Applying Constraint Programming and Neural Net Techniques to model warehouse capacity check

Tian Sion Hui
Knowledge Engineering Pte Ltd, Singapore
email:shtian2002@yahoo.com

Abstract — **This paper describes an application which employs constraint programming and neural net techniques to perform the function of a warehouse storage capacity check. By formulating storage capacity, existing and incoming package profiles as a constraint satisfaction problem, the system is able to determine if a warehouse can accommodate a particular incoming product type and quantity configuration, subject to specific physical and administrative storage constraints. Using constraint programming to work out the position placement for each incoming package, the results generated by this technique is then used to train a neural net to produce a warehouse capacity check model which is fast and accurate.**

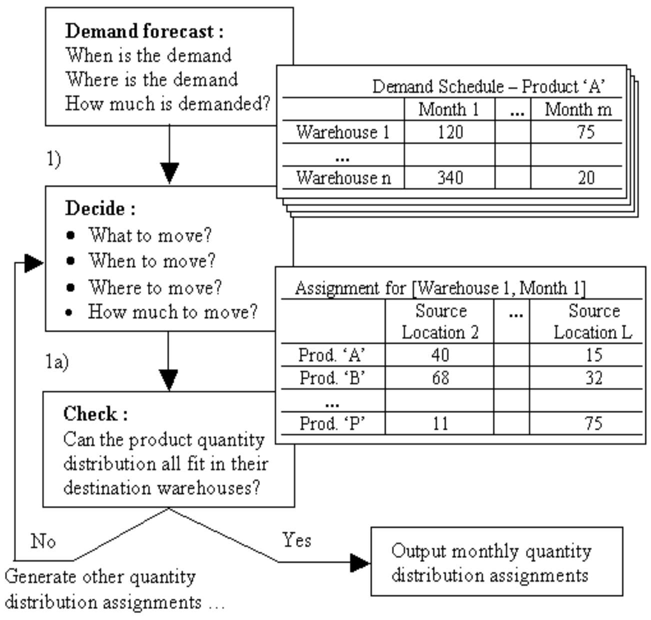
I. Introduction

One of the common logistic problem facing any large product manufacturing company is to ensure an efficient distribution and storage of manufactured goods to various warehouses around the world. In most cases, monthly fluctuations in demand for various types of goods may cause existing warehouse stocks to be depleted at varying rates. This situation raises a need to replenish depleted stocks, however, it is a constant challenge to ensure that:

1) Only the right quantity and types of products are shipped to the right places at the right time
2) Those quantities will all be able to fit perfectly into their respective destination warehouses, according to the local storage constraints, without any spillovers

Fig. 1 gives a broad overview of the problems faced. Given a schedule of demand forecast, the company has to plan their stock movements such that effort and cost of movements are minimized, together with the objective of meeting the targeted demands at the specified time and place. In the course of searching for the best combination of movements, a check has to be constantly made, to ensure that the quantities moved would be able to fit in the various destination locations, satisfying the various local storage constraints like floor layout, existing occupancy, etc.

As shown in Fig. 1, part 1) could be represented as a network flow model problem. The underlying theory and algorithms for network flows is elaborated by Ahuja, Magnanti and Orlin [2]. Bockmayr, Pisaruk, Aggoun [1]


Demand forecast :
When is the demand
Where is the demand
How much is demanded?
1)
Decide :
What to move?
When to move?
Where to move?
How much to move?
1a)
Check :
Can the product quantity distribution all fit in their destination warehouses?
No
Generate other quantity distribution assignments …
Yes
Output monthly quantity distribution assignments

Demand Schedule – Product ‘A’

	Month 1	...	Month m
Warehouse 1	120		75
...			
Warehouse n	340		20

Assignment for [Warehouse 1, Month 1]

	Source Location 2	...	Source Location L
Prod. ‘A’	40		15
Prod. ‘B’	68		32
...			
Prod. ‘P’	11		75

Fig. 1 – The process of determining monthly product type & quantity distribution to different warehouses.

provides some guidelines on formulating such problems using constraint programming techniques. In our application, the network nodes represent time-location combinations of demand and supply. The arcs represent the product quantities which flows between the nodes. This problem is then solved using the linear/constraint programming approach. In part 1a) of the problem, for every quantity distribution of a [product type, source location] being assigned to a particular destination location within a specific time period, a check is made to determine if the capacity of the destination warehouse is able to accommodate the quantities assigned for all product types from various sources. There are two ways to provide this check.

The first way is to rely on the experience of the warehouse crew, to provide a percentage figure that estimates the usable fraction, out of the remaining volume of the warehouse, which can be used to hold the incoming package quantities without violating local storage constraints. For example, in a particular warehouse, its remaining usable volume (less those occupied by existing packages) could be 1000 m^3. However, experience estimates that maybe only 700m^3 or 70% of this volume can actually be used to store incoming

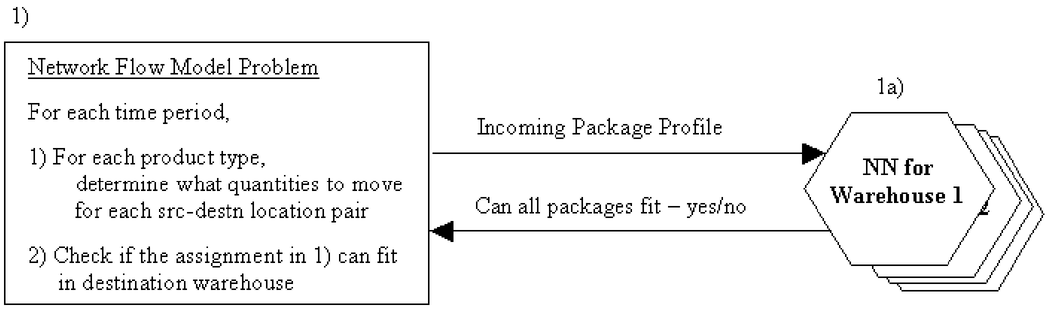

1)
Network Flow Model Problem
For each time period,
1) For each product type, determine what quantities to move for each src-destn location pair
2) Check if the assignment in 1) can fit in destination warehouse
Incoming Package Profile
Can all packages fit – yes/no
1a)
NN for Warehouse 1
2

Fig. 2 - Neural Nets trained to perform quantity distribution capacity check, for each warehouse

packages, taking into account warehouse-specific layout like positions of access-ways, walls and pillars, as well as any local storage constraints. Although this approach provides a quick check on whether a warehouse can store a particular quantity configuration of incoming products, the accuracy of this estimation depends heavily on the figures provided by the warehouse master. Due to the vast combinations of package quantities from different products that could come in different sizes, there is a tendency for the warehouse master to give a conservatively low percentage figure, to minimize the chances of encountering situations where the warehouse is unable to accommodate the incoming packages in any period of time. Moreover, the figures given for the same warehouse varies from person to person, and is therefore highly subjective, depending on the experiences and storage situations encountered by different warehouse planners. This results in under-utilization of storage space for all periods of time, which will in turn lead to a globally sub-optimal solution.

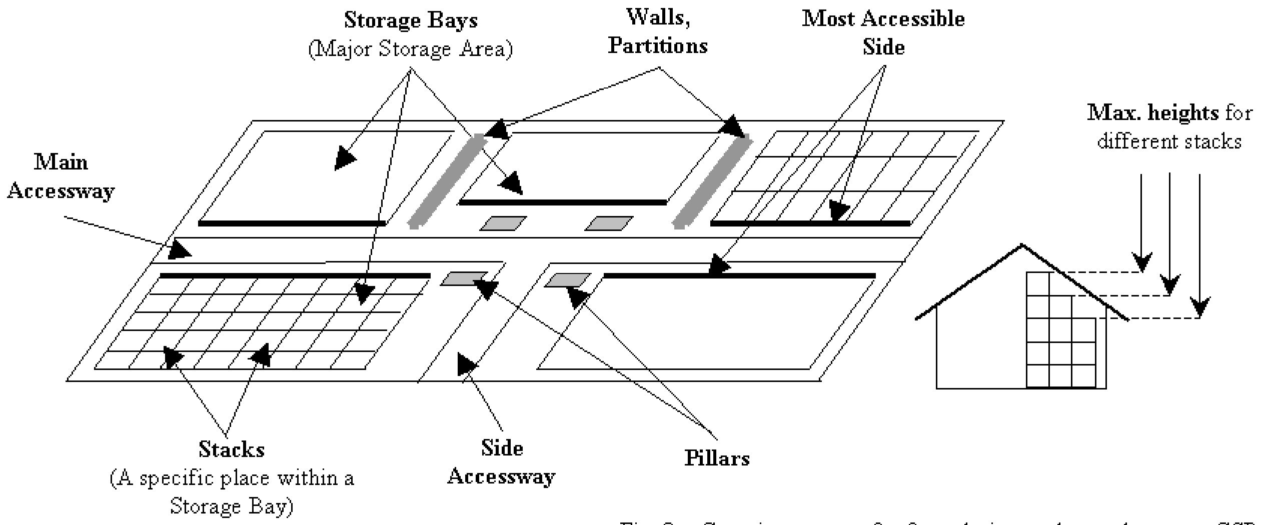
The second way to provide the capacity check is to formulate the incoming package placement as a constraint satisfaction problem (CSP). All storage rules are modeled as constraints to help guide the search. Given the remaining storable volume of the warehouse, this approach performs a search through all possible combinations of package placements. If a solution exists, this approach will find it and in the process provides a confirmation on whether all the incoming packages are able to fit into the warehouse. This is done by working out the placements for each package. The greatest disadvantage to this approach is the amount of time it takes to provide such a check. Because this is another optimization problem with its own goals, extra time is incurred in searching the entire solution space to determine the best package placements. Moreover, as seen in Fig. 1 under the ‘Assignment for [Location 1, Month 1]’ , there are potentially millions of quantity figures for different combinations of time periods, source and destination locations, for **each** of this assignment. Even if the package placement CSP can scan the entire solution space and provide an answer within an optimistic timing of 30 secs, the entire problem would still take many days to finish its processing!

Summarizing from the two approaches described above, the first way to compute the check is fast but highly inaccurate. The second way can determine the correct answer, but requires a lot of processing power, memory and time. This paper explores a third way to perform the warehouse capacity check using both Constraint Programming and Neural Net techniques. In this approach, the output generated from the CSP is used as training data, to train a series of neural nets to learn the capacity check function for each warehouse. Preliminary tests conducted indicates that for a trained neural net representing a particular warehouse, the output provides an answer that is more accurate than the first approach, but takes only a small fraction of a time incurred by the second approach. Fig. 2 above provides a high-level overview of the software design, using the suggested method.

Section 2 of this paper describes the nature of the Package Placement problem, and covers some of the storage constraints to be considered in the CSP. Section 3 describes neural net input data pre-processing. It also describes the choice of data attributes to be pre-processed to form the training input-output data for the neural net. Section 4 briefly covers the neural net architecture and its performance in terms of estimation accuracy and timing, when compared to other approaches. Section 5 gives the conclusion.

II. Warehouse planning problem domain

Assuming that the product demand schedule and the product distribution assignments are carried out as projected, there are two main sub-problems to be solved. Firstly, given that the various products demanded at different warehouses have to be satisfied, there is a need to choose packages from specific locations to meet the quantity demanded (for their respective products). Secondly, given a schedule of when new stocks are arriving for each month, there is a need to assign packages to store at suitable locations. The former sub-problem is treated as an Outgoing Package Selection problem while the latter, is an Incoming Package Placement problem. We are only concerned with the latter since in this case, we are not sure if the warehouse is able to fit various quantities of different product packages, when subjected to the warehouse storage constraints. The package placement is analogous to the bin packing problem where the objective is to pack rectangular items of different sizes into some bins/storage areas, and taking up as few bins/storage areas as possible. Both approximate and exact algorithms for 3D bin packing problems have been discussed by Martello, Pisinger, Vigo [3] and Scheithaur [5] respectively. Various heuristics for item packing into a single bin, along with their asymptotic worst-case performance presented by Li and Cheng [4] have provided some guidelines on their respective processing time limits. However, these papers addressed the package placement problem considering physical size constraints. In our problem, concerns such as grouping and ordering packages by their product types and expiry dates, are equally important in facilitating warehouse administration activities like maintenance and stock-taking. This necessarily implies that some space wastage is unavoidable to maintain proper package arrangements. The requirement is that space wastage should be kept to a minimum, so as to accommodate as many packages as possible. A constraint programming approach is taken and due to the requirement of minimizing space wastage, it becomes necessary to search the entire solution space until the algorithm terminates, and the optimal solution found. Warehouses in different locations have different layouts, therefore, there is a need to identify a few concepts to establish a generic coordinate system in which to locate a specific package. Fig. 3 below gives an overview of the concepts defined.



Storage Bays (Major Storage Area)
Walls, Partitions
Most Accessible Side
Max. heights for different stacks
Main Accessway
Stacks (A specific place within a Storage Bay)
Side Accessway
Pillars

Fig. 3 – Generic concepts for formulating package placement CSP

The storable floor area within a warehouse is divided into one or more major storage areas known as bays. The size and position of each bay is dependent on the physical layout of the warehouse like the locations of main/side access ways, pillars, walls and partitions. Each bay could be divided by conceptual grid lines into smaller storage areas known as stacks. The size and number of stacks within each bay is dependent on the type of products to be stored or currently existing in the bay. Each row/column of stacks have different maximum heights, depending on the cross-section profile of the warehouse. For each bay, the most accessible side helps in the modeling of the CSP by providing a preference on which direction to store all the products towards. The following describes some of the more important constraints which are taken into consideration when performing floor planning for incoming packages:

a) Place packages such that those of the same products and batch are clustered as close to each other as possible (i.e. packages should be placed on same or adjacent stacks in the same bay, as far as possible)
b) Position packages based on forecasted outgoing quantities for each product type. (i.e. those projected to leave in the more immediate weeks should be placed closer to the most accessible side)
c) Store packages such that each bay holds only 1 product type/batch as far as possible (this arrangement helps in administrative efforts to locate the relevant product packages, as well as routine activities like stock-taking)
d) To maximize floor-space area, always attempt to fill up each stack 'height-wise', before moving on to other stacks to place the other packages
e) If a storage bay holds more than 1 product type, extra floor space should be set aside to maintain separation between different product types

III. Neural Net training data and pre-processing

Each instance of the CSP involves finding locations for each incoming package for a specific warehouse. The main inputs include the coordinate system of each bay, the profile of the incoming packages, the arrangement and locations of existing packages. If the CSP is able to output a series of suitable locations to store all incoming packages, this implies that the warehouse still has the capacity to store all of them. Whenever the CSP cannot find a solution, this implies that the warehouse is unable to accommodate all the incoming packages without violating some storage constraints. Tables 1 and 2 summarizes the pre-processing involved to derive the input/output training patterns. In table

1, column 'CSP Inputs' describes the inputs for solving the package placement as a constraint satisfaction problem. 'Aggregated Inputs' specifies how the inputs are combined to represent salient features of the CSP. In table 2, 'Input Neuron Value' provides information on the final pre-processed values used for NN training. The second column provides a brief explanation on each input neuron.

Table 1 - Data Preprocessing

CSP Inputs	**Aggregated Inputs**
No. of storage bays (Nb), Dimensions of each bay gives volume for each bay: Vol {Bay1, Bay2, … , Bay_{Nb}}	Storable volume of warehouse: $Vol_{warehouse} = \Sigma$ {Vol_{bay1}, Vol_{bay2}, … , Vol_{bayNb}}
No. of existing packages (Ne), Dimensions of each existing package: Vol {Pkg1, Pkg2, …, Pkg_{Ne}}	Total volume of packages already in the warehouse: $Vol_{existing} = \Sigma$ {Vol_{pkg1}, Vol_{pkg2}, … , Vol_{pkgNe}}
No. of incoming packages (Ni), Dimensions of each incoming package: Vol { Pkg1, Pkg2, …, Pkg_{Ni}}	Total volume of packages coming into the warehouse: $Vol_{incoming} = \Sigma$ {Vol_{pkg1}, Vol_{pkg2}, … , Vol_{pkgNi}}
Existing Package Profile, No. of unique product types already in the warehouse (Nept)	A **set** of variables representing: Qty of each existing Product Type: $Qty_{existing}$ = {Qty_{Prod-A}, Qty_{Prod-B}, …, $Qty_{Prod-Nept}$} Qty of each incoming Product Type $Qty_{incoming}$ = {Qty_{Prod-A}, Qty_{Prod-B}, …, $Qty_{Prod-Nipt}$} Qty of each incoming, totally new Product Type: $Qty_{new} = Qty_{incoming} -$ {$Qty_{existing} \cap Qty_{incoming}$}
Incoming Package Profile, No. of unique product types coming into the warehouse (Nipt)	
CSP Outputs	**Output Training Value**
A set of locations for each incoming package Which_Bay {Pkg1, Pkg2, …, PkgNi} Which_Stack {Pkg1, Pkg2, …, PkgNi} Which_Height {Pkg1, Pkg2, …, PkgNi}	0 if |Which_Bay| and |Which_Stack| and |Which_Height| = ∅, 1 otherwise

Table 2 – Input Neuron Description

Input Neuron Value	**Description**
$Vol_{existing}$ / $Vol_{warehouse}$	This value represents the volume ratio of the existing packages versus the storable volume of the entire warehouse. It models part of the problem that indicates to what degree is the warehouse currently filled with existing packages. Basic statistical analysis and floor planner's experience seemed to suggest that there is a certain limit to the degree of warehouse occupancy beyond which it may not be possible to store anymore packages and yet still adhere to all the storage constraints for the warehouse, irregardless of the values of other inputs.
$Vol_{incoming}$ / ($Vol_{warehouse}$ - $Vol_{existing}$)	This value represents the volume ratio of the incoming packages versus the left-over capacity of the warehouse. It models part of the problem indicating the amount of additional packages to be stored, relative to the remaining storable volume of the warehouse. Subject to the storage constraints, there may be an upper limit to this value beyond which it becomes difficult to store extra packages.
$Vol_{incoming}$ / $Vol_{existing}$	This value represents the volume ratio of the incoming versus existing packages. It models part of the problem indicating the degree in which the volume of incoming packages exceeds those existing in the warehouse. Unlike the previous 2 values, it is possible for this value to exceed 1.
$\| Qty_{new} \| / \| Qty_{existing} \|$	This value represents the ratio of the number of totally new product types to be stored, versus the number of unique product types currently existing in the warehouse. It is possible for this value to exceed 1.
$\| Qty_{new} \| / \| Qty_{incoming} \|$	This value represents the ratio of the number of totally new product types to be stored, out of the number of distinct product types coming into the warehouse.
$\| Qty_{incoming} \| / \| Qty_{existing} \|$	This value represents the ratio of the number of distinct product types coming into the warehouse, versus the number of unique product types existing in the warehouse. Due to the fact that storage policy demands a clear, visible separation between packages of different product types, this indirectly implies a certain amount of volume wastage if the warehouse stores too many different product packages. Therefore, together with the previous 2 ratios, this value expresses the relationships for the number of unique

	product types the warehouse can hold, subject to the various storage rules.
Σ (Qty_{new}) / $\lvert Qty_{new}\rvert$ * Max (Qty_{new})	These 3 values represents the 'even-ness' of quantity distribution over each totally new/current existing/incoming product types respectively. It applies specifically to the cases where all 3 set cardinalities $\lvert Qty_{new}\rvert$, $\lvert Qty_{existing}\rvert$ and $\lvert Qty_{incoming}\rvert$ are greater than 1. For example, the 3rd value models part of the problem indicating the degree of quantitative differences for each incoming product type. The closer the value is to 1, the more 'similar' are the quantities in each type. These values model part of the problem in situations where there are large quantities incoming packages, it is usually easier store them when a greater proportion belongs to a particular type, with the remaining few belonging to the 'odd ones out' category, compared to situations where packages belonging to different product types are in equally great quantities, thereby incurring higher volume wastage to maintain type separation.
Σ ($Qty_{existing}$) / $\lvert Qty_{existing}\rvert$ * Max ($Qty_{existing}$)	
Σ ($Qty_{incoming}$) / $\lvert Qty_{incoming}\rvert$ * Max ($Qty_{incoming}$)	

From the tables above, an instance of a NN training pattern includes all the inputs described in the rows under the 'Input Neuron Value' column, as well as the single row under the 'Output Training Value' column. Due to the fact that there are millions of possible input combinations for the floor planning CSP, this implies there could also be as many possible training patterns that could be generated, which would have taken an immense amount of time. The basic advantage in training a neural net (to learn the floor space capacity check function) is that there is no need to provide all the training patterns in the entire problem domain, in order for it to perform well. However, it is helpful if the entire set of training patterns comprises well balanced representations from as many incoming quantity configuration and initial warehouse occupation scenarios as possible. For a fixed storage capacity of a particular warehouse, the following provides a description of the possible package placement scenarios under which groups of training patterns are generated:

A) Varying the volume of packages present in the warehouse
B) Varying the volume of packages to be stored
C) Varying the number of unique product types existing in the warehouse
D) Varying the number of unique product types in the packages to be stored
E) Varying the number of new product types to be stored in the warehouse

Each scenarios posses a quantitative factor that could be classified as 'low', 'medium' or 'high' categories. These categories maps to certain value range specific to the scenario it represents. For example in scenario B, the greatest amount of incoming packages ever encountered by the warehouse crew could be $600m^3$. Under their counsel, this amount could translate to value ranges of 0 to 100, 200 to 300 and 500 to 600, which corresponds to the 'low', 'medium' or 'high' categories respectively. If there are N scenarios each having c categories, there are N^c combinations of scenarios by varying the input categories. For our training, we used all the above scenarios each having 2 value-categories (low, high), resulting in 25 groups of training patterns to generate. For the patterns in each scenario group, we followed a '70-20-10 percentage rule' in which we randomly select 70%, 20% and 10% to produce the training, testing and verification sample sets respectively.

IV. NN ARCHITECTURE AND RESULTS

Using the NeuroShell application, we constructed a generic multi-layer feed-forward neural net with error back-propagation. Fig. 4 below depicts a simple architecture comprising 3 layers of fully connected nodes.

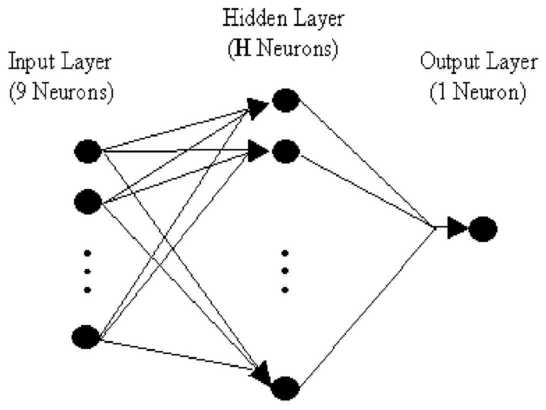


Fig. 4 – NN Architecture to learn the capacity check function for a warehouse

The input layer consists of 9 neurons each representing inputs which have been described in section 3. The activation function used is the default logistic function f(x) = 1/(1+exp(-x)), which produces output values within the range from 0-1. The output layer consists of just 1 neuron and has a linear activation function that maps to give a continuous value. Although the number of neurons in the hidden layer determines how well the capacity check can be modeled, we lack sufficient knowledge to specify a correct estimate. We used NeuroShell's default formula H = ½ (Input + Output neurons) + (No. of patterns in training set)$^{1/2}$ as our initial estimate. The recommended Gaussian activation function f(x) = exp(-x^2) is used in the hidden layer.

Table 3 – Performance comparison of different capacity check models, for a group of warehouses

Warehouse Capacity Check Model	**Correct Estimates (%)**	**Time Taken**
Usable Fraction Estimates	40 - 45	Fast (<= 1 secs)
Constraint Satisfaction Problem	100	Slow (> 10 mins)
Neural Net Estimates	71 - 89	Fast (<= 1 secs)

Table 3 above summarizes the performance of various warehouse capacity check models across all combinations of scenarios, for three warehouses of different layouts. The 'usable-fraction estimates' is a model that uses the percentage figure derived from the experience of the warehouse crew to predict the volume of incoming packages that could be safely accommodated by the warehouse without violating any storage constraints. The CSP model finds the placement solutions for each incoming package quantity configuration, thereby confirming whether the warehouse can hold the packages or not. Finally, 'neural net estimates' is the model which is trained using the outputs generated by the CSP model. The column labeled 'correct estimates' shows the performance range of the various models in correctly predicting both cases where the warehouses actually cannot hold the incoming package quantity, as well as those instances where it can accommodate those quantities.

As shown in table above, the CSP model produces the most accurate results, but is also the slowest as it has to painstakingly work out the placements of each incoming package in order to determine if the warehouse can/cannot accommodate them. The neural net model is able to predict correctly between 71-89% of the incoming package profile instances, and represents an improvement of 31-44% over the usable fraction estimates. This indirectly translates to better utilization of storage space because existing warehouses can actually accommodate more packages due to increases in accuracy of performing warehouse capacity check. Although there is still some gap in predictive accuracy between the Neural Net and the CSP models, the neural net has the greater advantage in run-time processing speed, because all the processing durations has already been 'pre-incurred' during the generation of training patterns using constraint programming techniques.

V. Conclusion

To relate to the bigger problem described earlier, in order to determine the optimal product type and quantity configurations to assign to each warehouse for each period, it is necessary for the network flow model to try thousands of configurations for each warehouse in each period. For the algorithm to work out these assignments within a reasonable amount of time, there must be a quick way to determine, as accurately as possible, which product and quantity configurations could be accommodated by each warehouse, without violating their respective storage constraints. The approach of training a neural net for each warehouse offers a quick and fairly accurate way to perform such capacity check function, without incurring extra processing time in working out exact placements during the running of the main algorithm.

The predictive accuracy may vary greatly depending on the type of warehouse, storage constraints and incoming quantities of different product types and package sizes. However, in the commercial world of logistics where accessible storage locations are limited and space rental is expensive, even a small improvement in warehouse storage utilization could translate to great cost savings for large manufacturing companies maintaining tens or hundreds of different warehouses around the world. To conclude, our exploratory efforts determines that for any logistics application dealing with problems of similar nature, it is worthwhile to train a neural net to model the storage constraints, and learn the capacity function check for each warehouse by combining both constraint programming and neural net techniques in the suggested way.

References

[1] A. Bockmayr, N. Pisaruk, and A. Aggoun, *Network flow problems in constraint programming. Principles and Practice of Constraint Programming*, CP'2001, Paphos, Cyprus, November 2001. Springer, LNCS 2239, 196 - 210. © Springer-Verlag

[2] R. K. Ahuja, T.L. Magnanti and J.B. Orlin. *Network flows : theory, algorithms and applications*. Prentice Hall, 1993.

[3] S. Martello, D. Pisinger and D. Vigo. *The Three-Dimensional Bin Packing Problem*. Operations Research: Mar-Apr 2000, Vol. 48, no. 2.

[4] K. Li and K. H. Cheng. On three-dimensional packing. SIAM Journal on Computing, 19:847-867, 1990.

[5] G. Scheithaner. A three-dimensional bin packing algorithm. J. Inform. Process. Cybernet, 27:263-271, 1991.