SOLVING THE PALLET LOADING PROBLEM USING A COLOURED PETRI NET APPROACH

Miquel Angel Piera[(a)], Catya Zuñiga[(b)]

[(a) (b)]Autonomous University of Barcelona, Dept. of Telecommunication and Systems Engineering, Barcelona, Spain

[(a)]miquelangel.piera@uab.es, [(b)] CatyaAtziry.Zuniga@uab.cat

ABSTRACT

There is a lack of commercial decision support tools that could help to deal with the best configuration of decision variables to optimize the performance of a system with a stochastic, dynamic and synchronous behaviour. In this paper a Coloured Petri Net model that formalizes the Pallet Loading problem and its optimization by integrating evaluation methods with search methods will be presented.

Most Pallet Loading optimization tools, provide very good solutions when the number and type of boxes to be placed in a pallet is considerably low, which is the case of most production industrial systems, since they are used to work with a reduced number of master boxes. However, there are other systems (ie. distribution warehouses) in which there are a high diversity of boxes to be placed in pallets. The model developed provides very good results by using certain heuristics that avoid the analysis of the whole search space, evaluating only the best scenarios.

Keywords: Coloured Petri Nets, Constraints, Coverability Tree, Simulation

1. INTRODUCTION

World-wide market competition, short cycle product time, together with random demands instead of steady demands, are some key-factors which have forced industry to change traditional rigid and/or non-automated production architectures (such as Flow Shop, Job Shop) towards Flexible Manufacturing Systems (FMS).

Despite flexibility to react to market fluctuations can easily be achieved by reprogramming production units (CNC machines), and transport resources, efficient flexibility can only be achieved by a correct coordination of all the entities (material and resources), that takes part on the production and transport processes. A key factor in this highly competitive market is the ability to respond rapidly to changes in the demand while minimizing costs.

Most research and development efforts have been focussed in the process of planning the efficient, cost effective flow and storage of raw materials, in-process inventory, finished goods and related information from point of origin to point of consumption. However, the picking of the final deliverable products into pallets can be a critical supply chain component for many distribution centers.

The complexity in transport, the changing mentality in logistics and the constant need to improve competitiveness provoke the necessity of developing new tools that could tackle the problem from a global point of view, considering operational, strategic and tactic decisions.

There are different methodologies that have been used traditionally to give response to planning problems. Modelling and Simulation techniques have proved very useful in strategic and tactic design. However, several limitations appear when trying to find a feasible solution to a NP-complete problem. A limited number of scenarios can be evaluated in an acceptable time interval.

A solution for these operational decision problems would be finding a modelling approach that could tackle these problems by integrating different approaches: search methods (AI, OR) with evaluation methods (Simulation).

Coloured Petri nets provide a framework for the construction and analysis of distributed and concurrent systems. A CPN model of a system describes the states which the system may be in and the transitions between these states. CP-nets have been applied in a wide range of application areas, and many projects have been carried out in industry. In contrast to most specification languages, Petri nets are state and action oriented at the same time – providing an explicit description of both the states and the actions, (Kristensen, 1998).

A Discrete Event System model has been developed to formalize the pallet packing problem (PLP), using the CPN formalism to analyze the best configuration that minimizes the number of pallets required for packing a certain amount of boxes.

2. A DISTRETE EVENT MODEL FOR PALLET MAKER

There are several modelling approaches to tackle the PLP, some of them are based in container loading approach which try to construct vertical 'walls' across the container (Lim, Rodrigues, and Yang, 2005). George and Robinson (1980) where the first to formulate a heuristic algorithm called the wall-building

method for the packing of up to 20 different box types into one container.

Gehring, Menschner, and Meyer (1990), proposed a method to pack rectangular boxes of different size into a shipping container of known dimensions. This problem specification does not allow for orientation constraints. The algorithm is also based on the wall-building method. In this method, a series of ranking rules such as box ranking rule, box position ranking rule are assumed. Bischoff and Dowsland (1982), developed a method based on filling the container by building layers across its width. Han, Knotta and Egbelu (1989) provided a heuristic based on the dynamic programming method and wall-building method. In his study, the objective was to maximize the volume occupancy without no orientation constraints. The algorithm calls for the boxes to be packed along the base and one vertical wall of the container. After a L-shaped packing is complete, a "new" container can be formed, which become the focus of the successive packing. Xue and Lai (1997), provided another algorithm based on the wall-building method, in which both the cargo and container must be rectangular. This algorithm integrated three heuristics:

- The ordering heuristic sorts the cartons according to the depth, quantity and surface area. Higher priority was assigned to cartons with larger values of the above characteristics.
- The placement heuristic determined the depth of the new layer.
- Layer-building heuristic.

Ngoi, Tay, and Chua (1994), designed a heuristic algorithm based on "spatial representation" techniques to solve the problem of packing rectangular carton boxes into a single rectangular container, which includes finding the best placement position and orientation for each box. By increasing the number of boxes to be fitted in the container, the matrix expands to accommodate additional information. This packing algorithm is independent of the ordering of the boxes which usually exists in other algorithms. However, the strategy divides the original container into many small empty spaces which are not suitable for holding big boxes in successive packing. Chua, Narayanan, and Loh (1998) described an improved method based on Ngoi, Tay, and Chua (1994), which allows the user to suggest the locations of certain boxes which exists in real-life packing problems.

Ivancic, Mathur, and Mohanty (1989), proposed an integer programming based heuristic approach to the three dimensional packing problem in which a container is packed considering to maximize a linear function of the boxes packed.

Most of these packing strategies were based on the wall building approach with some adaptations. The method itself is a greedy heuristics which can lead to weak final solutions. Other methods include using spatial representation, reducing the problem to maximum clique problem or an integer programming problem. For methods based on the spatial representation, empty volume search routines are used to generate small spaces, whereas the successful utilization of these small spaces is difficult. For methods based on graph theory and integer programming, some disadvantages appear due to model simplification aspects and the inherent difficulty of the combinatorial nature of these problems. On the whole, published works generally give successful implementations and provided some interesting insights into the various views on how successful packings can be best achieved, (Lim, Rodrigues, and Yang, 2005).

The PLP considered in this paper deals with n different types of boxes (rectangular shaped) with known dimensions cxi, cyi, czi (with i = 1 . . . . . n) which have to be placed onto a surface pallet of size scx * scy. The objective is assumed to be the maximisation of the number of boxes to be fitted inside the pallet, preserving (i.e. the maximisation of pallet utilisation) preserving several constraints such as the maximum weight supported in each pallet, the maximum height and floors recommended, etc.

Boxes may be rotated 90º so long as they are placed with edges parallel to the pallet's edges, i.e., the placement must be orthogonal, and they can be rotated only once. It is assumed, without loss of generality, that cxi, cyi, czi, scx, scy, gz, are positive integers, in which scx, scy defines the pallet position where the left hand corner of the box-i has been placed, and gz corresponds the pallet floor where box-i has been placed. It is also assumed that each box has a ''this side up'' restriction.

The palletizing problem can be seen as a DES by busing a proper abstraction level in which each event represents the placement of a certain box in the pallet surface, as it is represented in Figure 1.

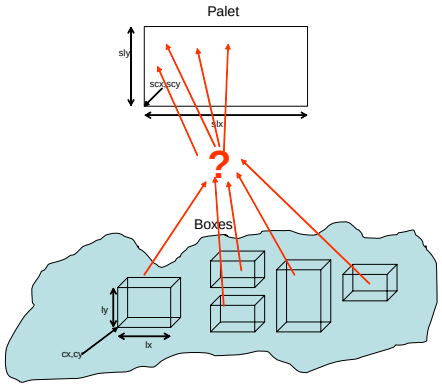

Figure 1: Placing boxes into a pallet

Attributes can be defined to describe the pallet configuration: the coordinates of each box placed in the pallet surface together with the coordinates of the fragmented space as the results of placing a box in the pallet.

As a consequence of placing a box in the pallet, fragmented surfaces will appear. Figure 2 illustrate this

situation, in which it is easy to identify two different surface areas in the pallet that should be evaluated to fit the next box in the pallet. Thus, events describing different possibilities for placing a box in a pallet should compute the new layout configuration of the pallet once the box has been placed: position and orientation of each box in the pallet, together with the computation of the dimensions of each new free fragmented space generated as a consequence of placing boxes.

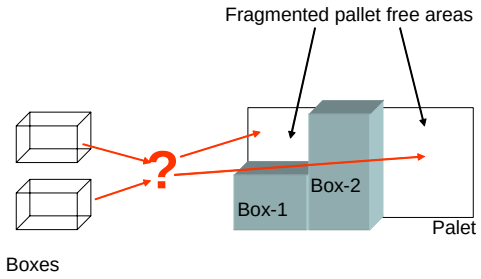

Figure 2: Fragmented areas in a pallet

3. MODEL

The Pallet loading model has been specified in the CPN formalism (Jensen, 1997) and it can be decomposed into 5 different set of transitions:

- Transitions T_{21} ... T_{24} : These transitions represent fitting a box into a free pallet area where the length *x* and/or *y* of the box are equal or shorter than the length *x* and *y* of the free pallet surface
- Transitions T_{31} ... T_{35} : These transitions represent fitting a box into a free pallet area where the length *x* and/or *y* of the box are longer than the length *x* and *y* of the free pallet surface
- Transitions T_{41} ... T_{48} : These transitions represent fitting a virtual box into a free pallet area where the length *x* and/or *y* of the box are equal or shorter than the length *x* and *y* of the free pallet surface
- Transitions T_{51} ... T_{58} : These transitions represent fitting a virtual box into a free pallet area where the length *x* and/or *y* of the box are longer than the length *x* and *y* of the free pallet surface

3.1. Colour specification

Table 1 and 2 summarizes the colours and places used to describe all the information required to fit boxes in a pallet using the abstraction level of the pallet maker process introduced in this section.

Table 1: Place specification

Place	Colour	Meaning
P1	1'(idc,cx,cy,cz,lx,ly,lz,cr,ce)	Boxes
P2	1'(scx,scy,slx,sly)	Free Pallet Surfaces
P3	1'(idc,cx,cy,cz,lx,ly,lz,ce)	Virtual Boxes
P4	1'(ge,gz,gsf,gncv,gnc)	Global Variables

Table 2: Colour specification

Colour	Definition	Meaning
idc	Integer	Box identifier
Cr	Integer	0: original orientation 1: rotated 90º wrt Z
Ce	Integer	0: not assigned 1: working 2: placed in the pallet
Cx	Real	Coordinate X where the box is located
Cy	Real	Coordinate Y where the box is located
Cz	Real	Coordinate Z where the box is located
Lx	Real	Box length in coordinate X
Ly	Real	Box length in coordinate y
Lz	Real	Box length in coordinate z
Scx	Real	Coordinate X where the surface is located
Scy	Real	Coordinate Y where the surface is located
Slx	Real	Surface length in coordinate X.
Sly	Real	Surface length in coordinate Y.
ge	Integer	0: A Box can be placed in the pallet 1: Box to be assigned 2: Looking for a surface 3: Evaluating the new fractioned surfaces
gz	Integer	Indicates the pallet floor
gsf	Real	Available surface in the pallet
Gncv	Integer	Number of virtual boxes
Gnc	Integer	Total number of boxes

3.2. Events examples

Figure 4 illustrates the event that formalizes fitting a box into a free pallet area if the length *x* and *y* of the box are shorter than the length *x* and *y* of the free pallet surface. It should be noted that under these circumstances, the fragmented new spaces in the pallet is incremented in two new free areas that can be used as free pallet areas to place future boxes.

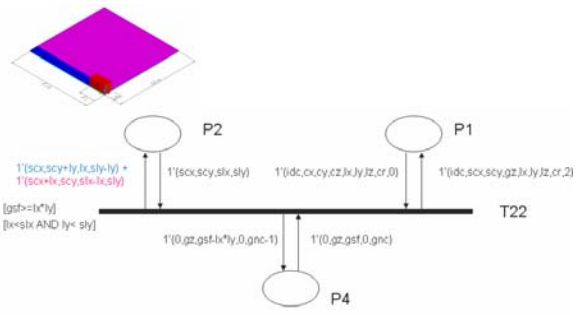

Figure 3: T22. Fitting a box with lx<slx and ly<sly

Node P1 holds the tokens associated to boxes while P2 holds the tokens representing the free areas in the pallet. Thus, in transition T22 (Figure 3) two tokens are generated to describe the two new free squares generated due to the space fragmentation.

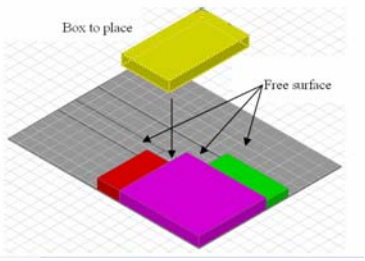

Figure 4: T31. Fitting a box with lx>slx and ly>sly

When a box does not fit in a free area, as shown in Figure 4, but there is a way to place it by compacting different free areas that can be attached altogether, the box is virtually decomposed as a sequence of smallest boxes (virtual box concept). A virtual box is the division of a box into small boxes which must be placed in contiguous free areas, Figure 5.

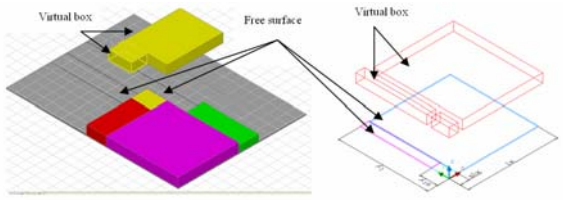

Figure 5: T31. Virtual boxes

Figure 6 illustrates the event that formalizes fitting a box into a free pallet area when the box lengths *x* and *y* are longer than the lengths *x* and *y* of a free pallet surface. Therefore, two new virtual boxes are generated and added in the place *P3* to describe the dimensions. P4 is used to control and coordinate the sequence of events to solve a virtual box placement before any other box could be chosen to be placed in the pallet. Colour *Ge* changes form *0* to *1* in the output arc to mark that the original box has been divided into one or more virtual boxes; the global free surface also is decremented in *sly*slx* which are the dimensions of the free surface used and the number of virtual boxes *gncv* is incremented in two, which are the two virtual boxes generated due to the firing of this transition.

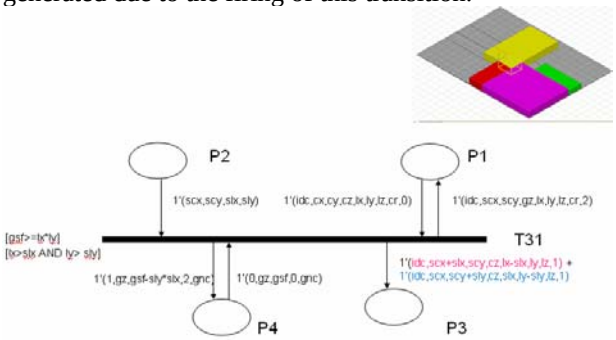

Figure 6: T31. Fitting a box with lx>slx and ly>sly

To place a virtual box, two different set of events have been considered: the first set represents the placement of a box that do not overcome a free surface, while the second set of events places a box which does overcome the surface and generate more virtual boxes.

Figure 4 which was used to introduce the concept of virtual box is used again to illustrate the idea of placing virtual boxes. It is assumed that part of the original box has been placed and it remains only one virtual box to be placed as shown in Figure 7.

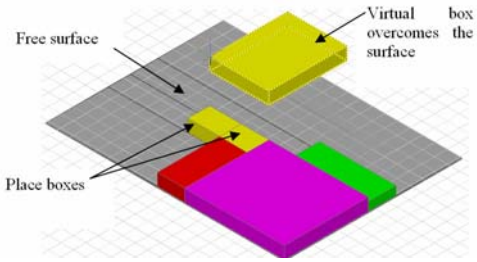

Figure 7: Fitting a box with lx>slx, ly<sly, scx=cx and scy=cy.

In this situation, transition T58 would be fired (see Figure 8), in which a virtual box is placed generating one virtual box, which is indicated by the output arc of place *P3*. Also another free surface is generated, output arc of place *P2*. Finally as explained in the transition *T31*, global free area is decremented and number of virtual boxes keeps steady because one virtual box is placed but another y is created.

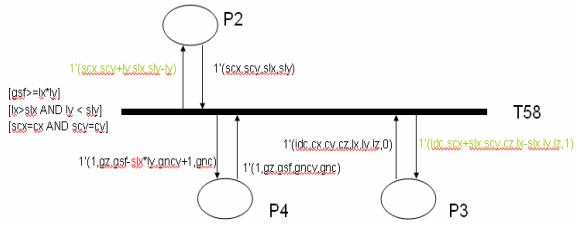

Figure 8: T58. Fitting a box with *lx>slx, ly<sly, scx=cx* and *scy=cy*.

Additionally to all these events that specify how to place a box into a free pallet area, there is a particular event that allows changing the orientation of the box in order to fit better in a free pallet area. This new event can be fired at any time but only once per each box. Figure 9 illustrates the arc expressions that describe this event.

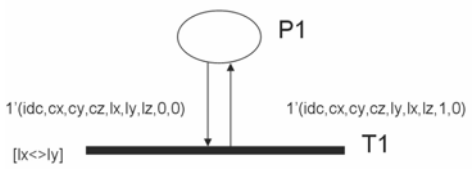

Figure 9: Rotation of a box to be fitted in the pallet

3.3. Coverability tree

The coverability tree relies on the computation of all reachable states and state changes of the system, and it is based on an explicit state enumeration.

By means of the coverability tree, behavioural properties of the system can be verified. Examples of such properties are the absence of deadlocks in the system, the possibility of always reaching a given state, and the guaranteed delivery of a given service. The coverability tree of CP-nets can also be applied to timed CP-nets. Hence, it is also possible to investigate the functional correctness of the proposed model when it is used as a DSS at the final part of a production system.

By using a CPN simulator that can support the evaluation of the coverability tree of the system described under different work loads (different boxes specified in place *P1*) it is possible to check the different combination in which boxes can be fitted in the pallet, and choose the one that minimizes the number of levels of boxes in the pallet.

The specification of the final state consists to force all the tokens in node *P1* to set the colour ce with value 2, mathematically represented by the vector:

$$Mf=[*'(*,*,*,*,*,*,*,2),*,*,*] \quad (1)$$

Thus, marking M_f can be interpreted that any state with all the tokens in place *P1* with colour *ce=2* can be considered the goal state since all the boxes has been fitted inside the pallet area.

3.4. Heuristics

By considering that the exploration of the whole coverability tree is quite expensive in terms of computer memory requirements and computational time, some heuristics have been designed to avoid the evaluation of certain sequence of events that will not lead a good solution. In Piera (2007) the main aspects of the CPN tool used to support heuristics and knowledge representation to improve the analysis of the coverability tree is presented. This tool has been used to get feasible results solving the pallet loading problem using a reduced number of different types by means of formalizing specific knowledge in terms of heuristics.

3.4.1. Place boxes with the same characteristics

An heuristic has been developed to implement in the model so boxes are placed by groups of the same type. Thus, if a box of certain type is chosen, the model is forced to place all boxes of this type before any other type of box could be chosen.

This heuristic is specified by the introduction of two more transitions to the model, two places and two colours. These colours are *tp*, type box which can take values of *0* if the type of box has never been picked and *1* if it has been picked The other colour added to the model is *nc*, the number of boxes of the same type.

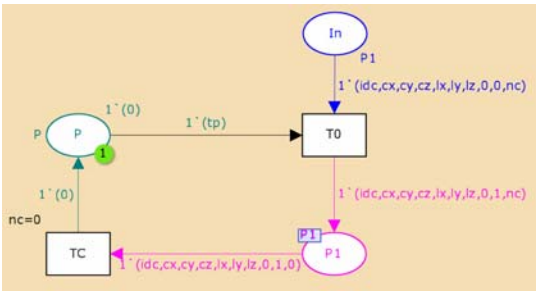

Figure 10: Introduction of the heuristic

Figure 10 illustrates the changes in the model introduced by the heuristic. Transition *T0* marks a type of box as used by changing *tp* from *0* to *1* while *TC* release this marking only in case all boxes of this type have been place.

4. RESULTS

Several tests to explore the coverability tree have been made based in the CPN tools environment, (CPN Tools home page). CPN tools is a graphical computer tool supporting CP-nets. It consists of three closely integrated components. The CPN editor supports the construction and editing of hierarchical CPN models. Simulation of CPN models is supported by the CPN simulator, while the CPN state space tool supports the state space method of CP-nets. In addition to these three main components, the tool includes a number of additional packages and libraries.

It is possible to attach a piece of sequential CPNML code to obtain the information required.

4.1.1. Scenario evaluations

In this test, seven boxes of two types have been introduced. The initial markings of the boxe place is 3`*(1,0,0,0,40,50,10,0,0)*++4`*(1,0,0,0,20,50,10,0,0)*. These boxes must be fitted in a surface of 100x100 units.

Eight feasible solutions were reached, but some of them take a longer way to reach the result, it means more transitions must be fired to reach the final state. Thus, two feasible solutions with the shortest path (number of events used) has been chosen (see table 3, figure 11)).

Table 3: Optimal solutions

Solution No. 1
1`(1,0,0,0,40,50,10,0,2)+1`(1,0,50,0,40,50,10,0,2)+ 1`(1,40,0,0,20,50,10,0,2)+1`(1,40,50,0,20,50,10,0,2) 1`(1,60,0,0,40,50,10,0,2)+1`(1,60,50,0,20,50,10,0,2) 1`(1,80,50,0,20,50,10,0,2)
No empty space
Solution No. 2
1`(1,0,0,0,40,50,10,0,2)+1`(1,0,50,0,40,50,10,0,2)+ 1`(1,40,0,0,40,50,10,0,2)+1`(1,40,50,0,20,50,10,0,2) 1`(1,60,50,0,20,50,10,0,2)+1`(1,80,0,0,20,50,10,0,2) 1`(1,80,50,0,20,50,10,0,2)
No empty space

As it is shown, these feasible solutions place the eight boxes into the pallet leaving no space unused. All necessary information such as coordinates and dimensions of each box is generated and can be checked by evaluating tokens' colours in place *P2*.

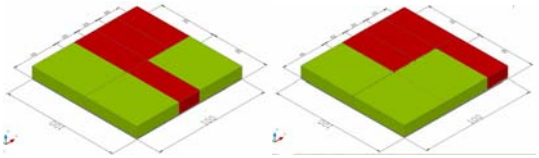

Figure 11: Two feasible solutions

4.1.2. Test results using heuristics

In this test, eight boxes of three different types were introduced. The initial markings of the box place is: *1`(1,0,0,0,60,40,10,0,0,2) ++ 1`(2,0,0,0,40,35,10,0,0,2) ++ 1`(3,0,0,0,30,20,10,0,0,4.*.

Again, the coverability tree was explored and two feasible solutions were reached, and both fired the same number of transitions to reach the final state, the information is shown in table 4, (see figure 12).

Using this information it can noticed that the 8 boxes are placed successfully inside the pallet and there is no free space unused. This means that a 100% pallet utilisation can be obtained with all boxes properly placed.

Table 4: Optimal solutions

Solution No. 1
1`(1,40,0,0,60,40,10,0)+1`(1,40,40,0,60,40,10,0)+ 1`(2,0,0,0,40,35,10,0)+1`(2,0,35,0,40,35,10,0)+ 1`(3,0,70,0,20,30,10,1)+1`(3,20,70,0,20,30,10,1)+ 1`(3,40,80,0,30,20,10,0)+1`(3,70,80,0,30,20,10,0)
No empty space
Solution No. 2
1`(1,0,0,0,60,40,10,0)+1`(1,0,40,0,60,40,10,0)+ 1`(2,60,0,0,40,35,10,0)+1`(2,60,35,0,40,35,10,0)+ 1`(3,0,80,0,30,20,10,0)+1`(3,30,80,0,30,20,10,0)+ 1`(3,60,70,0,20,30,10,1)+1`(3,80,70,0,20,30,10,1)
No empty space

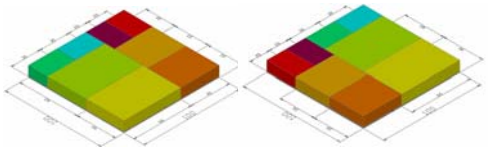

Figure 12: Two feasible solutions

5. CONCLUSIONS AND FURTHER WORK

A pallet maker model approach using the Coloured Petri Net formalism has been presented. The model has been evaluated under different work load conditions, obtaining better results than using commercial software tools as far as pallet utilization it concerns. To improve computational time, some heuristics have been introduced, in such a way that feasible solutions can be obtained.

Due to some shortages inherent to CPN tools, the proposed model has been codified under another logistic focussed CPN environment (see Piera 2004), in order to integrate the model in the DSS of picking process operation that appears in most of production industries.

REFERENCES

CPN Tools home page: http://www.daimi.au.dk/CPNTools/.

Bischoff, E., Dowsland, W.B., 1982, An application of the microcomputer to product design and distribution, *Operational Research Society Journal*.

Chua, C.K., Narayanan, V., Loh, J., 1998 Constraint-based spatial representation technique for the container packing problem, *Integrated Manufacturing Systems*.

Gehring, H., Menschner, K., Meyer, M., 1990, A computer-based heuristic for packing pooled shipment containers, *European Journal of Operational Research*.

George, J.A., Robinson, B.F., 1980, A heuristic for packing boxes into a container, *Computer and Operational Research*, 7, pp. 147–156.

Han, C.P., Knott, K., Egbelu, J.P., 1989, A heuristic approach to the three-dimensional cargo-loading problem, *Int. J. Prod. Res*., 27 (5), pp. 757–774.

Ivancic, N., Mathur, K., Mohanty, B.B.,1989, An integer programming based heuristic approach to the three dimensional packing problem, *Journal of Manufacturing and Operations Management*.

Jensen, K, 1997, *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use*, 2, Springer-Verlag. Berlin.

Kristensen, M.L., 1998, Christensen, S., Jensen, K., The practitioner's guide to coloured Petri nets, *Springer-Verlag*, pp. 98-132.

Lim, A., Rodrigues, B., Yang, Y., 2005, 3-D Container Packing Heuristics *Applied Intelligence, Springer Science + Business Media*, l22, pp. 125–134.

Ngoi, B.K.A., Tay, M.L., Chua, E.S., 1994, Applying spatial representation techniques to the container packing problem, *Int. J. Prod. Res*.

Piera, M. A., et al. 2004, Optimization of Logistic and Manufacturing Systems through Simulation: A Colored Petri Net-Based Methodology, SIMULATION, *Transactions of The Society for Modeling and Simulation International*, 8, pp. 121-129.

Piera, M.A., Mujica M.A. Guasch, 2007, An efficient CPN modeling approach to tackle the pallet packing problem, *in Proceedings of the 6th EUROSIM Congress on Modeling and Simulation*, September 9-13, Ljubljana (Slovenia).

Xue, J., Lai, K.K., 1997, Effective methods for a container packing operation, *Mathl. Comput. Modelling*.