A Genetic Programming-based Hyper-heuristic Approach for Storage Location Assignment Problem

Jing Xie, Yi Mei, Andreas Ernst, Xiaodong Li, Andy Song

***Abstract*— This study aims to solve real-world warehouse Storage Location Assignment Problem (SLAP) under grouping constraints by Genetic Programming (GP). Integer Linear Programming (ILP) formulation is used to define the problem. By the proposed approach, a subset of the items is repeatedly selected and placed into the available current best location of the shelves in the warehouse, until all the items have been assigned with a location. A heuristic called matching function is evolved by GP to guide the selection of the subsets of items. Our comparison between the proposed GP approach and the traditional branch-and-cut approach shows that GP can obtain near-optimal solutions (deviation of less than 0.4% from the optimum) on the training data within a short period of time. Moreover, the evolved heuristics can achieve good optimization result on unseen scenarios, comparable to that on the scenario for training. This shows that the evolved heuristics have good reusability and can be directly applied for slightly different scenarios without any new search process.**

I. Introduction

Warehouses are essential components in industry. The improvement of productivity in daily warehouse operations is of great importance. Typical activities in a warehouse include receiving, storing, picking and delivering. Among these operations, product storage and retrieval are considered as the most crucial and resource-consuming activities [1]. One problem associated with both activities is known as the Storage Location Assignment Problem (SLAP) [2].

Briefly speaking, SLAP aims to determine the optimal storage arrangement strategy for a warehouse so that the total picking cost, or the total picking travel distance, would be minimal. To achieve this goal, it would be ideal to be able to predict the future ordering status, which is unusual difficult to foresee. In practice, one can estimate the picking frequency of the items and correlations between them (the likelihood that they are ordered together) based on historical ordering lists from a certain period of time in the past, assuming the distributions of data remain the same in the near future. Consequently SLAP can be simplified as minimizing the frequency-weighted total distance from the storage locations of the items to the Pick-up/Drop-off (P/D) point subject to the correlation constraints and other case-based constraints. In general, the items with higher demand rates are picked more frequently, which should be stored in more accessible locations. In addition items with stronger correlation (e.g., they are more likely to be ordered simultaneously) should be located closer to each other [3].

Given the past ordering lists, the picking frequency can be calculated directly based on the occurrence of the items. However, the definition of correlation between items is not straightforward. To identify the correlation between items, the Bill Of Material (BOM) information is the most popular and convincing data to be used if available. Products or raw-materials appeared on the same BOM are expected to be picked together more frequently [4] [5] and thus have higher correlation. In the cases where BOM information is unavailable, the correlation may be defined according to the association rule of products found by data mining techniques [6] [7] or sophisticatedly developed measurements such as the cube-per-order index [8]. These studies are all conducted under an assumption that correlated items should be stored to closer locations. In a warehouse storing school wear or work wear, this assumption is no longer reasonable. Items with the same SKU number but different color and size are considered strongly correlated in this warehouse and these items are often required to be stored to adjacent bins instead of closer ones. However, simply assigning the same SKU items next to each other may lead to counter-intuitive solutions, especially when the picking frequencies of different items of the same product are far away from each other. For example, given T-shirts of a same style, size L may have a large demand of 1000 and the demand of size $XXXL$ may be only 10. Hence, a more applicable way is to split them into multiple groups and locate the items based on groups rather than putting all the items together. To this end, a new approach for SLAP with the grouping constraint on products is needed. In this paper we propose a novel method based on Genetic Programming (GP).

Various approaches have been used to solve SLAP in literature. Exhaustive search methods such as branch-and-bound are only applicable to small sized instances [9]. For large scale real-world scenarios, heuristic techniques such as simulated annealing [10] and tabu search [11] are often adopted to obtain near-optimal solutions within a reasonable time frame. All these methods aim to find solutions for a particular scenario, e.g., an optimal storage plan for a given set of items. However in practice, solutions obtained for one scenario may not be applicable for a different scenario, e.g. a future scenario. In this case, a decision rule that can be generalized to past, present and future scenarios would be much more desirable than a specific storage solution. A prominent Evolutionary Computation (EC) method, Genetic Programming is capable of learning rules or heuristics rather

Jing Xie, Yi Mei, Xiaodong Li and Andy Song are with the school of computer science and IT, RMIT University, Melbourne, Australia (email:{jing.xie, yi.mei, xiaodong.li, andy.song}@rmit.edu.au), Andreas Ernst is with Commonwealth Scientific and Industrial Research Organisation (CSIRO), Melbourne, Australia (email: Andreas.Ernst@csiro.au)

than finding solutions. For example GP has been successfully applied to the job shop scheduling problems [12][13], and demonstrated to be capable of evolving efficient reusable dispatching rules that outperform other frequently used human-designed rules on some criteria [14]. We expect that GP can also achieve competitive performance on SLAP, which does share similar properties of combinatorial optimization with job shop scheduling problems.

The rest of the paper is organized as follows: the problem is described in Section II. The proposed GP-based method is described in Section III. The experimental studies are carried out in Section IV. The comprehensibility of evolved heuristics and the comparison with manually constructed heuristics are discussed in Section V. Section VI concludes the study and discusses several directions of future work.

II. Problem Description

This section describes the SLAP with grouping constraint in a single level single block warehouse.

A. SLAP with Grouping Constraint

In SLAP, there is a set of products, each consisting of a number of different items with their own picking frequencies, to be located into a given warehouse. A simplified layout of the warehouse is shown in Fig. 1. The warehouse has a rectangular shape and a P/D point. M homogeneous shelves each consisting of C storage bins are placed in the warehouse, illustrated as columns of blocks. Each block indicates a storage bin. Each bin l is labelled with a unique integer from 1 to N. The width of aisles between shelves can be ignored. The distance from P/D point to a storage bin is defined as the Manhattan distance between these two points, which can be calculated by $V_l + H_l$ (V_l and H_l are vertical and horizontal distance between location l and P/D point). Then, the SLAP with grouping constraint aims at minimizing the frequency-weighted distance, which is defined as $\sum_{i=1}^{N} P_i D_i$, where P_i stands for the picking frequency of item i and D_i indicates the distance from the P/D point to the location of item i. This measurement is widely employed as an approximation of operational cost in warehouse in the field of operational research [9] [10] [15] [16]. The following constraints need to be satisfied:

- Each item is placed in exact one storage bin;
- Each product can be split into at most k groups and each item should be located adjacent to at least one of the other items in the same group. Here, only the bins on the same shelf and next to each other are considered to be adjacent locations. For example, in Fig. 1, location 1 and 2 are adjacent locations, while location 6 and 11 are not. This is referred to as *grouping constraint* in this paper.

B. An ILP Formulation

The $0-1$ decision variables of $x_{il}(i = 1, ..., N; l = 1, ..., N)$ and $y_{sl}(s = 1, ..., S; l = 1, ..., N)$ are defined to develop the ILP model of the described SLAP, where N is the number of items/bins and S is the number of products. x_{il} equals 1 if item i is assigned to location l, and 0 otherwise. y_{sl} equals 1 if location l is a starting point of product s, and 0 otherwise. Given $x_{il} = 1$ (item i is from product s), the location l is called a starting point of product s either no other item of product s is located to location $l-1$ or location l is the first bin on the shelf.

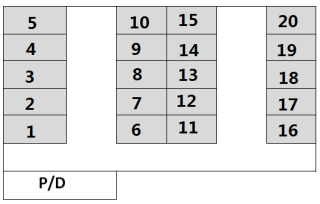

Fig. 1. Example of Warehouse Layout

Then, the ILP model of the SLAP with grouping constraint can be stated as follows:

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

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

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

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

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

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

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

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

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

where

- A_{is} is 1 if item i is from product s; 0, otherwise.
- B_{is} is 1 if item i is the most popular item in product s; 0, otherwise.

The objective function is to minimize the total picking frequency-weighted travelling cost. The distance of location l to the P/D point is calculated by the vertical and horizontal distance of these two points, which is $2(V_l + H_l)$. Constraint (2) ensures that each item is assigned to exact one storage location. Constraint (3) ensures that one storage location is occupied by exact one item. Constraint (4) ensures that each

product has no more than k starting points. Constraint (5) ensures that each location is either a starting point of a product or preceded by the location of another item from the same product. Constraint (6) ensures that the first bin of each shelf must be a starting point. Constraint (7) ensures that for each product, the item with the largest picking frequency is located at a starting location. Constraint (8) ensures that the number of starting points for each product does not exceed the total number of items in it.

C. Discussion

The value of parameter k affects the quality of the solution. Theoretically, a large k yields a good solution. In the extreme cases where $k \geq \sum_{i=1}^{N} A_{is} (\forall s = 1, ..., S)$, the correlation between items can be ignored and a trivial solution can be found by sorting the items from most to least frequent and putting them from the nearest location to the farthest. On the contrary, a small k is desirable by decision makers as it causes fewer management difficulties. However, when k is 1, no split is allowed and the described SLAP is NP-complete. A special case of the problem is a PARTITION PROBLEM [17], which is known to be NP-complete, by assuming the warehouse only has two shelves and all items have exactly the same picking frequency. Based on the above considerations, we set k to 2 to achieve a good tradeoff between quality of solutions and management difficulties.

III. A GP-BASED APPROACH

As described in Section II, the investigated SLAP involves two interdependent tasks: grouping and assigning. On one hand, the problem of grouping items is not evaluable until these items are assigned to locations. On the other hand, the assignment of items should not violate the grouping constraint. In this section, we present a GP-based hyper-heuristic approach to address the investigated SLAP, which has two phases. First, a matching function, which is also referred to as heuristics in previous sections, is evolved by GP to evaluate the candidate subsets of items. Then, a greedy assignment strategy is employed to locate all the items into the shelves repeatedly until all the items have been located. Specifically, at each step, the best subset of unallocated items identified by the matching function evolved by GP is selected and located into the current best empty location on the shelves. Details of the proposed method is presented in the following.

A. GP Representations

We use a tree-based GP to evolve the matching function. A matching function matches a set of sequential locations and a set of items (denoted as s') and returns a value reflecting the degree of suitability of locating the set of items to the set of locations. A higher value implies that it is more suitable to locate the set of items to the set of locations.

To evolve the matching functions, three arithmetic operator $+, -, \times$ and the following two simple customized functions are used.

TABLE I
TERMINAL SET OF GP

Terminal	Meaning
$Mean$	Average picking frequency of s'
$Card$	Cardinality of s'
Min	The picking frequency of the least popular item in s'
Max	The picking frequency of the most popular item in s'
Sum	The total picking frequency of s'
Std	The standard deviation of picking frequency in s'
AB	Number of bins available on the shelf
C	Number of bins on each shelf
Pr	This value is set to 1 if and only if the cardinality of s' is equal to AB; 0, otherwise
0,1	0 and 1

1) IF: This function accepts three parameters (α, β, γ). It returns β if $\alpha \geq 1$, and γ otherwise.
2) $\leq$: This function accepts two parameters (α, β). It returns 1 if $\alpha \leq \beta$, and -1 otherwise.

Information about the locations and items is summarized and used as terminals. The basic features of s' are summarized and described by several commonly used statistical properties such as sum, mean and standard deviation. This representation ensures that the matching function can handle item sets with arbitrary size. Details of the terminal set is given in Table I.

B. Fitness Evaluation

The evaluation of a matching function f consists of two steps: solution construction and fitness calculation. First, a solution is constructed by the greedy assignment strategy based on the matching function. It keeps identifying the best subset of the unallocated items using the matching function and locate it at the current best available location until all the bins have been occupied. It is described in Algorithm 1.

Algorithm 1 Solution Construction Procedure for an Evolved Matching Function f

1: Mark all the items as unlocated and all bins as empty;
2: **repeat**
3: Obtain the current best available location l_0 based on the status of the bins;
4: Identify the best subset of unlocated items using the matching function and locate them to sequential locations starting from l_0;
5: Mark all the items in the assigned subset as located, and set corresponding x_{il}'s to 1;
6: **until** All the items are located;
7: **return** x_{il}'s

In line 3, the empty bin with the shortest Manhattan distance to the P/D point is selected as l_0. In line 4, a pre-selection scheme is applied to remove the subsets of items

that is not optimal in the current scenario. For example, given l_0 with a distance of 2 to the P/D point, a second best location l_1 with distance of 4 to the P/D point, and a product $\theta = \{\theta_1, \theta_2, \theta_3\}$ with picking frequencies of $\{300, 200, 100\}$, there are seven different non-empty subsets of θ: $\{\theta_1\}$, $\{\theta_2\}$, $\{\theta_3\}$, $\{\theta_1, \theta_2\}$, $\{\theta_1, \theta_3\}$, $\{\theta_2, \theta_3\}$, $\{\theta_1, \theta_2, \theta_3\}$. Among these subsets, we can prove that some subsets must lead to better solutions than others without evaluating the entire solution based on the following property: For each pair of items (i, j) satisfying $A_{is} = A_{js} (\forall s = 1, \ldots, S)$ (that is, they belong to the same product), $P_i > P_j$ and $D_i > D_j$, a better solution can be obtained by exchanging the locations of item i and item j. In the above example, if the subset of $\{\theta_1, \theta_3\}$ is selected and placed to locations l_0 and $l_0 + 1$, respectively, the best location available for θ_2 is l_1. Then, we have $D_1 = 2$, $D_3 = 3$ and $D_2 = 4$. In this case, the pair of items (θ_2, θ_3) satisfies the above condition. Thus, exchanging θ_2 and θ_3 leads to a better solution, which can be obtained by selecting the subset of $\{\theta_1, \theta_2\}$ in the first place. In this sense, $\{\theta_1, \theta_2\}$ is better than $\{\theta_1, \theta_3\}$. Similarly, we have $\{\theta_1\}$ is better than $\{\theta_2\}$ and $\{\theta_3\}$, and $\{\theta_1, \theta_2\}$ is better than $\{\theta_2, \theta_3\}$. As a result, only (θ_1), (θ_1, θ_2) and $(\theta_1, \theta_2, \theta_3)$ are retained.

With the above pre-selection scheme, the number of subsets to be evaluated by the matching function can be much reduced. After the pre-selection, the remaining subsets are evaluated by the matching function and the subset with the biggest return value is selected. When there are multiple subsets with the same return value, the subset with biggest cardinality that is smaller than or equal to the number of the residual number of bins on that shelf is selected.

After the solution is constructed by the above greedy algorithm, the fitness of the matching function is then defined by Eq. (10).

$$Fitness = \varsigma(x) + \sum_{i=1}^{N} \sum_{l=N+1}^{N'} P_i \times 2(V_l + H_l + \delta) \times x_{il} \quad (10)$$

where the x_{il}'s are the decision variables of the constructed solution, $\varsigma(x)$ is defined by Eq. (2). Note that in the constructed solution, the number of items located on a shelf may exceed its capacity when the size of the located subset is larger than the residual number of bins of the shelf. In this case, a set of imaginary bins are introduced and indexed from $N + 1$ to N' (N' is a sufficiently large number, e.g., MN). The items outside the shelves are then punished by the penalty function $P_i \times 2(V_l + H_l + \delta) \times x_{il}$, where δ is the penalty coefficient.

IV. Experiments and Results

Various GP settings have been examined in preliminary experiments to find the best configuration. The results show that the search converges quickly at the early stage of evolution. Hence, the number of generations is set to 30 in the subsequent experiments, which is adequate for the convergence of the algorithm. The results also show that different setting of mutation and crossover rates do not have significant impact on the quality of the solution. The maximal depth of the tree is set to 7, because the computational time increases rapidly with the maximal depth of the tree but the quality of solutions change little when the depth is bigger than 7. The penalty coefficient is set to N, which is the total number of storage bins. The details of parameter settings are listed in Table II.

TABLE II
Runtime Parameters of GP

Parameter	Value	Parameter	Value
Population Size	1000	Elitism Rate	5%
Generations	30	Crossover Rate	76%
Penalty Coefficient (δ)	N	Mutation Rate	19%
Max-depth	7	Min-depth	4

TABLE III
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
-	-	-	-	-

We used the data collected from a real warehouse operation. This includes over 12000 unique items of nearly 500 different products. Each item has a color, size and total picking frequency within one year. As the real data is confidential, we exhibit an example data set in Table III for reference. It can be seen that the items from the same product can have significantly different picking frequency (e.g., No. 4 versus No. 5). Twenty artificial data sets are generated based on the real data by random sampling. Then, the proposed GP compared with the branch-and-cut on the data to evaluate its training performance. Additionally, to evaluate the test performance, the original data set is also randomly split into several equally sized subsets for cross validation.

A. Optimization Performance

In this section, we evaluate the training performance in terms of solution quality by comparing with the branch-and-cut method. The comparison is only conducted on small data sets due to the scalability issue of the branch-and-cut. The branch-and-cut is coded in Java using Gurobi 5.5.0 library [18] and run on a PC with Intel Core $i7 - 3770$ CPU with 8GB RAM. The GP system used to train matching functions is coded based on ECJ21 library [19] and run on the same computational platform. Twenty problems with different number of items ranging from 25 to 900 are randomly generated. Five independent runs are conducted for GP, and the best and average results are recorded.

TABLE IV
SOLUTION QUALITY OF THE PROPOSED HYPER-HEURISTIC METHOD

		Branch-and-cut		Proposed GP				
No.	N	$Best$	$ElapsedTime$	$Best$	$Mean$	Std	$Diff_{\min}$	$ElapsedTime$
1	25	1685	$0s$	1685	1685	0	0.000%	$2s$
2	25	2554	$0s$	2554	2554	0	0.000%	$3s$
3	25	2280	$0s$	2280	2280	0	0.000%	$3s$
4	25	2350	$0s$	2350	2350	0	0.000%	$4s$
5	64	7522	$5s$	7523	75264	3.58	0.013%	$12s$
6	64	9506	$4s$	9506	9506	0	0.000%	$13s$
7	64	9231	$10s$	9233	9237	2.91	0.022%	$10s$
8	64	11280	$75s$	11289	11294	6.67	0.080%	$11s$
9	100	31342	$24s$	31364	31384	18.41	0.070%	$20s$
10	100	15781	$332s$	15786	15789	1.82	0.031%	$19s$
11	100	32997	$321s$	33036	33064	17.71	0.012%	$20s$
12	100	15561	$17403s$	15616	15669	34.88	0.353%	$16s$
13	400	–	–	196636	196810	118.67	–	$348s$
14	400	–	–	134436	134772	207.74	–	$351s$
15	400	–	–	58575	58788	427.63	–	$267s$
16	400	–	–	290241	290541	246.43	–	$266s$
17	900	–	–	648221	649060	512.40	–	$1720s$
18	900	–	–	599890	600205	434.80	–	$1472s$
19	900	–	–	610375	611147	652.20	–	$1491s$
20	900	–	–	621154	621845	468.01	–	$2101s$

[1.] $Best$ is the best fitness during the run.
[2.] $Mean$ is the average fitness of during the run.
[3.] Std is the standard deviation of fitnesses.
[4.] $Diff_{\min}$ is the difference between best fitness obtained by the GP and the optimal fitness obtained by the branch-and-cut.

Table IV shows the results of the branch-and-cut approach and the proposed GP method. The difference $Diff_{\min}$ between the best solution obtained by the GP and the optimal solution is defined as follows:

$$Diff_{\min}(F_{\min}, F_{opt}) = \frac{F_{\min} - F_{opt}}{F_{opt}} \times 100\% \qquad (11)$$

where $F_{\min}$ and F_{opt} are the fitness values of the best solution obtained by the GP and the optimal solution, respectively. Note that the exact branch-and-cut approach is only applicable to the instances with $N \leq 100$ due to the restriction of computational resources. Thus, the corresponding columns and $Diff_{\min}$ are unavailable and marked as "–" for the larger instances with $N \geq 400$.

It is seen that the branch-and-cut approach obtained the global optima of small sized ($N \leq 64$) instances in a short time. However, its computational time increased significantly with the increase of N and became prohibitive on our experimental platform due to out of memory for $N > 100$. On the contrary, the proposed GP showed much better scalability, and managed to reach nearly optimal best performance ($Diff_{\min} < 0.4\%$) on the instances with $N \leq 100$. For the larger instances, although there is no other results to be compared with the GP, it can still be seen that the proposed GP showed quite stable performance, as the standard deviations are much smaller than the corresponding mean values. In summary, for the training data, the proposed GP method obtained nearly optimal solutions reliably and showed good scalability to the problem size.

B. Performance on Unseen Scenarios

The major goal of the proposed hyper-heuristic methodology is to find heuristics that can be reused on unseen situations (test data) by training on existing data (training data). The problem is investigated under the assumption that training data and test data have the same distribution, which means the demand rates of products do not change dramatically. To validate this, the original data is split into five subsets by random selection for 5-fold cross validation. Fig. 2 shows the cumulative distribution functions (CDFs) of picking frequencies of items in these subsets. We can see that these subsets have almost the same CDF curve, which means that they have very similar distribution on picking frequency. This is consistent with our assumption.

For evaluating a given matching function, four solutions are first constructed using the matching function, each for one training data set. Their fitnesses are then calculated by Eq. (10) and the mean of the four fitnesses is used as the fitness of the matching function. In each generation, the best individual is used to evaluate the test data. This approach is similar to the real scenario, e.g. given the past four year's data for training to find heuristics that can be used in next year. We conduct 30 independent runs and Fig. 3 shows the convergence curves of the average fitness of the GP individuals on the training and test data. It can be seen that the convergence curve on the test data is consistent with that

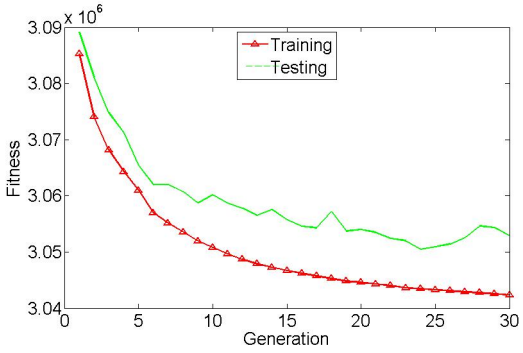

Fig. 3. Average fitness of the GP individuals on the training and test data

TABLE V
PERFORMANCE OF TRAINING AND TEST OF DIFFERENT DATA SETS

	Training		Test		
No.	ε	$Mean \pm Std$	ε	$Mean \pm Std$	$Diff$
1	100%	3046462 ± 8121	73.3%	3070166 ± 44932	0.78%
2	100%	3000394 ± 7906	86.7%	3008722 ± 11540	0.28%
3	100%	3105142 ± 7241	66.7%	3118174 ± 26309	0.42%
4	100%	2927105 ± 5977	90%	2931468 ± 9416	0.15%
5	100%	3133088 ± 8307	100%	3136116 ± 8868	0.097%

1. ε is the proportion of feasible solutions.
2. $Diff$ is the difference of average fitnesses for training and test, which is calculated by $(Mean_{test} - Mean_{training})/Mean_{training}$.

on the training data. This implies that if a matching function is considered to be better on the training data, it is also better on the test data. In other words, the quality of the matching function can generalize from the training data (past) to the test data (future).

Table V shows the details of the results. The i^{th} row indicates that the i^{th} subset is selected as the test data, and the other four subsets are used as the training data. It is seen that the feasible solutions can always be obtained on the training data ($\epsilon = 100\%$). For the test data, the probability of obtaining feasible solution is still high ($\epsilon \geq 66.7\%$), which means there are at least two feasible solutions out of three runs of the GP. Besides, the small $Diff$ value ($Diff \leq 0.78\%$) indicates that the test performance is very close to the training performance.

V. DISCUSSIONS

In this section, we firstly discuss several simple human-designed matching functions and compare them with the best heuristics evolved by the GP, as shown in Table VI. Secondly, we investigate the comprehensiveness of the evolved tree.

A. Comparison with Manually Constructed Heuristics

1) Heuristic 1: $Mean$: Intuitively, the items with higher picking frequencies should be located to better locations. Hence, we tried $Mean$ first. When using this heuristic, more than half of the 20 instances failed to get feasible solutions and obvious gaps can be observed comparing with the proposed GP method. Even for the best case, there still exists 1.26% difference.

2) Heuristic 2: $Mean \times Sum$: Another factor that worth considering is the Sum. Several calculations of $Mean$ and Sum such as $Mean + Sum$ and $Sum - Mean$ have been tried and we only exhibit the one with the best result. This heuristic performs better than the previous one. The results for 8 instances are almost as good as the proposed GP one (less than 1% difference). But it still requires considerable improvement considering that 6 instances failed to get feasible solutions.

3) Heuristic 3: $(1 + Pr) \times Mean \times Sum$: To increase the chance of getting feasible solutions, we add term Pr when constructing the third heuristic. When constructing solutions the chance of getting feasible solutions is increased as each time when there exists a set of items that can fill all the empty bins on the shelf for l_0, the return value for that set of items provided by the matching function is doubled. The results show that the number of feasible solution increase from 14 out of 20 to 19 out of 20. However, the solution quality of some instances is not as good as previous one.

4) Heuristic 4: $(1 + Pr) \times Mean \times Sum + Max \times Min$: In this matching function, term Max and Min is taken into consideration. The rough idea is putting items with similar picking frequency into same group, especially those popular items. Although some instance failed to get feasible solutions, the quality of the solutions for most of the instance is improved considerably.

As we can see, it is not a very easy task to design heuristics for good feasible solutions. The results generated by manually constructed matching functions are always worse (difference range from 0.04% to 46.43%) than the solutions obtained by the proposed GP method. Although for instance 2 and heuristic 3 or instance 10 and heuristic 4, of which the difference is below 0.1%, the proposed GP method still shows advantage in terms of ease of use as it only need simple terminals and functions. This task could be extremely difficult for human when there is not enough domain knowledge.

B. Analysis of Evolved GP Programs

Fig. 4 illustrates an example of evolved tree in Section IV-B. We can see that the terminals related to the picking frequency ($mean$, sum, max) of items have been identified as important factors in this function. For example, max is squared and has a great contribution to the return value. The multiplication of $mean$ and max or sum and max indicates that subsets with bigger $mean$, sum, max tend to be accepted. This is partially consistent with those manually constricted heuristics in Section V.

VI. CONCLUSIONS AND FUTURE WORK

In this study, we investigate a SLAP with grouping constraint of items in stock. The problem is defined with an ILP formulation to find the optimal assignment of items at minimum operational cost. Matching functions are evolved to address aforementioned novel SLAP. This method is demonstrated to be able to find near optimal solutions within a reasonable time period on different-scaled problems. Such

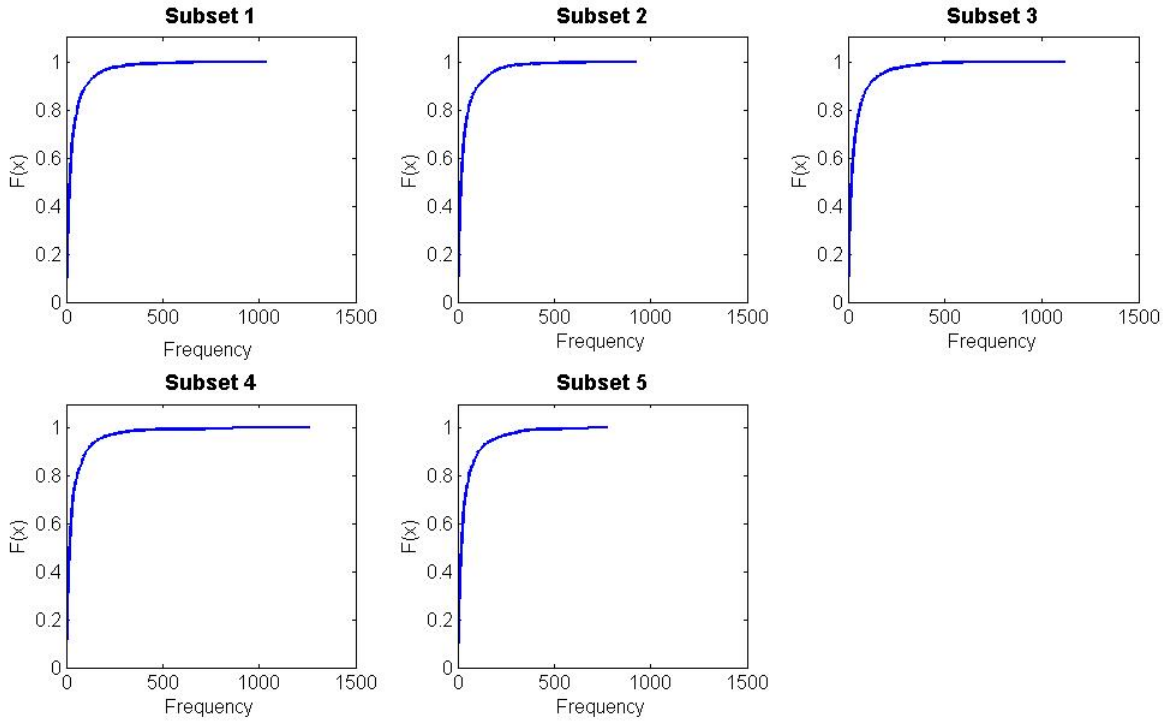

Fig. 2. Cumulative Distribution Functions (CDFs) of Frequency of the Items in the Subsets for Five-fold Cross Validation.

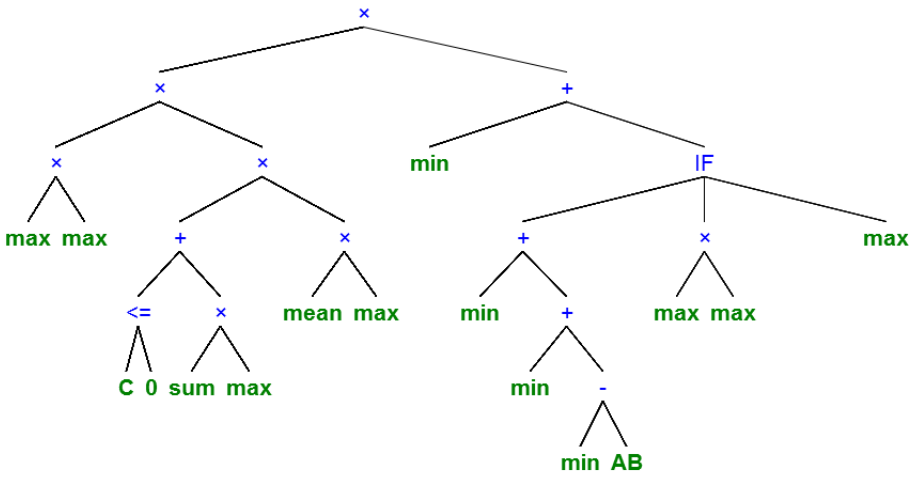

Fig. 4. Example of Evolved Tree

good training performance can also generalize to the scenarios with similar properties. This opens up the possibility of using this GP approach on SLAPs in dynamic environments in future study. For example, matching functions trained by last winter's data may also perform well for different seasons although the data for each season are not exactly the same.

In future, several possible extensions of this study can be developed. First, the selection of terminal sets can be further explored. In current approach, we include all the possible factors that might be useful when designing the methodology which may lead to a huge search space. Some redundancies will increase the difficulty to reach the global optimum. On the other hand, if the terminal set is too small, some essential factors may be excluded. Second, besides the correlation of items, dependencies of products will also be taken into consideration to build a more realistic model, as well as the relocation cost required for rearrangement.

REFERENCES

[1] JP van den Berg and WHM Zijm. Models for warehouse management: Classification and examples. *International Journal of Production*

TABLE VI

COMPARISON BETWEEN PROPOSED GP AND MANUALLY CONSTRUCTED HEURISTICS

No	N	Proposed GP	Heuristic 1			Heuristic 2			Heuristic 3			Heuristic 4		
			$Fitness$	$Diff_{GP}$	$Feasible$	$Fitness$	$Diff_{GP}$	$Feasible$	$Fitness$	$Diff_{GP}$	$Feasible$	$Fitness$	$Diff_{GP}$	$Feasible$
1	25	1685	1735	2.97%	Y	1698	0.77%	Y	1698	0.77%	Y	1698	0.77%	Y
2	25	2554	2642	3.45%	N	2582	1.10%	N	2555	0.04%	Y	2582	1.10%	N
3	25	2280	2399	5.22%	N	2289	0.39%	Y	2306	1.14%	Y	2289	0.39%	Y
4	25	2350	2380	1.28%	Y	2366	0.68%	Y	2366	0.68%	Y	2366	0.68%	Y
5	64	7523	7655	1.75%	Y	7576	0.70%	Y	7579	0.74%	Y	7585	0.82%	Y
6	64	9506	10171	7.00%	N	10131	6.57%	N	10134	6.61%	N	9725	2.30%	N
7	64	9233	9439	2.23%	Y	9254	0.23%	Y	9286	0.57%	Y	9248	0.16%	Y
8	64	11289	15638	38.52%	N	11395	0.94%	Y	11544	2.26%	Y	11399	0.97%	Y
9	100	31364	31794	1.37%	Y	31821	1.46%	Y	31839	1.51%	Y	31398	0.11%	Y
10	100	15786	16102	2.00%	Y	15854	0.43%	Y	15926	0.89%	Y	15797	0.07%	Y
11	100	33036	36092	9.25%	N	33449	1.25%	Y	33443	1.23%	Y	33297	0.79%	Y
12	100	15616	22347	43.10%	N	16155	3.45%	N	16068	2.89%	Y	17131	9.70%	N
13	400	196636	242378	23.26%	N	215714	9.70%	N	200337	1.88%	Y	198688	1.04%	Y
14	400	134436	209310	55.69%	N	145239	8.04%	N	141682	5.39%	N	135168	0.54%	Y
15	400	58575	109343	86.67%	N	70857	20.97%	N	60144	2.68%	Y	61612	5.18%	N
16	400	290241	367776	26.71%	N	291693	0.50%	Y	293913	1.27%	Y	291934	0.58%	Y
17	900	648221	662282	2.17%	Y	657537	1.44%	Y	658590	1.60%	Y	653285	0.78%	Y
18	900	599890	612510	2.10%	Y	606103	1.04%	Y	607272	1.23%	Y	603890	0.67%	Y
19	900	610375	623241	2.11%	Y	620811	1.71%	Y	621707	1.86%	Y	615063	0.77%	Y
20	900	621154	634193	2.10%	Y	628742	1.22%	Y	629294	1.31%	Y	624564	0.55%	Y

[1] $Diff_{GP}$ is the difference of fitnesses of heuristics and the proposed GP.
[2] $Feasible$ displays Y if the solution is feasible; otherwise N.

Economics, 59(1):519–528, 1999.

[2] Jinxiang Gu, Marc Goetschalckx, and Leon F McGinnis. Research on warehouse operation: A comprehensive review. *European Journal of Operational Research*, 177(1):1–21, 2007.

[3] EA Frazele and Gunter P Sharp. Correlated assignment strategy can improve any order-picking operation. *Industrial Engineering*, 21(4):33–37, 1989.

[4] 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*, 17(9):683–691, 2001.

[5] Jian Xiao and Li Zheng. A correlated storage location assignment problem in a single-block-multi-aisles warehouse considering bom information. *International Journal of Production Research*, 48(5):1321–1338, 2010.

[6] David Ming-Huang Chiang, Chia-Ping Lin, and Mu-Chen Chen. The adaptive approach for storage assignment by mining data of warehouse management system for distribution centres. *Enterprise Information Systems*, 5(2):219–234, 2011.

[7] David Ming-Huang Chiang, Chia-Ping Lin, and Mu-Chen Chen. Data mining based storage assignment heuristics for travel distance reduction. *Expert Systems*, 2012.

[8] James L Heskett. Cube-per-order index-a key to warehouse stock location. *Transportation and distribution Management*, 3(4):27–31, 1963.

[9] Gajendra Kumar Adil et al. A branch and bound algorithm for class based storage location assignment. *European Journal of Operational Research*, 189(2):492–507, 2008.

[10] Gajendra Kumar Adil et al. Efficient formation of storage classes for warehouse storage location assignment: a simulated annealing approach. *Omega*, 36(4):609–618, 2008.

[11] Lu Chen, André Langevin, and Diane Riopel. A tabu search algorithm for the relocation problem in a warehousing system. *International Journal of Production Economics*, 129(1):147–156, 2011.

[12] Su Nguyen, Mengjie Zhang, Mark Johnston, and Kay Chen Tan. A coevolution genetic programming method to evolve scheduling policies for dynamic multi-objective job shop scheduling problems. In *Evolutionary Computation (CEC), 2012 IEEE Congress on*, pages 1–8. IEEE, 2012.

[13] Su Nguyen, Mengjie Zhang, M Johnston, and K Tan. A computational study of representations in genetic programming to evolve dispatching rules for the job shop scheduling problem.

[14] Joc Cing Tay and Nhu Binh Ho. Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems. *Computers & Industrial Engineering*, 54(3):453–473, 2008.

[15] KK Lai, Jue Xue, and Guoqing Zhang. Layout design for a paper reel warehouse: A two-stage heuristic approach. *International Journal of Production Economics*, 75(3):231–243, 2002.

[16] GQ Zhang, J Xue, and KK Lai. A class of genetic algorithms for multiple-level warehouse layout problems. *International Journal of Production Research*, 40(3):731–744, 2002.

[17] Peter van Emde-Boas. *Another NP-complete partition problem and the complexity of computing short vectors in a lattice*. Department, Univ., 1981.

[18] Inc. Gurobi Optimization. Gurobi optimizer reference manual, 2013.

[19] David R White. Software review: the ecj toolkit. *Genetic Programming and Evolvable Machines*, 13(1):65–67, 2012.