm is applied to solve SLAP by minimizing the sum of storage space cost and the handling cost based on the class-based storage strategy [18]. An ILP model is developed in [19] to optimize the average picking costs in warehouses located in seismic areas with the consideration of the rack stability. These exact methods can guarantee the optimality of solutions but cannot scale well for large instances due to the high complexity, which makes them computationally prohibitive in industry applications. In contrast, heuristic approaches such as simulated annealing, tabu search, and genetic algorithms are widely used in real-world scenarios due to their abilities to obtain reasonably good solutions within the given time budget. For example, tabu search has shown its capability in making dynamic operation policies for product relocation [6]. Genetic algorithms have also been applied to address different storage location problems [20] [21] [22]. To achieve promising performance, a number of problem-specific search operators for the heuristic methods have been developed based on the insights of the characteristics of the problems.

All the aforementioned approaches, as well as the majority of the studies in literature, do not consider the grouping constraint, and thus deal with the assignment sub-problem alone. As a result, they are not effective for solving SLAP-GC, which consists of two interdependent sub-problems (*grouping* and *assignment*). Due to the interdependency, it is unwise to solve the two sub-problems separately. However, solving the two sub-problems simultaneously leads to a huge search space, and thus a poor scalability. Therefore, one needs to design effective heuristic approaches that can take the interdependency into account and solve the two sub-problems cooperatively. For this purpose, a GP method was proposed in [23] to evolve effective heuristics for the grouping sub-problem. The item groupings are selected one by one by using the current information about the unassigned items and the distribution of the empty locations in the warehouse. Then given this item grouping heuristic, the assignment sub-problem can be solved by a greedy heuristic.

Despite of the advantages over exact approaches shown in [23], the GP method is not flexible enough, as the assignment sub-problem is solved by a fixed heuristic. To achieve better solutions, we consider using a Bi-level programming approach [24] in this paper. Inspired by Stackelberg game [25], a Bi-level model involves two players: the leader (the upper level) and the follower (the lower level). The follower responds to the decisions made by the leader to optimize its own objective while the leader takes into account the optimal reaction of the follower to optimize its objective [26], and finally, both players accomplish their tasks. This programming approach provides a novel perspective on the problem structure and are gaining popularities in the field of logistics, for example, the optimization of facility locations [27] [28] [29] and the plannings of distribution networks [30] [31].

To develop a Bi-level model for SLAP-GC, we consider the grouping sub-problem as the upper level, and the assignment sub-problem as the lower level. The resultant BIGO model aims to find the optimal grouping and the corresponding optimal assignment under the optimal grouping.

IV. BI-LEVEL GROUPING OPTIMIZATION MODEL

In this section, we describe the proposed BIGO model for solving the grouping and assignment sub-problems of SLAP-GC. To facilitate the description, besides the notations presented in Table I, we list some more notations for the BIGO model in Table II.

TABLE II
THE NOTATIONS USED IN THE BI-LEVEL GROUPING OPTIMIZATION (BIGO) MODEL.

Notation	Description
N_p	number of items in the p^{th} product
G	an item grouping
$\mathcal{G}$	a set of all feasible item groupings
$n_p(G)$	size of the "larger" group of the p^{th} product in the grouping G
S	a solution of SLAP-GC
$S(G)$	a feasible solution under the grouping G
$\Omega_S(G)$	the set of all the feasible solutions under the grouping G
$S^*(G)$	the optimal solution under the grouping G
$\phi(S)$	fitness of the solution S
$e_{ij}(S)$	item placed in the j^{th} location of the i^{th} shelf in the solution S

Given the notations, the BIGO model is stated as follows:

$$\min_{\substack{G\in\mathcal{G},\\ S\in\Omega_S(G)}} \quad F(G,S)=\phi(S), \tag{10}$$

$$s.t.:\ S\in \arg\min_{S'\in\Omega_S(G)} \phi(S'), \tag{11}$$

where

$$\phi(S)=\sum_{i=1}^{M}\sum_{j=1}^{L}(i+j)f(e_{ij}(S)). \tag{12}$$

$F(G,S)$ is the objective function, which is the total picking frequency weighted distance $\phi(S)$ of a solution S. It is defined in Eq. (12), where $e_{ij}(S)$ indicates the item placed in the j^{th} location of the i^{th} shelf in the solution S. The distance from the location of $e_{ij}(S)$ to the P/D point is defined as the Manhattan distance, i.e. $d(e_{ij}(S))=i+j$, and $f(e_{ij}(S))$ stands for the picking frequency of the item $e_{ij}(S)$. M is the number of shelves, and L is the capacity (length) of each shelf.

In the proposed BIGO model, Eq. (10) is the upper level optimization, and Eq. (11) is the lower level optimization. The upper level optimization aims to find the optimal grouping G^*, while the lower level optimization is to find the optimal solution $S^*(G^*)$ under the grouping G^* in terms of the objective value $\phi(\cdot)$.

The optimal solution of SLAP-GC is composed of the optimal grouping G^* and the optimal item assignment $S^*(G^*)$

under this grouping. Therefore, the optimality can be guaranteed in the BIGO model. That is, the optimal solution to the BIGO model is the optimal solution to the original SLAP-GC.

Since each product cannot be divided into more than two groups, an item grouping G can be represented as a list of numbers $G = (n_1(G), \ldots, n_p(G))$, where $n_p(G)$ stands for the size of the "larger" group (i.e. the group with no fewer items than the other) of the p^{th} product in G, and $\lceil N_p/2 \rceil \leq n_p(G) \leq N_p$ $(p = 1, \ldots, P)$. Consequently, the number of items in the other group of product p is $N_p - n_p(G)$. For example, suppose there are two products, Shirt and Jacket, each with five items (S, M, L, XL and XXL). Then, the grouping G that divides Shirt into Shirt(XXL, XL, S) and Shirt(M, L), and Jacket into Jacket(S, M, XXL) and Jacket(XL, L) is represented as $G = (3, 3)$ since the larger groups of both products have 3 items. By always keeping the sizes of the larger groups in the representation, we can reduce the chance of getting duplicated solutions.

A solution S is considered to be a feasible solution under grouping G if the item groupings of all the products in S comply with grouping G. Suppose that the aforementioned two products are to be placed in two 5-bins shelves. Eqs. (13) and (14) show two solutions, where the element in the i^{th} row and j^{th} column stands for item placed in the i^{th} location of the j^{th} shelf. For example, Jacket(M) is placed in the fourth location of the first shelf in these two solutions. The groupings of S_1 and S_2 are $G_1 = (3, 3)$ and $G_2 = (4, 4)$ respectively. Therefore, S_1 is feasible under the grouping $G = (3, 3)$ while S_2 is not.

$$S_1 = \begin{pmatrix} \text{Shirt(M)} & \text{Shirt(XXL)} \\ \text{Shirt(L)} & \text{Shirt(XL)} \\ \text{Jacket(S)} & \text{Shirt(S)} \\ \text{Jacket(M)} & \text{Jacket(XL)} \\ \text{Jacket(XXL)} & \text{Jacket(L)} \end{pmatrix}. \tag{13}$$

$$S_2 = \begin{pmatrix} \text{Shirt(M)} & \text{Shirt(XXL)} \\ \text{Jacket(XL)} & \text{Shirt(XL)} \\ \text{Jacket(S)} & \text{Shirt(S)} \\ \text{Jacket(M)} & \text{Shirt(L)} \\ \text{Jacket(XXL)} & \text{Jacket(L)} \end{pmatrix}. \tag{14}$$

V. BI-LEVEL GROUPING OPTIMIZATION ALGORITHMS

In this section, we propose two heuristic search algorithms for finding solutions to the proposed BIGO model. First, we design the fitness evaluation for the groupings in the upper level optimization based on solving the lower level optimization. Then, we develop two search algorithms by hybridizing the designed fitness evaluation with different search frameworks. The first algorithm is a Multi-Start Random Search (MSRS) method, and the second is a tabu search method. In the following, we will describe the designed fitness evaluation and the two search algorithms in turn.

A. Fitness Evaluation of Item Grouping

Based on Eqs. (10) and (11), the fitness function of a grouping G can be set to the objective value of the optimal solution $S^*(G)$ under the grouping G. That is,

$$fit(G) = \phi(S^*(G)) = \min_{S \in \Omega_S(G)} \phi(S). \tag{15}$$

However, finding $S^*(G)$ is very hard, if not impossible, since the lower level optimization is NP-hard. When there are only two shelves (i.e. $M = 2$), and all the items have the same picking frequency, the lower level optimization, i.e. finding the optimal assignment under a given grouping, can be reduced to the NP-hard partition problem. In this situation, we design a local-search-based fitness evaluation to approximate the real optimal value with local optimal values. To this end, four different search operators, namely `inSlfSort`, `slfSort`, `groupSort`, and `itemSort`, are designed to conduct the local search. They sort the items and/or shelves to reach nearby local optimal solutions efficiently. The pseudo code of the local-search-based fitness evaluation is given in Algorithm 1. The local search starts from an initial feasible solution $S_0(G)$, which can be obtained in various ways such as using an initialization algorithm or adopting an examined solution during the search process. Then, at each iteration, the four search operators are randomly shuffled and applied to the current solution one by one. The local search is stopped if there is no improvement or the number of iterations reaches to the maximum number n_{max}.

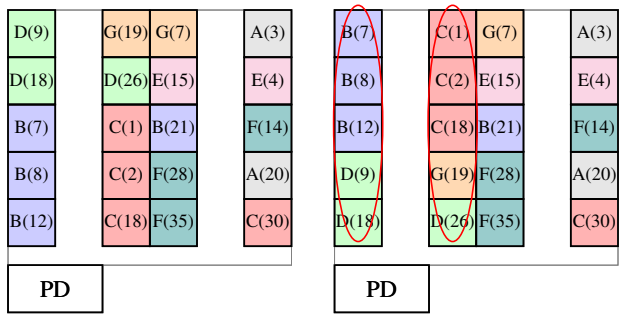
Algorithm 1 $(fit(G), S^*(G)) \leftarrow$ `Evaluate`$(G, S_0(G))$

INPUT: A grouping G and an initial feasible solution $S_0(G)$
OUTPUT: $fit(G)$ and the corresponding solution $S^*(G)$

```
Set the pool of search operators Ω ← {inSlfSort, slfSort, groupSort, itemSort};
S ← S_0(G);
n ← 0;
while n < n_max do
    S' ← S;
    Randomly shuffle Ω;
    for each sort ∈ Ω do
        Apply sort to S';
    end for
    n ← n + 1;
    if φ(S') ≥ φ(S) then
        break;
    end if
    S ← S';
end while
fit(G) ← φ(S), S*(G) ← S;
return (fit(G), S*(G));
```

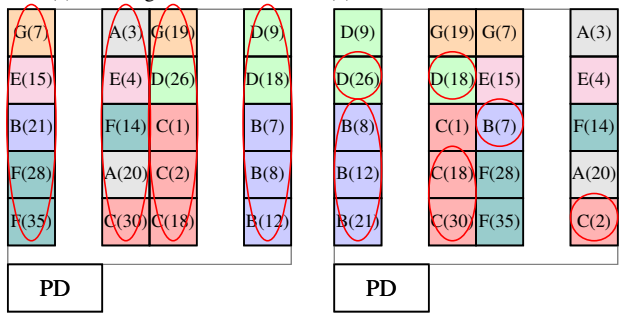
The details of the four search operators are described as follows:

1) `inSlfSort`: This operator sorts all the groups placed in the same shelf in the descending order of the average picking frequency, and thus, the groups with higher average picking frequency are always placed in the locations closer to the P/D point. Since `inSlfSort` does not change the number



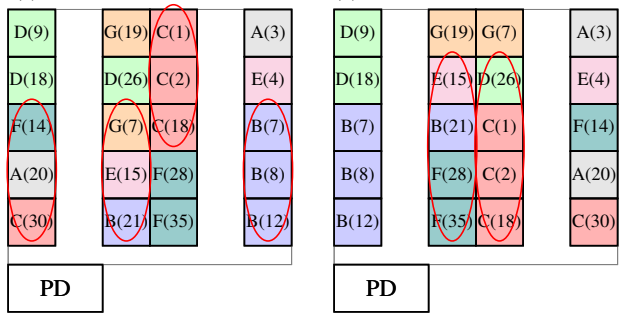
(a) The original solution.

(b) The solution after `inslfSort`.



(c) The solution after `slfSort`.

(d) The solution after `itemSort`.



(e) The solution after `groupSort` when $k = 3$.

(f) The solution after `groupSort` when $k = 4$.

Fig. 2. Illustrations to show how the sorting operators work. Fig. 2a is the original solution. Figs. 2b–2f are the solutions after applying different operators to the original solution. Each cell is labeled with the product name and picking frequency of the item assigned to it. The modified locations are circled by red ellipses.

of items placed in each shelf or the item groupings, it does not change the feasibility of the solution.

2) `slfSort`: This operator sorts all the shelves in the descending order of the average picking frequency of all the items placed on the shelf. After the sorting, the shelf with the higher average picking frequency is always closer to the P/D point. There is no movement of items cross the shelves, and thus, this operator does not change the feasibility.

3) `itemSort`: This operator sorts all the items of each product without changing the locations assigned to the product, and thus, the feasibility of the solution is not changed. After applying `itemSort`, for each product, the items with higher picking frequencies are always placed in the locations closer to the P/D point than the ones of the same product but with lower picking frequencies. Since `itemSort` only shifts items of same products, it does not change the groupings, and the feasibility of the resultant solution is maintained.

4) `groupSort`: The `groupSort` operator exchanges groups of items with the same size regardless of which shelf they are located in. Given a number k ($1 \leq k \leq L-1$), the operator looks for chunks of item groups whose sizes equal to k, and then sort them in the decreasing order of the average picking frequency. Note that a chunk can be composed of multiple adjacent item groups. In addition, the `groupSort` operator requires that there is no overlap among the selected chunks (they share no common item). If there are overlapping chunks, only one of them are selected randomly. Since all the selected chunks have the same size (k), the feasibility of the solution is not changed after the sorting. The `groupSort` operator starts with $k = 1$, and sorts all the single-item chunks. Then, k is increased by 1, and the `groupSort` operator continues to sort all the 2-item chunks. The whole process continues until all the $L-1$-item chunks have been sorted.

Fig. 2 illustrates how the four sorting operators work. The original solution is shown in Fig. 2a, in which each cell is labelled with the product name and picking frequency of the item assigned to it. For example, C(18) indicates that the item belongs to product C and has a picking frequency of 18. Figs. 2b–2f show the resultant solutions after applying the sorting operators, where the modified locations are circled by red ellipses.

As shown in Fig. 2b, after applying the `inslfSort` operator, the item groups with larger average picking frequencies are always closer to the PD point than other item groups in the same shelf (e.g. {B(7), B(8), B(12)} versus {D(9), D(18)} in the first shelf). As shown in Fig. 2c, the shelves are sorted by the `slfSort` operator so that the shelves with higher average picking frequencies are closer to the PD point. Fig. 2d shows the solution obtained by the `itmSort` operator. The locations of the products are not changed. However, the locations of the items of products B, C and D are sorted so that the items with larger picking frequencies are closer to the PD point. Figs. 2e and 2f show the solutions obtained by the `groupSort` operator for $k = 3$ and $k = 4$ respectively. In Fig. 2e, four 3-item chunks are selected and sorted. In Fig. 2f, two 4-item chunks are exchanged after the sorting. In fact, the `groupSort` operator selects another chunk in the last shelf, which is {A(3), E(4), F(14), A(20}. However, the location of this chunk is not changed after the sorting.

*B. Multi-Start Random Search (*MSRS*)*

The framework of MSRS is described in Algorithm 2. It repeatedly generates and evaluates groupings until the stopping criteria are met. In each iteration, a grouping G is first generated by randomly splitting each product into no more than two groups (line 5). Then, a feasible solution S_0 under the grouping G is initialized by `Init`(G), which will be described in Algorithm 3 (line 6). `Init`(G) is a greedy insertion algorithm and may not be able to obtain feasible solutions for some groupings. Therefore, the grouping generation and solution initialization are repeated until a feasible solution under the generated grouping is found. Then the grouping and the initialized solution are evaluated by `Evaluate`$(\cdot)$, i.e. the local-search-based fitness evaluation described in the last section. The best-so-far item grouping G^* and the optimal solution S^* will be updated every time a better grouping is

found (line 9–11). Finally, MSRS returns the best grouping and the corresponding best solution.

Algorithm 2 MSRS for SLAP-GC using the BIGO model

1: $G^* \leftarrow null$, $S^* \leftarrow null$, $fit(G^*) \leftarrow \infty$;
2: **while** Stopping criteria are not met **do**
3: $S_0 \leftarrow null$;
4: **while** $S_0 = null$ **do**
5: Randomly generate a grouping G;
6: $S_0 \leftarrow \texttt{Init}(G)$;
7: **end while**
8: $(fit(G), S^*(G)) \leftarrow \texttt{Evaluate}(G, S_0)$;
9: **if** $fit(G) < fit(G^*)$ **then**
10: $G^* \leftarrow G$, $S^* \leftarrow S^*(G)$, $fit(G^*) \leftarrow fit(G)$;
11: **end if**
12: **end while**
13: **return** (G^*, S^*);

Given a grouping G, the solution initialization algorithm $\texttt{Init}(G)$ is described in Algorithm 3. It is a greedy insertion heuristic. At first, all the shelves are empty and all the items are unassigned. For each product, the items are grouped based on G. Then, at each step, the heuristic selects the largest unassigned item group and insert it into the best feasible locations (i.e. closest to the P/D point). Note that such a greedy insertion may not lead to a feasible solution, and some groups may not be able to fit into the shelves without exceeding their capacities. In this case, the algorithm returns $null$, indicating that no feasible solution is found.

Note that MSRS randomly generate new groupings for the upper level optimization of the BIGO model, and we considered it as a baseline algorithm for SLAP-GC using the BIGO model.

C. Tabu Search

Proposed by Glover and Laguna [32] [33] [34], tabu search has been successfully applied to various real-world problems, such as scheduling problems [35] [36], planning [37] [38], static and dynamic assignment problem [39] [40], and graph theory [41] [42]. Since SLAP-GC is similar to these problems in many aspects (e.g. they are all combinatorial optimization problems), it is reasonable to expect tabu search to perform well in solving SLAP-GC.

The framework of the proposed tabu search algorithm for solving SLAP-GC using the BIGO model is given in Algorithm 4. First, the current grouping G is randomly initialized and the corresponding best solution S is obtained by $\texttt{Evaluate}(\cdot)$ (lines 1–6). The tabu list is initialized to be empty, and the best-so-far grouping G^* and solution S^* are set to the current ones respectively. Then, in each iteration, all the neighbors (G', S') in the neighborhood $\mathcal{N}(G, S)$ are evaluated (lines 11–19), and the best non-tabu neighbor is selected to replace the current grouping and solution (line 20). A widely used improved-best aspiration criterion is also employed in this implementation (line 13). When a new solution is better than the best-so-far solution, it will be accepted even the corresponding move is tabu. Then, the best-so-far result and

Algorithm 3 $S \leftarrow \texttt{Init}(G)$

INPUT: A grouping G;
OUTPUT: A solution S under grouping G;
1: Set $S \leftarrow \emptyset$;
2: Split each product according to the grouping G to obtain the item groups $\mathcal{E}_k(p)$, $k = 1, 2$ and $p = 1, \ldots, P$;
3: Set all item groups as unassigned;
4: **while** not all item groups are assigned **do**
5: Select the largest unassigned item group $\mathcal{E}^*$;
6: **if** there exist feasible locations in S for $\mathcal{E}^*$ **then**
7: Assign $\mathcal{E}^*$ to the best feasible locations in S;
8: Set $\mathcal{E}^*$ as assigned;
9: **else**
10: **return** $null$;
11: **end if**
12: **end while**
13: **return** S;

the tabu list are updated accordingly (lines 21–24). Finally, the best-so-far grouping G^* and solution S^* are returned.

Algorithm 4 Tabu search for SLAP-GC using the BIGO model

1: $S_0 \leftarrow null$;
2: **while** $S_0 = null$ **do**
3: Randomly generate a grouping G;
4: $S_0 \leftarrow \texttt{Init}(G)$;
5: **end while**
6: $(fit(G), S) \leftarrow \texttt{Evaluate}(G, S_0)$;
7: Initialize the tabu list *tabuList* $\leftarrow \emptyset$;
8: $G^* \leftarrow G$, $S^* \leftarrow S$;
9: **while** Stopping criteria are not met **do**
10: $G_{next} \leftarrow null$, $S_{next} \leftarrow null$, $bestFitOneItr \leftarrow \infty$;
11: **for each** $(G', S') \in \mathcal{N}(G, S)$ **do**
12: $(fit(G'), S^*(G')) \leftarrow \texttt{Evaluate}(G', S')$;
13: **if** G' is not tabu **or** G' satisfies the aspiration criterion **then**
14: **if** $fit(G') < bestFitOneItr$ **then**
15: $G_{next} \leftarrow G'$, $S_{next} \leftarrow S^*(G')$;
16: $bestFitOneItr \leftarrow fit(G')$;
17: **end if**
18: **end if**
19: **end for**
20: $G \leftarrow G_{next}$, $S \leftarrow S_{next}$;
21: **if** $fit(G) < fit(G^*)$ **then**
22: $G^* \leftarrow G$, $S^* \leftarrow S$;
23: **end if**
24: Update *tabuList* according to G_{next}, and remove the tabu elements whose tabu duration have reached the tabu tenure;
25: **end while**
26: **return** (G^*, S^*);

In Algorithm 4 (and in tabu search in general), two major issues are to be addressed: (1) the neighborhood definition and (2) the structure of the tabu list. In this paper, we will address them as follows:

1) Neighborhood Definition: In this work, the neighborhood is defined by two problem-specific move operators called `resizeInSlf` and `resizeXSlf`. The operators change the current grouping and solution by moving the items within the same shelf and cross different shelves respectively. The movements are carefully designed to keep the feasibility of the solution. The details of the two operators are given below.

a) `resizeInSlf`: This operator changes the grouping of the products whose two groups are placed in the same shelf. Suppose that two groups $\mathcal{E}_1$ and $\mathcal{E}_2$ belong to the same product and are placed in the same shelf. Without loss of generality, $\mathcal{E}_1$ is assumed to be placed closer to the P/D point than $\mathcal{E}_2$. We can carry out the following two operations without changing the feasibility: (1) move the last item of $\mathcal{E}_1$ to the front of $\mathcal{E}_2$, and shift forward all the items in between; (2) move the first item of $\mathcal{E}_2$ to the end of $\mathcal{E}_1$, and shift backward all the items in between. Since the number of items placed in the shelf does not change, the `resizeInSlf` operator does not change the feasibility of the solution.

b) `resizeXSlf`: In contrast to the `resizeInSlf` operator, the `resizeXSlf` operator is applied to groups of items that are placed in different shelves. This may involve multiple products in order to keep the feasibility of the solution, i.e., to keep the number of items placed in the shelves unchanged. For example, if one item is moved from shelf A to shelf B, one must move another item from shelf B back to shelf A without violating the grouping constraint. Here, given two shelves A and B, the following two scenarios are considered:

- There are two products so that both products have one group placed in shelf A, and the other placed in shelf B. Let $\mathcal{E}_{11}$ and $\mathcal{E}_{12}$ be the two groups of the first product, and $\mathcal{E}_{21}$ and $\mathcal{E}_{22}$ be the two groups of the second product, one can assume that $\mathcal{E}_{11}$ and $\mathcal{E}_{21}$ are placed in shelf A, and $\mathcal{E}_{12}$ and $\mathcal{E}_{22}$ are placed in shelf B without loss of generality. In this case, one can move one item from $\mathcal{E}_{11}$to $\mathcal{E}_{12}$ (from shelf A to shelf B), and move one item from $\mathcal{E}_{22}$ to $\mathcal{E}_{21}$ (from shelf B to shelf A), or vice versa.
- There is one product with two item groups $\mathcal{E}_{11}$ and $\mathcal{E}_{12}$, where $\mathcal{E}_{11}$ is placed in shelf A and $\mathcal{E}_{12}$ in shelf B. In addition, there is another single-item group $\mathcal{E}_2$ in shelf A or B. Without loss of generality, one can assume that $\mathcal{E}_2$ is in shelf A. Then, one can move an item from $\mathcal{E}_{12}$ to $\mathcal{E}_{11}$ (from shelf B to A), and move $\mathcal{E}_2$ to shelf B to fill the empty location.

2) Tabu List: In tabu search, the tabu list is designed to prevent the search from going back to the areas that have been explored recently. Therefore, a key issue is how to design the tabu list structure to properly represent the previously explored areas. The tabu list structure can be designed in various ways, such as solution-based tabu lists (forbid to go back to previously visited solutions) and operation-based tabu lists (forbid to reverse a previously conducted operation). In this paper, we adopt the operation-based tabu list structure due to its space efficiency and ability to represent search areas rather than exact solutions. We design two different operation-based tabu list structures, called TL_1 and TL_2. They are described as follows:

a) TL_1: In this tabu list, an element is defined as a previously modified *product* and *its previous grouping*, represented as a tuple $(p, n_p(G))$.

b) TL_2: In this tabu list, an element is defined as a previously modified *product*, represented as an ID (p).

A simple example is provided here to demonstrate how the two tabu lists are updated. Assuming that the current solution consisting of 4 products (Shirt, Jacket, Shoes and Trousers) is given as follows:

$$S = \begin{pmatrix} \text{Shirt(M)} & \text{Shirt(XXL)} & \text{Jacket(XL)} \\ \text{Shirt(L)} & \text{Shirt(XL)} & \text{Jacket(L)} \\ \text{Jacket(S)} & \text{Shirt(S)} & \text{Trousers(L)} \\ \text{Jacket(M)} & \text{Shoes(5)} & \text{Shoes(6)} \\ \text{Jacket(XXL)} & \text{Shoes(8)} & \text{Shoes(9)} \end{pmatrix}.$$

In this solution, the number of items of the products are 3, 3, 2 and 1, respectively. Therefore, the current grouping is $G = (3, 3, 2, 1)$.

Then, suppose after applying the move operators (e.g. `resizeInSlf` and `resizeXSlf`), we obtain the following new solution:

$$S' = \begin{pmatrix} \text{Shirt(M)} & \text{Shirt(XXL)} & \text{Jacket(XL)} \\ \text{Shirt(L)} & \text{Shirt(XL)} & \text{Jacket(L)} \\ \text{Jacket(S)} & \text{Shirt(S)} & \text{Shoes(6)} \\ \text{Jacket(M)} & \text{Trousers(L)} & \text{Shoes(5)} \\ \text{Jacket(XXL)} & \text{Shoes(8)} & \text{Shoes(9)} \end{pmatrix}.$$

In this new solution, the items Shoes(5) and Trousers(L) are swapped, and the grouping of the Shoes product is changed, i.e. $n_{\text{Shoes}}(G)$ is changed from 2 to 3. As a result, the new grouping is $G' = (3, 3, 3, 1)$. In this situation, the element for TL_1 is (Shoes, 2) and that for TL_2 is (Shoes).

The effectiveness of the two tabu list structures will be investigated in the experimental studies (Section VI), and the better one will be selected for the comparison with other algorithms.

VI. Experimental Studies

To evaluate the efficacy of the proposed BIGO model, we use a real-world dataset collected from a warehouse business owned by a local company based in Melbourne, Australia. The dataset includes the garment orders in four years [1]. From this dataset, we generate a variety of benchmark instances with different problem sizes and difficulties. Then, to verify the effectiveness of the proposed BIGO model, we test the proposed MSRS and tabu search algorithms on the benchmark instances, and their performance is compared with the branch-and-cut method and the previously proposed GP approach [23]. The branch-and-cut method is an exact method that can guarantee optimality. However, it can only be applied to small to medium sized instances due to the high complexity. The GP approach is the current state-of-the-art heuristic approach for SLAP-GC. In addition, we also compare between the MSRS method and the tabu search method to show the advantage of the tabu search framework (e.g. using historical information and prevent from revisiting recently explored areas) when using the BIGO model.

[1]This is the full dataset of four years available in the current ERP system of the actual warehouse.

A. Real-world Data and Benchmark Generation

Table III gives the summary of the real-world data year by year, where P is the number of products requested in the order in that year, and N_p is the number of items of the p_{th} product. $\max N_p$, $\min N_p$ and $\text{mean } N_p$ are the maximum, minimum and mean number of items in the products. The factor $\alpha = \frac{\hat{P}}{P}$ is the proportion of products with no more than two items, where $\hat{P}$ is the size of the subset of the original products containing no more than two items. It reflects the difficulty of the instance to some extent, and a higher α value indicates that the instance tends to be easier to solve, since there are more products with only one or two items. In an extreme case where $\alpha = 1$, all the products have only one or two items, and the grouping constraint can be eliminated.

From the table, one can see that there are more than 400 products requested in each year's orders, and each product contains 25 or 26 items on average. There are totally over 10,000 items requested each year. The number of items varies significantly from one product to another. The smallest product has only one item, while the largest product has 162 items. The α value is quite small (only about 5% of the products have no more than two items), indicating that the corresponding problem instances are difficult to solve.

Four real-world instances are generated based on these order information, each corresponding to the data in one year (one row in Table III). To construct the problem, we define the warehouse information including the number of shelves (M) and the capacity of each shelf (L) for each year of data. Here, we assume that the capacity of each shelf is fixed to 140, and the number of shelves varies from one year to another to fit the number of items requested in that year.

TABLE III
SUMMARY OF THE REAL-WORLD DATA SETS. P IS THE NUMBER OF PRODUCTS, AND N_p IS THE NUMBER OF ITEMS OF THE p_{th} PRODUCT. α STANDS FOR THE PROPORTION OF PRODUCTS WITH ONE OR TWO ITEMS. M AND L ARE THE NUMBER OF SHELVES AND THE SHELF SIZE RESPECTIVELY.

Year	Order Info.					Warehouse Info.	
	P	$\max N_p$	$\min N_p$	$\text{mean } N_p$	α	M	L
1	478	162	1	26	0.059	89	140
2	403	162	1	25	0.050	72	140
3	450	162	1	25	0.047	83	140
4	486	162	1	26	0.037	92	140

Note that these instances are too large and prohibitive for the branch-and-cut approach. To make meaningful comparison with the branch-and-cut approach, we further generate 48 benchmark instances with various problem sizes (from 100 to 900 items in total) and difficulties (i.e. different α values) by randomly sampling products and items from the real-world data. Due to space limit, the benchmark generation algorithm is not given here. Details can be found in the supplementary material.

For the sake of clarity, the four large instances based on the yearly data are referred to as the *real-world instances*, and the 48 benchmark instances generated by random sampling are referred to as the *benchmark instances*.

B. Experiment Setup

All the compared algorithms are implemented in Java. The tabu search is implemented using the library provided in [43]. The branch-and-cut approach is implemented using Gurobi 6.5 [44]. The GP algorithm is implemented using ECJ21 [45]. The experiments for the benchmark instances are carried out on a desktop computer with Intel Core i7-3770 CPU and 8 GB of RAM. The experiments for the real-world instances are conducted on cloud computing facilities provided by VPAC [46]. For each compared algorithm and each instance, 20 independent runs are conducted. The stopping criteria for the compared algorithms are set as follows.

1) Both the tabu search and MSRS method stop at a maximum running time. Based on our preliminary experiment, the running time for 100-item, 400-item, and 900-item instances were set to 10, 100, and 500 seconds, respectively. For the real-world instances, the running time was set to 70 hours on the cloud computing facilities.
2) The branch-and-cut approach is only applied to the 100-item and 400-item benchmark instances with a maximum running time of 2 hours.
3) Unlike the tabu search and MSRS, the GP algorithm [23] is a hyper-heuristic that evolves heuristics/rules that construct solutions by inserting item groups one by one. The GP algorithm trains rules using a set of SLAP-GC instances (training set), and the trained rules can then be used on any unseen test instances.
   To make a fair comparison, for each benchmark instance, the GP algorithm directly evolves rules on that instance, and its training (optimization) performance is compared with other algorithms. The maximal generation is set to 30, which can guarantee the convergence of the GP algorithm. For the large real-world instances, the training process of GP is too time consuming. To address this issue, we randomly select 200 rules obtained by GP from the benchmark instances, and test them on the real-world instances. For each real-world instance, the top 20 results are used for comparison with other algorithms.

C. Tabu Parameters

In Section V-C, two tabu list structures were proposed for the tabu search. It is unknown which one will be better for solving SLAP-GC. In addition, the tabu tenure is an important parameter that can significantly affect the performance of the tabu search. To decide the best tabu tenure value and tabu list structure for the tabu search for SLAP-GC, preliminary experiments are carried out to compare a range of tabu tenure values for both tabu list structures on several benchmark instances.

In the preliminary experiments, a small (100-item), a medium (400-item), and a large (900-item) instance are generated for parameter tuning using the same procedure descried in Section VI-A. For the β parameter, the values $\{0.1, 0.3, 0.5, 0.7\}$ are taken into account. In addition, it is reasonable to believe that the best tabu tenure value depends on the problem size, and a larger problem size should require a longer tabu tenure. Therefore, in the preliminary experiments, we set the

tabu tenure as $\beta \times P$, where β is a tabu tenure coefficient, and P is the number of products in the instance.

Table IV shows the mean and standard deviation of the results of the tabu search with the compared parameter settings over 20 independent runs. For each instance, the smallest mean and standard deviation values are marked in bold. The results shown in Table IV indicate that for all the three instances, the tabu list structure TL_1 and $\beta = 0.7$ obtained the best performance. Therefore, we set the tabu list structure to TL_1 and $\beta = 0.7$ for the tabu search algorithm in the subsequent experiments.

TABLE IV
THE MEAN AND STANDARD DEVIATION OF THE RESULTS OF TABU SEARCH WITH THE TABU LIST STRUCTURE TL_1 AND TL_2 AND DIFFERENT β VALUES OVER 20 INDEPENDENT RUNS. FOR EACH INSTANCE, THE SMALLEST MEAN AND STANDARD DEVIATION VALUES ARE MARKED IN BOLD.

				$\mathbf{TL}_1$		$\mathbf{TL}_2$	
Instance	M	L	β	mean	std	mean	std
1	10	10	0.1	11364.7	13.3	11360.9	9.3
			0.3	11359.2	3.8	11355.6	2.8
			0.5	11354.8	2.8	11360.0	3.2
			0.7	**11353.8**	**2.3**	11369.1	5.4
2	20	20	0.1	134741.7	74.4	134727.7	138.7
			0.3	134679.7	22.8	134716.8	23.7
			0.5	134671.4	21.5	134782.1	34.6
			0.7	**134658.5**	**18.8**	134890.8	45.7
3	30	30	0.1	667167.5	97.8	667180.1	47.9
			0.3	667146.7	42.0	667246.1	53.4
			0.5	667165.3	42.6	667359.8	78.3
			0.7	**667139.5**	**32.5**	667538.6	110.1

D. Experiment Results on Benchmark Instances

First, we compare the performance of the branch-and-cut approach, the GP algorithm [23] and the proposed MSRS and tabu search algorithms on the 48 benchmark instances, which can be categorized into *small instances* (100 items), *medium instances* (400 items) and *large instances* (900 items) There are 12 small instances, 12 medium instances and 24 large instances. Note that the categorization is purely based on problem size, which is an intuitive way to classify instances into difference difficulty levels.

The results of the compared algorithms on the 48 benchmark instances are shown in Tables V–IX. For each instance and each algorithm, the mean, standard deviation and best results over the 20 independent runs are reported, along with the deviation of the mean and best results to the corresponding lower bound, i.e. $\Delta_{\mathrm{LB}}^{fit}(\%) = \frac{fit-\mathrm{LB}}{\mathrm{LB}} \times 100\%$, where LB refers to the lower bound obtained by the branch-and-cut approach. For the branch-and-cut approach, the percentage deviation of its upper and lower bounds, i.e. $\Delta_{bc}(\%) = \frac{\mathrm{UB}-\mathrm{LB}}{\mathrm{LB}} \times 100\%$ is reported as well, where UB refers to the upper bound.

1) Small Instances: Table V shows the average performance of the compared methods on the 12 small instances. The instances have layouts of either 4×25 or 10×10 in the format of $M \times L$. For each layout, there are six instances with different α values. The Mann-Whitney U test is conducted between the results of the GP, MSRS and tabu search algorithms (with significance level of 0.05 and Bonferroni correction). The entry is highlighted in bold if the corresponding algorithm performs significantly better than both of the other two algorithms. The entry is marked with * if the corresponding algorithm performs significantly better than one of the other algorithms.

From Table V, we can see that the branch-and-cut approach can obtain both upper and lower bounds for all the small instances except Instance 9, on which no feasible solution (UB) is found. The $\Delta_{bc}(\%)$ values of some instances (e.g. Instances 1, 5, 10 –12) are very small, indicating that the branch-and-cut approach can obtain near optimal or optimal results for these instances. Among the heuristic search approaches (GP, MSRS and tabu search), it is clear that tabu search significantly outperforms the other two algorithms. Both MSRS and tabu search perform significantly better than the GP algorithm, and achieved less than 1% of deviation to the lower bound obtained by the branch-and-cut approach on most small instances (9 out of 12 instances for MSRS and all 12 instances for tabu search). This demonstrates the efficacy of the proposed BIGO model in solving SLAP-GC. Note that for Instance 1, both MSRS and tabu search consistently reached the upper bound (59793) obtained by the branch-and-cut approach in all the 20 runs. Therefore, it is possible that all of them reach the optimal results for Instance 1. In addition, we can observe from the table that for the MSRS and tabu search approaches, the value of $\Delta_{\mathrm{LB}}^{\mathrm{mean}}(\%)$ decreases with the increase of the α value. This partially demonstrates that the difficulty of the instance decreases with the increase of the α value.

Table VI exhibits the best (smallest) fitness values obtained in 20 runs of the compared algorithms on the small instances. The UB and LB of the branch-and-cut approach are also given for reference. For each instance, the smallest min value among the GP, MSRS and tabu search methods is highlighted in bold. The second smallest min value is marked with *. Overall, the best performance of the compared methods is consistent with the average performance. The tabu search consistently shows better performance than MSRS and GP, and MSRS performs better than the GP method for most instances. When comparing to the best solutions obtained by the branch-and-cut approach (UB), the tabu search obtained better results on seven small instances, while slightly worse results on only three instances in the best case. This shows the effectiveness of the BIGO model and the tabu search in solving SLAP-GC.

2) Medium Instances: Table VII shows the average performance of the compared methods on the 12 medium instances,. The instances have layouts of either 10×40 or 20×20. For each layout, there are six instances with different α values. Among these 12 medium instances, the branch-and-cut approach successfully obtained both upper and lower bounds for only five instances. Also, the UB values of the branch-and-cut approach on these instances are much worse than the results obtained by the GP, MSRS and tabu search algorithms. This indicates that the medium instances are difficult for the branch-and-cut approach to find promising solutions effectively.

Then we compare the heuristic search algorithms on these medium instances and find similar patterns to that of the small instances. The tabu search performs significantly better than GP and MSRS on all the 12 medium instances. The $\Delta_{\mathrm{LB}}^{\mathrm{mean}}$

TABLE V
THE AVERAGE PERFORMANCE OF THE 100-ITEM **SMALL INSTANCES**. THE MANN-WHITNEY U TEST IS CONDUCTED BETWEEN THE RESULTS OF THE GP, MSRS AND TABU SEARCH ALGORITHMS (WITH SIGNIFICANCE LEVEL OF 0.05 AND BONFERRONI CORRECTION). THE ENTRY IS **HIGHLIGHTED** IF THE CORRESPONDING ALGORITHM PERFORMS SIGNIFICANTLY BETTER THAN **BOTH** OF THE OTHER TWO ALGORITHMS. THE ENTRY IS MARKED WITH * IF THE CORRESPONDING ALGORITHM PERFORMS SIGNIFICANTLY BETTER THAN **ONE** OF THE OTHER TWO ALGORITHMS.

No.	M	L	α	branch-and-cut			GP		MSRS		Tabu	
				UB	LB	$\Delta_{bc}(\%)$	mean $\pm$ std	$\Delta_{\text{LB}}^{\text{mean}}(\%)$	mean $\pm$ std	$\Delta_{\text{LB}}^{\text{mean}}(\%)$	mean $\pm$ std	$\Delta_{\text{LB}}^{\text{mean}}(\%)$
1	4	25	0.00	59793	59256	0.91	60018.7 $\pm$ 102.7	1.29	* 59793.0 $\pm$ 0.0	0.91	* 59793.0 $\pm$ 0.0	0.91
2	4	25	0.14	76912	75503	1.87	76845.7 $\pm$ 135.8	1.78	* 76497.8 $\pm$ 83.2	1.32	**75929.3** $\pm$ 5.1	0.56
3	4	25	0.25	34832	27379	27.22	29087.9 $\pm$ 451.3	6.24	* 27675.0 $\pm$ 35.7	1.08	**27594.7** $\pm$ 41.5	0.79
4	4	25	0.33	21396	16422	30.29	17472.5 $\pm$ 467.0	6.40	* 16516.4 $\pm$ 11.8	0.57	**16506.1** $\pm$ 11.2	0.51
5	4	25	0.50	29160	29151	0.03	29951.4 $\pm$ 227.3	2.75	* 29284.1 $\pm$ 29.1	0.46	**29169.9** $\pm$ 2.2	0.06
6	4	25	0.53	32790	32663	0.39	33416.8 $\pm$ 932.6	2.31	* 32852.8 $\pm$ 26.4	0.58	**32736.1** $\pm$ 9.9	0.22
7	10	10	0.08	12200	9765	24.94	9877.6 $\pm$ 24.3	1.15	* 9831.0 $\pm$ 7.4	0.68	**9793.7** $\pm$ 1.2	0.29
8	10	10	0.19	14886	14775	0.75	14992.7 $\pm$ 23.4	1.47	* 14939.6 $\pm$ 13.6	1.11	**14885.6** $\pm$ 1.7	0.75
9	10	10	0.25	-	10971	-	11118.1 $\pm$ 35.4	1.34	* 11001.3 $\pm$ 3.3	0.28	**10988.3** $\pm$ 1.4	0.16
10	10	10	0.33	14898	14898	0.00	15106.0 $\pm$ 95.1	1.40	* 14944.2 $\pm$ 7.1	0.31	**14902.1** $\pm$ 1.2	0.03
11	10	10	0.48	11834	11834	0.00	11937.5 $\pm$ 77.6	0.87	11893.3 $\pm$ 9.6	0.50	**11841.2** $\pm$ 3.5	0.06
12	10	10	0.60	24505	24482	0.09	24816.7 $\pm$ 223.8	1.37	* 24533.2 $\pm$ 7.7	0.21	**24507.9** $\pm$ 2.7	0.11

values of the tabu search approach are all relatively small (less than 1%). Although worse than the tabu search, MSRS still performs significantly better than the GP method on nine out of the 12 medium instances while only outperformed by GP on three instances.

TABLE VI
THE min FITNESS VALUES OF THE 20 RUNS OF THE COMPARED METHODS ON **SMALL INSTANCES**. THE UB AND LB OF THE BRANCH-AND-CUT APPROACH IS GIVEN FOR REFERENCE. FOR EACH INSTANCE, THE SMALLEST min VALUE AMONG GP, MSRS AND TABU SEARCH IS **HIGHLIGHTED**. THE SECOND SMALLEST min VALUE IS STARRED (*).

No.	branch-and-cut		GP	MSRS	Tabu	Smallest
	UB	LB	min	min	min	$\Delta_{\text{LB}}^{\min}(\%)$
1	59793	59256	**59793**	**59793**	**59793**	0.91
2	76912	75503	76497	* 76361	**75924**	0.56
3	34832	27379	27983	* 27593	**27578**	0.73
4	21396	16422	16702	* 16498	**16495**	0.44
5	29160	29151	29652	* 29222	**29167**	0.05
6	32790	32663	33018	* 32772	**32732**	0.21
7	12200	9765	9834	* 9815	**9793**	0.29
8	14886	14775	14963	* 14911	**14882**	0.72
9	-	10971	11083	* 10994	**10986**	0.14
10	14898	14898	14978	* 14922	**14899**	0.01
11	11834	11834	* 11855	11875	**11835**	0.01
12	24505	24482	24604	* 24520	**24505**	0.09

Table VIII shows the smallest fitness values obtained by the compared methods on the medium instances. It can be seen that tabu search obtains the smallest min values among all the compared algorithms on all the medium instances. MSRS obtains smaller min fitness values than the GP method on seven out of the 12 medium instances.

3) Large Instances: Table IX shows the average performance of the compared methods on the 24 900-item large instances, which include four different layouts (10×90, 15×60, 20×45, and 30×30). For each layout, there are six instances with different α values. Due to the complexity of the large instances, the branch-and-cut method is no longer applicable. In fact, JVM (JAVA Virtual Machine) on the test platform runs out of memory for most of the cases. Overall, the tabu search still performs the best among the three compared

TABLE VIII
THE min FITNESS VALUES OF THE 20 RUNS OF THE COMPARED METHODS ON **MEDIUM INSTANCES**. THE UB AND LB OF THE BRANCH-AND-CUT APPROACH IS GIVEN FOR REFERENCE. FOR EACH INSTANCE, THE SMALLEST min VALUE AMONG GP, MSRS AND TABU SEARCH IS **HIGHLIGHTED**. THE SECOND SMALLEST min VALUE IS STARRED (*).

No.	branch-and-cut		GP	MSRS	Tabu	Smallest
	UB	LB	min	min	min	$\Delta_{\text{LB}}^{\min}(\%)$
13	-	317084	338352	* 322554	**321041**	1.25
14	387934	347721	* 352103	355406	**349786**	0.59
15	747116	368271	* 371841	372073	**369516**	0.34
16	-	192857	196214	* 195788	**194092**	0.64
17	197151	178079	* 180470	181861	**179280**	0.67
18	-	160712	* 162969	163330	**161634**	0.57
19	350797	338826	* 341385	342943	**339885**	0.31
20	441729	234583	237792	* 236008	**235308**	0.31
21	-	317153	321024	* 319609	**318891**	0.55
22	-	307822	309951	* 309348	**308660**	0.27
23	-	175234	176644	* 176481	**175572**	0.19
24	-	273963	276473	* 275665	**274993**	0.38

methods. It outperforms both of the other two methods on 21 of the 24 large instances. In addition, MSRS outperforms the GP method on 20 of the 24 instances. We observe that for the instances on which the tabu search performs the best, MSRS performs no worse than the GP method (either significantly better or no significant difference). There are only three large instances (Instances 30, 36, and 42) on which the tabu search failed to show the best performance. Note that these instances have relatively high α values (0.57, 0.57 and 0.56). We can see that they have the highest α values among the instances with the same layout. Recall that the α value partially reflects the difficulty of the instance, and a higher α indicates a simpler instance. Therefore, the three instances may be simpler than the other five instances with the same layout, and the advantage of MSRS and tabu search becomes less evident.

Table X compares the best performance of the compared methods on the 24 large instances. For each instance, the table also gives the number of products and the maximum and mean size of the products for references. For Instance 30 on which MSRS had the best average performance, one can

TABLE VII
THE AVERAGE PERFORMANCE OF THE 400-ITEM **MEDIUM INSTANCES**. THE MANN-WHITNEY U TEST IS CONDUCTED BETWEEN THE RESULTS OF THE GP, MSRS AND TABU SEARCH ALGORITHMS (WITH SIGNIFICANCE LEVEL OF 0.05 AND BONFERRONI CORRECTION). THE ENTRY IS **HIGHLIGHTED** IF THE CORRESPONDING ALGORITHM PERFORMS SIGNIFICANTLY BETTER THAN **BOTH** OF THE OTHER TWO ALGORITHMS. THE ENTRY IS MARKED WITH * IF THE CORRESPONDING ALGORITHM PERFORMS SIGNIFICANTLY BETTER THAN **ONE** OF THE OTHER TWO ALGORITHMS.

No.	M	L	α	branch-and-cut			GP		MSRS		Tabu	
				UB	LB	$\Delta_{bc}(\%)$	mean ± std	$\Delta_{\text{LB}}^{\text{mean}}(\%)$	mean ± std	$\Delta_{\text{LB}}^{\text{mean}}(\%)$	mean ± std	$\Delta_{\text{LB}}^{\text{mean}}(\%)$
13	10	40	0.00	-	317084	-	342720.1 ± 3052.7	8.08	* 324467.2 ± 703.6	2.33	**321842.3** ± 863.0	1.50
14	10	40	0.11	387934	347721	11.56	* 353429.4 ± 749.3	1.64	356618.1 ± 725.1	2.56	**349881.9** ± 60.8	0.62
15	10	40	0.22	747116	368271	102.87	376124.1 ± 4105.4	2.13	* 372450.1 ± 227.5	1.13	**369634.4** ± 76.9	0.37
16	10	40	0.40	-	192857	-	198672.1 ± 1585.7	3.02	* 196106.1 ± 142.6	1.68	**194177.9** ± 57.4	0.68
17	10	40	0.41	197151	178079	10.71	181889.4 ± 1149.6	2.14	* 182319.2 ± 251.7	2.38	**179338.2** ± 49.2	0.71
18	10	40	0.53	-	160712	-	* 163596.2 ± 376.9	1.79	163788.3 ± 224.1	1.91	**161718.6** ± 64.8	0.63
19	20	20	0.06	350797	338826	3.53	* 341515.8 ± 96.7	0.79	344157.1 ± 429.0	1.57	**339922.9** ± 36.9	0.32
20	20	20	0.10	441729	234583	88.30	240217.4 ± 1223.0	2.40	* 236113.3 ± 66.2	0.65	**235368.1** ± 35.5	0.33
21	20	20	0.26	-	317153	-	324931.2 ± 2042.5	2.45	* 319868.6 ± 105.9	0.86	**318974.0** ± 77.1	0.57
22	20	20	0.40	-	307822	-	312305.5 ± 2437.9	1.46	* 309657.0 ± 124.3	0.60	**308701.9** ± 28.9	0.29
23	20	20	0.46	-	175234	-	177487.7 ± 1472.4	1.29	* 176625.4 ± 81.9	0.79	**175601.7** ± 13.5	0.21
24	20	20	0.52	-	273963	-	277548.0 ± 850.5	1.31	* 275802.0 ± 88.2	0.67	**275029.5** ± 22.2	0.39

TABLE IX
THE AVERAGE PERFORMANCE OF THE 900-ITEM **LARGE INSTANCES**. THE MANN-WHITNEY U TEST IS CONDUCTED BETWEEN THE RESULTS OF THE GP, MSRS AND TABU SEARCH ALGORITHMS (WITH SIGNIFICANCE LEVEL OF 0.05 AND BONFERRONI CORRECTION). THE ENTRY IS **HIGHLIGHTED** IF THE CORRESPONDING ALGORITHM PERFORMS SIGNIFICANTLY BETTER THAN **BOTH** OF THE OTHER TWO ALGORITHMS. THE ENTRY IS STARRED (*) IF THE CORRESPONDING ALGORITHM PERFORMS SIGNIFICANTLY BETTER THAN **ONE** OF THE OTHER TWO ALGORITHMS.

No.	M	L	α	GP	MSRS	Tabu
				mean ± std	mean ± std	mean ± std
25	10	90	0.03	946656.7 ± 5009.0	* 938069.5 ± 1271.0	**922575.7** ± 128.5
26	10	90	0.19	819444.3 ± 4006.5	820352.0 ± 1060.0	**810266.3** ± 1780.5
27	10	90	0.30	1023981.6 ± 5407.5	* 1017074.7 ± 1113.7	**1012774.8** ± 3112.9
28	10	90	0.33	687530.3 ± 4713.7	* 682615.5 ± 775.3	**680184.3** ± 1293.5
29	10	90	0.45	774225.2 ± 5798.7	* 768617.7 ± 1181.9	**761268.8** ± 2232.2
30	10	90	0.57	* 820525.1 ± 3955.7	**818885.8** ± 782.9	824876.1 ± 4223.5
31	15	60	0.05	939930.9 ± 17197.9	* 891829.6 ± 595.5	**884329.1** ± 98.8
32	15	60	0.16	664661.7 ± 3758.0	663755.7 ± 635.4	**653696.5** ± 371.3
33	15	60	0.29	657982.1 ± 6174.2	* 651130.8 ± 713.1	**644007.6** ± 1113.2
34	15	60	0.33	1025081.8 ± 12961.8	* 1017069.1 ± 1170.7	**1006818.1** ± 1111.6
35	15	60	0.46	639202.2 ± 8906.7	* 632040.4 ± 584.4	**625477.1** ± 713.1
36	15	60	0.57	**651079.7** ± 852.5	* 654358.4 ± 728.9	657585.0 ± 2749.7
37	20	45	0.06	266480.0 ± 5513.5	264461.3 ± 144.1	**260628.1** ± 114.7
38	20	45	0.10	285562.3 ± 4682.2	* 280832.7 ± 202.5	**277421.3** ± 55.2
39	20	45	0.26	673607.3 ± 4520.0	* 671745.2 ± 320.9	**666923.4** ± 242.6
40	20	45	0.34	846090.4 ± 10846.3	* 837884.9 ± 527.7	**830619.5** ± 357.6
41	20	45	0.45	566952.4 ± 2440.0	* 564728.9 ± 477.7	**558523.2** ± 186.4
42	20	45	0.56	**648025.6** ± 2480.4	651660.4 ± 549.8	* 650959.4 ± 2426.6
43	30	30	0.09	715113.1 ± 2594.5	* 707268.6 ± 354.3	**701639.6** ± 92.9
44	30	30	0.12	709222.6 ± 5020.1	* 683525.9 ± 359.6	**678996.6** ± 38.7
45	30	30	0.26	904422.4 ± 6690.6	* 900490.8 ± 488.4	**893578.7** ± 282.8
46	30	30	0.33	837076.0 ± 11900.3	* 829445.7 ± 614.8	**822313.6** ± 199.3
47	30	30	0.49	535063.6 ± 3659.5	* 531984.3 ± 432.3	**526106.5** ± 67.1
48	30	30	0.54	544058.6 ± 2174.3	* 541775.5 ± 599.2	**536154.8** ± 83.1

TABLE X
THE SMALLEST FITNESS OF THE 20 RUNS OF THE COMPARED METHODS ON **LARGE INSTANCES**. THE NUMBER OF PRODUCTS P AND THE MAXIMUM AND AVERAGE NUMBER OF ITEMS PER PRODUCT ARE GIVEN FOR REFERENCE. FOR EACH INSTANCE, THE SMALLEST min VALUE AMONG GP, MSRS AND TABU SEARCH IS **HIGHLIGHTED**. THE SECOND SMALLEST min VALUE IS STARRED (*).

No.	P	N_p		GP	MSRS	Tabu
		max	mean			
25	40	70	22	939832	935494	**922381**
26	100	38	9	811144	817551	**807203**
27	111	48	8	1013161	1015451	**1008675**
28	110	40	8	679782	680123	**677147**
29	128	44	7	766954	764802	**756927**
30	295	14	3	**812037**	817452	814545
31	38	74	23	906285	890483	**884193**
32	97	36	9	661345	662068	**653131**
33	109	43	8	651257	649417	**642350**
34	112	33	8	1011551	1014342	**1004760**
35	134	41	6	632633	630625	**623880**
36	298	16	3	**649038**	652295	653593
37	88	47	10	263123	264141	**260431**
38	78	56	11	279853	280468	**277345**
39	103	45	8	670935	671131	**666439**
40	101	43	8	838218	836766	**830000**
41	130	41	6	564181	563664	**558245**
42	299	16	3	**645353**	650622	647250
43	57	54	15	707900	706358	**701470**
44	52	59	17	693006	682847	**678913**
45	108	35	8	900145	899567	**893214**
46	109	49	8	828049	827680	**822082**
47	130	43	6	531240	530880	**526033**
48	144	38	6	541652	540588	**536041**

see in Table X that MSRS did not get the best min fitness value. Instead, the GP method obtained the better min result than MSRS (812037 versus 817452). For Instances 36 and 42 on which the GP method had better average performance than the other two methods, it also achieved the smallest min results. Note that these three instances all have much larger number of products than the other instances. Hence, besides the difficulties of the instances, there could be two reasons for the BIGO algorithms not showing better performance than the GP method on these instances.

- First, the number of products in an instance has impact on the performance of the tabu search. The tabu search generates neighboring solutions for the current solution based on two operators, i.e. `resizeInSlf` and `resizeXSlf`, both of which are defined based on products. Thus, the neighborhood size increases with the increase of the number

of products. Furthermore, a larger number of products also leads to a longer tabu tenure. As we can see from Table X, Instances 30, 36, and 42 all have nearly 300 products, which is much bigger than the other instances. As a result, the tabu search fails to achieve the best average and minimum fitness values on these instances. Instance 48 also have the highest α value among Instances 43-48, but it only consists of 144 products, and the tabu search still obtained the best average and minimum results on it.

- Second, the number of products also has impact on the performance of Algorithm 1 (i.e. `Evaluate`$(\cdot)$). For example, `slfSort` reorganizes each shelf so that groups on each shelf are in descending order. Having more groups on a shelf requires more efforts to perform the sorting operation. In addition, `groupSort` will require more computational budget when there are more groups on the shelf.

Given the same total number of items, if an instance has a larger number of products, the average product size will be smaller. For example, the maximum product size is only 14 for Instance 30, while the maximum product size is 70 for Instance 25. Intuitively, instances consist of small products are usually considered as simple ones as the grouping constraint could be tackled more easily than instances with big products. However, the proposed BIGO algorithms show no advantages of tackling those problems comparing to the GP method on the large instances. Based on the analysis above, these two methods may perform better if given longer running time. However, more efficient operators may be designed in the future work for this type of scenarios.

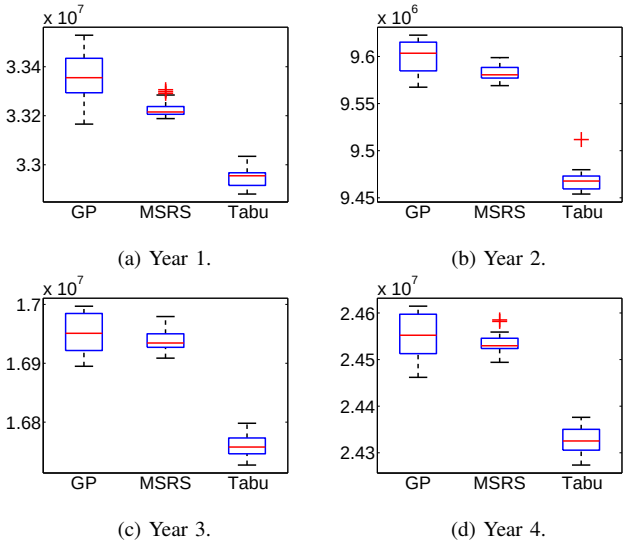
TABLE XI
SUMMARY OF THE STATISTICAL TEST RESULTS ON THE 48 BENCHMARK INSTANCES USING THE MANN-WHITNEY U TEST WITH SIGNIFICANCE LEVEL OF 0.05.

	Win	Draw	Lose
MSRS v.s. GP	38	5	5
Tabu v.s. MSRS	45	1	2
Tabu v.s. GP	45	0	3

4) Summary: Table XI shows the summary of the statistical test results for all the 48 benchmark instances including three comparisons: (1) MSRS versus the GP method; (2) the tabu search versus MSRS; and (3) the tabu search versus the GP method. The first comparison is to verify the efficacy of the BIGO model. The second and third comparison are to verify the efficacy of the tabu search. For each comparison, the Mann-Whitney U test is conducted between the compared algorithms with the significance level of 0.05. The "Win" column in the table records the numbers of instances on which the former algorithm performs significantly better than the later one while the "Draw" column corresponds to those with no significant difference, and the "Lose" column shows those with significantly worse performance. From the table, one can see that the MSRS approach outperforms the GP method on 38 of the 48 instances. When comparing tabu search against the other two methods, there are 45 instances on which tabu search performs significantly better. The GP method is better than the MSRS on only five of the instances while this number decreases to three when comparing against tabu search. These results clearly show that among the three compared methods, tabu search has the best overall performance on these test instances. In addition, the advantage of the MSRS and tabu search algorithms demonstrates the efficacy of the proposed BIGO model in solving SLAP-GC.

E. Experiment Results for Real-word Instances

To analyze the performance of the compared searching heuristics, Fig. 3 depicts the box plots of the 20 independent runs of the compared algorithms on the real-world instances. In addition, Table XII shows the pairwise statistical test results of the GP, MSRS, and tabu search methods on the real-world instances.



(a) Year 1. (b) Year 2.

(c) Year 3. (d) Year 4.

Fig. 3. Box plot of the 20 independent runs of the GP, MSRS, and tabu search methods on real-world instances.

TABLE XII
SUMMARY OF THE PAIRWISE MANN-WHITNEY U TEST ON *Real-world Instances* WITH SIGNIFICANCE LEVEL OF 0.05.

	Win	Draw	Lose
MSRS vs GP	2	2	0
Tabu vs MSRS	4	0	0
Tabu vs GP	4	0	0

One can observe from these figures that the tabu serach performs significantly better than both GP and MSRS on the real-world instances. This can also be observed from the results shown in Table III, i.e. tabu search performs significantly better than GP and MSRS on all the real-world instances. When comparing between the MSRS and GP method, it can be seen that MSRS achieved much better medium fitness values than the GP method on Year 1 and 2. Although the GP method can achieve better minimum fitness values than the MSRS approach, the variances of MSRS are much smaller than those of the GP method, indicating that MSRS has a more stable performance. In addition, one can see from Table III that the MSRS outperforms the GP method on two of the

four real-world instances, and there is no significant difference between MSRS and GP on the rest of the instances. All the aforementioned aspects indicate the effectiveness of the proposed BIGO model, as well as the efficacy of the proposed tabu search approach, on real-world instances.

VII. CONCLUSIONS

In this paper, a novel Bi-level Grouping Optimization (BIGO) model is proposed for solving the Storage Location Assignment Problem with Grouping Constraint (SLAP-GC). A challenging issue in SLAP-GC is that items of one product can only be divided into at most two groups. SLAP-GC consists of two challenging sub-problems, i.e. deciding the optimal grouping of items and the allocation of item groups in the warehouse. It is challenging to solve the above two sub-problems simultaneously. Thus a novel BIGO model is proposed in this paper to solve this problem more effectively. The proposed model transforms the problem into a simpler and more structured form. Based on the proposed BIGO model, different heuristics have been proposed to focus on the optimization of different levels of the proposed model. For the lower level optimization, an efficient local-search-based method is developed employing the domain knowledge of the problem. For the upper level optimization, a Multi-Start Random Search (MSRS) method and a tabu search algorithm are developed. A comprehensive experimental study is carried out to analyze the contributions of different components in the proposed BIGO model and the proposed search algorithms. From the experimental results, we also conclude that both MSRS and tabu search outperform the branch-and-cut approach and the state-of-the-art GP approach for SLAP-GC.

For future studies, more investigations are required on the relation between the efficiency of the proposed methods and the characteristics of the test instances in order to get a better understanding of the complexity of SLAP-GC. Moreover, the current approaches for the real-world instances still require relatively long running time. More efficient techniques are required for large-scale SLAP-GC problems for practical purposes. As one can see from the experimental results, the proposed BIGO algorithms, especially the tabu search, can still get relatively good performance when there are 900 items. The running time required by these BIGO algorithms will tremendously increase when the problem size becomes bigger. One possible approach to deal with this situation is to decompose the original problem into several sub-problems so that each sub-problem can be addressed efficiently and effectively.

REFERENCES

[1] K. J. Roodbergen, *Layout and routing methods for warehouses*. Erasmus University Rotterdam, 2001.

[2] J. v. d. Berg and W. Zijm, "Models for warehouse management: Classification and examples," *International Journal of Production Economics*, vol. 59, no. 1, pp. 519–528, 1999.

[3] J. Gu, M. Goetschalckx, and L. F. McGinnis, "Research on warehouse operation: A comprehensive review," *European Journal of Operational Research*, vol. 177, no. 1, pp. 1–21, 2007.

[4] L. Chen, A. Langevin, and D. Riopel, "The storage location assignment and interleaving problem in an automated storage/retrieval system with shared storage," *International Journal of Production Research*, vol. 48, no. 4, pp. 991–1011, 2010.

[5] P. Montulet, A. Langevin, and D. Riopel, "Minimizing the peak load: an alternate objective for dedicated storage policies," *International journal of production research*, vol. 36, no. 5, pp. 1369–1385, 1998.

[6] L. Chen, A. Langevin, and D. Riopel, "A tabu search algorithm for the relocation problem in a warehousing system," *International Journal of Production Economics*, vol. 129, no. 1, pp. 147–156, 2011.

[7] S. Önüt, U. R. Tuzkaya, and B. Doğaç, "A particle swarm optimization algorithm for the multiple-level warehouse layout design problem," *Computers & Industrial Engineering*, vol. 54, no. 4, pp. 783–799, 2008.

[8] J. Xiao and L. Zheng, "A correlated storage location assignment problem in a single-block-multi-aisles warehouse considering bom information," *International Journal of Production Research*, vol. 48, no. 5, pp. 1321–1338, 2010.

[9] S. Hsieh and K.-C. Tsai, "A bom oriented class-based storage assignment in an automated storage/retrieval system," *The international journal of advanced manufacturing technology*, vol. 17, no. 9, pp. 683–691, 2001.

[10] D. Ming-Huang Chiang, C.-P. Lin, and M.-C. Chen, "Data mining based storage assignment heuristics for travel distance reduction," *Expert Systems*, vol. 31, no. 1, pp. 81–90, 2014.

[11] B. Rouwenhorst, B. Reuter, V. Stockrahm, G. Van Houtum, R. Mantel, and W. Zijm, "Warehouse design and control: Framework and literature review," *European Journal of Operational Research*, vol. 122, no. 3, pp. 515–533, 2000.

[12] J. Gu, M. Goetschalckx, and L. F. McGinnis, "Research on warehouse design and performance evaluation: A comprehensive review," *European Journal of Operational Research*, vol. 203, no. 3, pp. 539–549, 2010.

[13] L. Chen and Z. Lu, "The storage location assignment problem for outbound containers in a maritime terminal," *International Journal of Production Economics*, vol. 135, no. 1, pp. 73–80, 2012.

[14] M. Ang, Y. F. Lim, and M. Sim, "Robust storage assignment in unit-load warehouses," *Management Science*, vol. 58, no. 11, pp. 2114–2130, 2012.

[15] P. Yang, L. Miao, Z. Xue, and L. Qin, "An integrated optimization of location assignment and storage/retrieval scheduling in multi-shuttle automated storage/retrieval systems," *Journal of Intelligent Manufacturing*, vol. 26, no. 6, pp. 1145–1159, 2015.

[16] J. C.-H. Pan, P.-H. Shih, and M.-H. Wu, "Storage assignment problem with travel distance and blocking considerations for a picker-to-part order picking system," *Computers & Industrial Engineering*, vol. 62, no. 2, pp. 527–535, 2012.

[17] J. C.-H. Pan, P.-H. Shih, M.-H. Wu, and J.-H. Lin, "A storage assignment heuristic method based on genetic algorithm for a pick-and-pass warehousing system," *Computers & Industrial Engineering*, vol. 81, pp. 1–13, 2015.

[18] G. K. Adil *et al.*, "A branch and bound algorithm for class based storage location assignment," *European Journal of Operational Research*, vol. 189, no. 2, pp. 492–507, 2008.

[19] M. Bortolini, L. Botti, A. Cascini, M. Gamberi, C. Mora, and F. Pilati, "Unit-load storage assignment strategy for warehouses in seismic areas," *Computers & Industrial Engineering*, vol. 87, pp. 481–490, 2015.

[20] M. Jin, X. H. Mu, L. C. Li, and F. P. Du, "Solving stereo warehouse storage location assignment problem based on genetic algorithm," in *Applied Mechanics and Materials*, vol. 307. Trans Tech Publ, 2013, pp. 459–463.

[21] B. Fahimnia, L. Luong, and R. Marian, "Genetic algorithm optimisation of an integrated aggregate production–distribution plan in supply chains," *International Journal of Production Research*, vol. 50, no. 1, pp. 81–96, 2012.

[22] M. Li, X. Chen, and C. Liu, "Pareto and niche genetic algorithm for storage location assignment optimization problem," in *Innovative Computing Information and Control, 2008. ICICIC'08. 3rd International Conference on*. IEEE, 2008, pp. 465–465.

[23] J. Xie, Y. Mei, A. T. Ernst, X. Li, and A. Song, "A genetic programming-based hyper-heuristic approach for storage location assignment problem," in *Evolutionary Computation (CEC), 2014 IEEE Congress on*. IEEE, 2014, pp. 3000–3007.

[24] J. F. Bard, *Practical bilevel optimization: algorithms and applications*. Springer Science & Business Media, 2013, vol. 30.

[25] G. P. Cachon and S. Netessine, "Game theory in supply chain analysis," in *Handbook of Quantitative Supply Chain Analysis*. Springer, 2004, pp. 13–65.

[26] J. F. Bard and J. T. Moore, "A branch and bound algorithm for the bilevel programming problem," *SIAM Journal on Scientific and Statistical Computing*, vol. 11, no. 2, pp. 281–292, 1990.

[27] H. Sun, Z. Gao, and J. Wu, "A bi-level programming model and solution algorithm for the location of logistics distribution centers," *Applied mathematical modelling*, vol. 32, no. 4, pp. 610–616, 2008.

[28] T. Drezner, Z. Drezner, and P. Kalczynski, "A leader–follower model for discrete competitive facility location," *Computers & Operations Research*, vol. 64, pp. 51–59, 2015.
[29] W. Hu, Y. Hou, L. Tian, and Y. Li, "Selection of logistics distribution center location for sdn enterprises," *Journal of Management Analytics*, vol. 2, no. 3, pp. 202–215, 2015.
[30] H. I. Calvete, C. Galé, and J. A. Iranzo, "Planning of a decentralized distribution network using bilevel optimization," *Omega*, vol. 49, pp. 30–41, 2014.
[31] B. Huang and N. Liu, "Bilevel programming approach to optimizing a logistic distribution network with balancing requirements," *Transportation Research Record: Journal of the Transportation Research Board*, no. 1894, pp. 188–197, 2004.
[32] F. Glover, "Tabu search-part i," *ORSA Journal on computing*, vol. 1, no. 3, pp. 190–206, 1989.
[33] ——, "Tabu searchpart ii," *ORSA Journal on computing*, vol. 2, no. 1, pp. 4–32, 1990.
[34] F. Glover and M. Laguna, *Tabu search*. Kluwer Academic, Boston, MA, 1997.
[35] J.-Q. Li, Q.-K. Pan, P. Suganthan, and T. Chua, "A hybrid tabu search algorithm with an efficient neighborhood structure for the flexible job shop scheduling problem," *The international journal of advanced manufacturing technology*, vol. 52, no. 5-8, pp. 683–697, 2011.
[36] K. A. Dowsland, "Nurse scheduling with tabu search and strategic oscillation," *European journal of operational research*, vol. 106, no. 2, pp. 393–407, 1998.
[37] C. Y. Lee and H. G. Kang, "Cell planning with capacity expansion in mobile communications: A tabu search approach," *Vehicular Technology, IEEE Transactions on*, vol. 49, no. 5, pp. 1678–1691, 2000.
[38] T. L. Ng, "Expanding neighborhood tabu search for facility location problems in water infrastructure planning," in *Systems, Man and Cybernetics (SMC), 2014 IEEE International Conference on*. IEEE, 2014, pp. 3851–3854.
[39] T. James, C. Rego, and F. Glover, "Multistart tabu search and diversification strategies for the quadratic assignment problem," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 39, no. 3, pp. 579–596, 2009.
[40] B. Xin, J. Chen, J. Zhang, L. Dou, and Z. Peng, "Efficient decision makings for dynamic weapon-target assignment by virtual permutation and tabu search heuristics," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 40, no. 6, pp. 649–662, 2010.
[41] S.-M. Pan and K.-S. Cheng, "Evolution-based tabu search approach to automatic clustering," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 37, no. 5, pp. 827–838, 2007.
[42] P. Gutturu and R. Dantu, "An impatient evolutionary algorithm with probabilistic tabu search for unified solution of some np-hard problems in graph and set theory via clique finding," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 38, no. 3, pp. 645–666, 2008.
[43] R. Harder, R. Hill, and J. Moore, "A java universal vehicle router for routing unmanned aerial vehicles," *International Transactions in Operational Research*, vol. 11, no. 3, pp. 259–275, 2004.
[44] I. Gurobi Optimization, "Gurobi optimizer reference manual," 2013. [Online]. Available: http://www.gurobi.com
[45] D. R. White, "Software review: the ecj toolkit," *Genetic Programming and Evolvable Machines*, vol. 13, no. 1, pp. 65–67, 2012.
[46] "Victorian partnership for advanced computing." [Online]. Available: http://www.vpac.org/

Jing Xie received the bachelor's degree in software engineering from Fudan University, Shanghai, China, and the bachelor's degree of science (Honors) from the University College Dublin - National University of Ireland, Dublin in 2011.

She is currently a Ph.D. candidate in the Department of Computer Science and IT, School of Science, RMIT University, Melbourne, VIC, Australia. Her research interests include evolutionary algorithms, meta-heuristics, and matheuristics for combinatorial optimization problems.

Yi Mei is a Lecturer at the School of Engineering and Computer Science, Victoria University of Wellington, Wellington, New Zealand. His research interests include evolutionary computation in scheduling, routing and combinatorial optimisation, as well as evolutionary machine learning, genetic programming, feature selection and dimensional reduction.

He has more than 40 publications, including the top journals in EC and Operations Research (OR) such as IEEE TEVC, IEEE Transactions on Cybernetics, European Journal of Operational Research, ACM Transactions on Mathematical Software, and top EC conferences (GECCO). As the sole investigator, he won the 2nd prize of the Competition at IEEE WCCI 2014: Optimisation of Problems with Multiple Interdependent Components. He received the 2010 Chinese Academy of Sciences Dean?s Award (top 200 postgraduates all over China) and the 2009 IEEE Computational Intelligence Society (CIS) Postgraduate Summer Research Grant (three to four recipients all over the world). He is serving as the committee member of IEEE ECTC Task Force on Evolutionary Scheduling and Combinatorial Optimisation, IEEE CIS Task Force on EC for Feature Selection and Construction and IEEE CIS Task Force on Large Scale Global Optimisation. He is a guest editor of the Genetic Programming Evolvable Machine journal, and co-chair of a number of special sessions in international conferences such as IEEE CEC. He is serving as a reviewer of over 20 international journals including the top journals in EC and OR and PC member of almost 20 international conferences.

Andreas T. Ernst received the Ph.D. degree in network optimization at the University of Western Australia in 1995. He has spent 20 years working at CSIRO on developing innovative operations research solutions for a range of operations research applications. He is currently a Professor with the School of Mathematical Sciences, Monash University, Clayton VIC, Australia, and the deputy director of MAXIMA, the Monash Academy for Cross and Interdisciplinary Mathematical Applications. His research interests focus on scheduling and optimization for large scale industrial applications, including high performance combinatorial optimization algorithms, parallel matheuristics and network optimization. Past projects have included optimization of coal supply chains, scheduling of recreational vehicles (motorhomes), rostering and research into hub location algorithms.

Xiaodong Li (M'03 - SM'07) received the B.Sc. degree from Xidian University, Xi'an, China, and the Ph.D. degree in information science from the University of Otago, Dunedin, New Zealand. He is currently an Associate Professor with the Department of Computer Science and IT, School of Science, RMIT University, Melbourne, VIC, Australia. His current research interests include evolutionary computation, neural networks, complex systems, multiobjective optimization, and swarm intelligence.

Dr. Li was a recipient of the 2013 ACM SIGEVO Impact Award. He serves as an Associate Editor for the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, Swarm Intelligence (Springer), and the International Journal of Swarm Intelligence Research. He is a Founding Member and currently the Vice-Chair of the IEEE Computational Intelligence Society (CIS) Task Force on Swarm Intelligence, and the Vice-Chair of the IEEE Task Force on Multimodal Optimization, and the Former Chair of the IEEE CIS Task Force on Large Scale Global Optimization. He was the General Chair of the SEAL'08, the Program Co-Chair of AI'09 and the IEEE CEC'2012.

Andy Song is a senior lecturer with the Department of Computer Science and IT, School of Science, RMIT University, Melbourne, VIC, Australia. His research area include machine learning especially evolutionary computing based learning on solving complex real-world problems including texture analysis, motion detection, activity recognition, event detection, and optimization. Recently he has been active in establishing cutting-edge techniques, which integrate machine intelligence, mobile and crowd sensing, to benefit transportation, logistics and warehouse industry. He collaborates with a range of industry partners.