# Scaling Up Solutions to Storage Location Assignment Problems by Genetic Programming

Jing Xie[1], Yi Mei[1], Andreas T. Ernst[2], Xiaodong Li[1], and Andy Song[1]

[1] School of Computer Science and IT, RMIT University, Melbourne, Australia
{jing.xie,yi.mei,xiaodong.li,andy.song}@rmit.edu.au
[2] Commonwealth Scientific and Industrial Research Organisation (CSIRO), Melbourne, Australia
Andreas.Ernst@csiro.au

**Abstract.** The Storage Location Assignment Problem (SLAP) is to find an optimal stock arrangement in a warehouse. This study presents a scalable method for solving large-scale SLAPs utilizing Genetic Programming (GP) and two sampling strategies. Given a large scale problem, a sub-problem is sampled from the original problem for our GP method to learn an allocation rule (in the form of a matching function). Then this rule can be applied to the original problem to generate solutions. By this approach, the allocation rule can be obtained in a much shorter time. When sampling the problem, the representativeness is a key factor that can largely affect the generalizability of the trained allocation rule. To investigate the effect of representativeness, two sampling strategies, namely the random sampling and filtered sampling, are proposed and compared in this paper. The filtered sampling strategy adopts more information about the problem structure to increase the similarity of the sampled problem and the entire problem. The results show that the filtered sampling performs significantly better than the random sampling in terms of both solution quality and success rate (i.e., the probability of generating feasible solutions for the large problem). The good performance of filtered strategy indicates the importance of sample representativeness on the scalability of the GP generated rules.

**Keywords:** Scalability, Storage Location Assignment Problem, Sampling Strategy, Genetic Programming.

## 1 Introduction

The Storage Location Assignment Problem (SLAP) [1] is an important optimization problem in warehouse management. It improves the overall operational efficiency (e.g. space utilization [2], total picking effort [3], relocation effort [4] or peak picking load [5]) by rearranging the inventory layout in warehouses. There are many factors to be considered in this problem. The popularity of products is crucial and has been used to determine whether a product can be placed to locations close to a loading zone or not. Also, products frequently ordered simultaneously are considered to have stronger demand dependencies and should be assigned to closer locations [6]. Other factors such as picking strategies [7], resource availability [5] or warehouse maintenance cost [8] may also be included and the problem can be extremely hard to solve.

G. Dick et al. (Eds.): SEAL 2014, LNCS 8886, pp. 691–702, 2014.

There are many approaches for solving SLAPs in literature. Comprehensive surveys are presented in [1] and [9]. The majority of the work in this area is focused on finding the exact location for each product in a warehouse. Deterministic methods such as branch-and-bound have been successfully applied to small instances [10]. Stochastic approaches such as Simulated Annealing [11], Tabu Search [4] or Genetic Algorithms [12] are more widely used for larger problems to get near-optimal results. These methods usually find a good trade-off between solution quality and computational budget. However, the solution found is only applicable to particular scenarios. In reality, the scenario can change dramatically and frequently. To cope with such changes, one often has to repeat the whole optimization procedure and get a completely new solution. Undoubtedly, this is expensive in both time and resource. As a result, a Genetic Programming (GP) approach has recently been proposed, which searches in the space of allocation rules in the form of matching functions instead of solutions for this problem [13]. In this way, one can optimize the matching functions for the past or current scenarios, and then apply it to any future scenario to generate decent solutions efficiently. In contrast to the traditional Integer Linear Programming (ILP), where the size of the search space becomes prohibitive when the problem size reaches 400, the GP method can still achieve good optimization performance. However, the average elapsed time increases from around 300 seconds to about 1700 seconds when the number of items increases from 400 to 900. In other words, the efficiency of this method deteriorates rapidly with increasing problem size. Based on the above preliminary studies, large scale problems are defined as the problems with more than 1000 items.

Unlike the solutions obtained by other methods in literature, the matching function obtained by this GP approach is reusable and efficient and in most of the cases it can get feasible and good solutions for similar sized unseen problems. In this paper, we attempt to extend the GP approach to problems with distinct problem sizes and investigate the re-usability of the obtained matching function in this case. Specifically, a subset of items is sampled from the entire item set to represent the original problem. Then, the GP method is applied to the representative problem so as to obtain the allocation rule in a much shorter time. Finally, the obtained allocation rule is applied to the original problem to generate the corresponding solution for it. A similar approach has been explored in [14] for bin packing problems on data drawn uniformly from one distribution. This experimental setup is not applicable when the representative problems are sampled from the original problem. To investigate the representativeness of these smaller problems, two sampling methods are developed and compared, namely the random sampling and filtered sampling. As the name indicates, the random sampling is a pure random sampling technique for the items. The filtered sampling, on the other hand, filters the items based on some criteria before the sampling to obtain a more representative subset of items. The comparative results show that the filtered sampling performs much better than the random sampling. This demonstrates the importance to consider the characteristics of the representative problem when sampling the subset of items.

The rest of the paper is organized as follows: first, the problem description of SLAP and the recently proposed GP approach are briefly introduced in Section 2. Section 3 discusses the two proposed sampling techniques that are used to generate the representative problem. The experimental studies are carried out in Section 4. Finally, the conclusions and future work are described in Section 5.

## 2 Background

Our research is conducted in a warehouse storing garments. This section presents a brief description of the problem and the recently proposed GP approach, which will be used to solve the large-scale SLAPs in this paper.

### 2.1 Problem Description

Suppose we have a set of products, each consisting of a number of items in different colors and sizes with their own picking frequencies, to be assigned to a set of locations in the given warehouse. Intuitively, the same products are preferred to be stored together, while putting items with huge popularity difference together can lead to inefficient solutions. The grouping constraint was introduced to get rid of this dilemma by allowing the split of one product into several subgroups so that they can be stored to different areas. The total picking-frequency-weighted distance is used as the approximation of the overall picking effort. Frequently picked items are expected to be assigned to locations that are closer to the loading zone of the warehouse. The Integer Linear Programming (ILP) model [15] of the problem is stated in Eqs. (1) – (9).

$$\min \ \varsigma(x) = \sum_{i=1}^{N}\sum_{l=1}^{N} 2\,P_i\,(V_l + H_l)\,x_{il} \tag{1}$$

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

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

$$\sum_{l=1}^{N} y_{sl} \leq 2, \quad s = 1, ..., S \tag{4}$$

$$x_{il} \leq \sum_{s=1}^{S} A_{is} \left( y_{sl} + \sum_{j=1}^{N} A_{js} x_{j,l-1} \right), \quad i, l = 1, ..., N \tag{5}$$

$$\sum_{s=1}^{S} y_{sl} = 1, \quad \forall\, l \bmod C = 1 \tag{6}$$

$$\sum_{s=1}^{S} B_{is} x_{il} \leq \sum_{s=1}^{S} B_{is} \left( \sum_{s=1}^{S} A_{is} y_{sl} \right), \quad i = 1, ..., N \tag{7}$$

$$\sum_{i=1}^{N} A_{is} x_{il} \geq y_{sl}, \quad s = 1, ..., S, \quad l = 1, ..., N \tag{8}$$

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

Eq. (1) is the objective function, which is to minimize the total picking frequency-weighted distance. Eqs. (2) and (3) indicate that each location is occupied by a unique item. Eq. (4) means that each product can be split into at most two subgroups, where $y_{sl}$ takes 1 if location $l$ is the starting point of a subgroup of the product $s$, and 0 otherwise. Eq. (5) ensures that items from the same subgroup are stored to adjacent locations, where $A_{is}$ equals 1 if item $i$ belongs to product $s$, and 0 otherwise. Eq. (6) indicates that the first bin of a shelf must be the staring point of a product. Eq. (7) is a tightening constraint that states that the most popular item of a product is always at the start of a subgroup. Finally Eq. (8) states that the start of a product group should actually have an item from that product (this constraint is not strictly speaking necessary to get a valid bound, but ensures that the $y$ variables have the correct meaning). Table 1 lists all the notations involved in this ILP model.

**Table 1.** Notations for the ILP Model

| Notation | Meaning | Notation | Meaning |
|---|---|---|---|
| $i$ | Index of items, $i = 1, ..., N$ | $l$ | Index of locations, $l = 1, ..., N$ |
| $s$ | Index of products, $s = 1, ..., S$ | $P_i$ | Picking frequency of item i |
| $M$ | Number of shelves | $C$ | Number of bins on a shelf |
| $V_l$ | Vertical distance of location $l$ to the loading zone | $H_l$ | Horizontal distance of location $l$ to the loading zone |
| $A_{is}$ | equals to 1 if item $i$ is in product $s$; 0, otherwise. | $B_{is}$ | equals to 1 if item $i$ is the most popular item in product $s$; 0, otherwise |
| $x_{il}$ | equals 1 if item $i$ is assigned to location $l$, and 0 otherwise. $(i = 1, ..., N; l = 1, ..., N)$ | $y_{sl}$ | equals 1 if location $l$ is a starting point of product $s$, and 0 otherwise. $(s = 1, ..., S; l = 1, ..., N)$ |

## 2.2 A Genetic Programming Approach

The general idea of the GP approach is relatively simple. GP is used to evolve matching functions which help to identify the most suitable subsets of product to an equally sized set of locations. A matching function takes the set of items and consecutive locations as input, and returns a value to reflect the degree of suitability to assign the given items to the corresponding locations. Given a matching function, the solution is constructed from scratch. At each step, the best-fit set of items to the nearest available location is identified and allocated to the set of locations starting from the nearest available location. The solution generation is completed after all the items have been assigned into the locations. When evolving the matching functions, the fitness of a matching function is set to the objective function value of the solution generated by it.

A standard tree-based representation is adopted by the GP. To handle item sets with arbitrary size, statistical data of the picking frequencies of these items and distances of locations to the loading zone are used as the terminal set of the GP method. Simple arithmetic and logic operators are applied to connect these terminals to form the GP tree. Details of this method can be found in [13].

## 3 Methodology

The complexity of the optimization process of the allocation rule by GP depends on the problem size, and is time consuming for large scale problems. Thus, it is impractical to apply the proposed GP directly on the large scale problems. To address this scalability issue, smaller sized problems are generated by sampling subsets of items out of the item set of the original large scale problem as its representatives. The allocation rules are evolved on the smaller sized representative problems, and then applied to the original problems to obtain the solution. For example, given an 1000-item problem, one can sample 200 items out of the total 1000 items to form a 200-item problem, and evolve the allocation rules for this smaller problem by the GP. In this way, the allocation rule can be obtained in a much shorter time.

As demonstrated in [13], the evolved allocation rules managed to obtain promising performance on unseen problems with the same sizes. However, it is unknown whether such property is maintained for those have different sizes, since the difference in problem size may lead to distinct problem structures, and thus different desirable properties of the allocation rules. Therefore, one cannot guarantee that an allocation rule optimized on smaller problems can be generalized well to larger problems. To evolve a more generic allocation rule with better re-usability regardless of the problem size, we propose and compare two sampling techniques, namely the *random sampling* and *filtered sampling*, to generate representative problems for the original problem. Here, the re-usability of an allocation rule is defined as its ability of getting feasible and high quality solutions when applied to problems with different sizes.

### 3.1 Random Sampling

Random sampling is the most intuitive way of sampling. Given a list $I$ of the items, the algorithm of random sampling is as described in Algorithm 1. For example, given the 8-item list shown in Table 2, a subset $\{1, 5\}$ consisting of the first and fifth item is randomly sampled. Ease of implementation is the main advantage of this method. It simply places all the items in a giant sequence and picks the items randomly until the size limitation $Limit$ has been reached. The algorithm starts with an empty list $X$, the number of sampled items $Limit$ and the original list of items. In each iteration, a random number $r$ is generated, and the $r_{th}$ item in the current item list is selected. Then, the selected item is removed from the item list and is inserted into list $X$. The procedure terminates after $Limit$ iterations.

### 3.2 Filtered Sampling

In our previous study [13], we have observed the following behaviours of the proposed GP:

1. It consistently obtained feasible solutions during the optimization.
2. It can obtain feasible and good solutions on most of the unseen problems.

**Table 2.** Example Data Set

| Item No. | SKU | Color | Size | Picking Frequency |
|---|---|---|---|---|
| 1 | $AK001$ | $BLACK$ | $S$ | 221 |
| 2 | $AK001$ | $BLACK$ | $M$ | 1070 |
| 3 | $AK001$ | $BLACK$ | $L$ | 293 |
| 4 | $AK001$ | $WHITE$ | $S$ | 15 |
| 5 | $AK001$ | $WHITE$ | $M$ | 2200 |
| 6 | $AK001$ | $WHITE$ | $L$ | 378 |
| 7 | $BL78$ | $BLUE$ | $XS$ | 735 |
| 8 | $BL78$ | $BLUE$ | $S$ | 467 |
| - | - | - | - | - |

**Algorithm 1.** Random Sampling Procedure

1: Initialize an empty item list $X$;
2: Initialize $Limit$ to a positive integer;
3: **repeat**
4: Assign $r$ with a random number between $[0, size(I)]$;
5: Add $I_r$ to item list $X$;
6: Remove $I_r$ from $I$;
7: $Limit := Limit - 1$;
8: **until** $Limit == 0$;
9: **return** $X$;

3. When there exists a product containing a large number of items, and if a smaller number of items in this product is firstly picked and the rest is not allowed to be split. Then none of the shelves is capable of holding the rest items without violating any constraint. As a result, an infeasible solution occurs.

Based on the above observations, we deduce that the number of items in the products plays an important role in the generalizability of the obtained allocation rule. In order to increase the probability of obtaining an allocation rule that can deal with products with a large number of items, a filtered sampling method is proposed to remove the products with single item. It is described in Algorithm 2. In Step 3, products with one item are firstly deleted as those products are not required to be split and thus can hardly provide information for the learning procedure. In the algorithm, at each step, a set of $\lambda$ consecutive items are picked instead of only one item so as to increase the probability of selecting the items in the same product. The parameter $\lambda$ can be specified to a sufficiently large integer, e.g., 20 or 30, in practice.

## 4 Experiments and Results

The test problems are generated from two raw data sets, each with more than $10,000$ items. Each raw data set is randomly split into five exclusive subsets of items, each

**Algorithm 2.** Filtered Sampling Procedure

Initialize $X$ to an empty item list and $I$ to the list of all items;
Initialize $Limit$ to a positive integer;
Delete from $I$ those products with only one item;
Initialize parameter $\lambda$;
$inter = \lambda$;
**repeat**
    Assign $r$ with a random number between $[0, size(I) - inter]$;
    Add $I_r, ..., I_{r+inter}$ to item list $X$;
    Remove $I_r, ..., I_{r+inter}$ from I;
    $Limit := Limit - inter$;
    $inter := \min\{inter,\ Limit\}$;
**until** $Limit == 0$;
**return** $X$;

consisting of over $2,000$ items. Overall, 10 test problems have been generated for experimental studies, labelled from $p_1$ to $p_{10}$ ($p_1 \sim p_5$ for the first raw data set, and $p_6 \sim p_{10}$ for the second). When sampling the subsets to form the representative problem for each test problem, the number of sampled items is set to $400$. The sampling is repeated five times and thus five representative problems are generated. During the optimization process, the fitness of an allocation rule is defined as the average fitness of the solutions generated by applying the rule to the five representative problems. Finally, the obtained best-fit allocation rule is evaluated by applying it to the original problem and calculating the objective value of the corresponding solution. Thirty independent runs were conducted for each test problem, and the best, worst and average results are recorded.

### 4.1 Random Sampling

Table 3 shows the experimental results for the random sampling method. It lists the number of infeasible solutions obtained in the 30 independent runs and the best, worst and average fitnesses achieved when applying the allocation rule obtained from the representative problems to the original problem. For each test problem, the best known fitness $\alpha$ is obtained by optimizing the allocation rules on the full problem without using representative subproblems. It can be seen that on $p_6 \sim p_8$, the obtained allocation rules failed to get feasible solutions in all the 30 runs. $p10$ shows the best performance in terms of feasibility. In terms of the percentage deviation of the best fitness from the best-known fitness $Diff^{\alpha}_{Min}$, it can be observed that the random sampling obtained promising results on $p_1$, $p_3$ and $p_5$, which are less than $1\%$. Due to the high success rate, the random sampling achieved the best average results on $p_{10}$.

**Table 3.** The Results of Random Sampling

| $Set$ | $\alpha$ | $\beta$ | $Min$ | $Diff^{\alpha}_{Min}$ | $Max$ | $Diff^{\alpha}_{Max}$ | $Avg \pm (Stdv)$ | $Diff^{\alpha}_{Avg}$ |
|---|---|---|---|---|---|---|---|---|
| $p_1$ | 3059169 | 23 | 3078208 | 0.62% | 6673130 | 118.14% | $3716687.90 \pm (3.48E05)$ | 21.49% |
| $p_2$ | 2977428 | 24 | 3029585 | 1.75% | 3877362 | 30.23% | $3133622.43 \pm (2.73E06)$ | 5.25% |
| $p_3$ | 3092760 | 25 | 3120909 | 0.91% | 4463370 | 44.32% | $3254232.40 \pm (5.10E06)$ | 5.22% |
| $p_4$ | 2912623 | 27 | 2943177 | 1.05% | 4259747 | 46.25% | $3036241.17 \pm (1.91E06)$ | 4.24% |
| $p_5$ | 3114503 | 12 | 3130649 | 0.52% | 10086239 | 223.85% | $3530287.10 \pm (1.42E05)$ | 13.35% |
| $p_6$ | 1900213 | 30 | 2337416 | 23.01% | 3898660 | 105.17% | $2788313.30 \pm (9.90E05)$ | 46.74% |
| $p_7$ | 3265076 | 30 | 4135615 | 26.66% | 17004364 | 420.80% | $7389082.73 \pm (1.92E05)$ | 126.31% |
| $p_8$ | 3686151 | 30 | 4339916 | 17.74% | 23698704 | 542.91% | $7331767.53 \pm (2.32E05)$ | 98.90% |
| $p_9$ | 3400729 | 27 | 3436107 | 1.04% | 13076871 | 284.53% | $4313234.23 \pm (2.38E05)$ | 26.83% |
| $p_{10}$ | 3006363 | 8 | 3059604 | 1.77% | 3844230 | 27.87% | $3118322.03 \pm (1.32E06)$ | 3.72% |

[1.] $\alpha$ is the best fitness known so far, obtained by optimizing the set directly.

[2.] $\beta$ is the number of infeasible solutions achieved in total 30 runs.

[3.] $Diff^{B}_{A} = \frac{A-B}{B} \times 100\%$.

### 4.2 Filtered Sampling

Tables 4 and 5 give the results using the filtered sampling method. Four experiments with $\lambda = 10$, 20, 50 and 100 are conducted for each set to compare with random sampling. The tables record the number of infeasible solutions in 30 runs for each set and for each $\lambda$ configuration. The $\lambda$ value of the row for random sampling is denoted by "$-$". The minimum, maximum and average fitness values and the differences of these fitnesses to the best known result for the corresponding test problem are also calculated. In addition, some statistical data related to the representative problems are presented for further discussion. Each representative problem is to allocate 400 items into a warehouse with 8 shelves, each consisting of 50 bins.

To have a clearer understanding of the results in Table 4 and 5, we firstly compare the overall performance of the random sampling method and the filtered sampling method. Table 6 shows the average number of infeasible solutions, the average difference of the best, worst and average fitness obtained by the filtered sampling with different $\lambda$ settings. It shows that the filtered sampling can get more feasible solutions in general. For $p_1$, $p_2$, $p_3$, $p_4$, $p_5$, $p_6$ and $p_9$, the number of infeasible solutions obtained by the filtered sampling is nearly half of that of random sampling. In addition, the solution quality is much better when using filtered sampling. This can be observed by columns $Diff^{\alpha}_{Min}$, $Diff^{\alpha}_{Max}$ and $Diff^{\alpha}_{Avg}$ as the filtered method consistently achieved smaller percentage deviation from the best known results.

Then we investigate the impact of parameter $\lambda$ on the performance of the method. This parameter is used in Algorithm 2 to determine the size of chunks taken after randomly selecting a start point in the data set. The larger the $\lambda$ is set, the more likely it can generate representative problems with products containing larger number of items. This can be observed in Table 4 and 5 where $\gamma_3$ increases with $\lambda$ for most of the test problems.

**Table 4.** Performance Comparison Using Different Representative Problems Generated using Different $\lambda$ Configurations for Filtered Sampling Method

| | | | Results for the Original Sets | | | | | | Representative Problems | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $set$ | $\lambda$ | $\beta$ | $Min$ | $Diff^{\alpha}_{Min}$ | $Max$ | $Diff^{\alpha}_{Max}$ | $Avg$ | $Diff^{\alpha}_{Avg}$ | $\gamma_1$ | $\gamma_2$ | $\gamma_3$ | $\gamma_4$ | $\gamma_5$ |
| | 10 | 19 | 3049702 | −0.31% | 3256170 | 6.44% | 3144771.80 | 2.80% | 123.00 | 3.26 | 16.60 | 21 | 13 |
| | 20 | 19 | 3044269 | −0.49% | 4719413 | 54.27% | 3261829.43 | 6.62% | 115.40 | 3.49 | 21.20 | 25 | 18 |
| $p_1$ | 50 | 7 | 3031050 | −0.92% | 3106600 | 1.55% | 3066181.73 | 0.23% | 103.60 | 3.97 | 25.40 | 31 | 22 |
| | 100 | 5 | 3046582 | −0.41% | 3100820 | 1.36% | 3062909.17 | 0.12% | 64.80 | 6.20 | 34.00 | 37 | 32 |
| | – | 23 | 3078208 | 0.62% | 6673130 | 118.14% | 3716687.90 | 21.49% | 210.20 | 1.92 | 8.20 | 9 | 8 |
| | 10 | 1 | 3004222 | 0.90% | 3064512 | 2.92% | 3037839.73 | 2.03% | 130.80 | 3.07 | 16.20 | 21 | 11 |
| | 20 | 4 | 2996131 | 0.63% | 3051311 | 2.48% | 3025666.50 | 1.62% | 129.00 | 3.14 | 18.80 | 25 | 15 |
| $p_2$ | 50 | 26 | 3020389 | 1.44% | 3745966 | 25.81% | 3145886.40 | 5.66% | 125.40 | 3.29 | 18.40 | 22 | 15 |
| | 100 | 7 | 2989703 | 0.41% | 3072195 | 3.18% | 3034130.20 | 1.90% | 60.00 | 6.69 | 34.60 | 38 | 28 |
| | – | 24 | 3029585 | 1.75% | 3877362 | 30.23% | 3133622.43 | 5.25% | 202.40 | 1.99 | 8.20 | 10 | 7 |
| | 10 | 13 | 3109296 | 0.53% | 4278922 | 38.35% | 3223621.93 | 4.23% | 134.40 | 2.99 | 13.40 | 17 | 11 |
| | 20 | 10 | 3106868 | 0.46% | 7285031 | 135.55% | 3342338.00 | 8.07% | 130.60 | 3.09 | 19.40 | 23 | 17 |
| $p_3$ | 50 | 20 | 3099309 | 0.21% | 3261410 | 5.45% | 3177372.20 | 2.74% | 124.40 | 3.24 | 18.40 | 28 | 15 |
| | 100 | 13 | 3102046 | 0.30% | 3756533 | 21.46% | 3171127.93 | 2.53% | 62.00 | 6.51 | 32.80 | 35 | 30 |
| | – | 25 | 3120909 | 0.91% | 4463370 | 44.32% | 3254232.40 | 5.22% | 204.80 | 1.97 | 8.80 | 11 | 7 |
| | 10 | 22 | 2926350 | 0.47% | 3307064 | 13.54% | 2990174.07 | 2.66% | 134.60 | 2.97 | 15.60 | 18 | 12 |
| | 20 | 22 | 2919410 | 0.23% | 3177614 | 9.10% | 2964727.20 | 1.79% | 123.80 | 3.29 | 17.40 | 22 | 15 |
| $p_4$ | 50 | 11 | 2926303 | 0.47% | 3257099 | 11.83% | 2962987.50 | 1.73% | 122.80 | 3.30 | 22.20 | 28 | 17 |
| | 100 | 3 | 2925870 | 0.45% | 2967114 | 1.87% | 2942840.63 | 1.04% | 63.00 | 6.38 | 31.00 | 34 | 27 |
| | – | 27 | 2943177 | 1.05% | 4259747 | 46.25% | 3036241.17 | 4.24% | 213.60 | 1.89 | 8.60 | 10 | 7 |
| | 10 | 7 | 3129402 | 0.48% | 3446347 | 10.65% | 3161727.47 | 1.52% | 132.80 | 3.05 | 15.60 | 18 | 12 |
| | 20 | 7 | 3128977 | 0.46% | 3403266 | 9.27% | 3168708.33 | 1.74% | 124.40 | 3.23 | 20.80 | 26 | 16 |
| $p_5$ | 50 | 5 | 3131274 | 0.54% | 3732355 | 19.84% | 3166259.27 | 1.66% | 127.60 | 3.20 | 21.00 | 33 | 16 |
| | 100 | 1 | 3131965 | 0.56% | 3165643 | 1.64% | 3148795.97 | 1.10% | 65.00 | 6.18 | 34.60 | 37 | 28 |
| | – | 12 | 3130649 | 0.52% | 10086239 | 223.85% | 3530287.10 | 13.35% | 211.60 | 1.91 | 8.00 | 9 | 7 |

[1.] For each representative problem, the number of products, the average number of items a product has and the maximum number of items a product contains are already known and can be denoted as $\phi$, $\varphi$ and $\Omega$ correspondingly. $\gamma_1$ is the average of $\phi$ for 5 representative problems. $\gamma_2$ is the average $\varphi$ for 5 representative problems. $\gamma_3$, $\gamma_4$ and $\gamma_5$ are respectively the average, maximum and minimum of $\Omega$.
[2.] The definition for $\alpha$, $\beta$ and $Diff$ are the same as in Table 3.

The assumption of proposing filtered sampling is that the GP method can learn to deal with grouping constraints if enough information is provided in representative problems. In other words, when the representative problems include products containing a larger number of items, the GP method can handle the grouping constraint better. The results shown in Tables 4 and 5 are consistent with this assumption. For each of the test problems, we have four rows for different $\lambda$ configuration in filtered sampling. In terms of $\beta$, $Diff^{Min}_{\alpha}$,$Diff^{\alpha}_{Max}$ and $Diff^{\alpha}_{Avg}$, the row with the smallest value is considered as the winner. The comparative results are shown in Table 7. It is obvious that $\lambda = 100$ obtained the best performance, as it was the winner for 7 out of the total 10 test problems in terms of $\beta$, $Diff^{\alpha}_{Max}$ and $Diff^{\alpha}_{Avg}$.

**Table 5.** Performance Comparison Using Different Representative Problems Generated using Different $\lambda$ Configurations for Filtered Sampling Method (Cont.)

| | Results for the Original Sets | | | | | | | | Representative Problems | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $Set$ | $\lambda$ | $\beta$ | $Min$ | $Diff^{\alpha}_{Min}$ | $Max$ | $Diff^{\alpha}_{Max}$ | $Avg$ | $Diff^{\alpha}_{Avg}$ | $\gamma_1$ | $\gamma_2$ | $\gamma_3$ | $\gamma_4$ | $\gamma_5$ |
| | 10 | 30 | 2015539 | 6.07% | 3338767 | 75.70% | 2853695.37 | 50.18% | 74.40 | 5.39 | 25.40 | 31 | 21 |
| | 20 | 20 | 1911381 | 0.59% | 2709338 | 42.58% | 2008070.67 | 5.68% | 66.40 | 6.07 | 28.40 | 40 | 23 |
| $p_6$ | 50 | 15 | 1923527 | 1.23% | 2034104 | 7.05% | 1953522.73 | 2.81% | 57.20 | 7.28 | 42.20 | 52 | 29 |
| | 100 | 4 | 1924827 | 1.30% | 4206497 | 121.37% | 2024823.67 | 6.56% | 25.60 | 15.77 | 69.40 | 77 | 57 |
| | – | 30 | 2337416 | 23.01% | 3898660 | 105.17% | 2788313.30 | 46.74% | 107.60 | 3.75 | 16.60 | 19 | 15 |
| | 10 | 30 | 3386434 | 3.72% | 13010004 | 298.46% | 5235240.37 | 60.34% | 60.20 | 6.66 | 33.00 | 38 | 29 |
| | 20 | 30 | 3348193 | 2.55% | 8896970 | 172.49% | 4538060.60 | 38.99% | 59.20 | 6.86 | 39.60 | 53 | 28 |
| $p_7$ | 50 | 30 | 3285883 | 0.64% | 6158601 | 88.62% | 3900587.93 | 19.46% | 50.60 | 8.25 | 52.80 | 66 | 41 |
| | 100 | 24 | 3325634 | 1.85% | 5276951 | 61.62% | 3624195.83 | 11.00% | 21.80 | 18.51 | 85.20 | 100 | 70 |
| | – | 30 | 4135615 | 26.66% | 17004364 | 420.80% | 7389082.73 | 126.31% | 83.80 | 4.81 | 23.80 | 31 | 19 |
| | 10 | 30 | 4004036 | 8.62% | 26155280 | 609.56% | 6869958.07 | 86.37% | 47.00 | 8.52 | 35.40 | 47 | 27 |
| | 20 | 30 | 3908860 | 6.04% | 23733476 | 543.86% | 6821483.63 | 85.06% | 47.20 | 8.55 | 35.40 | 43 | 30 |
| $p_8$ | 50 | 30 | 3797211 | 3.01% | 14658422 | 297.66% | 5387505.03 | 46.16% | 41.60 | 9.91 | 50.60 | 55 | 44 |
| | 100 | 18 | 3873500 | 5.08% | 6886889 | 86.83% | 4343174.63 | 17.82% | 17.40 | 23.07 | 82.60 | 92 | 76 |
| | – | 30 | 4339916 | 17.74% | 23698704 | 542.91% | 7331767.53 | 98.90% | 54.40 | 7.41 | 20.80 | 22 | 20 |
| | 10 | 7 | 3428812 | 0.83% | 4178877 | 22.88% | 3485899.07 | 2.50% | 85.40 | 4.69 | 22.80 | 30 | 19 |
| | 20 | 10 | 3415128 | 0.42% | 3690611 | 8.52% | 3454856.63 | 1.59% | 76.00 | 5.33 | 29.00 | 39 | 23 |
| $p_9$ | 50 | 10 | 3414876 | 0.42% | 3830894 | 12.65% | 3472938.93 | 2.12% | 91.80 | 4.38 | 29.40 | 33 | 23 |
| | 100 | 6 | 3425275 | 0.72% | 3606875 | 6.06% | 3460712.57 | 1.76% | 33.40 | 12.22 | 65.40 | 74 | 54 |
| | – | 27 | 3436107 | 1.04% | 13076871 | 284.53% | 4313234.23 | 26.83% | 128.00 | 3.15 | 16.20 | 21 | 14 |
| | 10 | 6 | 3041814 | 1.18% | 3476412 | 15.64% | 3083405.33 | 2.56% | 127.60 | 3.15 | 15.00 | 21 | 12 |
| | 20 | 12 | 3028730 | 0.74% | 3555192 | 18.26% | 3097883.93 | 3.04% | 128.60 | 3.17 | 16.00 | 20 | 14 |
| $p_{10}$ | 50 | 14 | 3027124 | 0.69% | 3709570 | 23.39% | 3089759.90 | 2.77% | 106.20 | 3.90 | 22.00 | 27 | 18 |
| | 100 | 11 | 3022651 | 0.54% | 3097094 | 3.02% | 3049117.07 | 1.42% | 63.60 | 6.34 | 31.60 | 38 | 30 |
| | – | 8 | 3059604 | 1.77% | 3844230 | 27.87% | 3118322.03 | 3.72% | 213.00 | 1.89 | 8.60 | 10 | 8 |

[1.] For each representative problem, the number of products, the average number of items a product has and the maximum number of items a product contains are already known and can be denoted as $\phi$, $\varphi$ and $\Omega$ correspondingly. $\gamma_1$ is the average of $\phi$ for 5 representative problems. $\gamma_2$ is the average $\varphi$ for 5 representative problems. $\gamma_3$, $\gamma_4$ and $\gamma_5$ are respectively the average, maximum and minimum of $\Omega$.
[2.] The definition for $\alpha$, $\beta$ and $Diff$ are the same as in Table 3.

In summary, the experimental studies demonstrate that the filtered sampling strategy performs much better than random sampling on the large scale test problems in terms of both the capability of generating feasible solutions and the quality of the generated solutions. The improvement of the filtered sampling with increasing $\lambda$ values indicates the importance of keeping products with larger numbers of items in the representative problems. In other words, the consistency in the product size is playing an important role in retaining the re-usability of the allocation rules obtained by the GP approach.

**Table 6.** Comparison of Overall Performance of Random and Filtered Sampling

| No. | Average of Filtered Sampling ($\lambda = 10, 20, 50, 100$) / Random Sampling | | | |
|---|---|---|---|---|
| | $\beta$ | $Diff^{\alpha}_{Min}$ | $Diff^{\alpha}_{Max}$ | $Diff^{\alpha}_{Avg}$ |
| $p_1$ | 12.5 / 23 | −0.53% / 0.62% | 15.91% / 118.14% | 2.44% / 21.49% |
| $p_2$ | 9.5 / 24 | 0.85% / 1.75% | 8.60% / 30.23% | 2.80% / 5.25% |
| $p_3$ | 14 / 25 | 0.38% / 0.91% | 50.20% / 44.32% | 4.39% / 5.22% |
| $p_4$ | 14.5 / 27 | 0.41% / 1.05% | 9.08% / 46.25% | 1.80% / 4.24% |
| $p_5$ | 5 / 12 | 0.51% / 0.52% | 10.35% / 223.85% | 1.50% / 13.35% |
| $p_6$ | 17.25 / 30 | 2.29% / 23.01% | 61.68% / 105.17% | 16.30% / 46.74% |
| $p_7$ | 28.5 / 30 | 2.19% / 26.66% | 155.30% / 420.80% | 32.45% / 126.31% |
| $p_8$ | 27 / 30 | 5.69% / 17.74% | 384.48% / 542.91% | 58.85% / 98.90% |
| $p_9$ | 8.25 / 27 | 0.60% / 1.04% | 12.53% / 284.53% | 2.00% / 26.83% |
| $p_{10}$ | 10.75 / 8 | 0.79% / 1.77% | 15.07% / 27.87% | 2.45% / 3.72% |

[1.] The definition for $\alpha$, $\beta$ and $Diff$ are the same as in Table 3.

**Table 7.** Numbers of Wins for Different $\lambda$ Configuration

| $\lambda$ | $\beta$ | $Diff^{\alpha}_{Min}$ | $Diff^{\alpha}_{Max}$ | $Diff^{\alpha}_{Avg}$ |
|---|---|---|---|---|
| 10 | 2 | 0 | 0 | 0 |
| 20 | 1 | 3 | 1 | 2 |
| 50 | 0 | 5 | 2 | 1 |
| 100 | 7 | 2 | 7 | 7 |

[1.] The definition for $\alpha$, $\beta$ and $Diff$ are the same as in Table 3.

## 5 Conclusions and Future Work

This paper extends the GP approach with two sampling methods to solve a SLAP with grouping constraints. It adopts a relatively simple concept of generalization from subproblem to the original problem for the purpose of using the GP approach for handling large-scale SLAPs efficiently. The major idea is to evolve the allocation rules on smaller sized representative problems sampled from the original problem. Two sampling methods, random sampling and filtered sampling, are developed to generate the representative problems. This paper conducts a comprehensive experimental study of these two sampling methods and the results demonstrate that the filtered sampling performs better in terms of both feasibility and solution quality. It also shows that the GP method can learn to deal with hard constraints regardless of the problem size if adequate critical information is provided.

In future, several possible extensions of this study could be developed. The size of products in representative problems has been identified as an important factor for the reusability of the allocation rules obtained by the GP method in this paper. More factors are to be determined to develop an automatic procedure for generating better representa-

tive problems. For example, the size of representative problems affects the efficiency of the training procedure and the performance of the evolved allocation rules. Besides, the proposed method has only applied to a SLAP with grouping constraints. More realistic models are to be developed by including the constraints that have not been considered in the current model. For example, more accurate approximations of operational effort rather than the simple picking-frequency-weighted distance will be designed in the future. This can be achieved by using order information instead of picking frequency. We may also consider routing problems in warehouse instead of simply Manhattan distance between two points. In addition, developing new techniques for solving this problem in dynamic environments is another promising direction.

## References

1. Gu, J., Goetschalckx, M., McGinnis, L.F.: Research on warehouse operation: A comprehensive review. European Journal of Operational Research 177(1), 1–21 (2007)
2. Chen, L., Langevin, A., Riopel, D.: The storage location assignment and interleaving problem in an automated storage/retrieval system with shared storage. International Journal of Production Research 48(4), 991–1011 (2010)
3. Önüt, S., Tuzkaya, U.R., Doğaç, B.: A particle swarm optimization algorithm for the multiple-level warehouse layout design problem. Computers & Industrial Engineering 54(4), 783–799 (2008)
4. Chen, L., Langevin, A., Riopel, D.: A tabu search algorithm for the relocation problem in a warehousing system. International Journal of Production Economics 129(1), 147–156 (2011)
5. Montulet, P., Langevin, A., Riopel, D.: Minimizing the peak load: an alternate objective for dedicated storage policies. International Journal of Production Research 36(5), 1369–1385 (1998)
6. Frazele, E.A., Sharp, G.P.: Correlated assignment strategy can improve any order-picking operation. Industrial Engineering 21(4), 33–37 (1989)
7. Lin, C.-H., Lu, I.-Y.: The procedure of determining the order picking strategies in distribution center. International Journal of Production Economics 60, 301–307 (1999)
8. Kofler, M., Beham, A., Wagner, S., Affenzeller, M., Achleitner, W.: Re-warehousing vs. healing: Strategies for warehouse storage location assignment. In: 2011 3rd IEEE International Symposium on Logistics and Industrial Informatics (LINDI), pp. 77–82. IEEE (2011)
9. Rouwenhorstp, B., Reuter, B., Stockrahm, V., Van Houtum, G.J., Mantel, R.J., Zijm, W.H.M.: Warehouse design and control: Framework and literature review. European Journal of Operational Research 122(3), 515–533 (2000)
10. Adil, G.K., et al.: A branch and bound algorithm for class based storage location assignment. European Journal of Operational Research 189(2), 492–507 (2008)
11. Adil, G.K., et al.: Efficient formation of storage classes for warehouse storage location assignment: a simulated annealing approach. Omega 36(4), 609–618 (2008)
12. Zhang, G.Q., Xue, J., Lai, K.K.: A genetic algorithm based heuristic for adjacent paper-reel layout problem. International Journal of Production Research 38(14), 3343–3356 (2000)
13. Xie, J., Mei, Y., Andries, E., Li, X., Song, A.: A genetic programming-based hyper-heuristic approach for storage location assignment problem. In: Proceedings of the 2014 IEEE Congress on Evolutionary Computation, pp. 3000–3007. IEEE Computational Intelligence Society (2014)
14. Burke, E.K., Hyde, M.R., Kendall, G., Woodward, J.R.: The scalability of evolved on line bin packing heuristics. In: 2007 IEEE Congress on Evolutionary Computation, pp. 2530–2537. IEEE Computational Intelligence Society (2007)
15. Wolsey, L.A.: Integer programming, vol. 42. Wiley, New York (1998)