A Constrained Fuzzy Knowledge-Based System for the Management of Container Yard Operations

Abbas, AHHJ, Al Bazi, A & Palade, V

Original citation & hyperlink:
Abbas, AHHJ, Al Bazi, A & Palade, V 2018, 'A Constrained Fuzzy Knowledge-Based System for the Management of Container Yard Operations' *International Journal of Fuzzy Systems* , vol 3, no. 16, pp. (in press)
https://dx.doi.org/10.1007/s40815-018-0448-9

DOI 10.1007/s40815-018-0448-9
ISSN 1562-2479
ESSN 2199-3211

Publisher: Springer

The final publication is available at Springer via http://dx.doi.org/10.1007/s40815-018-0448-9

A Constrained Fuzzy Knowledge Based System for the Management of Container Yard Operations

Ali Abbas[1] . Ammar Al-Bazi[1] . Vasile Palade[1]

Abstract The management of container yard operations is considered to be a very challenging task by the yard operators due to many uncertainties inherent to such operations. The storage of the containers is one of those operations that require proper management for efficiently utilising the container yard, reducing the retrieval time and the number of re-handlings.

The challenge of the problem faced in this paper appears when newly arrived containers of a different size, type and weight need to be stored in a yard that holds a number of pre-existing containers. This challenge becomes even more complex when the departure time of a container is unknown, as is the case when the container is collected by a Third Party Logistic (3PL) company without any prior notice of the collection date/time and then delivered to customers.

The aim of this study is to develop a new constrained fuzzy knowledge based system for the management of container yard operations that takes into consideration a number of real life factors and constraints. One of these factors is the duration of stay for the topmost containers of each stack when the containers are stored. Because the duration of stay for containers in a yard varies dynamically over time, an 'ON/OFF' strategy is proposed to activate/deactivate the duration of stay factor constraint if the length of stay for these containers varies significantly over time.

As a methodology used in this research, a number of tools and techniques are utilised for developing the proposed system including: Discrete Event Simulation for the modelling of container storage and retrieval operations, a Fuzzy Knowledge Based Model (FKBM) for the stack allocation of containers, and a Heuristic Algorithm called 'Neighbourhood' Algorithm for the container retrieval operation.

Results show that by adopting the proposed 'ON/OFF' strategy, 5% of the number of re-handlings, 2.5% of the total retrieval time, 6.6% of the total re-handling time and 42 % of the average waiting time per truck are reduced.

Keywords Constrained Fuzzy Knowledge Based System . 'ON/OFF' Strategy . 'Neighbourhood' Algorithm . Container Yard Operations

1 Introduction

With the growth in the international container shipping worldwide owing to the offshoring of manufacturing, there has been an increased interest in improving the operations in the container terminals [1].

Container terminals involve a number of operations including rail side, container-yard side, and the gate-side operations. The most important operation is the container-yard side involving the storage and retrieval of the containers.

The management of container yard operations is a complex task, and this complexity is due to inherent uncertainties in the storage and/or retrieval operations of the containers. The storage operation of containers is a very important task for achieving efficient utilisation of container yards. Proper storage operation leads to a reduction of the container yard operations cycle time [2].

A number of problems are faced in such storage operations where the departure time of containers is unknown, which include storage space allocation and location assignment [3].

Ali Abbas
abbasa2@uni.coventry.ac.uk

Ammar Al-Bazi
aa8535@coventry.ac.uk

Vasile Palade
ab5839@coventry.ac.uk

[1] Faculty of Engineering, Environment and Computing, Coventry University, Coventry, UK

The storage space allocation problem studies the assignment of containers to a block or a bay, while the location assignment (stacking) problem involves the allocation of containers to stacks [4]. The location assignment problem can be considered quite complex because of the uncertainty regarding which container departs first. [5] and [6] have both performed studies concerning the location assignment problem for containers with an unknown departure time where they considered a number of real-life criteria such as grouped containers which can be stored in the same block and stack (i.e. discharged from the same vessel, belonging to the same weight class, picked up by the same customer, and retrieved in the same period), and the arrival time of containers (i.e. the length of time the container has been in the yard).

However, additional factors together with more real-life constraints need to be taken into consideration during the storage operation, and hence this study was established to achieve this.

This paper aims at developing a new constrained fuzzy knowledge based system for container storage and retrieval operations where the departure time of containers is unknown. It introduces a new methodology that can be used to solve complex real-life storage and retrieval problems.

The developed system considers additional real-life factors and constraints that lead to more realistic container yard operations. This system considers three factors including the number of containers in each stack, the similarity between containers in each stack (i.e. containers belonging to the same customer), and the duration of stay of the topmost container in each stack. The constraints considered in this system are weight (full and empty), size, and type. Based on these factors and constraints, stacks are allocated for container storage.

The rest of this paper is organised as follows: Related work is reviewed in Section. 2. Section 3 presents the container storage problem. The research methodology is presented in Section 4. Section 5 provides the experimental part, results analysis and discussion, followed by the conclusion and future work in the final section.

2 Previous Work on Container Storage Operations Given Unknown Departure of Containers

The container storage problem in a container yard is an important part of yard management, thus, container yard operations have been of interest to researchers. So, in this section, the existing approaches for container storage problems when the departure time of the container is unknown are reviewed.

A fuzzy logic model taking into consideration a set of criteria such as the distance of the block to the gate, the block utilisation, the stack height, and the difference in the estimated time for dispatching the container was developed [7]. The developed container stacking model aimed at reducing the relocation ratio of containers with random departure times Another researcher [8], discussed ordered stacking and random stacking strategies for the slot assignment of containers in both single and twin storage areas with unknown departure time for the containers. [5] presented a fuzzy optimisation model was presented for storage space allocation to try to balance the total number of containers in a yard given the uncertain departure time. A number of criteria were utilised such as grouped containers which can be stored in the same block and stack. [9] suggested an efficient Genetic Algorithm (GA) to solve the storage space allocation problem to reduce the container storage and retrieval times. The storage operation considered the type and size of container and the fact that the pick-up time was known. However, there were containers with unknown departure times at the moment of planning, or containers with known departure times beyond the planning horizon. [10] established a model for allocating export containers to container yard blocks for balancing the workloads and reducing the transportation distance. During the allocation operation, retrieval time, type and number of containers were used as constraints.

[11] addressed the optimisation of a block stacking system for reducing the number of blocks relocated. The algorithm considered the arriving unit load type during the relocation operation. [12] analysed both segregation and non-segregation strategies. In a segregation strategy, cargoes from different ships were

separated to reduce the number of extra movements of containers and the operation cost. [13] improved further the segregation strategy by considering the arrival pattern of containers. [14] proposed a methodology for the container stacking problem which estimated the number of re-handlings required. [15] introduced methods for container storage which could handle whether or not newly-arrived containers were mixed with pre-existing containers when the departure time of the containers was uncertain. The methods were introduced to estimate the number of containers which had to be relocated. [6] considered the probability for the departure of containers from the and functions were developed to achieve the optimal storage with the aim of minimising unproductive movements of the containers. The arrival and departure rates of containers and the storage yard characteristics were considered in the optimisation process.
Although some of the previous works above have shown serious attempts in modelling and solving container storage problems, the modelling of these works considered only limited number of factors and real-life constraints.
In this paper, additional factors and constraints are considered that reflect the real-life situation. These input factors include, the number of containers in each stack, the similarity between containers (i.e. containers belonging to the same customer), and the duration of stay of containers. The constraints include container size, type and weight (empty and full).

3 Problem Description

The management of the container yard including the storage and retrieval operations for containers is a complex task. This complexity appears when containers in the yard need be stored in stacks taking into consideration different constraints including the type, size and weight of the container together with the number of containers, the similarity between containers, the duration of stay and where the departure date/time is unknown.
A container terminal utilises reach stackers to handle containers. The reach stacker is used to store, re-handle, retrieve and upload containers onto trucks within the container yard. See Fig. 1 for the layout of a container yard with pre-existing containers.

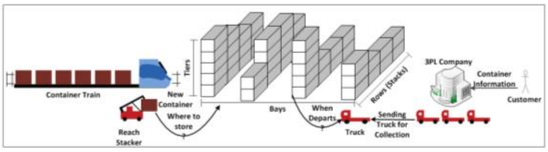


Fig. 1 Schematic representation for the layout of a container yard

In Fig. 1, the storage operation starts when a container train arrives at the terminal with a number of new containers. The reach stacker starts unloading containers from the train and stores them in the yard.
The problem appears when deciding where to store the newly arrival containers that have different weight, size and type (Import Containers) with already existing containers, especially when the departure time of containers (pre-existing and newly arrived containers) is unknown. The departure time of containers is unknown because the containers are collected by trucks that belong to Third party Logistic (3PL) companies. Customers deal directly with these 3PL companies to deliver their containers. The 3PL companies have a limited number of trucks for container collection and transportation to customers. Customers that have containers in the yard inform the 3PL companies to collect their containers without any prior notice given to the yard operators. The 3PL companies send trucks to the terminals for collection without any advance notice which makes the storage operation challenging.
In the next section, the tools and techniques used in modelling the container yard management system are discussed.

4 Development of 'FKB_CYM' – the Constrained Fuzzy Knowledge Based System for Container Yard Management

In order to solve the container-stack allocation problem based on an unknown departure time, the Fuzzy Knowledge Based technique is used. This technique is required because the arrival time of trucks to take the containers to their destination is unknown. A Heuristic Algorithm named 'Neighbourhood' is then used to model the re-handling operation of containers. The Discrete Event Simulation approach is utilised to mimic the arrival, storage and retrieval operations for the containers. The framework of the proposed 'FKB_CYM' system is explained below.

4.1 Framework of the 'FKB_CYM' System

This section presents the framework for the proposed Fuzzy Knowledge Based system. The system framework is comprised of the input, process and output components as shown in the Fig. 2. The input component consists of the specification and container yard information.

The process component involves a collection of techniques that work together to process the inputs. Finally, the output component includes a number of Key Performance Indicators which are categorised based on the operational time, yard criteria, truck utilisation, and resource utilisation.

In the input component, the specification information consists of a number of input parameters such as container yard definition, number of pre-existing containers, number of customers, number of companies, number of trucks, truck travel time, number of container trains, and the inter-arrival time of container trains. In addition, it includes the storage and retrieval time per row and bay, as well as the re-handling time per row, bay, and tier. The Container yard information involves a number of the related factors alongside real-life constraints which include: the container size, type, and weight (both empty and full) of the topmost containers in each stack. The factors which need to be considered are the number of containers in each stack, the duration of stay (i.e the length of time the topmost container has stayed in each stack) and the similarity of containers in each stack (i.e. containers which belong to the same customer). All this information is fed into the system to generate the required outputs.

However, the duration of stay of different containers is changing dynamically over time which requires an 'ON/OFF' strategy to determine whether or not the factor is taken into account in subsequent processing. This strategy will be discussed in section 4.4.2.

The process component is comprised of three modules including the Heuristic Algorithm, the Fuzzy Knowledge Based module and the Discrete Event Simulation techniques.

The process starts when the container yard information is fed into the Fuzzy Knowledge Based module and the specification information is fed into both the storage allocation operation module and collection operation module to be processed. The specification information, which includes the container yard definition (i.e. number of bays, number of rows, number of tiers), number of pre-existing containers and the number of new containers in each train is fed into the storage allocation operation to initiate the storage of containers in the yard. Also included in the specification information is the number of trucks that collect the containers to deliver to customers, which is fed into the collection operation. Using the input information, the Fuzzy Knowledge Based module determines (i.e. allocates) the stack in which to store the container. It achieves this by first calculating an acceptability level (α_i) for each stack, then, the container is allocated to the stack which has the highest acceptability level. The container is stored in the allocated stack and the yard information will be updated.

The heuristic algorithm is then applied to re-handle any container which is on top of the required one. When the collection process for a container takes place, the required container is retrieved and uploaded onto a truck. Once the collection process is completed, then the container yard information will be updated accordingly.

The discrete event approach is used in general to simulate the arrival and departure processes of both trains and vehicles alongside with other yard operations including storage and retrieval ones. The events of all entities including containers as temporary entities and trains, reach stackers, and vehicles as permanent ones. See Fig. 2 for the proposed framework

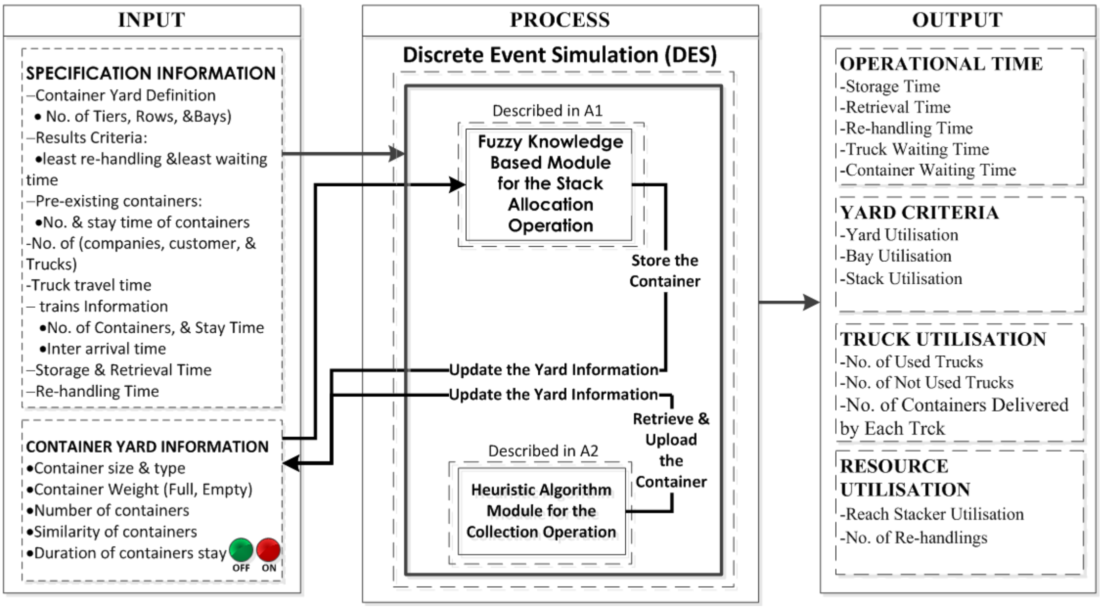


Fig. 2 The 'FKB_CYM' System Framework

The output module as shown in Fig. 2 includes the number of retrievals, operational time, space utilisation, truck utilisation and resource utilisation. In this case, the operational time includes for the container, storage time, retrieval time, re-handling time and waiting time, together with the truck waiting time. The space utilisation includes the yard utilisation, bay utilisation, and stack utilisation. The resource utilisation includes the reach stacker utilisation and the number of re-handlings of containers. The truck utilisation consists of the number of used and unused trucks, and the number of containers delivered by each truck used.

In the next section, each of the techniques used in the core of the 'FKB_CYM' system are explained in more detail.

4.2 The 'FKB_CYM' Components

This section discusses all the techniques used to develop the proposed 'FKB_CYM' system. This core system uses both Fuzzy Knowledge based and Discrete Event techniques to imitate the storage, retrieval, and re-handling operations for containers, together with the Heuristic Optimisation module to ensure a near optimal collection/retrieval operation. See Fig. 3 for the 'FKB_CYM' core components.

The core components start responding when incoming containers arrive by train and need to be stored in stacks within the container yard. The storage process is restricted by a number of real-life constraints.

The Fuzzy Knowledge Based module assesses the location to store the incoming container by using fuzzy reasoning which takes into account the constraints, and subsequently assigns an acceptability level value (α_i) to each stack. The constraints of the system are the weight (i.e.both empty and full), size and type differences for the containers in each stack.

Inputs from the container yard information are regarded as crisp inputs, which need to be fuzzified using fuzzy sets, which are represented by their respective membership functions, in order to apply the fuzzy knowledge based module. The fuzzy inference component which includes aggregation, will manipulate the given information in fuzzy format within the defined fuzzy rules.

The fuzzy output will then be de-fuzzified using one of the methods [19] and [20], to calculate the acceptability level value (i.e. crisp value) of each stack to be used for the allocation of incoming containers. The stack with the highest acceptability level value will then be used for container storage, while simultaneously satisfying all inputs and conditions. Once the container is stored, the system updates the yard information for the next incoming container.

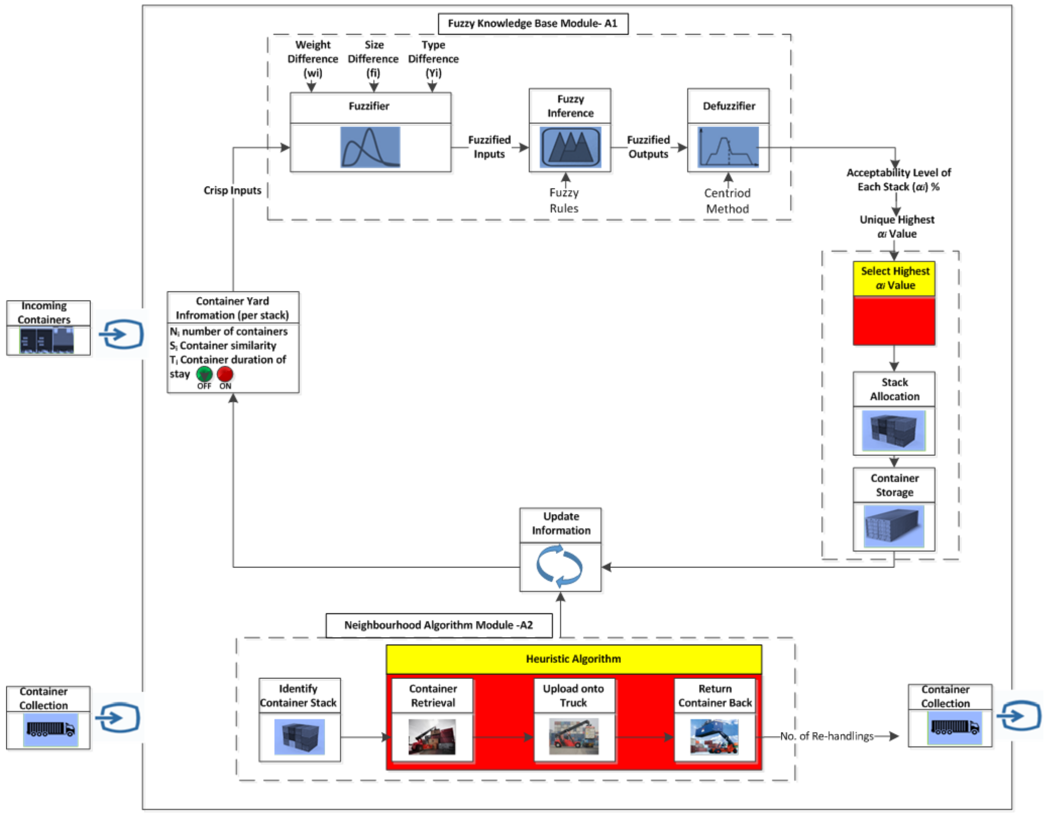


Fig. 3 Core of the 'FKB_CYM' System

An 'ON/OFF' strategy is used to activate/de-activate the duration of stay constraint (i.e. length of stay) for containers, to prevent it being used in the calculation if the value varies significantly over time. However, the imposed constraints play an important role in the storage process as they are providing the system with crisp sets. If the constraints for either the containers or stacks do not match, the acceptability levels for those stacks will be zero.

A container collection process (i.e. retrieval operation) occurs when a truck arrives for collection and the required container stack has been identified for retrieval. The collection operation is optimised using the proposed 'Neighbourhood' Heuristic Algorithm. This algorithm consists of a set of instructions to execute container retrieval, so that the container can be loaded onto the truck. The container retrieval process initiates the re-handing operation if any container is on top of the required one. In the 'Neighbourhood' Algorithm, containers are retrieved and stored as close as possible to the stack that contains the required container. Then, after the required container is loaded onto the truck, all the re-handled containers are returned to the original stack and the yard information updated accordingly.

When needed for departure, the collection process is used to allocate the first available truck based on the minimum waiting time for the container. This reduces the number of trucks used by companies for transporting containers to customers.

The collection process might happen during the storage (i.e. Allocation) operation. When these two operations take place at the same time, the storage operation will then be stopped and the collection operation will be carried out, because the collection process has priority over the allocation process. Once the collection process is completed, then the allocation process will be resumed.

In the following section, the Fuzzy Knowledge Based model components will be explained in more detail.

4.2.1 Fuzzy Knowledge Based Module (FKBM) for Storage Operation

This Fuzzy Knowledge Based Module consists of a number of stages, including the

fuzzification process, fuzzy rule implementation and de-fuzzification stage. These stages will be discussed in detail.
The acceptability level of storage **(α)** is the output from the model, which is an arbitrary value that reflects the value of the current stack in the decision process. This arbitrary value is defined as the acceptability level of an incoming container to the stack i (α_i). For every stack i available in the container yard, a value α is generated based on the input factors and constraints which are discussed below. The acceptability level allows for the assessment of the most suitable stack location for the incoming container. The stack that has the highest acceptability level value will be allocated to store the new container. Three types of factors are considered in this module including:

Factor 1: Number of Containers in the Stack

The first input (N) considered for use in this module is the number of containers in stack i (N_i). The effect of n_i on the output is that more containers currently in the stack will result in a lower acceptability for the new incoming container to the stack i (α_i). If the truck arrival that will collect a container is unknown, the probability of service time being longer, owing to the number of re-handlings that would need to happen for a condensed container stack, would be high. Equally, when the number of containers in a stack is high, the number of re-handlings will be high in that stack. Therefore, input N_i is implemented to consider the number of containers for every stack i.

Factor 2: Similarity of Containers

The second input (S) to be implemented in this module is the similarity of the incoming container to the containers that are already stored in the stack i (S_i). The effect of s_i on the output is that more similarity within the containers of the stack will result in higher acceptability of a new incoming container for the stack i (α_i). The attribute included in determining the similarity of containers is the customer (i.e. containers that belong to the same customers).

Factor 3: Duration of Stay of Containers (DoS)

The third input (T) is the total duration of stay of the containers within the stack i (T_i). The effect of T_i on the output is that the longer the duration of stay of containers in the stack, the lower the acceptability for a new incoming container for the stack i (α_i). Based on work by [6], it is known that the longer duration of stay correlates directly with a higher probability of departure on the next time unit. It is assumed that as time passes, when a container is not collected, the probability of departing in the future is increased, since the length of stay of the containers will be updated. If there is no significant difference between the lengths of stay of containers identified then an 'ON/OFF' strategy is introduced to deactivate and reactivate this factor as appropriate.
In addition to the above, three constraints (W, F & Y) are considered by the FKBM. These include the difference in weight (W_i), size (F_i) and type (Y_i) between the incoming container and the topmost container in the considered stack *i*. W_i is determined by subtracting the weight of the incoming container from the weight of the container in the topmost location of stack *i*. Similarly, F_i & Y_i is determined by subtracting the size and type of the incoming container from the size and type of the container in the topmost location of stack *i*. In this study, three sizes of containers are included which are 20ft (Small), 30ft (Medium) and 40ft (Large) with different types for each size.
In the FKB Module, three stages of operations are performed to identify an appropriate level of container storage, which are described in the following sections.

The Fuzzification Stage

Fuzzification is the stage where fuzziness is introduced to the inputs (control variables) and the output (solution variable). Fuzzy sets and related membership functions are assigned to each variable along with linguistic definitions [16] and a triangular "shape" will be used for all the membership functions.
Firstly, the output variable (α_i) is assigned a triangular membership function with six linguistic variables. The triangular

membership function of the output variable (α_i) is defined with six linguistic variables, and there are six fuzzy sets with their respective membership functions as shown in Fig. 4a. These fuzzy sets include '*Very Low*', '*Low*', '*Medium Low*', '*Medium*', '*Medium High*', and '*High*'.

For the first input variable (N_i), there are three linguistic variables with assigned triangular membership functions. The triangular membership function is defined, three fuzzy sets (linguistic variables) decided for the n_i are '*Low*', '*Medium*', and '*High*'. In Fig. 4b, the membership function of input (N_i) is presented.

The second input variable is the S_i. Similar to T_i and N_i, S_i have triangular shaped membership functions. The linguistic variables (levels) determined for s_i are '*Low*', '*Medium*', and '*High*'. Fig. 4c represents the membership function of s_i.

The third input variable considered in this paper is (T_i). Fuzzy sets have triangular membership functions, there are three linguistic variables (levels) that are selected for T_i; '*Low*', '*Medium*' and '*High*' as shown in Fig. 4d.

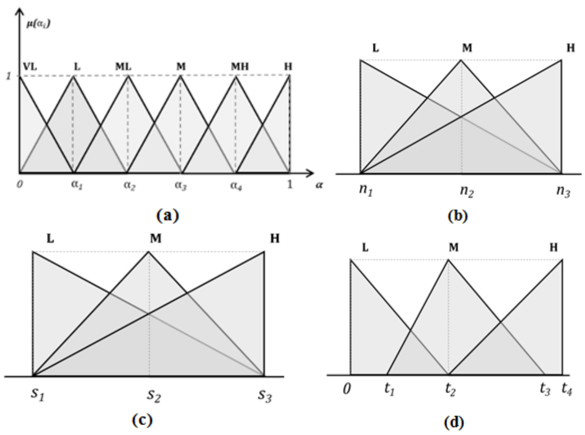


Fig. 4 a) The fuzzy membership function of the output. **b)** The fuzzy membership function of the input factor (N). **c)** The fuzzy membership function of the input factor (S). **d)** Fuzzy membership function of the input factor (T).

The three constraints w_i and F_i & Y_i have only one set called '*Accept*' or crisp membership functions.

The graphical representation of their mem bership functions are presented in: Fig. 5a for W_i, Fig. 5b for F_i and Fig. 5c for Y_i. W_i , F_i *and* Y_i have the same membership function.

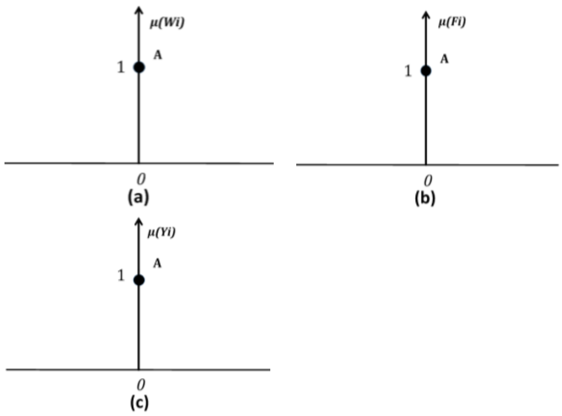


Fig. 5 The defined crisp membership functions of the constraints: **a)** The membership function of the weight. **b)** The membership function of the size. **c)** The membership function of the type.

The Fuzzy Inference- Fuzzy Rules Determination Stage

To define the relationship between the inputs and the output, fuzzy rules have been determined. These rules define the outcome of the interaction of each input variable on the output [17]. For this purpose, the selected input variables (N_i, T_i, and S_i,) and their interactions are analysed and the rules are determined. A total of 27 different rules are identified with respective levels for each input factor. The rules follow the 'If-Then' structure. The rules are decided based on expert opinions, which in this case, are based on the literature, observation and logic regarding the effect each input variable has on the output. In addition, the rules are proposed to reflect the location availability for the incoming container to minimise the number of re-handlings of containers during the retrieval operation. Table 1 provides all the fuzzy rules defined in this study.

Table 1 The defined fuzzy rules

Rule No.	*Ni*	*Si*	*Ti*	*ai*
1	L	L	L	H
2	L	M	L	VH
3	L	H	L	VH
4	L	L	M	H
5	L	M	M	VH
6	L	H	M	VH
7	L	L	H	MH
8	L	M	H	H
9	L	H	H	H
10	M	L	L	M
11	M	M	L	M
12	M	H	L	MH
13	M	L	M	ML
14	M	M	M	M
15	M	H	M	M
16	M	L	H	ML
17	M	M	H	ML
18	M	H	H	ML
19	H	L	L	L
20	H	M	L	L
21	H	H	L	ML
22	H	L	M	L
23	H	M	M	L
24	H	H	M	L
25	H	L	H	VL
26	H	M	H	VL
27	H	H	H	VL

In this stage, an aggregation process is applied. The aggregation includes manipulating the given information in fuzzy format within the defined rules. Upon completing the rules, the aggregation is implemented with the *minimum* operator [18]. Eq. (1) is introduced for the proposed approach for container stack allocation. For each rule j, a truncated value (T_j) is calculated.

$$T_j = min\left\{ \begin{matrix} \mu_{(\widetilde{N})}\, n_i, \mu_{(\widetilde{S})} s_i, \mu_{(\widetilde{T})} t_i, \mu_{(\widetilde{W})} w_i, \\ \mu_{(\widetilde{F})} f_i,\ \mu_{(\widetilde{Y})} y_i \end{matrix} \right\} \qquad (1)$$

Previously, the special condition of W_i, F_i & Y_i is discussed. As our operator is *minimum*, in any rule, if degree of membership of a given value for $W_i, F_i\ and\ Y_i$ are computed to be *0*, the final output for all T_j will also be *0*.

The De-fuzzification Stage

The de-fuzzification step involves the operations to transform the fuzzy output set into a crisp output. There are various methods for de-fuzzification including centre of gravity, mean of maximum and centre average, etc. [19] and [20]. For this study, the centroid (i.e. a specific implementation of the centre strategy of gravity method) is used for the de-fuzzification process.

The strategy finds the centre value (y_j) for each rule by using the truncated value reflected on the output fuzzy sets. Then, the overall centre of gravity is computed. Consider the truncated value T_j and the output $\tilde{\alpha}$ where the rule defines the outcome to be the level-*p*. The centre value is given by the following Eqs. (2 to 5) applied with Fig. 6. Upon finding the corresponding centre values for each of the rules j (y_j) as defined, the crisp output value defined as (y^*) is computed with the centre of gravity method as shown in Eq. (6).

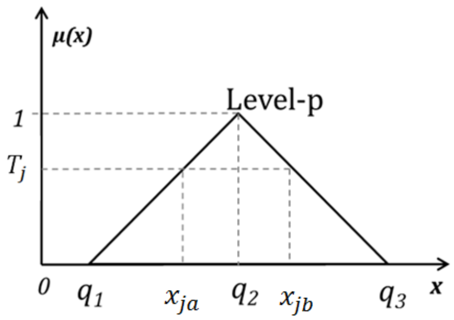


Fig. 6 Truncated value on the output fuzzy set

$$y_j = \frac{x_{ja} + x_{jb}}{2}, \qquad where; \qquad (2)$$

$$T_j = \frac{x_{ja} - q_1}{q_2 - q_1} = \frac{q_3 - x_{jb}}{q_3 - q_2}, \qquad where; \qquad (3)$$

$$x_{ja} = q_1 + T_j(q_2 - q_1) \quad and\ \ x_{jb} = q_3 - T_j(q_3 - q_2) \qquad (4)$$

$$\therefore \quad y_j = \frac{x_{ja} + x_{jb}}{2} = \frac{q_1 + q_3 + T_j(2q_2 - q_1 - q_3)}{2} \qquad (5)$$

$$y^* = \frac{\sum_{j=1}^{l} y_j\, T_j}{\sum_{j=1}^{l} T_j} \qquad (6)$$

4.2.1.1 The Proposed 'ON/OFF' Strategy

As mentioned before, the Fuzzy Knowledge Based Module (FKBM) has three input factors including the number of containers in each stack (N), similarity of containers in each stack (S), and duration of stay for the topmost containers in each stack (T). FKBM has three constraints as well, which are, for the containers in each stack, the weight (i.e. both Full and Empty), size, and type of container. Based on these factors and constraints the acceptability level value of each stack is computed. The stack with the highest acceptability level value is allocated to store the container. To provide the acceptability level values for the stacks, one of the input factors (i.e. Duration of Stay) is provided to the system and this changes dynamically over time. As the duration of stay for containers increases and varies over time, an 'ON/OFF' strategy is proposed to activate/deactivate the duration of stay factor in the system if there is a significant difference in the lengths of stay for the topmost containers in all the stacks. See Fig. 7 for the 'ON/OFF' strategy for the duration of stay factor.

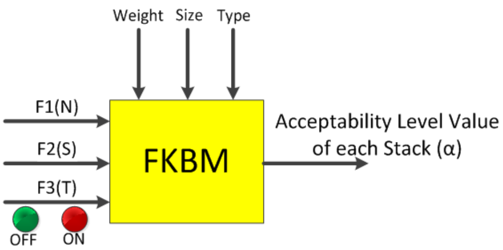


Fig. 7 The 'ON/OFF' strategy of duration of stay factor

When the duration of stay factor is activated (i.e. ON) to the system, all factors (N, S, and T) are used to calculate the acceptability level values for container storage operation. But when the duration of stay factor is deactivated (i.e. OFF) to the system, only the two factors (N and S) are used to calculate the acceptability level values for container storage operation (i.e. for stack allocation).

The decision of how the combination of different linguistic variables for each input factor affect the output (i.e. acceptability level values) is determined by the defined fuzzy rules. For this purpose, 27 fuzzy rules are identified as stated in Table (1), which define the outcome of the interaction of each input factor on the output. When the duration of stay factor is activated (i.e. ON) with the other two factors to the system, all defined rules (27 rules) are fed to the fuzzy inference engine to calculate the output (i.e. acceptability level values for each stack) for container storage operation.

However, when the duration of stay factor is deactivated (i.e. OFF) to the system, the other two factors (N and S) are utilised to calculate the acceptability level values for the stacks. In this case, the number of defined fuzzy rules is reduced to 9 and the acceptability level values are updated as shown in Table 2 below.

Table 2 The reduced fuzzy rules

Rule No.	*Ni*	*Si*	*Ti/OFF*	*αi*
1	L	L	L	~~H~~ M
2	L	M	L	~~VH~~ MH
3	L	H	L	~~VH~~ H
4	L	L	M	H
5	L	M	M	VH
6	L	H	M	VH
7	L	L	H	MH
8	L	M	H	H
9	L	H	H	H
10	M	L	L	~~M~~ ML
11	M	M	L	M
12	M	H	L	MH
13	M	L	M	ML
14	M	M	M	M
15	M	H	M	M
16	M	L	H	ML
17	M	M	H	ML
18	M	H	H	ML
19	H	L	L	~~L~~ VL
20	H	M	L	L
21	H	H	L	~~ML~~ M
22	H	L	M	L
23	H	M	M	L
24	H	H	M	L
25	H	L	H	VL
26	H	M	H	VL
27	H	H	H	VL

In Table 2, when the duration of stay factor is deactivated (OFF), only the rules highlighted in red will be used by the system. In this case only the number of containers and the container similarity factors will be used to calculate the acceptability level values for the stacks in the container storage operation. The highlighted column in green displays the linguistic variables for the duration of stay factors. The highlighted rows in red displayed in the second and third columns are the linguistic variables for the number of containers and container similarity factors. The rows highlighted in red in the last column are the linguistic variables for the output (i.e. acceptability levels). The linguistic variables of the output are updated based on the linguistic variables for the two input factors as shown in the above table.

The Incremental of Container Length of Stay

Once stored in the yard, the length of stay for a container is incremented continually until it departs. The updating process for the container duration of stay must be executed each time a container is stored in, or departs from the yard. This assists the decision of when to store newly-arrived containers with pre-existing ones. After a period of time, each of the containers in the yard will have different lengths of stay. See Fig. 8, which illustrates the differing lengths of stay for containers over time.

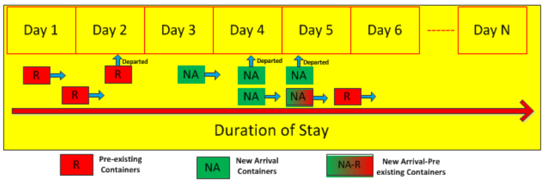


Fig. 8 The Time Incremental for the Container Length of Stay

In Fig. 8, there can be seen a number of pre-existing containers which have been stored in the yard for a period of time (i.e. Containers in Red). When a container arrives to be stored with pre-existing ones, the new containers will be stored based on the acceptability level values obtained from the FKBM as explained in Sect. 4.2.1. While the new arrivals are being stored (i.e. Containers in Green), some of the pre-existing containers may depart. In the passing of time, those new containers will become pre-existing (i.e. Containers in half Green and half Red), the duration of stay for the containers will be updated and each could have its own quite different duration of stay.

The Increment of Length of Stay Approximation Algorithm

Due to the variety of topmost containers in each stack in terms of length of stay, the duration of stay factor is updated and provided to the system dynamically over time. The notations of the length of approximation algorithm are defined below then followed by its steps in details.

DoS: Duration of Stay of topmost container in each stack

t_o: Minimum DoS in hours

t_{Max}: Maximum DoS in hours

d: DoS in day

d_o: Minimum DoS in day

d_{Max}: Maximum DoS in day

t_n: DoS between t_o and t_{Max}

d_n: DoS between d_o and d_{Max}

The steps of the Algorithm are explained below in detail:

Step 1: Obtain durations of stay for the topmost container for all stacks

Step 2: Calculate the possibility percentage for container storage (storage success)

Step 2.1: Approximate the duration of stay (DoS) of container

Step 2.1.1: If $t_o <$ DoS $\leq t_1$, then approximate the DoS to d_o

Step 2.1.2: If $t_1 <$ DoS $\leq t_n$, then approximate the DOS to d_n

Step 2.1.3: If $t_n <$ DoS $\leq t_{n+1}$, then approximate the DoS to d_{n+1}

Step 2.1.4: If $t_{n+1} <$ DoS $\leq t_{Max}$, then approximate the DoS to d_{Max}

Step 3: Check the approximated durations of stay

Step 3.1: Consider the stacks that have the same approximated duration of stay values as possible (success) stacks for storage

Step 3.2: Calculate the number of different durations of stay

Step3.3: Calculate the possibility percentage for container storage (number of different durations of stay / total number of stacks in the yard)

Step 3.4: If the possibility percentage for container storage (success) is ≥ a specific percentage, then go to Step 4

Step 3.5: If the possibility percentage for container storage (success) is < a specific percentage, then go to Step 5

Step 4: Activate the duration of stay factor (ON).

Step 5: Deactivate the duration of stay factor (OFF).

Obtaining the duration of stay for the topmost container in each stack was the first step of the algorithm, then, the next step was the calculation of the possibility percentage for container storage (i.e. the chance of the container being successfully stored in a stack). To calculate the possibility percentage for container storage, the approximation of the duration of stay for containers was necessary. Fig. 9 shows the duration of stay approximation process.

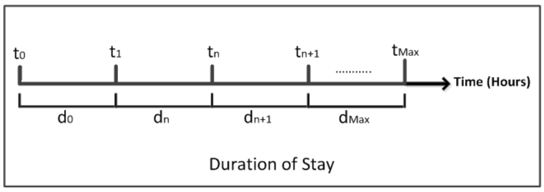


Fig. 9 The length of stay progression approximation

In Fig 9., when the duration of stay is t_1 hours or less, then the DoS is approximated to d_o days, however, when the duration of stay is more than t_1 hours and less than or equal to t_n

hours, then the duration of stay is approximated to d_n days. If the duration of stay is more than t_n hours and less than or equal to t_{n+1} hours, then the duration of stay is approximated to d_{n+1} days. But, when the duration of stay is more than t_{n+1} hours or equal to t_{Max} hours, then the duration of stay is approximated to d_{Max} days. The next step was to check the approximated duration of stay for the topmost container of all stacks, and to consider the stacks that have the same approximated duration of stay values, as possible (success) stacks for storage. This checking was important in order to calculate the number of different durations of stay for containers in the yard. The possibility percentage for container storage was calculated as the number of different durations of stay, divided by the total number of stacks in the yard. If the possibility percentage for the container (success) is greater than or equal to a specific percentage (i.e. provided by the user), then the DoS factor is activated (i.e. ON) to the system. However, if the percentage of storage possibility is less than a specific percentage (provided by the user), then the DoS factor is deactivated (i.e. OFF) to the system.

4.2.2 The Neighbourhood Algorithm for Container Re-handling

When departing from the yard, in order to be retrieved the container has to be topmost in the stack. However, if there are other containers on top of it, they have to be re-handled from the current stack (original stack) to another stack to free the way for retrieval. Once the target container is retrieved, the re-handled containers are moved back to the original stack [21], [22] and [23].
In this study, for the container re-handling operation, the strategy presented by [24] is used when containers need to be moved from the original stack to the next nearest stack. This Algorithm is considered in this paper in order to reduce the total retrieval time for the containers (i.e. the travel distance for the reach stacker). The 'Neighbourhood' Algorithm searches for a stack where there is an available slot. At the same time, in case the stack is not empty, it checks if the topmost container has the same size and type (i.e. constraints of the container) as the container being re-handled. The Algorithm searches for a stack by looking first in stacks immediately next to the original stack (i.e. stacks that are neighbours). This will result in finding the closest stack possible to the original stack that complies with the constraints of the container.
The first step of this algorithm is to search for an available slot in the closest stack to the original stack. If the found stack is empty, the container is re-handled to that stack. If the found stack is not empty, the 'NACR' algorithm checks that the size type and weight of the container being re-handled is the same as the topmost container in that stack., If it is, then the container is re-handled to the stack, if not, the container is not re-handled and the algorithm searches for another stack. If all stacks are full, the container being re-handled will wait until a slot becomes available.

The steps of the algorithm are summarised as below:

Step 1: Search for an available slot in the closest stack to the original stack.

Step 2: If the found stack is empty, then go to

Step 3, else then go to step 4.

Step 3: Re-handle the container to the stack and then go to step 7.

Step 4: If the found stack is not empty, then go to step 6 else then go to step 5.

Step 5: If all stacks are full, then go to step 1.

Step 6: Check the size, type, and weight of the container being re-handled with the topmost container in that stack.

> *Step 6.1: If the container being re-handled has the same size of the topmost container then go to step 6.2, else go to step 1.*
> *Step 6.2: If the container being re-handled has the same type of the topmost container, then go to step 3, else go to step 1.*
> *Step 6.3: If the container being re-handled has the same weight or less of the topmost container, then go to step 3, else go to step 1.*

Step 7: Terminate in case the retrieval operation is completed, else repeat steps 1-6

5. Experimental Study, Results and Discussion

In order to test the behaviour of the developed system, two scenarios are designed which would either consider or not the Duration of Stay (DoS) factor in processing within the system. The proposed 'On/OFF' strategy is applied if the Duration of Stay (DoS) factor is considered. The two scenarios were tested with pre-existing containers in the yard and used the Fuzzy Knowledge Based Module for the storage strategy and the Neighbourhood Heuristic Algorithm as a re-handling strategy.

This section reports the results of testing the developed system for container yard operations against the mentioned real-life scenarios. The performance of the system was evaluated both with and without the DoS factor being used in the calculation. The developed system was coded using the Visual Basic for Applications (VBA) language within MS Office Excel.

5.1 Developed Scenarios

In this section, two scenarios were designed to test the behaviour of the developed system. The first scenario was defined by inclusion of the duration of stay factor where the yard consists a number of pre-existing containers. The second scenario is similar to the first scenario except that the Duration of Stay factor was not considered.

5.1.1 Input Parameters

The input parameters for testing the developed system are presented. In the container yard management system, different resources are utilised including a container yard, a reach stacker, container trains, and trucks. The container yard is divided into 8 bays, each bay consists of 6 rows and each row (stack) holds up to 5 containers. The container yard had a number of pre-existing containers. The pre-existing containers had been stored in the yard for 2 to 10 days.

The number of container trains was 3 to 5 trains a day for 1 week. Each train had 50 to 70 containers with varying weight size and type. The inter-arrival time between trains was 4 hours. .

For each container, the values for parameters used were: Weight: empty or full, Size: small, medium or large, Type: 2 of small size, 3 of medium size and 4 of large size.

The storage time for each container in the first bay from the train side was set to 3 minutes, and the storage time per extra bay was set to 2 minutes. In order to activate/deactivate the duration of stay factor in the system, the duration of stay was set to 40%. When the difference in length of stay of the containers was 40% or above, then the duration of stay factor is activated (ON) otherwise it is OFF. When the required container was at the top of the target stack, the retrieval time was set to 2 minutes, but when there was a container on top of the required one, the re-handling time of that container was set to 1 minute per row and 2 minutes per bay. The containers were picked up and delivered to customers by 7 Third Party Logistic (3PL) companies. Each company had 2 to 20 trucks with 15 customers and each customer had 3 to 10 containers in each train. The travel time for each truck to deliver containers to customer and return was set from 60 to 200 minutes. The results are presented in the next section with figures and comments.

5.2 Results

The results of the two scenarios with the input parameters described above are presented in this section. The performance of the 'ON/OFF' strategy is presented in the figures below showing the container re-handling and retrieval time and the average waiting time per truck.

Fig. 10 shows the total number of re-handlings of containers in both scenarios. As can be seen, the total number of re-handlings is reduced by 5% when the DoS factor was considered by the system. When the DoS factor was considered to make the decision for allocating stacks for container storage, the system allocated the correct stacks for containers to be stored.

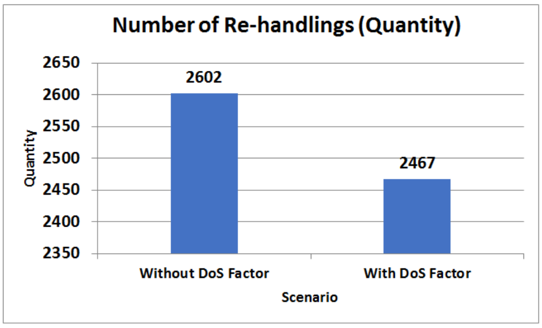


Fig. 10 The total number of re-handlings

Figs. 11a, 11b and 11c show the number of re-handlings per stack, row and bay respectively for both scenarios. Comparing the number of re-handlings achieved in stacks when the duration of stay factor was not considered, the following results can be observed. The highest number of re-handlings can be seen at both stacks 13 and 38 (i.e. 70 re-handlings) while the lowest number of re-handlings can be seen at stack 14 (i.e. 34 re-handlings). This meant that the number of stored containers during the operation at stacks 13 and 38 was higher than the other stacks, while in stack 14 the number of stored containers was low.

However, when the duration of stay factor was considered, the highest number of re-handlings was at stack 47 (i.e. 76 re-handlings) while the lowest number of re-handlings was at stack 30 (i.e. 32 re-handlings). This meant that the number of stored containers during the operation was higher than the other stacks, but in stack 30 the number of stored containers was low.

Regarding the rows, when the duration of stay factor was not considered, the highest number of re-handlings can be seen at row 6 (i.e. 457 re-handlings), while at row 5 the number of re-handlings was the lowest (i.e. 404 re-handlings).

However, when the DOS factor was considered, the highest number of re-handlings can be seen at row 1 (i.e. 430 re-handlings), but at row 4 the number of re-handlings was the lowest (i.e. 372 re-handlings).

The number of re-handlings at bay 7 was the highest (i.e. 345 re-handlings) when the DOS factor was not considered, and it was the lowest at bay 8 (i.e. 308 re-handlings).

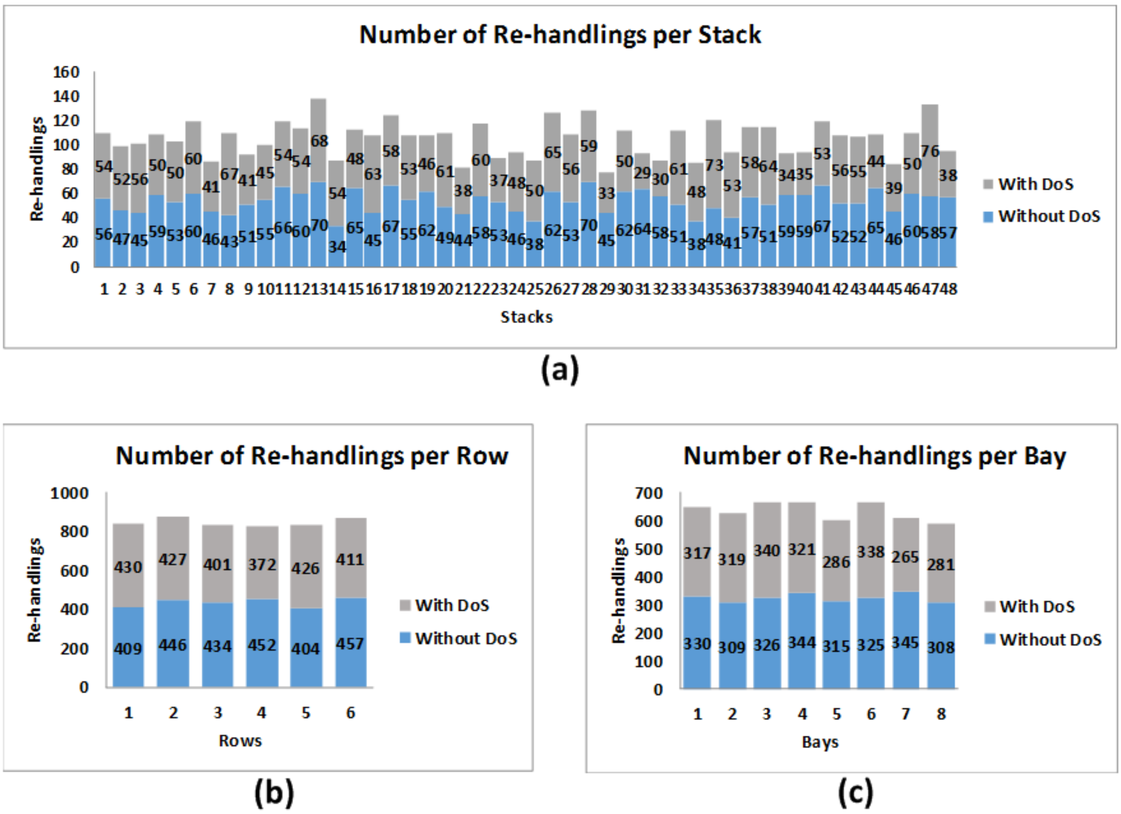


Fig. 11 Number of re-handlings of containers. **a** shows the number of re-handlings per stack, **b** shows number of re-handlings per row, **c** shows the number of re-handlings per bay

Although, when the DOS factor was considered, at bay 3 the number of re-handlings was the highest.

And at bay 7, the number of re-handlings was the lowest. When the DoS factor was not considered for allocating stacks for container storage, the system allocated stacks for containers based on the other two factors (i.e. Number of containers & Similarity of containers in each stack) without taking into consideration the duration of stay (i.e. time spent) for the containers in the yard.

Figs. 12a, 12b, 12c and 12d show the average utilisation of: bays, rows, tiers and stacks respectively. As it can be seen in Fig. 13a, the highest utilisation is at bay 3 when the duration of stay factor was not considered, but it was the highest at both bays (i.e. 2 and 6) when the duration of stay factor was considered.

However, the lowest utilisation is at bay 7 when the duration of stay factor is activated, and it is the lowest at bays 1 and 8 when the duration of stay factor is deactivated.

Fig. 12b shows that the utilisation in rows 3 and 4 is the highest when the DoS factor is activated, but in row 4 was the lowest.

However, the utilisation in all rows is almost the same in both scenarios with a trivial defference between them. Regarding the tier utilisation, tier 1 has the highest and equal utilisation in both scenarios, but it is the lowest in tier 5 in both scenarios. This means that the number of stored containers during the storage operation in tier 1 is higher than tier 5 in both scenarios. With regard to stack utilisation, stack 26 is the highest when DoS factor is activated, and stack 13 is the highest when the DOS factor is deactivated. Although, the difference in stack utilisation is trivial in both scenarios, the number of stored containers during the stoarge operation is almost the same in both scenarios in all stacks. In general, the utilisation of the stacks has been reduced by adopting the DoS factor for the purpose of easier retrieval.

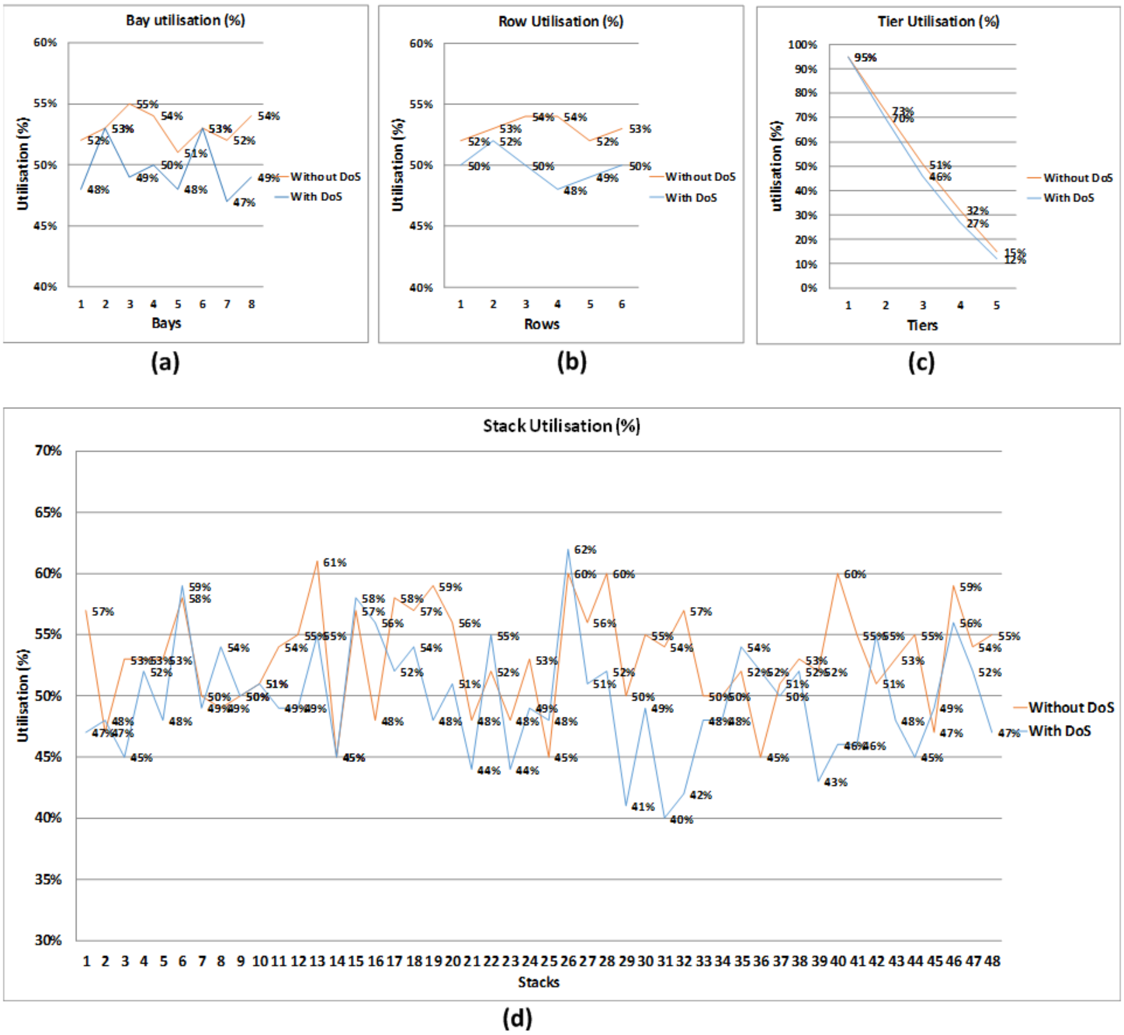


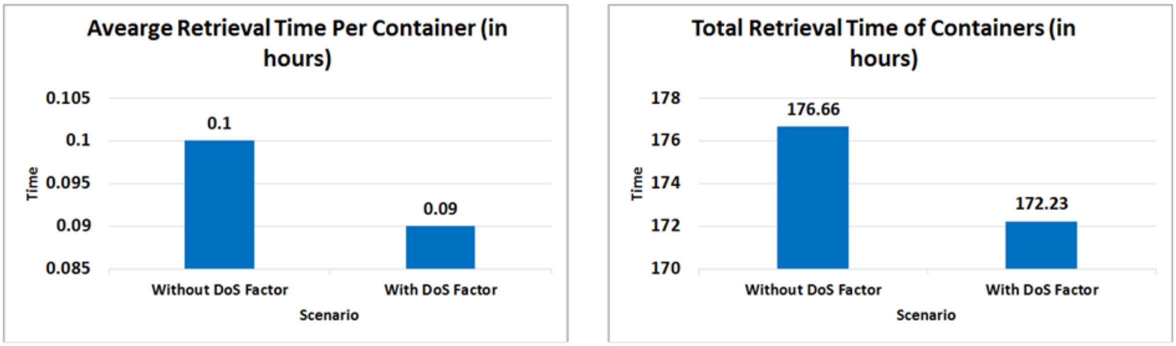
Fig. 12 The average utilisation in the yard. **a)** The average utilisation of bays, **b)** The average utilisation of rows, **c)** The average utilisation of tiers, **d)** The average utilisation of stacks.

All the stored containers during the storage operation are retrieved after a period of time based on a random departure that followed a triangular distribution. Figs. 13a and 13b show the average and total retrieval time per container respectively for all containers for both scenarios respectively.

As it can be seen in Fig. 13a, the average retrieval time per container was minimised when the DoS factor was activated. This was caused when the DoS factor was activated to make the decision for allocating stacks for container storage, the system allocated the correct stacks for containers to be stored and distributed, in which the utilisation of stacks

When the DoS factor was activated, the total retrieval time for all containers was also reduced, which can be seen in Fig. 13b.

After each container was retrieved, the container was then uploaded onto a truck and delivered to customers. Fig. 14 shows the average waiting time per truck. The waiting time of a truck was calculated by the formula: re-handling time of containers + uploading time onto the truck



(a) (b)

Fig. 13 The average and total retrieval time of containers. **a)** The average retrieval time per container, **b)** The total retrieval time of containers

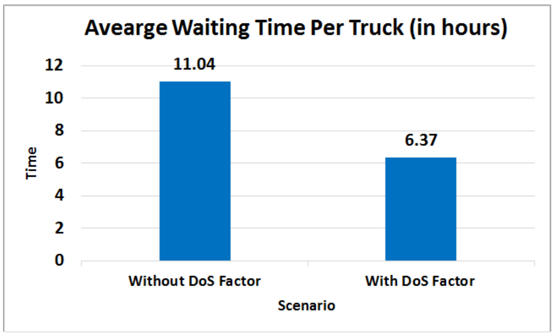


Fig. 14 The average waiting time per truck

As shown in Fig. 14, the average waiting time per truck was 11.04 hours when the DoS factor was deactivated, but when the DoS factor was activated, the waiting time was reduced to 6.37 hours per truck because the retrieval operation for the containers was easier and faster.

In order to retrieve a container underneath other containers, the containers on top of the one that need to be retrieved and re-handled to other stacks. Fig. 15 shows the re-handling time for containers during the retrieval operation for both scenarios.

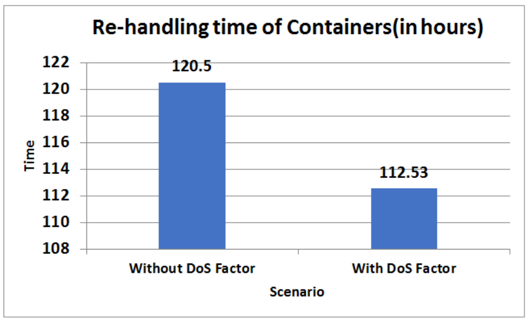


Fig. 15 Re-handling time of containers

When the DoS factor is activated, the re-handling time of containers was reduced by 6.6 % as shown in Fig. 15. This meant that as well as the number of re-handled containers being less when the DoS factor was activated, the number of re-handled containers was also less then the re-handling time.

6. Conclusion and Future Work

In this paper, the problem of storing newly arrived containers in the yard with pre-existing containers was solved using an innovative fuzzy knowledge based system. An ‘ON/OFF’ strategy was developed to respond efficiently to variations in the length of stay factor. A constrained fuzzy knowledge based system was developed for container yard operations management to reduce the number of re-handlings and operational time of containers with an unknown departure time. This system took into consideration the unknown departure time by taking into account the duration of stay for the containers (i.e. the time spent in the yard).

The system was developed for stacking newly arrived containers with pre-existing ones, allowing the operations to be analysed. The results indicated that when a stack that had more containers, the number of re-handlings at that stack was high. When the Duration of Stay (DoS) factor was activated, the results were more robust than when it was deactivated. The operational time and the number of re-handlings were reduced when the DoS factor was activated during the storage operation for containers (i.e. the stack allocation for container storage). This meant that when the DoS factor was activated, containers could be allocated to the correct stacks which led to reduced operational time and number of re-handlings during the retrieval operation.

As a second stage of this work, a Genetic Algorithm (GA) will be combined with the current Fuzzy Knowledge Based System for optimisation purposes. This will be used to select optimal rules from the fired fuzzy rules for each container in the stack, which will be fed to the fuzzy inference engine to assess the stacks and obtain optimised acceptability levels for the container storage operation.

References

1. Galle, V., Boroujeni, S.B., manshadi, V,H., Barnhart, C., Jaillet, P.: the Stochastic Container Relocation Problem. ArXiv Preprint arXiv:1703.04769 . 1-47 (2017)
2. Zehendner, E., Feillet, D., Jaillet, P.: An Algorithm with Performance Guarantee for the Online Container Relocation Problem. European Journal of Operational Research. 259 (1), 48-62 (2017)
3. Chen, L., Lu, Z.: The Storage Location Assignment Problem for Outbound Containers in a Maritime Terminal. International Journal of Production Economics. 135 (1), 73-80 (2012)
4. Zhen, L., Jiang, X., Lee, L. H., and Chew, E. P.: A Review on Yard Management in Container Terminals. Industrial Engineering and Management Systems. 12 (4), 289-304 (2013)
5. Liu, Y., Kang, H., Zhou, P.: Fuzzy Optimization of Storage Space Allocation in a Container Terminal. Journal of Shanghai Jiaotong University (Science). 15 (6), 730-735 (2010)
6. Sauri, S., Martin, E.: Space Allocating Strategies for Improving Import Yard Performance at Marine Terminals. Transportation Research Part E: Logistics and Transportation Review. 47 (6), 1038-1057 (2011)
7. Ries, J., González-Ramírez, R. G., Miranda, P.: A Fuzzy Logic Model for the Container Stacking Problem at Container Terminals. eds Int. Conf. Computational Logistics. Springer. 93-111 (2014)
8. Lawrence, W., Chwan-Kai, K.: A Comparison of Stacking Efficiency for various Strategies of Slot Assignment in Container Yards. Journal of the Eastern Asia Society for Transportation Studies. 4(1), 303-318 (2001)
9. Bazzazi, M., Safaei, N., Javadian, N.: A Genetic Algorithm to Solve the Storage Space Allocation Problem in a Container Terminal. Computers & Industrial Engineering. 56 (1), 44-52 (2009)
10. Hu Li, B., Xiao, T., Zhang, L., Mi, W., Yan, W., He, J., Chang, D.: An Investigation into Yard Allocation for Outbound Containers. COMPEL-the International Journal for Computation and Mathematics in Electrical and Electronic Engineering. 28 (6), 1442-1457 (2009)
11. Jang, D., Kim, S. W., Kim, K. H.: The Optimization of Mixed Block Stacking Requiring Relocations. International Journal of Production Economics. 143 (2), 256-262 (2013)
12. De Castilho, B., Daganzo, C. F.: Handling Strategies for Import Containers at Marine Terminals. Transportation Research. B 27 (2), 151- 166 (1993)
13. Kim, K. H., Kim, H. B.: Segregating Space Allocation Models for Container Inventories in Port Container Terminals. International Journal of Production Economics. 59 (1), 415-423 (1999)
14. Kim, K. H.: Evaluation of the Number of Rehandles in Container Yards. Computers & Industrial Engineering. 32 (4), 701-711 (1997)
15. Huynh, N.: Analysis of Container Dwell Time on Marine Terminal Throughput and Rehandling Productivity. Journal of International Logistics and Trade. 6(2), 69-89 (2008)

16. Zadeh, L. A.: The Concept of a Linguistic Variable and its Application to Approximate reasoning—I. Information Sciences. 8 (3), 199-249 (1975)
17. Zadeh, L. A.: A theory of approximate reasoning. Machine Intelligence. 9, 149–194 (1979)
18. Zimmermann, H.J.: Fuzzy Set Theory and its Applications. Springer Seience+Business Media, New York (2001)
19. Zadeh, L. A.: Fuzzy sets. Information and Control. 8, 338–353 (1965)
20. Zimmermann, H.J.: Fuzzy Set Theory — And Its Application. Kluwer, Boston (1991)
21. Imai, A., Nishimura, E., Papadimitriou, S., Sasaki, K.: The containership loading problem. International Journal of Maritime Economics. 4, 126–148 (2002)
22. Imai, A., Sasaki, K., Nishimura, E., Papadimitriou, S.: Multi-objective simultaneous stowage and load planning for a container ship with container rehandle in yard stacks. European Journal of Operational Research. 171, 373–389 (2006)
23. Güven, C., Eliiyi, D. T.: Trip Allocation and Stacking Policies at a Container Terminal. Transportation Research Procedia. 3, 565-573 (2014)
24. Ji, M., Guo, W., Zhu, H., Yang, Y.: Optimization of loading sequence and rehandling strategy for multi-quay crane operations in container terminals. Transportation Research Part E: Logistics and Transportation Review. 80, 1–19 (2015)