December 2002

A HEURISTIC FOR SHELF SPACE DECISION SUPPORT IN THE RETAIL INDUSTRY

Andrew Lim
National University of Singapore

Zhang Qian
National University of Singapore

Brian Rodrigues
Singapore Management University

A HEURISTIC FOR SHELF SPACE DECISION SUPPORT IN THE RETAIL INDUSTRY

Andrew Lim
School of Computing
National University of Singapore
alim@comp.nus.edu.sg

Zhang Qian
School of Computing
National University of Singapore
zhangqia@comp.nus.edu.sg

Brian Rodrigues
School of Business
Singapore Management University
br@smu.edu.sg

Abstract

In the competitive retail industry, one of the keys to gaining the competitive edge is an efficient shelf allocation system. Indeed, shelf space is often the retailers' scarcest resource. A good model for shelf allocation is an underlying key to a meaningful and robust decision support system in retail management. In this paper, we develop a Shelf Space Allocation Model (SSAP) with extensions that cater for a realistic variety of constraints and requirements as experienced in industry. To obtain near optimal solutions to this problem, we use a hybrid of heuristics in a 5-phase "Squeaky Wheel" Optimization with Local Search method. Our experiments achieve more than 99.59% performance level on average in quite reasonable timeframes.

Introduction

In the competitive retail industry, one of the keys to gaining the competitive edge is an efficient shelf allocation system. Indeed, shelf space is often the retailers' scarcest resource. With the number of brand lines continually increasing, allocating products on the supermarket shelf in the best possible arrangement poses a great challenge to the industry.

Proper shelf space planning provides the basis for making category-specific merchandising decisions. The traditional space management tool employed is the planogram, which provides a shelf layout of products. The purpose of plannogramming is to provide a workable method by which merchandising plans can be communicated efficiently. Retailers and merchandisers focus on developing effective visual merchandising plans to maximize profits on a store-by-store basis. With effective plannogramming, it is possible to leverage every inch of selling space available and to capitalize on available data to meet financial objectives.

With well-managed shelf space allocation, a retailer can improve the return on inventory investment as well as raise consumer satisfaction by reducing the likelihood of products being out-of-stock. More importantly, the retailer can increase sales and profit margins while reducing manpower costs.

Moreover, efficient space management can encourage impulse buying and boost incremental sales. With the improvement of shelf appearance and layout, less time is needed for shelf hygiene. Other benefits are derived, for example, product mix maximization; price and product line up; product rotation; and forward movement of dated products.

Manufacturers also enjoy benefits that include improved brand image, enhanced brand and category integrity and the provision of attractive and coherent product presentations.

For the customer, products are more easily visible and convenient to locate while prices and products are clearly lined up for quick and easy identification.

Retail shelf space management is in the core of retail operations management decision support systems (DSS). Requiring high volumes of data related to marketing and space configurations, this decision problem can be complex and ill structured. Any good DSS will require a good and robust shelf allocation model, especially in view of the dynamic nature of the problem, which is driven by frequent retail product line changes and limited implementation times for new product configurations. Such a system should, ideally, also have performance metrics to analyze sales volumes relative to space.

Within the retail industry, user interest in shelf space allocation is found to be very high. A search of ABI/INFORMS resulted in over 500 references and numerous recent articles in practitioner journals such as Advertising Age, Supermarket Business, Beverage World, and Electronic Business.

Currently, generating shelf layout is largely a manual process. Due to the problem's complexity, only relatively simple heuristic rules have been developed and applied in industry for retailers to plan their shelf space allocation (Zufryden 1986). The simplicity of current approaches, however, has an adverse effect on solution performance (Yang 2001). Moreover, we find that available software systems for SSAP require significant human interaction. From an academic perspective, interest in this area appears to be weak. We find that there is a lack of work done on this problem as is evidenced in Yang and Chen (1999), where we find 64% references are dated.

The basic objective of shelf space allocation is to improve the financial performance of the retail store (Buttle 1984). Drèze et al. confirm that product location on shelves has significant impact on sales whereas the number of facings had less impact (1994). Models of SSAP in the past thirty years have been studied and developed to meet certain objectives associated with this problem. For example, Anderson and Amato proposed a model that considers optimal brand selection (1974). Many of the models, however, ignore some real world constraints. For instance, some models ignore the integer nature of solutions by working with real numbers. Many are far too theoretical and complex to be applied in retail practice (Corstjens and Doyle 1981) (Corstjens and Doyle 1983) (Zufryden 1986). On the other hand, we find simplified versions of the problem. For example, in (Zufryden 1986), the display area of each was limited to multiples of the area unit values. In most other instances, cross effects between products are ignored and non-space variables of demand and sales are not taken into consideration. In the most recent work on the SSAP, Yang (Yang 2001) uses a model based on a knapsack problem and presents a heuristic to solve instances of shelf space allocation problems. However, this particular heuristic has limitations when the problem size increases to sizes encountered in industry.

As the short survey above indicates, there is a need to build comprehensive models for the SSAP within a DSS for retail management that can be of practical use. The ultimate objective is to devise effective software to replace and automate the manual process of allocating products for display.

Basic Problem Definition

The main objective of the SSAP is to maximize a certain profit function when allocating different products onto shelves. All other objectives, such as shelf space minimization, sales or sales turnover maximization, or the maximizing of the total number of all products allocated, can be treated in a similar way.

The basic problem is defined in as follows: Given m shelves and n products, we want to achieve maximum profit through allocating the products onto the shelves. The rationalizations and conditions of allocation are as follows:

1. **Capacity Constraint**. Each shelf has different capacity T_k, which limits possible allocations of products.
2. **Facing Size Constraint**. Different types of products vary in their facing size a_i.
2. **Lower Bound Constraint**. In practice, in order to attract basic levels of customers, stores are required to display all types of goods in some categories no matter whether the goods are profitable or not. In this case, there is a minimum quantity of allocation that must be satisfied for each product.
4. **Upper Bound Constraint**. Drèze (Drèze et al. 1994) reported the number of facings of a product had less of an impact when a certain threshold is maintained. An upper bound might also be set for each type of product at a later stage of its life cycle or to induce a mix of products allocated. Hence, retailers can deliberately use a maximum, or upper bound, for the number of same class of products that can be allocated into shelves including those products that are profitable.
5. **Allocation Constraint.** For each shelf, each type of product can be only allocated in 0 or 1,2,3…*k (kÎ N)* pieces
6. **Profit Assumption and Constraint**. The profit gained for product i displayed on shelf k is assumed to be linear with allocated amount of facings of product i on shelf k. Different products in different shelves enjoy distinct profits, which means

that the profit of a particular product changes when the allocated shelf changes. This also implies that identical shelves receive distinct profits for various products.

The problem can be formulated as follows. The input data are:

m: total number of shelves available
n: total number of products sold in a store
T_k: the initial length for shelf k
a_i: the length of a facing of product I displayed on any of these shelves, $i=1,..., n$
L_i: the lower bound on amount of facing of product displayed among all shelves
U_i: the upper bound on amount of facing of product displayed among all shelves
P_{ik}: the profit gained when one facing of product i displayed in shelf k

The decision variables are:

x_{ik}: the amount of facings of product i displayed on shelf k

Given the constraints and assumptions stated above, the objective of SSAP is to find a solution set $\{x_{ik} \mid i=1,...,n, k=1,...,m\}$, the formulation is

$$\text{Maximise } \mathbf{P} = \sum_{i=1}^{n} \sum_{k=1}^{m} p_{ik} x_{ik}$$

Subject to:

$$\sum_{i=1}^{n} a_i x_{ik} \le T_k \quad k=1,\ldots,m.$$

$$L_i \le \sum_{i=1}^{n} x_{ik} \le U_i \quad i=1,\ldots,n.$$

$$x_{ik} \in N \cup \{0\} \quad i=1,\ldots,n;\ k=1,\ldots,m.$$

The SSAP is a generalized "Multi-Constraints Knapsack Problem" and hence NP-hard. It is however, more difficult and complex.

Recent Work

In the most recent work on the SSAP, Yang (2001) has developed a heuristic algorithm that extends a method often applied to knapsack problems to solve SSAP. The profit of each item per displayed length on a particular shelf is treated as a weight, and the ranking order of weight is used as a priority index in the process of space allocation. The algorithm consists of three phases.

First, a ***preparatory phase*** checks the feasibility of a particular problem and builds the set of priority indexes.

Second, an ***allocation phase*** allocates available space to items one by one following the order of priority. This phase is further divided into two sub phases, which, respectively, assure the constraints of lower bound and of upper bound for the number of facings of a product is not be violated.

Thirdly, in a ***termination phase***, the objective value of the final solution is calculated.

An ***adjustment phase*** consisting of three methods can be adopted to improve performance. Adjustment 1 attempts to improve the solution by swapping one facing for a pair of products allocated on the same shelf. Adjustment 2 is aimed at interchanging one

facing for a pair of products allocated on two shelves. Adjustment 3 is an extension of Adjustment 2. After interchanging the number of facings between two products on two shelves, there may still be shelf space that can be reallocated to some other product.

Using a self-defined set of testing data, Yang claims a mean profit of solutions after the Allocation Phase as 98.2% of the optimal mean profit. After using the adjustment phase, the average profit ratio of the solutions obtained by his improved heuristic to optimal solutions was 99.6%.

The test data given by Yang, however, cannot be applied in most cases because of practical limitations. The limitation in the design of the test data is that the total available shelf space is generally more than sufficient to allocate all products in the upper bound. Moreover, the lower and upper bound as well as the capacity constraints have little or even no impact on the resulting feasible solution at all.

The scale of the test data used by Yang is also relatively too small to be applied to any realistic industry situation. Lower and upper bound variances of between 0-3 that are quoted can hardly have any impact on solutions obtained. Moreover, all shelves are assumed to have the same lengths T_k and all products have some lower L_i and upper bounds U_i. These assumptions seem to be unrealistic. Furthermore, there is only one group of test data given, which hence cannot represent performance in industrial applications.

Moreover, the Adjustment Phase given by Yang cannot help much in improvement if the length of the facing among products varies widely, which is often the case in practical applications.

Objectives and Model Extension

In order to make the problem more realistic, relevant and applicable to the retail industry, our objective is provide extensions to the SSAP model which can accommodate common industrial applications. We consequently develop a methodology that can solve the extended model.

In our model, we incorporate more than one objective since it is common to find this requirement in applications. For example, there are cases where the objective for a retailer is not just to maximize profit per se, but also to optimize shelf space.

Also, there exist many human and other factors that must be taken into account. For example, suppliers may influence retailers as to the amount of shelf space allocated to their brands and the position of the brands on the shelf. These, and others, are catered for in the following additional requirements and constraints, suggested in a recent study (University of Nottingham 2001), to which we append rationalisations.

1. **Product Line Constraint**. The store management defines the product line for display. This is handled by changing corresponding lower bound constraints.
2. **Physical Allocation Constraint**. A product line is allocated to shelves in such a way that it can be physically placed onto those shelves. For example, products cannot be taller than the shelf to which they are allocated.
3. **Packaging Constraint**. If a product is placed on the shelves in packaging, we allow for two cartons to be placed on the shelf. If only one carton were allowed, the product might sell out, causing a disruption to the sales until a new carton was allocated. If two cartons or more are catered for, whenever there is only a single carton left, an additional new carton can be supplied without interrupting sales.
4. **Packing Requirement.** For similar reasons, when products are completely removed from their packaging before being placed on the shelf then provision must be made to allow for 11/2 packages to be accommodated on the shelf. This allows for the shelf to be replenished from stock without having to return half empty packages to the store.
5. **Position Constraint.** The position of the product on the shelf is an important factor. It is thought by some in the industry that products at eye level sell better than products at, say, floor level. However, candy, for example, could be placed at a lower level without affecting its sales. Similarly, products at the start of an aisle tend to sell better than products in the center of an aisle. If customers are looking for a particular product, they will usually pick up the first one they encounter on their entry into the aisle.
6. **Supplier Requirement.** Suppliers may be able to influence the retailer with regards the amount of shelf space allocated to their brands and the position of the brands on the shelf.

7. **Multi-allocation Requirement.** In some cases, it may be better to place a particular brand in a block (using several shelves) rather than using a single shelf to display the brand.

Given the constraints and requirements stated above and the basic problem definition, the objective of SSAP is to find a solution set $\{x_{ik} \mid i=1,...,n;\ k=1,...,m\}$, which will optimize the objectives given.

For this extended version of SSAP, a hybridisation of meta-heuristics is used. In our approach, a mixture of an effective initial construction strategy and a meta-heuristic "Squeaky Wheel Optimization" with embedded local search is adopted. Experimental results show that the near optimal planogram can be produced in reasonable time.

Hybridization Methodology

Our hybridization methodology employs a greedy algorithm followed by "Squeaky Wheel" Optimization (SWO), and three new adjustment methods for the local search. Our approach has several dominant enhancements over the original version of SWO, which is a general approach to optimization, proposed in 1997 by Clements et al.

SWO Overview

In SWO, a construction algorithm first processes each element of a solution in an order that is determined by some priorities assigned to each element based on various criteria, thereby constructing a feasible solution. Then, the solution is examined to determine which elements of are positioned disadvantageously. These elements are deemed to "squeak" because they contribute negatively to the objective function of the solution.

These "trouble" elements are then advanced to the front in the ordered priority list so that the construction algorithm handles them earlier and better when the next solution is constructed. This process of constructing, analyzing and reordering is repeated, producing a variety of candidate solutions to the problem at hand. In favorable situations, near optimal or even optimal solutions are found with this procedure.

The core idea of SWO is to form a Construct-Analyze-Prioritize three-component cycle. The "Constructor" uses priorities assigned to construct a solution, employing a greedy algorithm. The "Analyzer" assigns a numerical value as a "blame" factor to each element that has contributed to the shortcomings in the solution constructed in previous step. The "Prioritizer" then modifies the priority list according the blame factor assigned for each element, by moving elements with greater blames to the front of the list. This ensures that the problematic elements can be taken care of earlier and hence better in next iteration, when the Constructor constructs a new solution based on this new priority list. These three steps are repeated until a termination condition is satisfied.

Enhanced 5-Components Design

This section describes the application of a hybrid approach that we have developed by combining the SWO technique with local search and heuristics for the SSAP. Since the greedy heuristic by itself may not generate good solutions, SWO fine-tunes the solutions by repeatedly strengthening the constructor. SWO reorders the priority list, so that the "trouble" elements can be handled earlier. This process can be viewed as jumping between two search spaces: the Solution and Prioritization spaces, where a small change in Prioritization Space will result a large transformation in the Solution Space.

Because of this feature, SWO is able to find good solutions very rapidly. However, there are also many deterministic limitations that need to be considered carefully during algorithm design. These limitations are mentioned for future work in the latest paper of Joslin & Clements (1999).

We have designed a new 5-phase SWO with Local Search to overcome possible limitations and to make this approach more applicable to SSAP. Two new main components are added to the core SWO cycle: a "Special Constructor" that is only used for generating the initial solution, and a "Local Search Adjuster" which serves to enhance constructed solutions. This refined 5-phase cycle is illustrated in Figure 1.

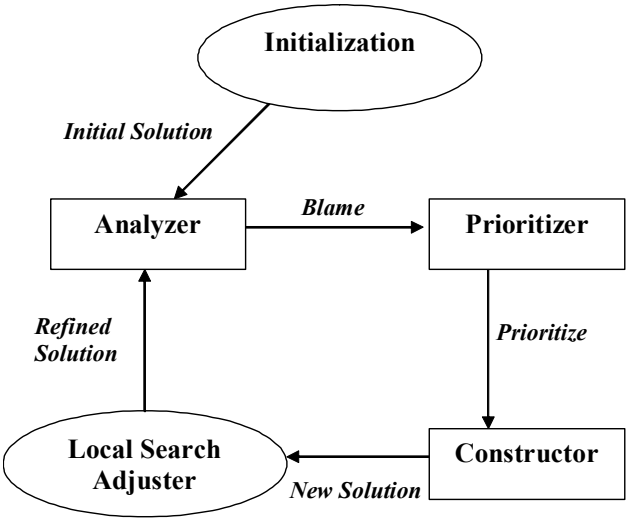

Figure 1. The Five Components Cycle

Local Search. SWO effectively avoids getting trapped in local optimal because of the jumping between the two search spaces. Although this characteristic of making large and coherent moves is strength of SWO, it is at a same time a limitation. This feature makes SWO poor at making small "tuning" moves in the solution space. Hence, we combine SWO with local search so as to be able to look for improvements in the vicinity of good solutions.

Another reason for employing local search is that for some problems, "sacrificial" elements are necessary. In such problems, these elements ideally should be handled badly in order to achieve a good overall solution. SWO alone is unable to achieve good solutions under such situations. In SWO, high blame is assigned to these "sacrificial" elements. As a result, in next iteration, the constructor will handle these elements well, leading the search further away from optimal solution. For such cases, the analysis is actually limiting SWO's ability to converge to good solutions.

By inserting a local search before analysis, the sacrificial elements are handled more effectively and will be not blamed highly by the analyzer. Furthermore, the adoption of adjustments during the local search will allow the constructor to avoid spending excessive time evaluating unnecessary alternatives when the previous configuration already retains favorable characteristics of a good solution.

Initialization. Although SWO finds solutions rapidly, it does not guarantee the feasibility of solutions generated. For SSAP with multiple constraints, the constructed solution tends to violate several of the hard constraints.

To improve the feasibility of solutions, it is important that the initial solution is feasible and can not be error prone. Therefore, instead of using just a simple greedy algorithm suggested by SWO for constructing initial solutions, we require a more complex heuristic.

Blame-Factor Design. The effectiveness of the blame factor is the key to the success of SWOL. We have found that using a satisfaction factor seems to be one of the most appropriate. A satisfaction factor can be described as a subjective value relating to how satisfactory a current allocation of a product and a shelf is, when compared with its most favorable allocation.

To control the scalability of construction so that feasible solutions are constructed, other dynamic values are also combined with blame factor.

We also attempt to control the effect miss-assigned blame has on the quality of solutions. Due to way the initial solutions are constructed, it is possible for the lower bound of solutions to be violated. Whenever this happens, we have an indication that in the current solution, the blame value assigned for some products are too low to fulfill products' lower bound requirement. To remedy this situation, we assign extra blame value for these elements so as to make their blame value high enough to force reallocation in the next iteration.

Algorithm Implementation

Initial Constructor. In this stage, we use Yang's initial heuristic to get the initial feasible solution. This allocation algorithm consists of two steps. The algorithm first checks if the problem is feasible to solve. The first round of allocation aims to fulfill lower bound constraint, allocating products according to unit profit, p_{ik} / a_i. The p_{ik} profit value here is calculated by a complex evaluation of three profit and bonus-value sets in order to handle the extensions to the SSAP (see Section 2). The second round uses a greedy algorithm to allocate each type of product until its upper bound reached.

Analyzer. We have experimented with various methods of assigning blame. The blame factors that results in achieving near optimal value performance comprises of a mixture of the satisfaction factor, allocation number and an infeasibility penalty.

Satisfaction factors are assigned in such a way so that a poorly allocated product in current iteration will be very likely blamed highly so as to a get better allocation in next iteration.

The allocation number is the current amount of facings for product i in shelf k. It is used to ensure that the solution constructed will not be far away from infeasibility since the starting point (initial solution) is feasible. Each following reallocation will not change the total number of allocation for each product among all shelves; it will only change the distribution among shelves. However, when the allocation is changed, it is possible that the resulting solution becomes infeasible. When this happens, an extra allocation number is added to these "exceptional" products with an extremely high blame value so as to ensure they can be reallocated with a lower bound quantity in next iteration reallocation.

Prioritizer. The prioritizer builds an object to record all information about above combined evaluation for each nonzero x_{ik}. Next, it assigns allocation numbers for each blame factor recorded in this object, together with information about current product or shelf and its ideal shelf or product. Then, it goes on to construct the priority list using the object by sorting the products in order of blame value from highest to lowest. For products with zero-value blame factors, random order will be assigned so that they will be allocated differently for subsequent iterations. A restart will be applied whenever some solutions are detected to form a cyclic loop within solution space.

Constructor. The constructor builds a sequence of allocation based on the priority list powered by the prioritizer. The constructor takes an object one at a time from the beginning to end of priority list. There are many different options involved in allocation.

Local Search. The local search consists of a greedy starting algorithm with three our proposed adjustment methods designed to improve the solutions. The first adjustment is the multi-shift method, which improves the initial solution by swapping multiple facings for a pair of products allocated on the same shelf. The second is the multi-exchange adjustment method. This method interchanges multiple facings for a pair of products allocated on two shelves. The third improvement multi-add & exchange in the search is an extension of the second. After interchanging a number of facings between two products on two shelves, there may be enough shelf space to be reallocated to some other product. The third adjustment will fit products into the shelf space freed up by the interchange.

These three adjustment methods are extensions of the corresponding Yang's three-adjustment methods (refer Section 1.2 "adjustment phase"). Our new adjustments are aimed to increase profit through a "many-to-many" products shift, exchange, and add and exchange between one or two shelves, while Yang's adjustments are limited to "one-to-one" product operations.

Experimental Results

Our 5-phase SWO with Local Search can achieve an average of 99.59% of the upper bound, with the maximum reached of 99.92%. The results are achieved using realistic test data within reasonable computational time limits.

Test Data Design

Simulated problems were generated to test the performance of our proposed adjustment methods. In the previous sections, we discussed the deterministic limitations on the parameter sets that we adopted (sample size being extremely small, scale of constraints being too small and some most important constraints ignored, etc). In order to have an accurate and complete

performance comparison between these two sets of adjustment methods, new parameter sets are simulated. There are a total of five sets of parameters involved for the problem.

(m,n): the order pair of numbers of shelves and products
T_k: the length of the shelf k
L_i: the lower bound for the amount of facings of product i
U_i-L_i: the difference between upper bound and lower bound for the amount of facings of product i
a_i: the length per facing of product i

In order to make our samples general enough to cover all reasonable situations, we use random generator that will generate a random value for each of five sets of parameter, within the set range by a normal distribution. Moreover, in order to see how the selection of parameter ranges can affect the solution performance, different ranges are tested for each parameter set. Table 1 shows the values and ranges of these parameters.

Table 1. Parameter Values Used in Examples

Set	Value	Range
(m,n)	*m= 5,10,...,100*	*n =2*m, 4*m, 6*m, 8*m, 10*m*
T_k	*Random (t/2,t)*	*T =(L/2,U)*a*n/m*
a_i	*Random (1,a)*	*a =(50, 500), a=50,100,150,200…300*
L_i	*Random (0,L)*	*L=(0,20), L=0,5,10…50*
Ui-Li	*Random (0,U)*	*U=(0,200), U=0,5,10,15,20…200*

In total, 173 (100+20+11+11+31) cases were designed, and for most of the test cases, at least 100 problems were solved in order to get the average performance. For small problem sizes, 200, or even 500 problems were tested. These random problems were solved on a Pentium III 450 PC with 128MB Memory.

Performance Testing

The evaluation of this hybridization algorithm is shown in Table 2. On average, 100 problems are solved to obtain an average for each *(m,n)* values, T_k is randomly selected within range (2000*n/m, 4000*n/m) while a_i is also generated randomly in range (1-150), L_i and *Ui* are randomly selected within range (1-40) and (L, L+80). We use P_H/P_O to represent the profit ratio of the heuristic to the optimal method (upper bound). The upper bound of each problem P_O was obtained using a max min cost network flow model

From Table 2, we can conclude the 5-phase SWO with Local Search obtains an almost optimal solution, especially compared with 91.02% performance of the knapsack heuristic used by Yang.

Table 2. Average P_H/P_O for Two Heuristics

(m, n)	Knapsack	Hybrid
(5,10)	85.93%	99.57%
(10,10)	86.59%	99.33%
(5,50)	89.97%	99.54%
(10,50)	91.26%	99.69%
(5,100)	93.04%	99.83%
(10,100)	94.42%	99.64%
(50,100)	95.95%	99.55%
Overall Average	**91.02%**	**99.59%**

Parameter Testing

Experiments were been done on different parameters sets (refer to Table 2). We show only performance of SWO with Local Search under different parameters sets in Figure 2.

Problem Size. The problem size had a high impact on the solution performance. Results show that the more complex and difficult a problem is, the better our approach performs when compared to the other heuristics.

Range of Lower and Upper Bounds. When the range of L_i increases, the performance of our approach increases. However, SWOL improves much faster than the old ones with the increase in the range of L_i. When $U_i - L_i$ increases within a certain limit, the performance improves.

Range of Shelf Capacity and Product Size. Experiments show that T_k greatly affects the performance. The optimal values of all algorithms increase with the increase of T_k. However, our algorithm performs very well when T_k is small, which is often the case in real life. When the range of product size increases, the quality of the performance of our approach is increased while Yang's heuristics suffers from poorer performance.

CPU Time

Last but not at least, the CPU time is reasonable for the methods used. Even for the large case with 50 shelves and 100 products, feasible and near optimal solutions can be obtained in reasonable time, see Table 3 for average CPU time consumed by our new proposed method.

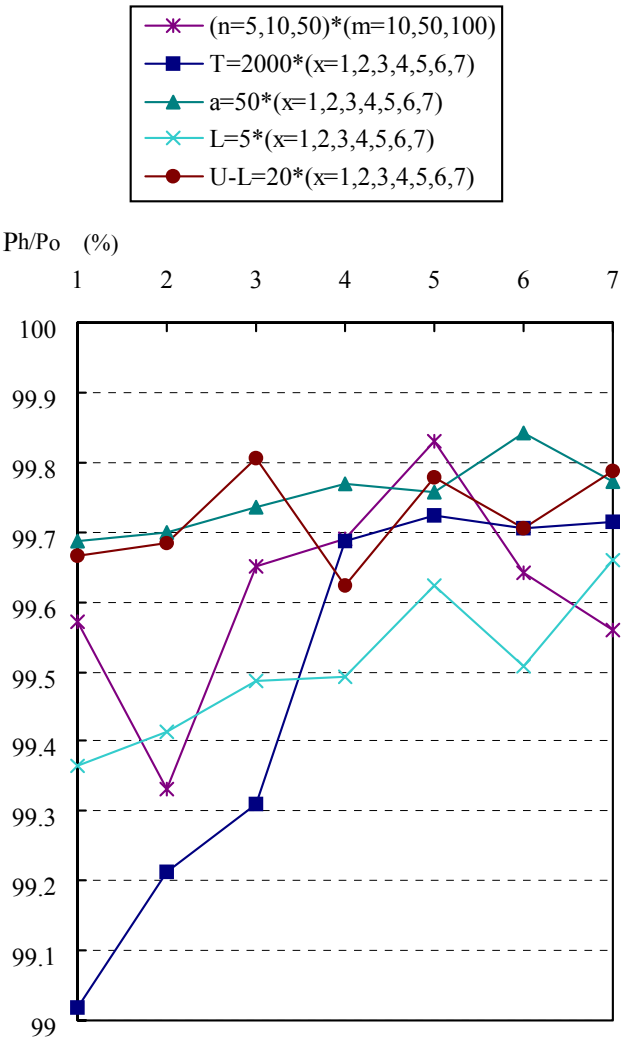

Figure 2. SWOL Performance under Different Parameter Set Values

Table 3. Average CPU Time for the Hybrid Heuristics

(m, n)	Hybrid (second)
(5,10)	1.550
(10,10)	4.138
(5,50)	47.136
(10,50)	83.316
(5,100)	160.662
(10,100)	437.457
(50,100)	1082.176

Conclusions and Future Work

We have surveyed existing work on the SSAP. These have been limited by their applicability or by their simplifications. Some models were highly complex. We have rationalized this problem providing a new SSAP with extensions that cater for more realistic situations in the retail industry. To obtain good quality solutions to this problem, we use a hybrid of heuristics in a 5-phase "Squeaky Wheel" Optimization with Local Search method. To make testing representative of possible problems, 173 simulated test cases (each 100 problems to get average) are designed with various combinations of different problem sizes and different scales for each of the five parameter-set. All the test data are generated using a random number generator. An upper bound value is computed to compare the relative optimality of these algorithms.

For comparison, we worked out Yang's improved heuristic that achieves a 90.4% on average performance, while a near optimal 99.59% performance has been achieved by our new approach. This new 5-phase SWO with Local Search is a hybridisation of an Initialisation Strategy, "Squeaky Wheel" Optimization and Local Search. The 5 phases adopted refer to Initialisation, Analysing, Prioritizing, Constructing, and Adjusting phases. The Initializer makes use of the initialisation strategy of knapsack model while the Adjustor is a local search technique employing three hill-climbing adjustments methods.

Although significant extensions have been proposed for the SSAP that has increased the complexity of the problem greatly, the extended problem can be handled by our 5-phase SWO with Local Search well.

The application of algorithms given here can be easily made amenable for use by retail managers. Most important factors that will affect their choices of shelf allocation, are captured by our extended models. Furthermore, our simulated data has covered a wide range of problem instances so that it is envisaged that retailers can achieve similar near-perfect performance results within a similar time frame.

Some of the areas that can be looked into for future research would include:

1. Validation of results with real data.
2. The integration of SSAP with other models within retail decision support systems.
3. Relaxing the linearity assumption of profit to shelf (Section 1.1 6) in the objective function, since this assumption does not apply to all situations.
4. Developing a larger even more comprehensive model to cater for more general retail shelf allocation requirements.
5. Improving on the heuristics given in SWO with local search.

References

Anderson, E. E., and Amota, H. N. "A Mathematical Model for Simultaneously Determining the Optimal Brand Collection and Display Area Allocation," Operations Research (3:6), 1979, pp. 58-63.

Buttle, F. "Retail Space Allocation, International Journal of Physics," Distribution and Material Management (14:4), 1984, pp. 3-23.

Clements, D., Crawford, J., Joslin, D., Nemhauser, G., Puttlitz, M., and Savelsbergh, M. "Heuristic Optimization: A hybrid AI OR approach," In Proceedings of the Workshop on Industrial Constraint Directed Scheduling. 1997.

Corstjens, M., and Doyle, P. "A Model for Optimising Retail Space Allocations," Management Science (27:7), 1981, pp. 822-833.

Corstjens, M., and Doyle, P. "A Dynamic Model for Strategically Allocating Retail Space," J. of the Operational Research Society (34:10), 1983, pp. 943-951.

Drèze, X., Hoch, S. J., and Purk, M. E. "Shelf Management and Space Elasticity," J. of Retailing (70:4), 1994, pp.301-326.

University of Nottingham: Investigation of Novel Methods to Optimise Shelf Space Allocation, http://www.cs.nott.ac.uk/~gxk/positions/advertinfo.htm

Joslin, D. E., and Clements, D. P. "Squeaky Wheel Optimization" in Journal of Artificial Intelligence Research (10), 1999, pp. 353–373.

Yang, M., and Chen, W. "A study on shelf space allocation and management," International J. Production Economics (60-61), 1999, pp. 309-317.

Yang, M. "An Efficient Algorithm to Allocate Shelf Space", European J.of Operational Research (131), 2001, pp. 107-118.

Zufryden, F. S. "A Dynamic Programming Approach for Product Selection and Supermarket Shelf-Space Allocation," J. of Operational Research Society (37:4), 1986, pp. 413-422.