V. A FRAMEWORK FOR THE ROBUST DESIGN OF UNIT LOAD STORAGE SYSTEMS

Marc Goetschalckx
Georgia Institute of Technology

Pratik Mital
Georgia Institute of Technology
Edward Huang
George Mason University

Abstract

The unit load storage assignment problem determines the assignment of a set of unit loads with known arrival and departure times to a set of unit storage locations in a warehouse. The material handling device(s) can carry at most one unit load at the time. In this research it is assumed that each of the storage locations can be accessed directly without load relocations or rearrangements and that the travel times between the storage locations and from and to the warehousing docks can be computed in advance. The objective is to minimize the total travel time of the material handling device for performing a number of storage and retrieval operations. This type of storage system is in widespread use and implemented in both mechanized and automated systems. It is by far one of the most common storage system architectures for unit loads.

The formulation of this problem belongs to the class of Assignment Problems (AP) but finding the optimal solution for the most general variant is provably hard for large problem instances. A classification of the different variants of the APs for unit loads will be presented. The size of the instance problem is proportional to the product of the number of loads and the number of locations and the number of periods in the planning horizon and is typically very large for real world problem instances. Efficient solutions algorithms only exist for product-based

storage policies or for the very special case of a perfectly balanced warehouse for load-based storage policies. However, for load-based storage policies the integrality property is not satisfied in general. This results in very large binary programming problems that to date cannot be solved to optimality. However, the formulations have special structure that can be exploited to design efficient solution algorithms. Properties and the special structure of the formulation will be presented. A specialized compound solution algorithm combines primal and dual approaches and heuristics to reduce the optimality gap. Initial computational experience will be shared. It is anticipated that the solution algorithm can either be directly implemented in commercial warehouse management systems or that it becomes a tool to evaluate the performance of commercially implemented storage policies.

The above formulation is the sub problem in a decomposition algorithm for the design of unit load storage systems that identifies the tradeoffs between efficiency and risk of the performance of the storage system. Different risk measures such as the standard deviation and the downside risk can be used. An example based on realistic data values shows that in this case operator-controlled systems are less expensive and more risky than automated systems. However, if the same level of risk is mandated then the automated system is less expensive.

1 Introduction

One of the most common types of warehousing systems is the unit load storage system. Many of the following definitions have been established over many years and have been summarized in Goetschalckx [1]. A *unit load* is a collection of materials so arranged that it can be stored, handled, and controlled as a single entity. Common examples of unit loads are pallet loads, drums, intermodal containers, totes, and baskets. In addition some materials are naturally aggregated into a unit load such as coils of sheet metal or feedstock rolls of paper. Pallet loads are the dominant type of unit load used in warehousing systems. The importance of the intermodal shipping container in international trade has been documented by among others Donovan and Bonney [2] and in the Economist [3].

It is assumed that all unit loads have dimensions so that they fit in a standard three-dimensional rectangular "box" enclosure. Note that this box is not a real container, but rather limits the dimensions of the unit loads. It is also assumed that all unit loads have a weight that falls in the safe operating range of the material handling device and that for all practical purposes unit loads can be treated as having identical weights.

Since the unit loads are standardized, the storage locations where they are being held in the warehousing system and the material handling devices that move the unit loads can also be standardized. The material handling transport moves and corresponding

transportation devices can be divided into two classes: *single-load* and *multi-load* transportation devices. Common examples of single load devices are the counterbalanced industrial forklift truck and the crane in an automated storage and retrieval system (ASRS). For unit-load moves, the cost and the necessary resources for handling and storing of a unit load are identical for all unit loads. A common example of multi-load material handling devices is a tugger pulling carts that hold one or more unit loads. This material handling device is conceptually identical to a railroad train where several locomotives pull a number of railroad cars that hold intermodal shipping containers. For multi-load moves, the cost of the move of the material handling devices has to be allocated to the unit loads that are being transported. Once the unit load(s) have been delivered to their destination, the status of the material handling device becomes "unloaded". Depending on the operating mode of the warehousing system and on the next task for the material handling device, the cost of the unloaded travel may have to be added to the cost of the next move of a unit load.

The unit load is a material handling and storage concept and the resources required to move or store the unit load are not dependent on which materials are stored or moved in the unit load. The individual types of material are often denoted as the *products*. In warehousing systems, most often the notion of *stock keeping unit* (SKU) is used. An SKU denotes a very specific type of material in a specific package that is tracked individually in the warehousing system. An example of and SKU may be a box of 8 men's blue cotton dress shirts of a neck size 7.5 inches, sleeve length of 30 inches, with cuff buttons. Different item features such as style, size and color and different package types will yield different SKUs. Each SKU has a unique identifier such as a unique number or string; the term *part number* is also often used. This unique identifier is often displayed at the outside of the package as a linear or two-dimensional bar code which allows rapid scanning and identification of the package and the material in the package. A very common example of bar coding for individual items in retail stores is the Universal Product Code (UPC). The same product may be present and tracked in the warehousing system as more than one SKU, e.g. a particular type of men's shirts may be tracked in full pallet loads in the reserve storage area and also as cartons in the pick area. In summary, the notion of product focuses on the characteristics of the material itself, while the notion of SKU combines the characteristics of the material with the characteristics of the material handling package.

A unit load is said to be *homogeneous* if it holds or contains a single stock keeping unit. The pallet loads arriving from a supplier at a distribution center often are homogeneous. A unit load is said to be *mixed* or heterogeneous if it holds or contains multiple SKUs. The pallet loads shipped from a distribution center to a single customer often are mixed pallets or heterogeneous pallet loads.

2 Design Methodology

1.1 Hierarchical Design

Three types of decisions can be distinguished in the design of unit load storage systems. At the highest level are decisions that determine the material handling and storage technology, the high level organization of the warehouse such as the presence of a forward picking area or not, the type of storage policy, the location of the input and output points of the storage systems, and the presence of cross aisles in the storage system or not. These decisions are typically investigated independently in a sequential manner under direction of the designer. For instance, in the example presented below an operator-controlled storage system was compared with an automated storage and retrieval system. The storage system was designed independently for each of these technologies and the final selection decision was made by the designer. These decisions will be referred to as the technology decisions.

The second type of decisions determines the configuration and layout of the storage system and the decisions at this level assume that higher level decisions yield binding constraints. Typical examples of these decisions are the number of aisles, the number of vertical levels and the number of columns in the storage system. These decisions will be referred to as the configuration decisions.

The third type of decisions determines tactical and operational characteristics of the storage systems. Again it is assumed that decisions at higher levels create binding constraints for the decisions on this level. Examples of these decisions are the number of material handling devices, the number of human operators, and the assignment of the unit loads to individual unit storage locations. The uncertainty of the future environmental conditions for the storage system is modeled using scenarios and the decisions at the third level are made independently for each individual scenario. These decisions will be referred to as the assignment decisions.

1.2 Mathematical Formulations

1.1.1 Configuration Model

The configuration or master model is responsible for determining the best tradeoff between the expected efficiency of the system, and the risk. The uncertainty of the future conditions is modeled using scenarios. In this section a generic risk measure will be used. The configuration model will determine the number of aisles, number of levels, and number of columns (stacks) in an aisle for a given technology alternative.

The following notation will be used in the master model.

S set of possible future scenarios, indexed by s

Decision Variables

NA	number of two-sided aisles
NC	number of columns of storage locations in a rack face
NL	number of levels (rows) of storage locations in a rack face
Y	set of possible discrete system configurations, indexed by *y*. Each discrete system configuration specifies at minimum *NA*, *NL*, and *NC*
WW	warehouse width (measured perpendicular to the travel aisles)
WD	warehouse depth (measured parallel to the travel aisles)
WH	warehouse height

Objectives

p_s	probability of scenario *s* (given parameter)
z_{sy}	total system cost of a system configuration *y* in future scenario *s* discounted to the current time
$E[z \mid y]$	expected value of a random variable z over all scenarios for a system configuration *y*
$VAR[z \mid y]$	variance of a random variable z over all scenarios for a system configuration *y*
$SD[z \mid y]$	standard deviation of a random variable z over all scenarios for a system configuration *y*
$RISK[z \mid y]$	generic risk measure of a random variable z over all scenarios for a system configuration *y*
$DR_n(z \mid y, TL)$	downside risk measure of order *n* of a random variable over all scenarios for a system configuration *y* and a target level *TL*

Parameters

MAXW	maximum warehouse width
MAXD	maximum warehouse depth
MAXH	maximum warehouse height
SLW	width of a storage location for a single unit load along the travel aisle including all clearances

SLH	height of a storage location for a single unit load along the travel aisle including all clearances
SLD	depth of a storage location for a single unit load perpendicular to the travel aisle including all clearances
TAW	width of a single travel aisle
I_t	inventory of unit loads that must be stored in the storage system during period t
κ	trade-off parameter between risk and cost measures

Excerpts of the configuration model for the robust system design are given next.

$$\min_{Y} \quad E[z \mid y] + \kappa \cdot RISK[z \mid y] \tag{1}$$

$$s.t. \quad E[z \mid y] = \sum\nolimits_{S} p_s \cdot z_{sy} \tag{2}$$

$$NA \cdot (2 \cdot SLD + TAW) \le WW \qquad WW \le MAXW \tag{3}$$

$$NC \cdot SLW \le WD \qquad WD \le MAXD \tag{4}$$

$$NL \cdot SLH \le WH \qquad WH \le MAXH \tag{5}$$

$$NA \cdot NL \cdot NC \ge \max_{t} \{I_t\} \tag{6}$$

$$y \ discrete, z_{sy} \ge 0 \tag{7}$$

Based on the values of technology-dependent parameters the costs that are independent of the various scenarios are determined in the sub problem by a simple calculation, since in the sub problem the configuration is considered a given parameter. Typical examples are the area-dependent construction cost for the storage system and the purchasing cost of the cranes in an automated storage system with aisle-captive cranes. Note that this calculation strategy avoids having a quadratic objective for area costs in the master problem. The configuration problem may have a quadratic objective in function of the scenario costs if the variance measure of risk is used.

The most important constraint in the configuration model is the storage capacity constraint, which ensures that the total number of installed storage locations is larger than the required number of locations. This is a cubic constraint.

1.1.2 Vector Assignment Model

The decisions at the third level include the assignment of individual unit loads to unit storage locations. This model to determine the assignment will be called the assignment of sub model. Each assignment has a corresponding assignment cost that depends on the configuration decisions made at the higher level. For example, the time and corresponding cost to reach a particular location in a rack depends on the dimensions of the rack and a number of technology parameters such as vehicle speeds. Secondary costs such as the cost of the required number of material handling trucks or the cost of the required number of personnel may also be included. The planning horizon is assumed to consist of a number of equal and sequentially numbered time periods. The arrival and departure time of all the unit loads that are stored during the planning horizon are assumed to be the integer indices of the respective time periods and assumed to be given for each individual scenario. Since individual unit loads are modeled in the formulation, the model can accommodate any storage pattern, including non-stationary patterns. The scenarios are assumed to be independent and the sub model will be solved sequentially for each scenario to determine the scenario cost.

The following notation will be used in the assignment model. Since the models for each scenario are solved separately, the scenario subscript has been omitted for notational simplicity.

c_{ij}	assignment cost of unit load i to unit storage location j
x_{ij}	assignment variable equal to one if unit load i is assigned to unit storage location j, zero otherwise
arr_i, dep_i	Arrival and departure period of individual unit load i in the storage system, the unit load occupies a storage location starting with the arrival period and up to and including the departure period
$\mathbf{b}_i$	occupancy vector for unit load i. The vector has as dimension the number of periods in the planning horizon. The vector element b_{it} is equal to one if unit load i is stored (present) in the storage system during period t, zero otherwise.
B	occupancy matrix, consisting of the occupancy vectors of all the unit loads
I	identity matrix

$$\min \quad \sum_{i=1}^{M}\sum_{j=1}^{N} c_{ij}x_{ij} \tag{8}$$

$$s.t. \quad \sum_{j=1}^{N} x_{ij} = 1 \qquad \forall i \tag{9}$$

$$\sum_{i=1}^{M} \mathbf{b}_i x_{ij} \le \mathbf{1} \qquad \forall j \tag{10}$$

$$x_{ij} \in \{0,1\} \tag{11}$$

The objective function computes the aggregate assignment cost. The first constraint assures that each unit load gets assigned to exactly one storage location. Since this constraint ensures that every unit load is stored or serviced, it will be denoted as the service or assignment constraint. The second constraint represents a vector of constraints that ensure that at most one unit load is stored in a location during any time period in the planning horizon. Again the dimension of the vector is the number of time periods in the planning horizon. This constraint will be denoted as the capacity constraint. The final constraint ensures that the decision variable can only have binary values. A graphical illustration of the problem is shown in the next figure.

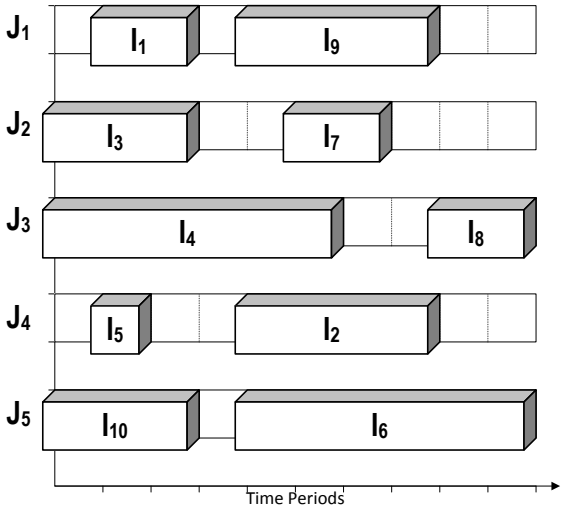

Figure 1: Gantt Chart of Unit Load Storage Assignments.

Since each unit load arrives in the system during a particular time period and remains stored in the system up to and including their departure period, the occupancy vector of a unit load has the consecutive ones property, introduced by Fulkerson and Gross (1965). In other words, the sequence of elements with value one consist of a single uninterrupted block of elements in the vector. An example of the structure of the

occupancy vectors is shown next for a case with five unit loads, indicated by the columns, and three time periods in the planning horizon, indicated by the rows.

$$\mathbf{B} = [\mathbf{b}_i] = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix} \tag{12}$$

Property 1

The assignment decision variables are not automatically integer in the optimal solution.

Hence the decision variables have to be explicitly forced to be binary assignment variables and the formulation has to be solved with an integer programming solver. This formulation belongs to the class of assignment formulations. It is neither a standard assignment formulation (AP), such as in the case of product dedicated storage policy, nor a three-dimensional assignment problem (3DAP), where the decision variable has three indices including one for the time period. This variant of the assignment problem is denoted as the vector assignment problem or VAP.

The VAP exhibits the block diagonal structure shown below. Each of the storage locations has the identical constraint structure ***B*** and the locations are connected by the service constraints, represented by the identity matrices ***I***. All right hand side parameters are equal to 1.

$$A = \begin{bmatrix} I & I & I & .. & I \\ B & 0 & 0 & .. & 0 \\ 0 & B & 0 & .. & 0 \\ .. & .. & .. & .. & .. \\ 0 & 0 & 0 & .. & B \end{bmatrix}$$

Consider a problem instance in which 1000 unit loads have to be assigned to a storage system that has 100 storage locations and the number of periods in the planning horizon is equal to 100. This instance has 100,000 binary variables, 1000 assignment constraints, and 10,000 capacity constraints for a single instance. This problem has to be solved repeatedly for each of the instances.

Because of the size of the problem instances corresponding to realistic storage system planning problems, simply submitting the instance to a mixed integer programming solver such as CPLEX yields unacceptable solution times.

1.3 Efficient Solution Algorithm for the Vector Assignment Problem

1.3.1 Algorithm Overview

A compound algorithm has been developed to solve the VAP efficiently. In the first phase the linear relaxation of the VAP is solved. If the solution contains fractional assignments, cuts are added iteratively until no further cuts can be identified. If the solution still contains fractional assignments, then the second phase commences.

In the second phase, the first step computes several heuristic primal feasible solutions. Loads are sorted by increasing departure time and decreasing duration of stay and assigned to their least expensive location. The second heuristic sorts the loads by increasing departure time and assigns them the location with smallest gap between the last departure in that location and the arrival time of the load. The third heuristic sorts the loads by increasing arrival time and then assigns them to the cheapest open location. This is the standard closest open location (COL) storage policy. A fourth heuristic sorts the assignments based on the ratio of the assignment cost over the duration of stay. Assignments of unassigned loads are made by this increasing ratio. The feasible solution with the lowest cost is retained and denoted as the incumbent.

All the heuristics made tradeoffs between reaching feasibility, i.e. assigning all loads to a storage location, and efficiency, i.e. assigning each individual load to a location that is inexpensive for this load. Their performance depends on the characteristics of the instance, such as instance size and "tightness" of the capacity constraints.

The expanded linear relaxation of phase one provides a lower bound and the best feasible solution created by the heuristics provides an upper bound on the solution value. The relative gap can now be computed. In the remainder of phase two, the problem is solved with Lagrangean relaxation. The algorithm terminates when the sub gradient solution algorithm for the Lagrangean relaxation can no longer improve the solution or when the relative gap becomes smaller than a specified threshold value. The sub problem will be solved for every scenario in every iteration of the master problem, so efficient and quick solutions of the sub problem are required. A threshold value of 3% to 5% provides a sufficient quality.

The most novel elements of the algorithm are the cuts for the VAP used in phase 1 and solution of the Lagrangean relaxation in phase 2. Some additional details are provided on those algorithm steps.

1.3.2 Constraint Elimination

The following property applies to the matrix $\boldsymbol{B}$ for a single location. In general, if for any row m another row n exists such that row n has an element equal to one in each column where row m has an element equal to one, then row m can be eliminated from the constraint matrix since it is redundant.

Property 2

For any row m of the matrix ***B*** it is sufficient to check rows m-1 and m+1 to see if they have an element equal to one wherever row m has an element equal to one. If neither one of these two rows satisfies that condition, then no such row exists and row m is non-dominated and cannot be eliminated. Hence, the effort for the row elimination procedure is linear in the number of time periods in the planning horizon.

Note that if a row is eliminated from the B matrix, it is eliminated for all the locations in the problem.

1.3.3 Set Partitioning Intersection Graph and Valid Inequalities

The VAP belongs to the class of Set Partitioning Problems (SPP), where the loads are partitioned into sub sets that can be feasibly assigned to a single location. The following facets of the standard SPP have been identified in a number of papers: the maximal complete subgraph or clique, the odd hole, and the odd anti-hole. The first three of these facet classes of the SPP have been investigated for the special case of the VAP.

The columns of the constraint matrix A are denoted by ak, with k between 1 and M·N. The intersection graph GA has one node for each column of A and an edge between the corresponding columns for every pair of non-orthogonal columns of A, i.e. it has an edge (m,n) if and only if am·an>0. The node-arc incidence matrix of GA is given by AG. A clique is a maximal complete subgraph. The clique matrix AC of GA is the incidence matrix of the set C of all cliques in GA (rows of AC) versus the nodes of G (columns of A). Three polyhedra can be defined and ordered by their size, where CH denotes the convex hull.

$$P_L = \{x : Ax \le 1, x \ge 0\} \tag{13}$$

$$P_C = \{x : A_C x \le 1, x \ge 0\} \tag{14}$$

$$P_I = CH\{x : Ax \le 1, x \in \{0,1\}\} \tag{15}$$

$$P_I \le P_C \le P_L \tag{16}$$

Two activities, i.e. storing two particular unit loads, are called "competing activities" if they cannot be assigned to the same resource, i.e. unit storage location, because they require the resource during at least one common time period. In other words, two competing activities i and j have non-orthogonal occupancy vectors, i.e. $b_i \cdot b_j > 0$.

Property 3

There exist only two types of edges in the intersection graph G. Either both nodes of the edge correspond to the same activity and the resources are different, or both nodes correspond to the same resource, but for competing activities. These edges are called an "activity edge" or "resource edge", respectively.

When traversing an edge from the node corresponding to variable x_{ij}, the end node will have either the same activity i, or the same resource j and an activity competing with i.

The following graphical notation will be used in the remainder of this paper. Each node can be labelled by a two tuple, indicating the activity and the resource, respectively. The activity will be denoted by a number, the resource by a letter. The two end nodes of an edge can have only one element of their tuples different. The VAP graph is illustrated in the next figure where activity edges are indicated by dashed lines and resource edges are indicated by solid lines.

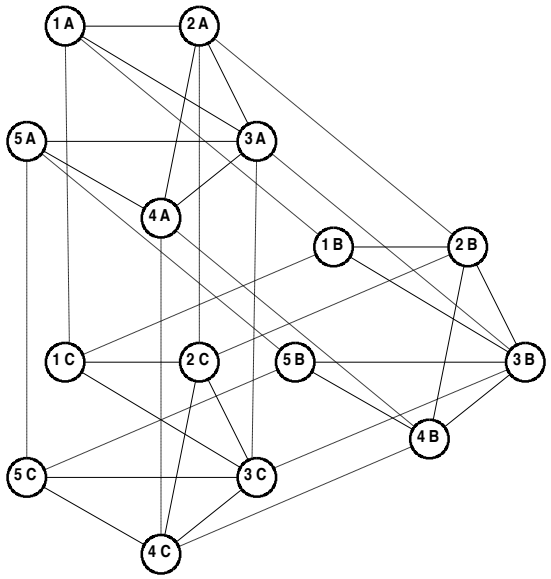

Figure 2: Intersection Graph of the Example.

Property 4

Each row of the constraint matrix A of the VAP corresponds to a clique of G_A and the clique matrix A_C is identical to the matrix A.

Hence A is the smallest matrix which uniquely defines the VAP problem and each constraint of A is a facet of P_I. The above property assures that the linear relaxation P_L is as tight as possible to P_I without adding constraints and that no other clique facets

exist. But from Property 1, it is clear that P_I is a proper subset of P_L, i.e. $P_I \subset P_L$, and other facets must exist.

Property 5

VAP has valid inequalities based on odd holes only for odd holes of cardinality 7. The VAP does not have valid inequalities based on odd anti holes. The only even anti hole subgraph the VAP has is of cardinality six.

The generic form of the valid inequality corresponding to a one hole of cardinality 7 is given next.

$$x_{ai} + x_{bi} + x_{bj} + x_{cj} + x_{ck} + x_{dk} + x_{di} \leq 3 \tag{17}$$

One example of an odd hole of cardinality seven is shown in the next figure. It was identified during the solution of the linear relaxation of the VAP_1000_100_100 instance, which had 1000 unit loads, 100 locations, and 100 periods. The first line in each node is the tuple identifying the node. The second element in the first line corresponds to the location. The second line in each node gives the arrival and departure time of the corresponding unit load. The last line in each node gives the optimal continuous value of the assignment variable for that node. Note that loads 72 and 510 do not have any overlapping residency periods.

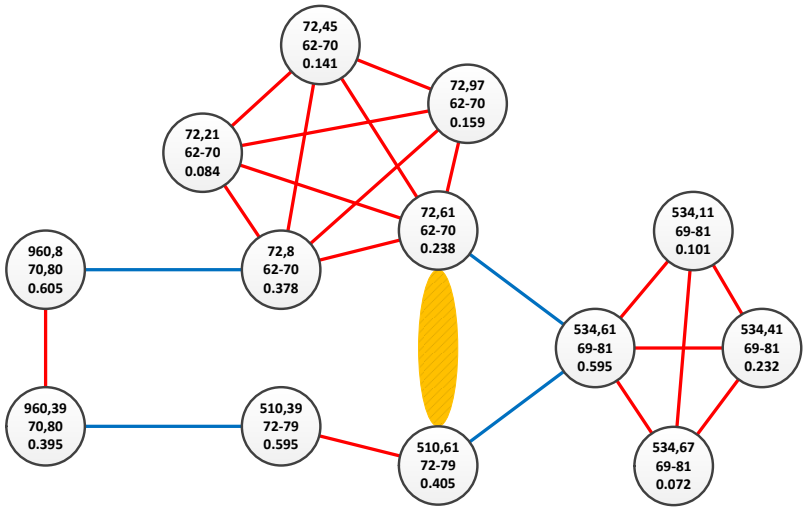

Figure 2: Example of a Seven Hole in the Intersection Graph of VAP_1000_100_100.

The corresponding cut is given by.

$$x_{534,61} + x_{72,61} + x_{72,8} + x_{960,8} + x_{960,39} + x_{510,39} + x_{510,61} \le 3$$

1.3.3 Lagrangean Relaxation with Dual Adjustments Procedures

The relaxation of the VAP model in which the assignment constraints have been relaxed will be denoted as *LAR(x)*. It is given next.

$$\min \quad \sum_{i=1}^{M}\sum_{j=1}^{N} c_{ij}x_{ij} + \sum_{i=1}^{M} \mu_i \left(1 - \sum_{j=1}^{N} x_{ij}\right) \tag{18}$$

$$s.t. \quad \sum_{i=1}^{M} \mathbf{b}_i x_{ij} \le \mathbf{1} \qquad \forall j \tag{19}$$

$$x_{ij} \in \{0,1\} \tag{20}$$

The objective function can be rearranged to the following. This problem is separable by location, i.e. each location can be solved for independently. In addition, an individual unit load will be stored in a location or not all. The optimal assignments will be automatically integer.

$$\min \quad \sum_{i=1}^{M}\sum_{j=1}^{N} \left(c_{ij} - \mu_i\right)x_{ij} + \sum_{i=1}^{M} \mu_i \tag{21}$$

For a given value of μ_i, solving this problem is equivalent to finding the shortest path in the following acyclic graph from an artificial start node (s) to an artificial terminus node (t). Each unit load corresponds to a node in the graph and the nodes are sorted and indexed by increasing departure time. There is an arc from s to each unit load node and from each unit load node to t. In addition, if the two unit loads do not overlap in their residence in the system, then there is an arc from the node with the lowest index to the node with largest index. The acyclic graph corresponding to the VAP example is shown in the next figure. Note that there is arc from node 1 to nodes 4 and 5 since the residency of load 1 does not overlap with the residency of loads 4 and 5. The solution can be found with an adaptation of Dijkstra's algorithm for the shortest path problem. If the final length of the path to node t is negative, then the node (unit loads) on the shortest path will be stored in this location.

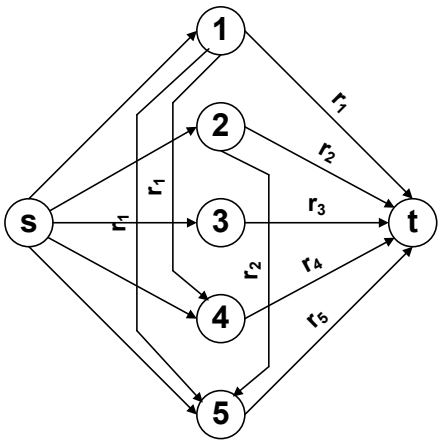

Figure 1: Acyclic Graph for a Single Unit Location for the VAP_5_3_3 Example

A node is a candidate to be included in the shortest path if its reduced cost $r_{ij} = c_{ij} - \mu_i$ is negative. Nodes on the shortest path are stored in this location if the length of the path from s to t is negative. If the Lagrangean variable for a load has a small positive or negative value, it is unlikely that the node will be included on the shortest path and equivalently be stored in this location. If the variable has a large positive value then it is advantageous that the load is stored in this location.

Again, note that the solution of the longest path problem with Dijkstra's algorithm generates naturally binary solutions for a single location for any value of the dual variables. For the optimal values of the dual variables the lower bound provided by the Lagrangean relaxation is stronger (larger) than the bound value provided by the linear relaxation.

After the shortest path problem has been solved for every location, the total number of times a load is stored in the locations can be computed. This count is equal to one for a feasible solution. If the count exceeds one, the Lagrangean variable μ_i should be decreased; if the count is less than one, the Lagragean variable should be increased. Two algorithms for this dual adjustment will be shown next. The first algorithm adjusts the dual variable in a finite number of discrete steps. If at its conclusion the solution is not feasible, i.e. each load is not assigned exactly once, then the subgradient dual adjustment procedure is executed. Since the formulation is a Lagrangean relaxation, the largest objective function value provides a lower bound to the original problem. Since the relaxed assignment constraints are equality constraints, the Lagrangean variables μ_i are unrestricted in sign. The Lagrangean problem will be denoted as *LAR(μ)*

$$\max LAR(\mu) = \max_{\mu} \left\{ \begin{array}{ll} \min & \sum_{i=1}^{M} \sum_{j=1}^{N} (c_{ij} - \mu_i) x_{ij} + \sum_{i=1}^{M} \mu_i \\ s.t. & \sum_{i=1}^{M} \mathbf{b}_i x_{ij} \le \mathbf{1} \qquad \forall j \\ & x_{ij} \in \{0,1\} \end{array} \right\} \tag{22}$$

$$LAR(\mu^*) \le VAP(x^*) \tag{23}$$

Again, note that the solution of the longest path problem with Dijkstra's algorithm generates naturally binary solutions for a single location for any value of the dual variables. For the optimal values of the dual variables the lower bound provided by the Lagrangean relaxation is stronger (larger) than the bound value provided by the original linear relaxation.

1.2 Numerical Experiment

1.1.1 Solution of the VAP Sub Problem

The investigation of the compound solution algorithm for the VAP problem is currently being executed. Some of the largest problems solved so far have dimensions as indicated in the next table.

All numerical experiments were executed on computer with Intel Pentium I7 920 2.67 MH processor with 4 cores and 12 GB of RAM. CPLEX 12.6 was used as to solve mixed integer sub problems generated. Solution times linearly depend on the number of scenarios. The parallel structure of the processor was not exploited but CPLEX uses all the cores available. The times reported are for a single scenario. Even for the largest instance, all heuristics require less than one second. The number of constraints reported is the number before the row elimination procedure has been run. For the first two instances, the linear programming relaxation generated integer solutions.

Table 1: Instance Characteristics and Solution Times for the VAP Sub Problem

Instance Name	# Loads	# Locations	# Periods	# Vars	# Constr.	Time (s)
VAP_100_20_100	100	20	100	2,000	2,100	0.24
VAP_200_25_200	200	25	200	5,000	5,200	1.07
VAP_1000_100_100	1000	100	100	100,000	10,100	124.23

1.1.1 Solution of the Robust Master Problem

The same problem instance as described in Mital et al. (2013) was used to investigate the performance of the master problem that makes tradeoffs between efficiency and risk. Two measures of risk were investigated: standard deviation and downside risk where the target level is set equal to the expected value. The number of scenarios is equal to 3.

The master problem, which had a cubic constraint, is solved to global optimality by complete enumeration over two of the decision variables in the cubic constraint, since these master configuration variables can only assume a very limited set of integer values. The whole problem is solved with 20 minutes of CPU time per technology. Two technologies were compared: an automated storage and retrieval system (ASRS) indicated by "auto" in the graphs and a system with operator-controlled industrial vehicles, indicated by "manual" in the graphs. Because all possible configurations are examined by the complete enumeration in the master algorithm and risk measures are computed for the population of scenarios, all Pareto-optimal configurations are identified for this instance. In the end two Pareto-optimal configurations were obtained from a total of 450 possible configurations. This means that 99.6% of the feasible configurations could be eliminated from further evaluation.

The two graphs show the Pareto-optimal configuration for the two technologies and for the two risk measures. The indices in the technology title indicate in sequence the number of aisles, number of levels, and the number of columns in one aisle, respectively.

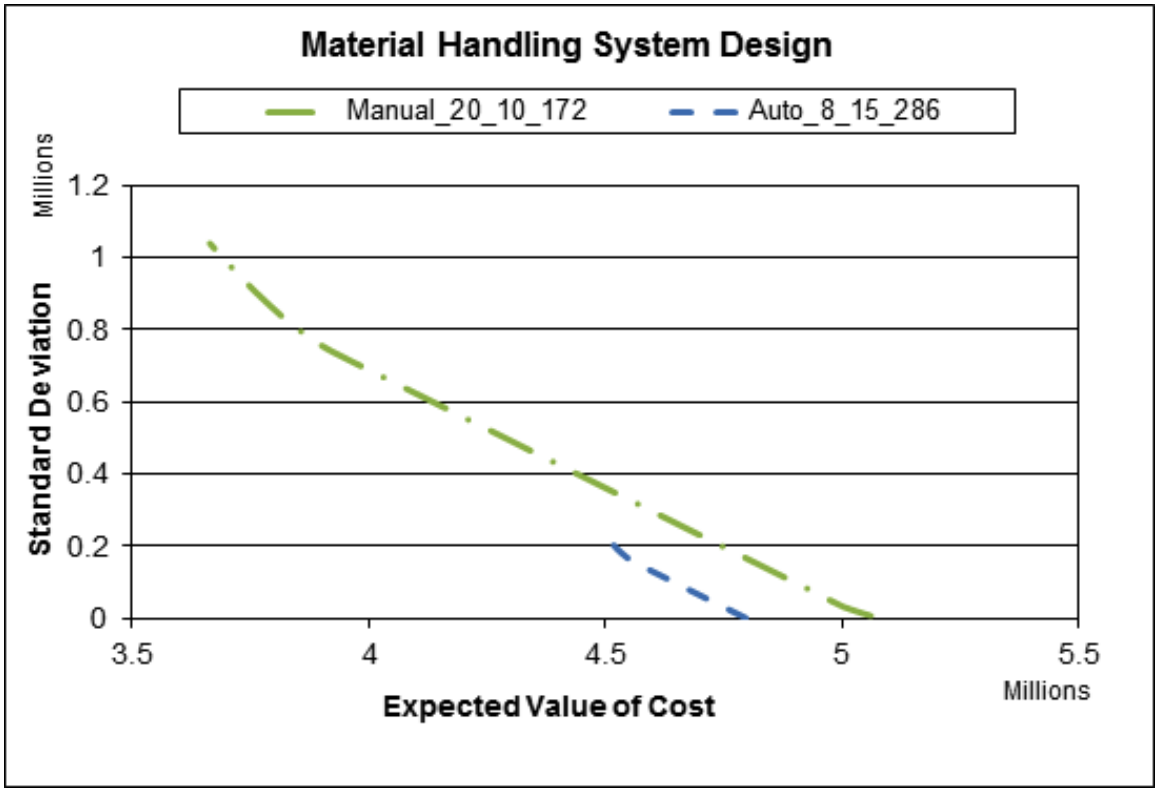

Figure 2: Efficiency versus Standard Deviation Risk Measure.

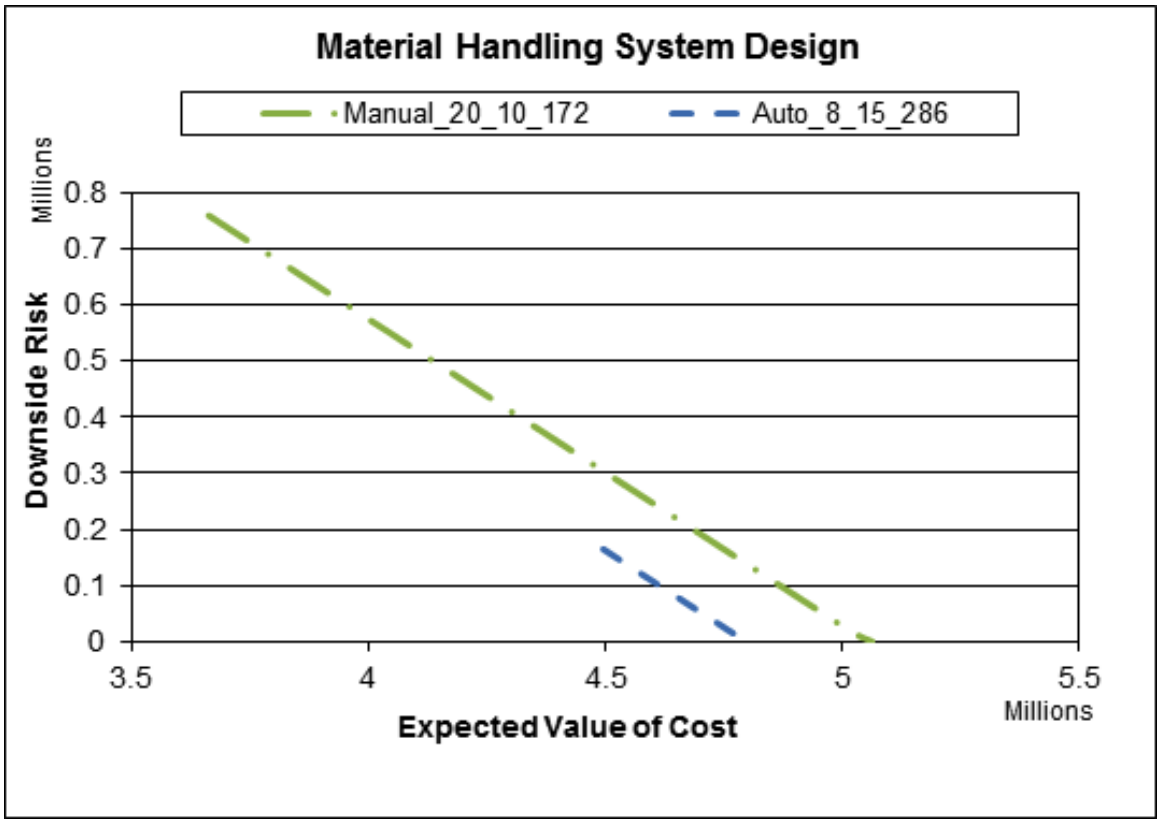

Figure 2: Efficiency versus Downside Risk Measure.

Several observations can be made. First, the operator-controlled system is less expensive than the automated system, i.e. it has a lower expected value of the cost. The automated system has a lower variability of the costs and thus a lower risk level. Furthermore, for equivalent risk levels that are feasible for both technologies, the automated system has a lower expected cost. A second observation is that both risk measures select identical Pareto-optimal configurations. The risk curves for both risk measures looked very similar. For this instance, it does not make any difference which of the two risk measures the decision maker selected. In Mital et al. (2013) different target levels for the downside risk were investigated and different Pareto-optimal configurations were found. Further details can be found in their paper.

2 Conclusions

A comprehensive framework was developed to design unit storage systems. The framework can accommodate any arrival and departure pattern of the unit loads,

including non-stationary patterns. Uncertainty is modeled through the means of scenarios. The framework can accommodate any scenario generation system. The framework finds all Pareto-optimal configurations of the storage system comparing efficiency (expected value) with various risk measures of the scenario costs. The risk measures studied are standard deviation and downside risk. The framework uses a hierarchical approach where at the master level the system configuration variables are determined, such as the number of aisles, number of levels and number of columns in the each aisle. The sub problem then computes the optimal performance for each given configuration and for all scenarios. In essence, the design framework allows for generality and flexibility at the cost of the size of the sub problem.

The sub problem is a very large binary linear programming problem that cannot be solved without algorithm specialization by contemporary mixed integer programming solvers. A compound algorithm has been developed. It includes several primal heuristics. It also solves iteratively the linear programming relaxation strengthened with problem-specific cuts. Finally, it also solves the Lagrangean relaxation created by relaxing the assignment constraints. The Lagrangean master problem is solved by a customized algorithm that adjusts the dual variable in discrete steps followed by the standard sub gradient algorithm which further adjusts the dual variables in a continuous manner. The compound algorithm terminates when the gap between the primal feasible solution and the best bound is smaller than a cut-off value. This value was set to 5%.

The technology specification and final technology selection is still the responsibility of the design engineer and the tradeoffs are clearly illustrated by the risk versus efficiency graphs that show all Pareto optimal configurations.

Currently a full numerical experiment is being conducted. The solution times for the instances investigated so far range from 20 to 900 minutes for each technology on standard contemporary computers. These times are acceptable for this strategic design.

Three avenues for further research exist. The first area focuses on the efficient solution of the VAP sub problem, including the identification of more valid inequalities and acceleration of the dual adjustment procedures in the Lagrangean relaxation. The second area focuses on efficient solution procedures for the master problem that makes the efficiency versus risk tradeoffs. Identification of all Pareto optimal configurations for other risk measures is also a research area. Finally, the integration of this design framework for the storage function of the warehouse with the systematic design procedures for the whole warehouse system by McGinnis [9] will create computational support for modeling based warehouse design.

References

[1] Goetschalckx, M., (2012), "Unit Load Storage Systems", Chapter 2 in *Warehousing in the Global Supply Chain*, Eds. R. Manzini, Springer London.

[2] Donovan, A. and J. Bonney, (2006), *The Box that Changed the World: Fifty Years of Container Shipping – An Illustrated History*, Commonwealth Business Media Inc.

[3] ---, "The Humble Hero," Economist, 18-May-2013, www.economist.com, last accessed on 01-Nov-2013.

[4] Lerher, T., M. Srami, (2012), "Designing Unit Load Automated Storage and Retrieval Systems," in *Warehousing in the Global Supply Chain*, pp. 211-232, Manzini (Ed.), 2012, Springer Verlag, London.

[5] Lerher, T., M. Borovinšek, I. Potrč, and M. Šraml, (2013), "A Multi-Objective Optimization Approach For Designing Automated Warehouses," *Proceedings of the 12th International Material Handling Research Colloquium (IMHRC) 2012*, pp. 280-302.

[6] Goetschalckx, M., E. Huang, and P. Mital, (2013). "Robust Material Handling System Design Based on the Risk versus Cost Tradeoff," P*roceedings of the 12th International Material Handling Research Colloquium (IMHRC) 2012*, pp. 186-200.

[7] Goetschalckx, M., E. Huang, and P. Mital, (2013), "Trading off Supply Chain Risk and Efficiency through Supply Chain Design" Proceedings of the Conference on Systems Engineering Research CSER 2013, Atlanta 20-22 March 2013. *Procedia Computer Science*, Volume 16, Pages 658-667 (2013) Elsevier, C. Pareidis and D. Bodner Eds.

[8] Mital, P, M. Goetschalckx, and E. Huang, (2013), "Robust Material Handling System Design with Standard Deviation, Variance and Downside Risk as Risk Measures," Submitted to *International Journal of Production Economics*.

[9] McGinnis, L. F., (2012), "An Object Oriented And Axiomatic Theory Of Warehouse Design," in Progress of Material Handling Research: Volume XII Eds B. Montreuil, A. Carrano, K. Gue, R. de Koster, M. Ogle, and J. Smith, pp. 328-347.