MASTER THESIS

MSC INDUSTRIAL ENGINEERING AND MANAGEMENT

Analysis and Improvement of baggage handling in CPH Airport using simulation tools

Beatriz Nebot Gracia
Supervised by David Pisinger

*Academic year 2018/*2019

TABLE OF CONTENTS

1 INTRODUCTION

1.1 Air transport and Copenhagen airport

Air transport is becoming a more popular way of transport in the recent years. More than four million people travelled in 2017 from more than 10000 airports all over the world and using an air network of 15 million of kilometres. [1]

European and Chinese airports are the largest contributors to global aviation growth with a 21.5 and 21.6% increase in the number of passengers in 2017, respectively. [1]

According to ACI [2], this increasing trend in air transport will continue, and passenger traffic in Europe will be more than doubled by 2040 (+112.3%), accommodating 4.36 billion passengers.

One of the busiest hubs in Northern Europe is Copenhagen Airport, which has been in continuous growth since the financial crisis of 2009. The passenger traffic was up to 29,177,833 travellers in 2017 and this number is expected to increase for 2018. [1]

Regarding the growing passenger scenario, airport capacity will probably be a bottleneck for the aviation sector. On top of some major building projects are being constructed in Copenhagen Airport (CPH Airport) and other investments to manage the passenger increase, the need for efficiency and optimising the existing facilities is clear.

1.2 Baggage Handling System

One of the important resources is the *baggage handling system* (BHS). With the increase in air passenger travel, it also comes an increase in the total luggage. Therefore, it is critical to ensure that no bottlenecks occur, while minimising the baggage travel time and maximizing throughput performance measures.

The *baggage handling system* (BHS) is the conveyor-based network that transfers baggage from its origin to its destination, including the sorting, storing, distribution and security procedures such as the X-ray scanning of all the bags.

The process that luggage follows in the BHS depends on the origin and destination of the luggage, and according to that criteria baggage can be classified in *inbound baggage*, *outbound baggage* or *transfer baggage*. The bags whose origin is the airport, bags from passengers departing, are considered in *outbound* handling. *Inbound baggage* handling is responsible for the bags of passengers arriving at its final destination airport. And the bags from passengers transferring from an aircraft to another in the same airport is *transfer baggage*. At each airport, the outgoing baggage flow is the input from the check-in counters plus the input from transfer baggage.

A simple scheme of the luggage flow is found in the following image:

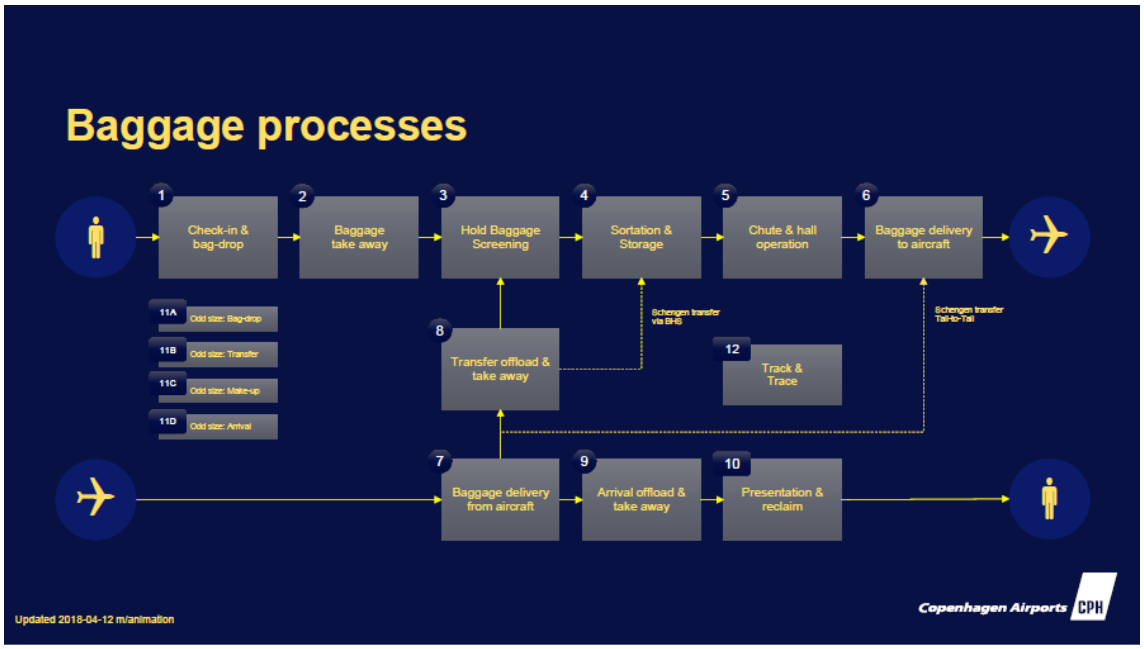


Figure 1. Baggage processes in an airport. Source: Copenhagen Airports

Now, the process each type of baggage follows will be explained in detail, giving details for Copenhagen Airport.

To start with, inbound baggage is the input from the passengers departing from the airport. Passengers go to the check-in counters assigned to their flight, their information is registered in the system and a barcode is attached to every item before entering the BHS. In the last years, *self-service drop check-in counters* have been introduced, where the passengers process their bags themselves, resulting in a lower check-in time and usually less queues for this type of counters. The bags from each of the counters are transported in a take-away band. The same take-away band gets the baggage from a group of check-in counters. In Terminal 2 in Copenhagen Airport the group consists of eight desks but can be multiplied in case the entrance from the take-away band to the BHS is blocked or interrupted.

After the check-in area, bags proceed through X-ray security scanning machines to ensure that no dangerous goods or threats are loaded into the aircraft. If something suspicious is found in this first screening level, the bag goes through a second screening level. If there is a threat identified in the bag in the second level, it will go to a manual handling area for dangerous goods. Otherwise, if no threat is identified in the second level of screening, the bag will go again into the main line, along with all the bags that passed the first security screening area. In the main line, the bag barcode is scanned by an *Automatic Tag Reader* (ATR) and then, the control system assigns its corresponding destination. The destination can be either a chute or an *early baggage storage* (EBS).

The output pier or *chute* is the baggage unloading zone for a specific flight. It is usually assigned two hours and a half before flight departure, excepting overseas flights, when it is assigned three hours and a half before departure. A visualization of chutes can be found on the following image:

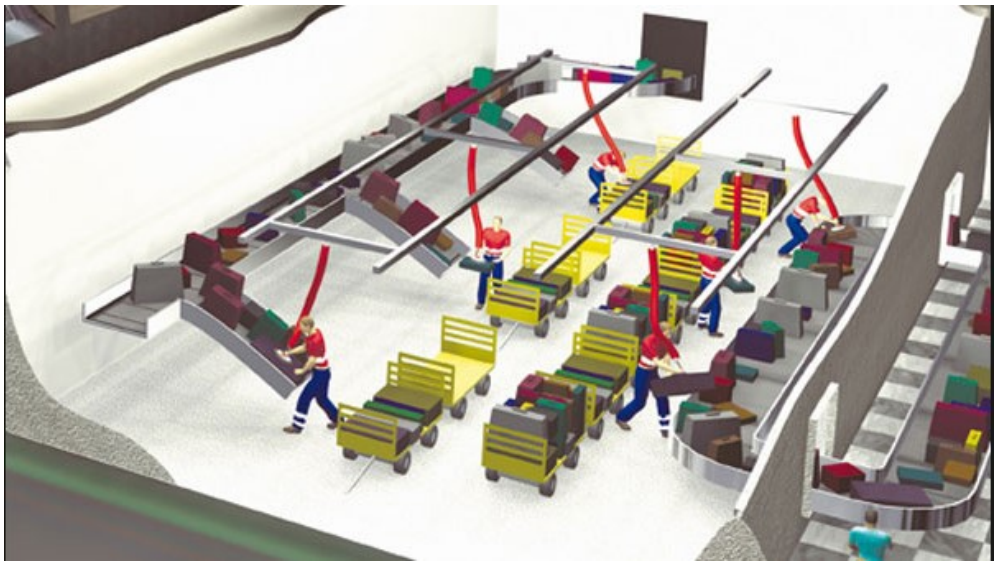


Figure 2. Visualization of chutes. Source: http://www.vaculex.com/solutions/airports/

Therefore, if the bag arrives within that time window, it will be pushed to the chute. But, in the case that the bag arrives before that time window it will be pushed into the early baggage storage for later re-sorting.

A visualization of an early baggage storage with belts is shown in the following image:

Figure 3. Visualization of Early Baggage Storage. Source: http://en.logan-ksec.com/web-60-155.html

The early baggage storage is the area where bags arriving before their chute time window are kept. In CPH airport the buffer area is divided into 14 belt lines with different length, so each of them can keep a different quantity of bags stored. When a bag arrives before the time window established for its chute, it is assigned to one of the early storage belts.

When baggage is being *sorted* (or re-sorted after being in a EBS) it is loaded into a tray carousel system that lifts the tray and unloads the bag when it reaches the chute it is allocated to. A visualization of the sorting system is shown in the image:

Figure 4. Visualization of the sorting system in the BHS. Source: http://www.yyc.com/en-us/calgaryairportauthority/projectsprograms/CalgaryAirportAuthority/NewInternationalTerminal/Blog/TabId/785/ArtMID/1885/ArticleID/2/YYCs-New-Baggage-Handling-System.aspx

Another input to the sorting system is transfer baggage. Transfer baggage is unloaded from the aircraft and stored into the facility if the connecting flight unloading zone has not been assigned. Depending on the bags origin, transfer baggage is scanned on the first security level as the check-in counters, on a separate security screening for transfers or not scanned (in the case of Schengen transfers).

At many airports, some transfer baggage also follows special processes, for example bags with short time connections. These bags have a direct transfer identification and are transported directly from the landing flight to the departing flight bypassing the standard procedures for transfer baggage. They are unloaded from the aircraft and stored in separate containers (without being included in the sorting system), with or without a manual resorting before being loaded into the connecting aircraft.

When the baggage has arrived at its final destination, bags are unloaded from the aircraft and taken to the passenger luggage claim area. As it is a much simpler process and it is not so strictly time constrained, it is almost never the bottleneck of the baggage handling system.

Since baggage handling involves many labour processes, it is one of the main cost drivers in ground handling. Therefore, the need for increasing the efficiency in the process is clear. On the other side, quality is an important requirement in baggage handling in the airports. Poor quality in luggage handling is regarded as one of the most important factors in customer satisfaction [3] with the airport, even considering that some of the problems may be caused by other players, such as ground handlers or airlines. Furthermore, the consequences of problems in baggage handling may not only impact the passenger, but may also impact a chain of events, resulting in blocking aircraft parking, usage of handling equipment for a longer duration than planned, or on the worst case, causing terminal and roadway congestion due to a delay in processing the baggage [3].

1.3 Purpose and contributions

Regarding Copenhagen Airport BHS, there are many areas in the baggage handling system that could be studied to find, if possible improvements or, to confirm that the current performance is indeed efficient. Two of them, identified by the Operations Management department of CPH Airport will be covered in this thesis.

The first of them is the check-in counters area. It is important to ensure that the maximum throughput is obtained for any input of luggage at the service check-in desks. Different intensity baggage input scenarios will be studied, with the purpose of finding a strategy that maximizes the total input for a group of check-in desks while making a fair distribution of the input of each of them (similar waiting times).

The second point of the baggage handling network that could be improved is the early baggage storage area. Each bag goes to one belt or another depending on remaining time until its flight departs. The objective will be to improve performance distributing the belts depending on different criteria or time segments. It will also be investigated whether the efficiency of the BHS can be improved by using the EBS as a buffer.

These two problems will be explained in further detail in following sections.

1.4 Outline of the document

After the present section introducing the Baggage Handling System and the two problems that will be addressed in the thesis, a more detailed explanation of each of them will be given in section 2.

Once the reader is familiar with the focus of the thesis, a literature review will be presented in section 3. As there is no precedent literature for neither the check-in counter algorithm and the Early Baggage Storage analysis, the literature review will be focused on simulation case studies within the airport field and related problems. It will firstly introduce the most relevant studies on simulation within the terminal airport field, to secondly continue with research related with the baggage handling system. To continue, research about specific parts of the BHS will be summarized, and problems related to the present thesis focus will be highlighted. That is the reason why there will be a specific section showing all the topics related to the check-in counters window algorithm, and another one for the early baggage storage similar topics. Finally, literature on simulation will also be included to reference the main guidelines and procedures on the methodology.

Although both problems intend to analyse and find, if possible, improvements to the baggage handling system, each of them can be studied independently. The check-in desks and the Early Baggage Storage are areas physically separated one from the other, and the result of the improvement of one of them, despite improving the overall performance, does not have a direct impact in the other. Then, they will be studied and analysed separately, and one section will be dedicated for each of the problems.

The fourth section will be dedicated to the simulation methodology, justifying why it is appropriate and needed for the problems presented. And the simulation study of the check-in

counters window allocation algorithm and the improvement of the early baggage storage will be presented separately in sections 5 and 6, respectively. Included in the last-mentioned sections will be the results extracted from each of the studies will be presented in along with an analysis discussion of the results obtained and a possible improvement of performance, if applicable. A comparison of the system behaviour with each of the strategies or algorithms proposed will be shown, discussing if any possible improvements were found.

On the last section of the document, general conclusions and recommendations the luggage handling system in CPH airport will be given, as well as an overview of what further research could be done in this area.

2 PROBLEM DEFINITION

2.1 Check-in counters

The check-in counters are the main input entrance for the baggage handling system. Physically, they are distributed in groups, being each desk an input to the main belt or take-away band, following a “comb shape”. The physical distribution of the check-in counters in Copenhagen Airport terminal 3 is shown in the following image.

Figure 5. Check-in counters at Copenhagen Airport terminal 3

Each group of eight counters loads baggage in the same take-away band, what can be represented in the following figure, where the “comb shape” can be seen:

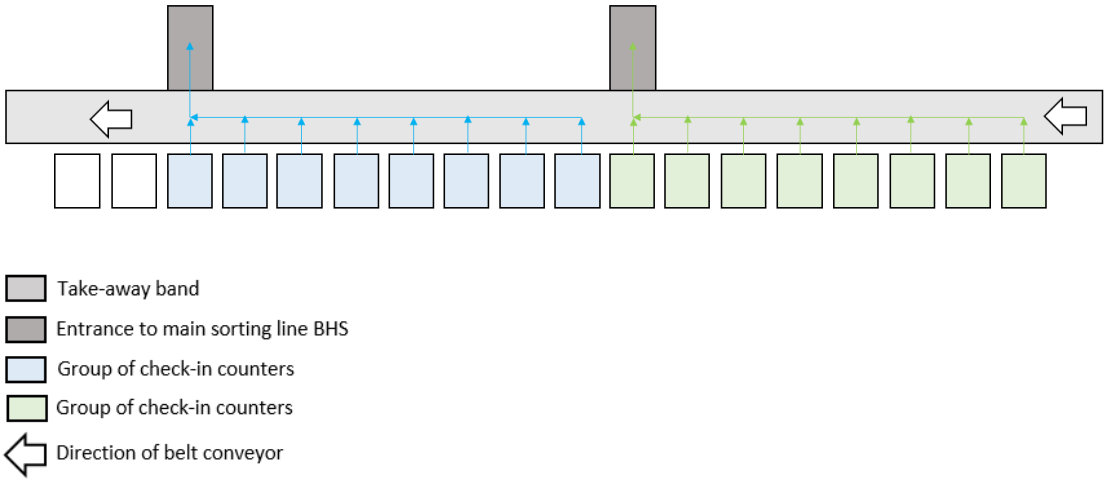


Figure 6. Representation of flow in check-in counters.

The introduction of several self-service check-in bag counters has resulted in a higher utilization of the check-in counters capacity for shorter periods. One consequence is that the last check-in counter, relative to the belt direction, has a higher waiting time to include a suitcase in the belt conveyor, thus creating queues and longer waiting times for the rest of passengers. In the future,

additional self-service bag drops are expected to be installed, so this problem will be even a bigger challenge.

The objective aims to compensate and reduce the waiting times at the check-in counters belonging to the same take-away band, while maintaining a high utilization in the periods of high luggage load.

The problem that will be analysed is how to address or which strategy to follow for inputs of each check-in counter to the group take-away band following a fair distribution with the maximum throughput. The methodology will develop several algorithms for allocating luggage *windows* for take-away bands.

The *window* algorithm bases on the idea that the conveyor belt is divided in spaces or windows of fixed size moving in the belt direction. Therefore, a window is a space where a bag can be loaded. The window will be empty or free until a bag is loaded and occupies the space.

When referring to a fair distribution, it is understood that it is a fair decision criterion regarding to passengers. Therefore, baggage waiting times for each of the counters should be as equal as possible and their maximum waiting times should not differ considerably. Another success criteria that the algorithm must accomplish, following internal policies of the airport, is a *take-away time* of 20 seconds for 96% of the luggage.

It is understood by *take-away time* the time it takes since a bag arrives to the check-in counter until it is loaded in the tape belt. Note that time is measured from the time it arrives to the counter so, in case there is queue for the specific counter, the time waiting in the counter will not be included in the take-away time.

2.2 Early Baggage Storage

2.2.1 Early Baggage Storage allocation

The *Early Baggage Storage (EBS)* is the area where bags without *chute* allocated wait until its it is allocated.

A *chute* is the physical space where baggage will remain before it is loaded into the aircraft. It is a steep slope that connects the main sorting belt of the BHS with a lower area, used to keep the bags until loaded into containers ready to be carried into the aircraft. Each flight is assigned to a chute within a time window, so all the baggage that arrives to the BHS within the chute time window will go to the allocated chute. If it arrives before that time window, it will go to the early baggage storage.

In CPH airport the EBS is divided into 14 belt lines with different length, so each of them can keep a different quantity of bags stored. The question is: In which EBS line should the bag be stored when it arrives before its output pier time window? And, what would be a good criterion to empty each of the belts to make sure that the bags arrive to its respective chutes on time?

Constraints

On the one side, the time constraint is clear. If a bag is hold in the early baggage storage, it is because at the time it entered the system, its corresponding chute was not allocated, therefore it entered the BHS in the "early" time window. Then, if a bag is needed to be loaded to a chute, the earliest time the belt can be emptied for the items to be re-sorted is when the specific chute time window for the bag is opened (2:30 h before departure, 3:30 h for overseas flights).

But, it is important to not forget that the purpose of the baggage handling system is to ensure that inbound baggage arrives on time to its respective aircraft. And as follows, if a bag is hold in the early baggage storage, it must be released on time to be sorted and loaded in its allocated output pier. This means that it must arrive to the chute before closing time, which is 20 minutes (30 min for overseas departures). The latest time the belt could be opened is approximately 5 min before the closing time of the pier, so there is time for the sorting system to identify and allocate all the items in the system to its corresponding destination pier. Time constraint is represented in the following figure:

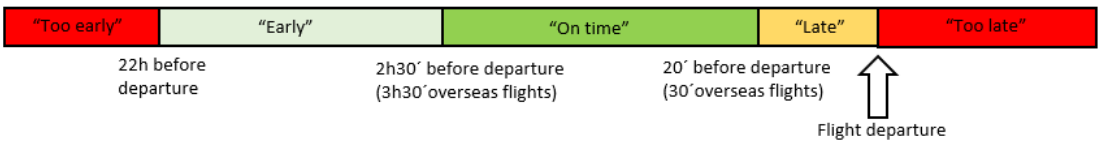


Figure 7. Representation of Early Baggage Storage time constraints

On the other side, there are physical system constraints. The fact that it is a tape belt gives an estimated capacity for the number of bags that can be stored, there is no exact positioning of the bags in the belt. The reason is that bags have different sizes and shapes, occupying a different length in the early baggage belt, on the top of the fact that there is no established space separating them. Hence, the storage capacity of each of the bags is an estimate.

Another important characteristic of the belt storage system is that it must be fully emptied to be re-sorted. As no positioning bag track is kept inside the EBS, only the EBS where a bag is stored is known, the belt must be fully emptied. In the best scenario possible, if just a part of it could be emptied, a margin error in length should always be considered.

It is important to note that every time one of the EBS is emptied for re-sorting, the system load is increased. As a consequence, making it desirable to diminish the number of times the emptying operation is performed.

Therefore, the solution must accomplish the assurance of on-time bags, at the same time as avoiding stress of the sorting machine and finding an even distribution of the bags along the belts.

Summary

Summarizing the early baggage allocation problem, the objective is to analyse and find, if possible, a strategy to allocate the baggage to the transfer belts and a decision criterion to when emptying the belts so that it improves the nowadays performance of the EBS, considering physical system constraints and time constraints.

2.2.2 Early Baggage Storage used as a buffer area

In the recent years with the opening of several direct lines among Europe, Copenhagen Airport has been less used for both domestic passengers and as a transfer via [1]. In the year 2017 transfers continued having negative growth, falling to 8.4%.

The fact that there are less transfers is reflected on the low utilization of the early baggage storage out of peak season. On an average day, the number of bags in EBS keeps stable with a moderate peak around midday, which doesn´t reach the full capacity of the system. It can be seen in the following graph:

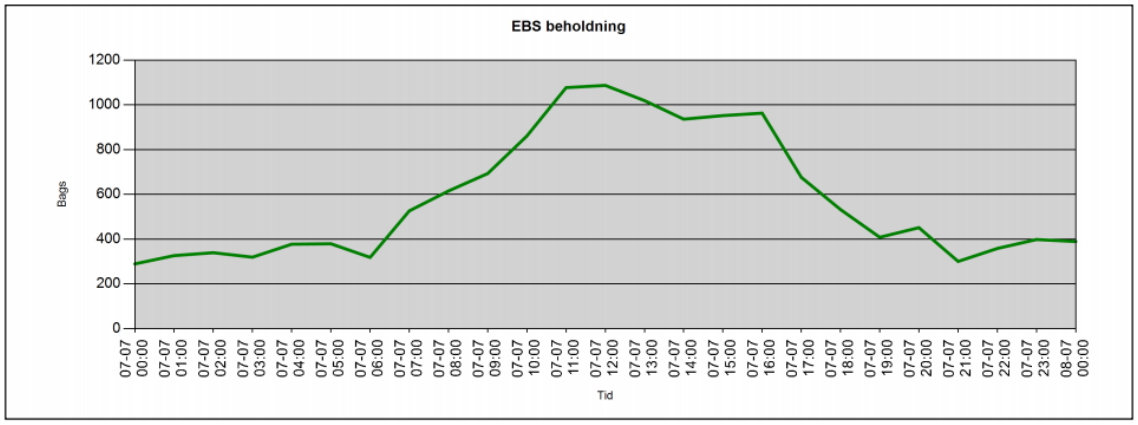


Figure 8. EBS utilization during an average day. Source: Copenhagen Airport

During all non-peak season in Copenhagen airport, the early baggage storage utilization does not reach its maximum capacity, resulting in some of the belts even empty at some points of the non-peak season.

The fact that the Early Baggage Storage does not reach its designed maximum capacity, along with the facts that Copenhagen Airport is very close to the city with limited expanding capacity, and the increase need in chutes arises the question of if it could be possible to use the early baggage storage space as a buffer. In other words, using it as a *buffer* would mean to use it as a storage space where bags would be kept before being moved to a chute, taking into consideration that chute window opening times are lower.

On the baggage handling system, the number of chutes is limited, and it is sometimes not enough for the demand of flights. Having a buffer would result in a good solution for keeping the bags for a specific flight in a space, that is not necessarily a chute. Logically, all the bags would have to be moved to its chute before flight departure, so they can be loaded into the aircraft, but the chute time window would be reduced, and it would decongest the chute allocation.

The present study will look into the possibility of using the Early Baggage Storage as a buffer, and how long chute opening window could be reduced without affecting the performance of the system.

3 LITERATURE REVIEW

3.1 Baggage handling system

Baggage Handling Systems have been widely studied and documented in literature due to the tremendous development of air transport industry in the recent years. While some research has been focused on the overall performance of the system, other studies have analysed specific areas of the network system.

Global approach to airport terminal simulation

Among the studies of the overall performance of the system, Verbraeck and Valentin in [3] created a simulation library, commercially available in the simulation software eM-Plant, that intends to create a standard set of simulation building blocks for modelling an airport terminal. With the mission of being the most adaptable, the library is divided into four types of blocks: infrastructure, passenger, passenger behaviour and control building blocks. This separation between infrastructure -static- and passenger behaviour -dynamic-, plus the infrastructure and control differentiation, makes the standardization easier. However, for studying a specific part of the terminal, as it is the case of the study in this document, blocks cannot be reused for being general. Similar study was applied to Toronto International Airport using ARCTERM simulations in [4] to model passenger capacity and behaviour.

Moving to one area of the terminal, a complete analysis of the passenger check-in was conducted on Buffalo Niagara International Airport [5], including the data collection process, simulation model and the scenario analysis to find improvements with different procedures and dispositions.

One of the most common studied problems is the airport baggage sorting stations problem. Ascó et al. [6] with the objective of improving planning decisions analysed the geometry, flight service time, station capacity and distance to the aircraft gate. Their effort contrasted with [7], which developed an algorithm for assigning flights to areas in the airport. The impact of grouping flights with close departure times was analysed and it was shown that it generated peak periods alternated with non-peak periods. Therefore, it resulted in low BHS utilization on the low load periods and created congestion in the peak times.

Baggage Handling System

Regarding baggage handling systems, the specific physical disposition and procedures of the airports have been studied. Examples of microscopic discrete-event based models are Riga Airport [8] and Santiago International Airport in Chile [9]. Both cover the process from the check-in until baggage loading. Taking a closer look into the areas in which the model is divided in the first case are check-in area, additional security check area and equipment area; while in the second the model is divided in counters, behaviour and baggage loading area. As it can be seen in the previous simulation model results - [8] and [9]-, the identification of bottlenecks when studying the whole system is clear, and the areas where improvements could be made are identified. In the case of the present document, it was Copenhagen Airport who identified the areas that needed improvement. The reason for choosing them was the repeated experience of high queue times in the furthest check-in desks in the belt direction and high overload of the sorting system in peak periods due to emptying the early baggage storage.

Specific areas within the BHS

Other research has been conducted in some specific areas of the BHS.

For example, different studies were conducted in the X-ray scanning after that 100 percent of bags scan was a requirement [10] and its impact was analysed in [11]. Consequently, with the objective to adapt the system to the new requirement without diminishing its performance, it resulted in research to find a cost effective layout to place the scanning [12] as well as in dynamic routing strategies study [13].

Other areas studied include the tracking of the bags with RFID ([14], [15]) or the scheduling and routing selection of destination code vehicles [16].

When modelling baggage handling systems, the work of Le et al. [17] provides a standard set of measures to assess the performance of the luggage network system through discrete event simulation. It offers a foundational set of metrics to follow for input modelling and data acquisition. Besides, it presents techniques for output simulation analysis which include steady-state conditions, warm-up and cool-down times, simulation running time, system recovery time, number of replications, as well as it discusses the most common performance analysis measures. These techniques for steady-state conditions will be taken into account for the present study of check-in counters discrete simulation.

One line of study is an accelerated-time simulation of the BHS. It is useful to test in advance the baggage system design for the airport. The known model ATISBAT in [18] is a reference when testing designs due to its flexibility and adaptability to various physical distributions of existing airports or possible future designs. It models the whole process of luggage from its registration in the check-in counters to the aircraft.

It is important to note that in many simulation studies, as it is the case of [18] and many other, the luggage process from the aircraft to the baggage claim is not considered. Since all the luggage from the same plane is carried to the baggage area, it does not require classification and does not have any time constraints, it is a much simpler process.

Specific areas within the BHS: Check-In counters

Regarding the input of the system, Le et al. [19] developed a loading algorithm to obtain near optimal input operating conditions for each of the check-in inputs, while preventing blockages in the inputs and minimising baggage travel time. On the case of Copenhagen Airport check-in counters this input will be estimated based on the real behaviour, and the focus will be on how to obtain the maximum throughput with the input stated. In the case of the Early Baggage Storage, the input will be real data obtained from the Airport.

The result that simulation is needed for check-in counters problem is also seen in [20], where simulation output is compared to traditional queuing theory. The study highlights that peaks in arrival times are the reason for not using analytical methods in the check-in counters. In other words, queuing times cannot be predicted with queuing theory due to the peaks in arrival times. The case is also tested with Amsterdam Airport Schiphol. The principle of queuing theory will be applied to the present thesis in the estimation of arrival rate of bags for check-in counters problem.

The check-in problems are many times self-driven by the flight schedule. That results in the problem of defining a pattern and profile of input of bags into the system. This topic was covered in [21], [22], [23] with the conclusion that the check-in bag arrival can be estimated with an exponential distribution.

Regarding the problem addressed in this document within check-in counters, the process can be assimilated to conveyor merges. Johnstone et al. [24] lead an investigation into the design and control of merging bottlenecks of conveyor-based baggage handling systems. Their study enveloped two merging control algorithms with the physical layout of the merge. The first of the algorithms was the common implemented windowing method. It is based on the idea that there are windows of fixed size moving through the merge. The control was based on placing a bag in each window. Contrarily, the second algorithm was based on windows of variable length. Their results showed that that the layout influences the throughput of the merge for both algorithms and that the second performed significantly better when the layout was in a preferred position. Due to the simplicity of the windowing method, it is broadly used in airports, including CPH Airport. The present study will be based in this method and will find the best heuristic possible to maximize the windows used.

Other authors [22] focused on analysing the performance of conveyor networks with merging configuration following the *queuing theoretic model (QTM)* by Arantes and Deng [25] with the objective of maximizing overall throughput. The results validated the robustness of the QTM for the study and the fact that the conveyor speed does not have a significant impact on the performance of the network. Similar conclusion was obtained by Xu and Piratesh [21], showing that there was no significant change in throughput after the conveyor reached a certain speed.

Specific areas within the BHS: Early Baggage Storage

Regarding the transfer belts problem, Taiwan Taoyuan International Airport [26] modelled a similar study. The formal modelling language *System Modelling Language (SysML)* is used to model the baggage handling system. In the mentioned airport, when the suitcases have entered the system and have passed the security screening, they are transported to a main sorter, where each item is identified by a barcode scanner. Then, they are assigned to either a buffer zone or to its unloading zone. The unloading zones are usually assigned to flights two hours prior to its departure. After a time, the luggage in the buffer will be moved to the main conveyor to be sorted again. The buffer areas in Taiwan Airport have a time window of 30 min assigned. When there is no free space in the buffer, the baggage travels to two carousels and waits for ground staff handling. The simulation scenarios in the study explore different time windows to obtain a trade-off between two important and inversely related objectives: average number of items re-entering the sorting system and number of manually operated baggage in the carousels. From this study, the importance of the time threshold is stated and thus, it will be considered in the present study. However, the present study does not have the same capacity in each buffer area, which makes the optimization of the early baggage storage more complex.

3.2 Simulation

The nature of airport terminals is complex due to the interrelation of many factors and the stochasticity of its behaviour. That is the reason why simulation has been broadly used in research within the airport resources.

In modelling, Rossetti´s six-phase methodology [27] is used as a reference and it established the following steps for creating and developing a valid simulation model: problem formulation, simulation model building, evaluate and iterate, and implementation. Although it is intended to help with the commercial software Arena, its process to build and implement a simulation model can be applied to any commercial software. This methodology will be followed to develop the simulation models in the present study.

The second reference most used related to simulation is Law´s [28] simulation and modelling analysis guidelines, which gives guidelines to create a simulation model, including all statistics and variables needed to make it represent the real behaviour.

Lastly, for discrete-event simulation Banks et al. is used as a reference [29].

4 SIMULATION EXPERIMENT

4.1 Introduction

In this section the choice of methodology and simulation software will be explained. Both areas of the baggage handling system that were identified in CPH Airport for improvement will be analysed using this methodology. Each of them will have its own simulation model that will be explained in detail in its own problem section and its respective construction and test of the model will developed according to Rossetti´s methodology [27].

Both models are simulated as discrete event-based used a microscopic simulation approach. A justification of why this methodology is used is also provided on this chapter, as well as the election of the simulation software.

4.2 Simulation type and software

SIMULATION AND WHY SIMULATION

Simulation can be described as the mimicking of a real-world system or process. The real system, object of the study, is represented by a simulation model so that the representation mimics the behaviour and qualities of the system [27]. To represent the real behaviour of the system in a simulation model, logical blocks connected among themselves are used, using mathematical, logical and symbolic relationships among them. This representation is usually the next best thing to observing the real system and it enables the study and experimentation with the internal interactions of the system.

The simulation model is used to generate artificial history and data of the system, and the observation and analysis of that artificial history added to the creation of variant scenarios, to draw inferences concerning the operating characteristics of the real system. The reason why these experiments are not conducted in the real system is the impossibility or impracticality, often due to cost or time.

It is clearly seen that simulation tools are needed in this thesis study cases to improve the performance of CPH Airport due to the impracticality and inconveniency of the experiment realization in the real system. The baggage handling system cannot decrease its performance to test possible designs to improve throughput. Note that analytical methods cannot be used either due to the high complexity of the system.

MICROSCOPIC SIMULATION

There are three levels of detail in which simulation can be implemented: macroscopic level, mesoscopic level and microscopic level.

In macroscopic models the level of detail is low, high level of aggregation is used in the model and details are omitted. It makes them suitable for supporting strategic decisions primarily.

On the other hand, microscopic models make attention to details, require much more information to be implemented and they usually incorporate many considerations.

Mesoscopic modelling is a mixture between the last two approaches of simulation, and it consists in models with a high level of detail, but with a low level of flow behaviour and flow interactions description.

As the level of detail required to analyse the shown problems is high, a microscopic approach is used in the simulation models.

DISCRETE-EVENT SIMULATION

There are three major methodologies used to build simulation models: system dynamics, discrete event modelling and agent-based modelling, being the last two the most used in a microscopic level.

To start with, system dynamics assumes a high abstraction level and is used for strategic problems such as market adoption rates and social process dependency.

Discrete-event simulation represents systems that can naturally be described as a sequence of operations, which makes it useful for manufacturing, logistics and healthcare field. The object of the simulation are distinct entities ("items" or "things") that move on a sequence of discrete, separate events [30], [29]. Regarding to time steps of the simulation, in discrete-event interval between events is dependent on when events occur, and model only recalculates when events occur, and the items are automatically routed to the first available branch in the model.

Contrary, agent-based time steps are sometimes constant or defined by time intervals while the routing of the individual components of the system (agents) is dependent on their behaviour. The behaviour of each of the agents is defined, together with the connections between them and the environmental variables.

Therefore, in the case of the baggage handling system, where the suitcases follow established operations in a conveyor network, a discrete event approach will be used.

SIMULATION SOFTWARE

A problem specific simulator will be used for each of the baggage handling study areas. The simulator for the check-in counters window algorithm will be developed in C++, and the Early Baggage Storage simulator will be developed in Python. The reasons for choosing to develop a specific simulator for the areas are:

- Discrete event modelling can be done using variables and numbers to associate with its respective states.
- Math and logic operations can be calculated within the simulation.
- Output representation of choice. Output files can be done in text or csv files, while other visual representations can be done after simulation ends.
- C++ code has a high flexibility.
- Inclusion of many built-in functions in Python which gives a lot of flexibility and ease of use. Among them, the inclusion of lists to represent objects.

- Large number of variables/states to take into consideration: commercial software with embedded code limits the possibilities regarding to variable tracking changes.
- Change in number of physical objects for different what-if analysis: in commercial software physical objects are represented by “drawing” each of them. A change in the number of physical objects would require a change in the global logic of the system and many manual steps for the what-if analysis.
- Simulation running time is considerably faster than commercial software, where visualization is showed for all running time.

The only disadvantage of using a simulator in C++/Python is that the visualization of the process is not possible, and variables need to be printed to show and follow the behaviour of the system. At the same time, not having a visualization could also be presented as an advantage as simulation time can be analysed and its results will be obtained faster than if visualization of the process was presented.

Note that running time is the time that it takes for the program to read the code lines, which will always be faster than the simulation run in a visualization process, even if it is an accelerated simulation.

5 CHECK-IN COUNTERS PROBLEM: FINDING A WINDOW ALLOCATION ALGORITHM

5.1 Conceptual model description

To ease the understanding of the simulation model, the real system will be described in this section. An explanation of the assumptions made will also be included.

When a passenger arrives to a service check-in counter for his flight, his/her data is registered in the system and the luggage is dropped into the conveyor belt. The check-in counters are physically distributed in groups of eight that lead to the same take-away band, which finally feeds the baggage handling system not visible to the customer. It is shown in the following figure:

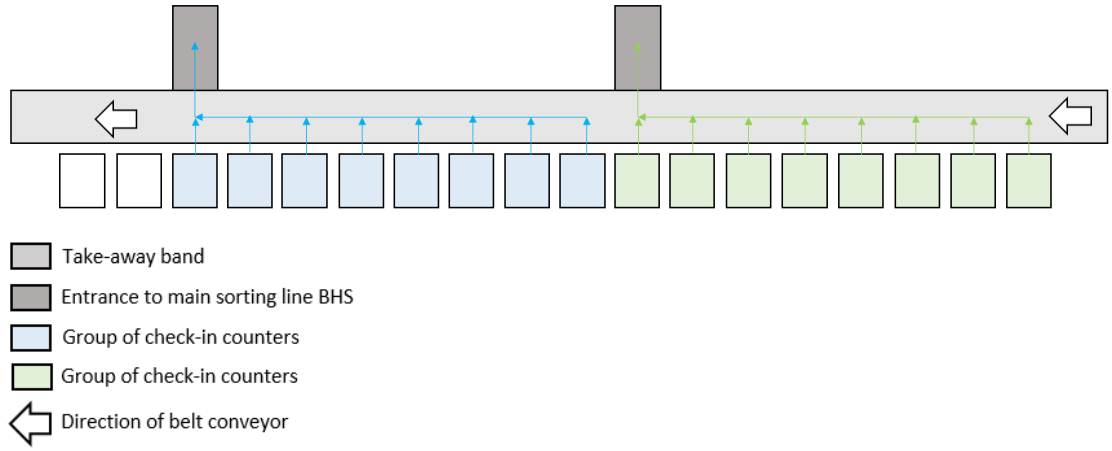


Figure 9. Normal grouping of counters in Copenhagen Airport. Groups of eight check-in desks.

Therefore, in case there is technical problem in one of feeding lines of the take-away bands, the system would need to load the luggage of 16 counters in the same take-away band (shown in the figure). And the same way, for 24 check-in desks in case two consecutive take-away conveyors are congested or collapsed.

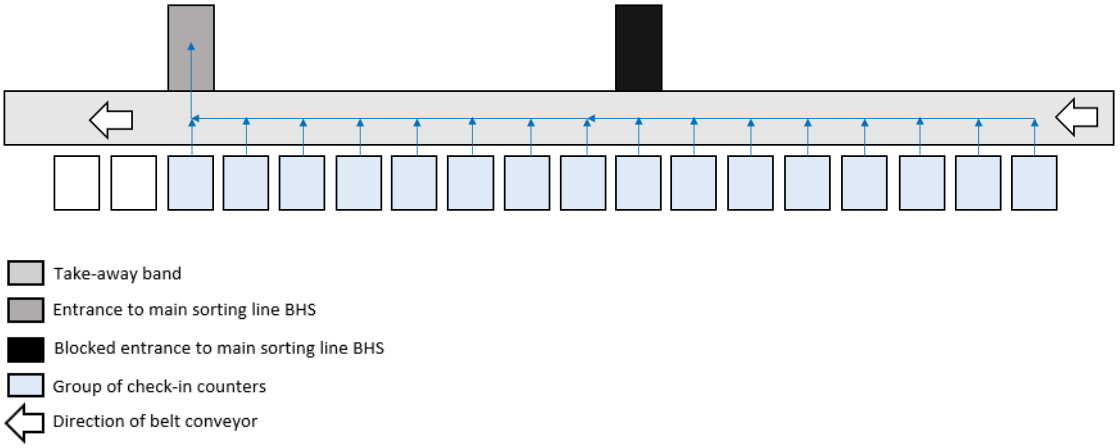


Figure 10. Grouping of 16 check-in counters loading in the take-away band.

When there are eight or more loading baggage to the same take-away belt, the question of ordering and prioritizing which belts should load first arises. The present strategy followed in Copenhagen Airport BHS is FCFS (First Come First Served). In other words, the item waiting in the check-in counter that first finds a "free space" in the conveyor belt is loaded, independently of its relative position to the rest of the counters.

It is understood that a "free space" is an empty window following the fixed-size window algorithm. It is assumed that there is a pulse attached to the take-away band. A window corresponds to a space in the tape where a bag can be placed. In this simulation it is assumed that a window is 1.5 meters. The belt runs at 0.5 m/s, so a window passes a desk every 3 seconds, resulting in a tape capacity of 1200 windows per hour.

The assumptions that will be taken into consideration in the simulation model will be the following:

- Conveyor speed will not be considered as a variable for the study. It has been proven in [25] that conveyor speed does not have a significant influence on the throughput of the network and [21] also showed that throughput did not change considerably after conveyor reached certain speed.
- The problem will not model the arrival of luggage to the check-in desks. In the model it will be assumed that the luggage appears or not at the check-in desk at a specific point of time, allowing the study to focus on finding a suitable heuristic for the baggage input relevant for the study.
- Each of the heuristics proposed will be tested for different values of luggage arrival intensity. Studies [21], [22], [23] on check-in and queuing problems at airport terminals have shown that the best check-in arrival time is modelled using an exponential distribution. Therefore, choosing different values of baggage arrival intensity, the behaviour of the real system will be studied, as different intensities cover different points along the exponential distribution behaviour.
- It is assumed that a single bag fits in the window-size of 1.5 meters, special luggage sizes are checked-in in designated counters, so special cases will not be included in the study.
- Check-in information processing time is considered negligible. This previous proves will be disregarded to focus on the window allocation algorithm. Due to the stochasticity of the arrival of bags to the system, check-in information time could also be included once the bag has arrived at the desk.

There are also other considerations regarding the data available:

- Copenhagen Airport uses a PLC to control the baggage handling system. Therefore, any strategy that can be programmed into a PLC is a possible implementation into the system.
- Physical changes in the disposition of the baggage handling system will not be considered. Eight check-in counters will be grouped and load to the same belt conveyor. So, the study will consider the combinations of 8, 16 and 24 desks loading to the same take-away band.
- In the desks focus of study (check-in counters in terminal 3 in Copenhagen Airport) there are sensors to detect the presence or absence of a bag in the counter. Therefore, the presence or not of a bag is known, as well as the waiting time for the bag, that can be

calculated by the system using the data from the sensors. This also means that the waiting time is known for the bag at the counter, but not for the rest of the bags that are on the queue. It is not even known how many bags are in the queue and the waiting time of each one.

To sum up the process considering the assumptions made, a bag would be the agent to study, that goes through a process of the following events or steps:

1. Arrival of a bag to the designed counter
2. Queue if the check-in counter is occupied (contains another bag). If the check-in desk is free, then it goes directly to step 3.
3. Bag is moved to the check-in desk.
4. Bag waits until there is a window for it
5. Bag is loaded into the conveyor belt
6. Bag is transported with the conveyor belt in its direction
7. Bag arrives to the main line in the BHS

The process described can also be seen in the activity diagram included in Appendix 9.1 Check-in counters: window algorithm problem. Activity diagram.

5.2 Model building and data collection

The model has been implemented in C++, according to the conceptual model explained and the assumptions and considerations before established.

The code for the simulator can be found in the appendix 9.2. “Check-in counters simulator in C++”. The model can be separated into different parts:

1. **Initialization of the problem:** inclusion of library files, definition of parameters and variables used in the simulation. Among the parameters *stress rate*, *counters* and *horizon* are included.

   The *stress rate* is a parameter that will change the overall performance of the system, understood as a percentage of the maximum capacity. Thus, a value of “100” for stress rate indicates that the system is working equal (or close, as it is dependent on stochasticity) to its designed capacity of 1200 bags per hour. From the process perspective, it changes the arrival rate of luggage.

   Counters is the parameter that indicates the number of check-in desks loading luggage to the take-away band. It will take the values 8, 16 and 24.

   Horizon is the parameter related with the simulation time that indicates the number of iterations of the process. One iteration is a time window move from one position to the following one in the conveyor belt. The simulation time will be 500 hours, so the horizon will be 600000 iterations as shown:

$$horizon = 500\ h * \frac{60\ min}{1\ h} * \frac{60\ s}{1\ min} * \frac{1\ iteration}{3\ s} = 600000\ iterations$$

where one iteration corresponds to 3 seconds as:

$$conveyor\ speed = \frac{window\ length}{iteration\ time}$$

Being the window length and conveyor speed established values:

$$iteration\ time = \frac{window\ length}{conveyor\ speed} = \frac{1{,}5\ m}{0{,}5\ \frac{m}{s}} = \ 3\ s$$

2. **Definition of counter arrival rates:** defines a different and random arrival rate for baggage at each of the counters.

   As it has been shown in research [21], [22], [23], the arrival rate of luggage to a counter before flight departure follows an exponential distribution. This means that few passengers check-in their luggage as soon as the counters just opens and more and more arrive as time passes by, with the number of passengers arriving growing exponentially.

   If the arrival rate of a counter follows an exponential distribution, now regarding the situation of a group of counters at a point in time assigned to different flights (some counters could be assigned to the same flight for big aircrafts), it will be seen that each of them is in a different point of their own exponential distribution. In other words, each of them would have a different arrival rate of luggage independent of the rest of the counters, excepting counters for same flight and category (first class counters or bag drop would also have different arrival rates even for the same flight assigned).
   To model the stochasticity of having an arrival rate for each counter independent of the rest of them, a random function will be used. It will assign an arrival rate from 0 (desk closed) to 100% to each of the counters. Furthermore, the arrival rate of all the counters will be changed every hour to properly model time stochasticity such as the fact that each of the counters changes arrival rate in its own exponential distribution and that counters could be used for new flights, closed or joined flights in the same counter.

3. **Choice of strategy to use for the window allocation algorithm**. When running the simulation, the heuristic that will be used is chosen. Each of them will be explained in the following section.

4. **Luggage arrival to the counters**. Reached this point, the process itself is represented by iterations.

   In each iteration, two cases are considered for luggage arrival: a bag arriving to the check-in desk or not. On the top of it, there is a queue that models the bags which must wait to be checked-in because there is another bag in the counter at that moment.

   Then, if a bag arrives and there is no bag in the counter or in the queue, it will directly arrive to the counter to be checked-in and loaded into the take-away belt, if possible. It will be loaded or not in the tape belt depending on the heuristic established. In general, a bag is loaded into the conveyor belt when there is a free window and it is allocated or

reserved to the specific counter. In the case that a bag arrives and there is already a bag in the desk waiting to be loaded, the arriving bag will wait in the queue until the counter is free and available. One item from the queue will moved to the check-in counter when it is available again, which in the real behaviour follows the First In First Out queuing theory.

To model the stochasticity of bag arrival, the probability that a bag arrives to a specific counter will be established by the formula:

$$P(bag\ in\ the\ i^{th} counter) = \frac{arrival\ rate\ of\ i^{th} counter}{\sum_{i=0}^{counters} arrival\ rate\ \ of\ i^{th} counter} * \frac{stress\ rate}{100}$$

Being the probability of a bag in the window in the take away band the stress rate:

$$P(bag\ in\ the\ takeaway\ band) = \frac{stress\ rate(\%)}{100}$$

The deduction of the formula above comes from the generalization of the formula if all the counters had the same arrival rate:

$$P(bag\ in\ the\ i^{th} counter) = \frac{1}{number\ of\ counters} * P(bag\ in\ the\ takeaway\ band)$$

And from the sum of probabilities that states that the probability of a bag appearing in the take-away band is the sum of the probabilities of that bag appearing in any of the check-in desks:

$$P(bag\ in\ a\ window\ in\ the\ takeaway\ band) = P(bag\ in\ the\ 1^{st} counter) + P(bag\ in\ the\ 2^{nd} counter) + \cdots + P(bag\ in\ the\ n^{th} counter)$$

Note that considering bags or passengers in the simulation would not affect the representation of the real behaviour of the system. It is because passengers usually have none, one or two items with them, and being the passenger with just one item the typical case. The other two cases would be included in the simulation by the stochasticity itself, as the creation of two consequent bags in consequent iterations in the same counter would model the passenger with two items of baggage. Likewise, the passenger with no luggage is included in an iteration where no luggage arrives.
It is not needed to mention that no luggage can arrive to closed check-in counters, whose arrival rate was defined to be zero previously.

5. **Assignment of windows**. Windows will be reserved or allocated to a specific counter before they reach the counter's position. Window allocation is what differentiates the algorithms and what explains the performance of the baggage handling system input. The fact that windows are assigned means that a counter would only be able to load a

bag if the window is reserved for itself. And if the bag cannot be loaded, it will wait in the counter until it can be taken in the tape belt.

6. **Loading of bags**. It is the event when the items in the desk are loaded in the band, the physical movement of the bags is represented.

7. **Conveyor movement**. It represents the physical movement of the belt in its designed direction, and in the simulation model, it is modelled by the movement of one position of all windows. That is, every window will move one position in the belt direction. Having in mind the fact that the length of every window (1.5 metres) matches with the length of the space designed for the counter, there will be one window "in front of" every counter, and when moving, a window will be placed in front of the next desk following the take-away band direction.

8. **Statistics**. Once finished all the process, output data is printed in a file, so the behaviour of the system can be showed.

5.3 Experiment planning and description

Different heuristics or strategies will be implemented to study the performance of the baggage handling system. The explanation of each of them and the expected performance and comments is given in this section.

UTOPIA

This first strategy will be used as a reference. When used, all luggage that arrives at the check-in desks is moved to the take-away band, independently if there is a bag or not on the belt, as if bags could be placed one on the top of the other, making a pile as high as needed.

As its name indicates, it is an algorithm impossible to perform in the real system, but it is useful as an upper bound for luggage throughput. When making experiments, knowing what the maximum throughput would be depending on system stress rate can be used to compare how good the rest of the strategies perform. In other words, for a specific stress rate, it is useful to compare the throughput of a strategy compared to its upper bound.

FIRST COME FIRST SERVED (FCFS)

It is an assignment method commonly used when supply or resources are expected to not be enough for its demand. It consists in assigning the resources in the order of arrival of people or requests. Therefore, the person who arrives first and finds a free window in the conveyor belt will be able to place his/her bag first.

It is the strategy currently used by Copenhagen Airport and its performance will be studied to be compared to any other possible strategies that could improve it.

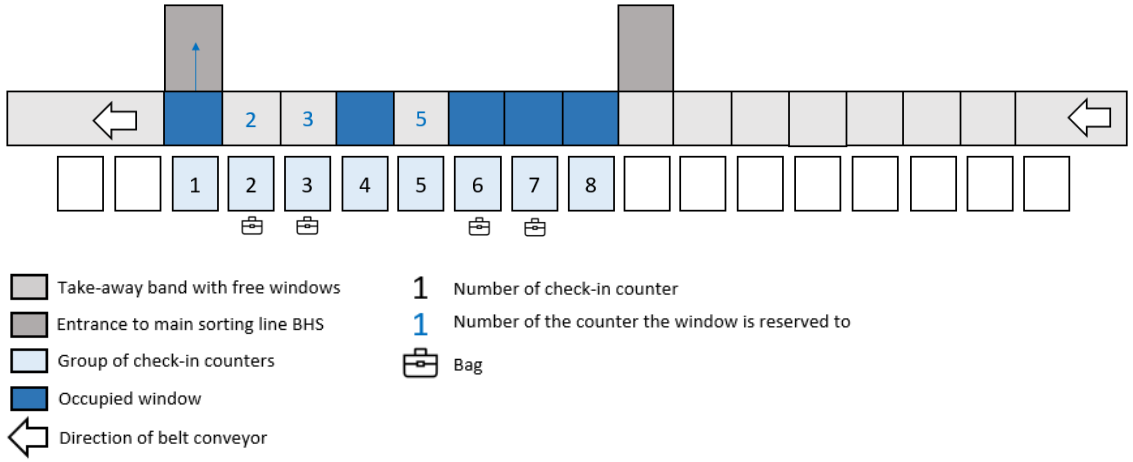


Figure 11. Representation of FCFS strategy

FAIR STRATEGY

Considering that one of the objectives is obtaining an approximately equal waiting time for all the counters, the fairest distribution possible would be that a counter has the right to place a bag in the tape conveyor every N^{th} window, being N the total number of counters connected to the same take-away band. That is, every N windows there will be a window "reserved" for a specific counter and it can either place or not a bag in it.

As one can easily deduce, the disadvantage of this strategy is that there will be empty windows if counters do not place bags in the windows when it is the time to do so. The performance of this strategy and the overall throughput will be analysed to see if this disadvantage is considerable or not. The reservation of windows is shown in the figure:

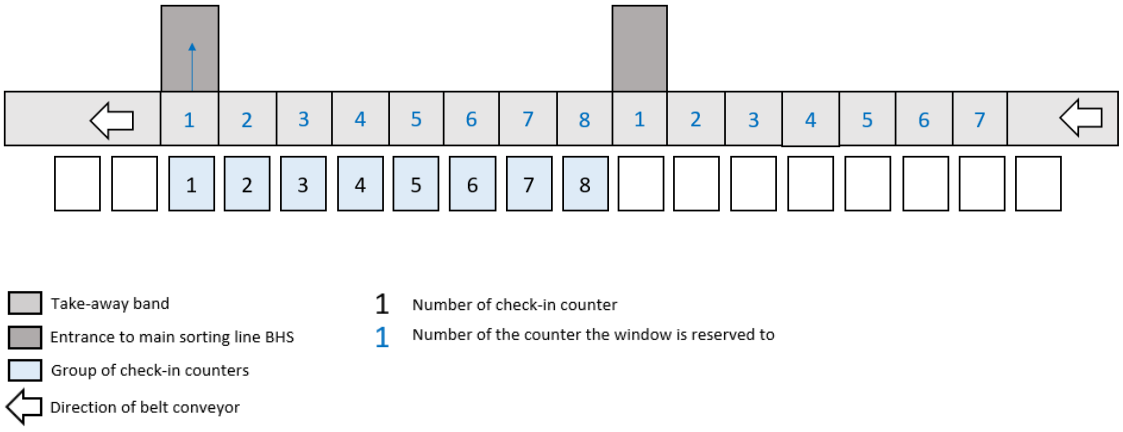


Figure 12. Representation of fair strategy.

FORWARD RESERVATION

With this algorithm, windows in the take-away band will be reserved by the counters when a condition is met. There can be different forward reservation strategies depending on the prioritization of counters reserving.

A general rule for forward reservation strategies is that a desk can only reserve a window if it has a bag being checked-in. So, in case there is bag in a counter, the desk can reserve a free window (a window that is empty and has not been reserved yet) for it. On the contrary, if there isn´t any bag on the counter, the counter cannot reserve a window.

Another important factor about reservation is the number of windows that are in the system, and the amount of them that can be reserved. As explained, the length of take-away is divided into sections of the same length, and each is considered a window. Then, there will always be a window in front of every counter. So, the windows that are in front of the desks will be available for reservation. Furthermore, on the top of the windows physically in front of the counters, more windows can be reserved, windows that "will come" in the direction of the conveyor but are not in front of the desks yet. It is important to note that when two or more counters have the same priority for reservation, the choice will be random among them.

FORWARD RESERVATION: THE FURTHERST COUNTER RESERVES FIRST

Given a conveyor belt direction and a FCFS strategy, the counters at the beginning of the line in the direction of the conveyor belt will be more probable to load baggage into the tape rather than the ones which are furthest, since the closer counters "see" more empty windows than the furthest. To try to diminish this difference, this strategy will be tested. Not the counters without a bag cannot reserve a window. An example is given in the figure:

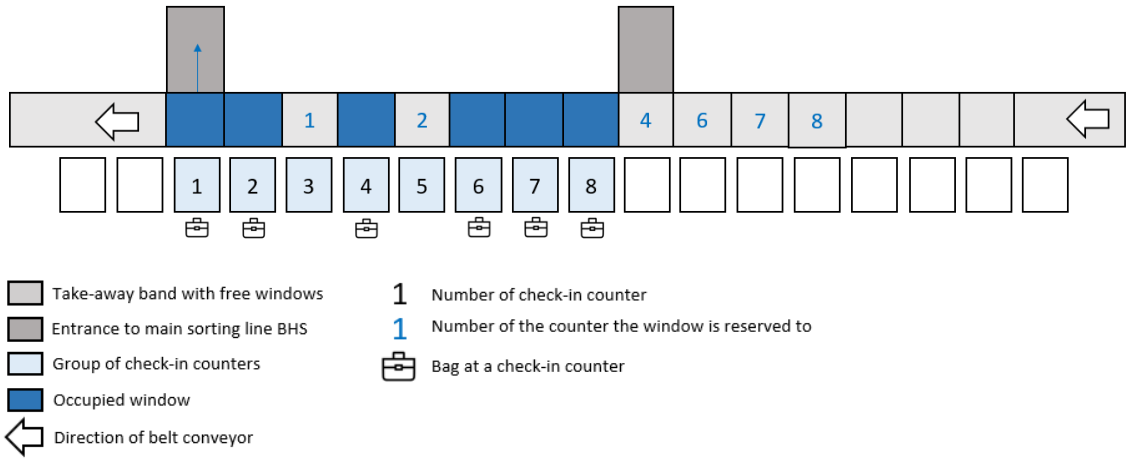


Figure 13. Representation of forward reservation: the furthest counter reserves first strategy

FORWARD RESERVATON: THE HIGHEST AVERAGE WAITING TIME RESERVES FIRST

This strategy is implemented to prioritize the reservation of the counters with highest average waiting time. The average waiting time of every desk is a dynamic value that changes every iteration. It is the division among the total waiting time and the total number of bags already loaded into the BHS.

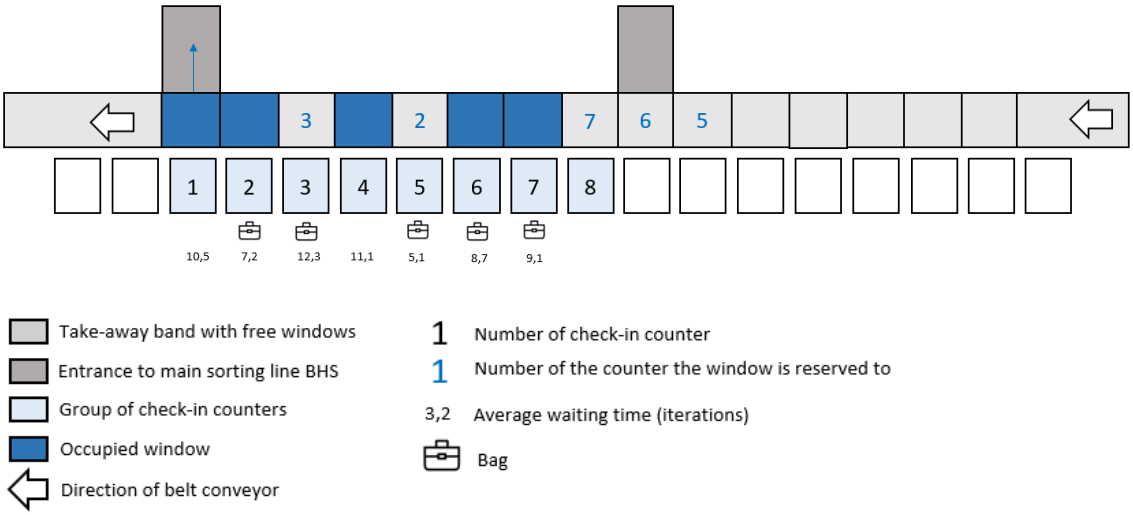


Figure 14. Representation of forward reservation: the highest waiting time reserves first strategy

FORWARD RESERVATION: THE HIGHEST MAXIMUM WAIT RESERVES FIRST

To equal the maximum waiting times, in this strategy the priority of reservation will be given to the counter with whose bag is waiting the longer. Note that the waiting times will always be integers.

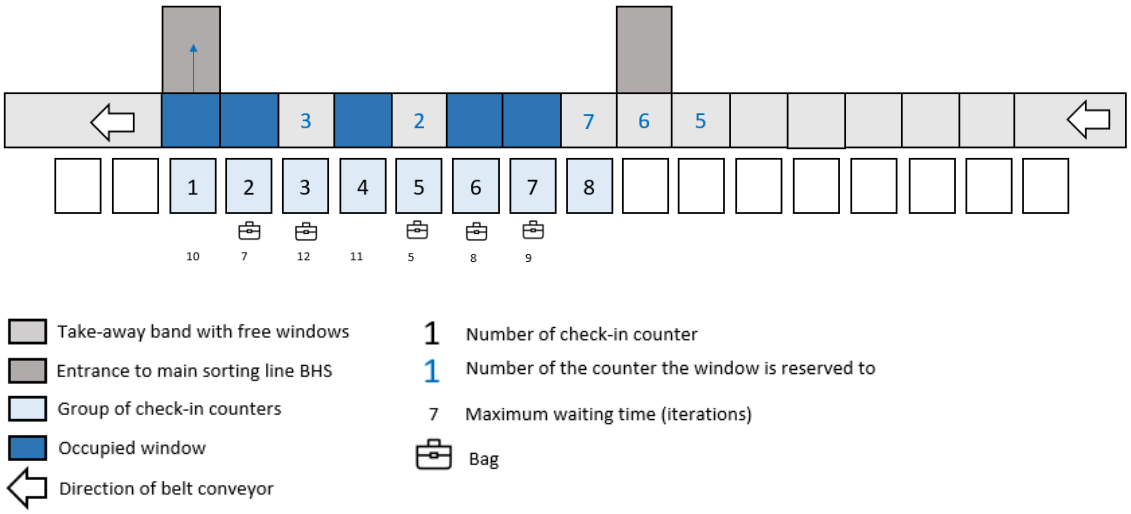


Figure 15. Representation of forward reservation: the highest maximum wait reserves first strategy

FORWARD RESERVATION: THE HIGHEST COMBINATION OF AVERAGE WAIT AND MAXIMUM WAIT RESERVES FIRST

This strategy will be a mix of the last two strategies, and the counter which will reserve first will have a higher combination of average wait and maximum wait, being both criteria equally important.

The formula for its prioritization is the following:

$$priority\ value\ for\ a\ counter = 0{,}5 * average\ wait\ of\ the\ counter + 0{,}5 * maximum\ wait\ of\ the\ counter$$

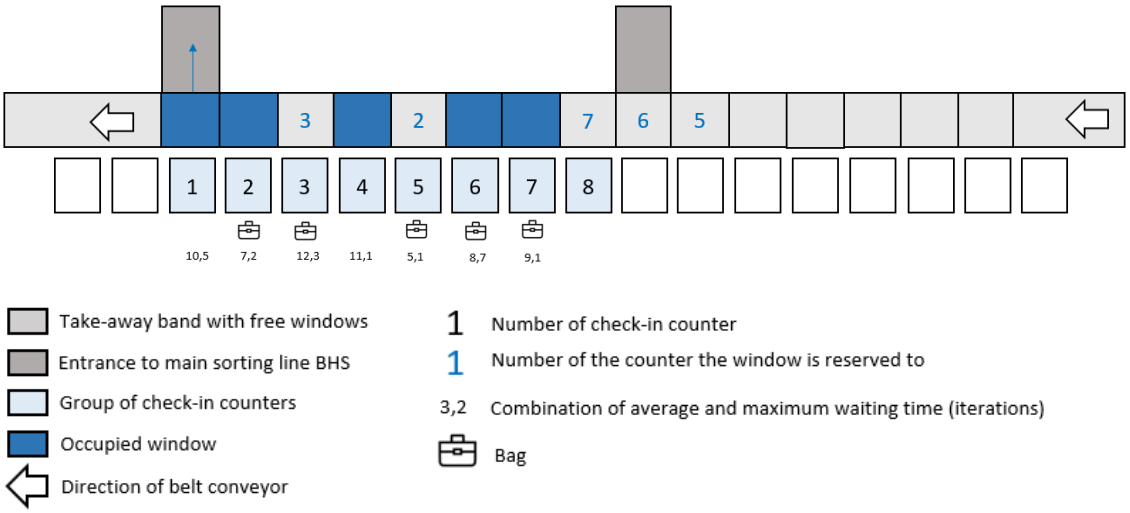


Figure 16. Representation of forward reservation: the highest combination of average wait and maximum wait reserves first strategy

5.4 Experimental design

To correctly represent the behaviour of the system, the simulation time is going to be 500 hours. In other words, the simulation model will represent the behaviour of 500 hours of the real check-in counters system. As explained before, it will be controlled with the "horizon" value, which corresponds to 600000 iterations, being one iteration (3 seconds) the movement of the conveyor belt one position in its direction. In other words, one iteration includes all discrete events regarding bag arrival, window reservation, loading of the corresponding bags into the conveyor belt and movement of the windows one position in belt direction.

Each of the iterations will be tried one at a time, with different values of stress rate. Then, relevant results regarding the influence of the stress rate in the strategies will be obtained and a comparison of the performance of the strategies is expected to highlight which strategy can work well for all baggage arrival scenarios.

5.5 Verification and validation

Once the model was built, it was verified that it reflected the behaviour of the system. To validate that the model was performing accurately, checks were done both with a global and a more focused scope. With a global perspective, the results obtained from the simulation were checked against the expectations for them, validating indeed that they were coherent in each of the strategies. With a more focused scope, the model was validated for individual bags in each of the strategies, verifying that the bags followed the right sequence of discrete events representing the reality.

5.6 Results

In this section the check-in counters study results will be presented. The results include a comparison of the strategies, followed by a check of the strategies against different criteria and a conclusion on which is the best strategy.

Firstly, the comparison of strategies will be shown for different stress rates of the system. The simulation results will be given for a system using 24 counters, since a system with 24 counters is expected to reflect better the strategy performance.

Secondly, the window allocation strategies will be tested against different criteria:

- Test of the strategies against different number of counters in the system: it will be checked that the performance shown in the study with 24 counters is the same with 8 and 16 counters.
- Test of the forward reservation strategies against different number of windows available for forward reservation.
- Test of the strategies against Copenhagen Airport criteria of performance. It will be checked that the strategies accomplish a take-away time of 20 seconds for 96% of the baggage.

Finally, a conclusion on the best strategy will be given.

5.6.1 Comparison of strategies with different stress of the system

In this section the performance of the different strategies will be compared with respect to three criteria:

- **Throughput of each counter (measured in bags)**: it is the total number of bags moved into the take-away belt along the whole simulation time (500 hours).
- **Average wait of each counter (in windows)**: it is the average number of windows that a bag in the specific counter must wait to get a free window and be loaded in the tape belt.
- **Maximum wait of each counter (in windows)**: it is the maximum number of windows that a bag arrived at the counter waited to be loaded in the take-away band.

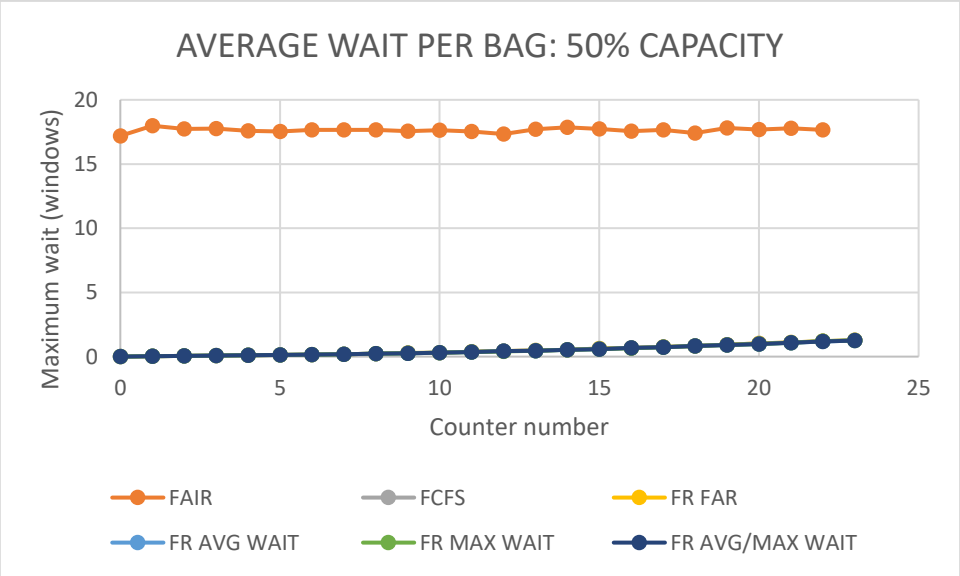
The comparison of strategies will be done with different stress of the system, to accurately define the behaviour of each of the strategies dependent on the general bag arrival rate or stress rate, being the designed capacity of the system 1200 bags/hour. The number of counters used for comparing the strategies will be 24, as the waiting behaviour is more representative. Note that the conveyor belt is moving to the right on the simulator, so counter number 0 is the first in the direction of the conveyor belt, and number 23 will be the last.

5.6.1.1 Comparison of strategies at low system capacity

When the system has a low input (until approximately 50% of the designed capacity), all the luggage arriving at the check-in desks is easily

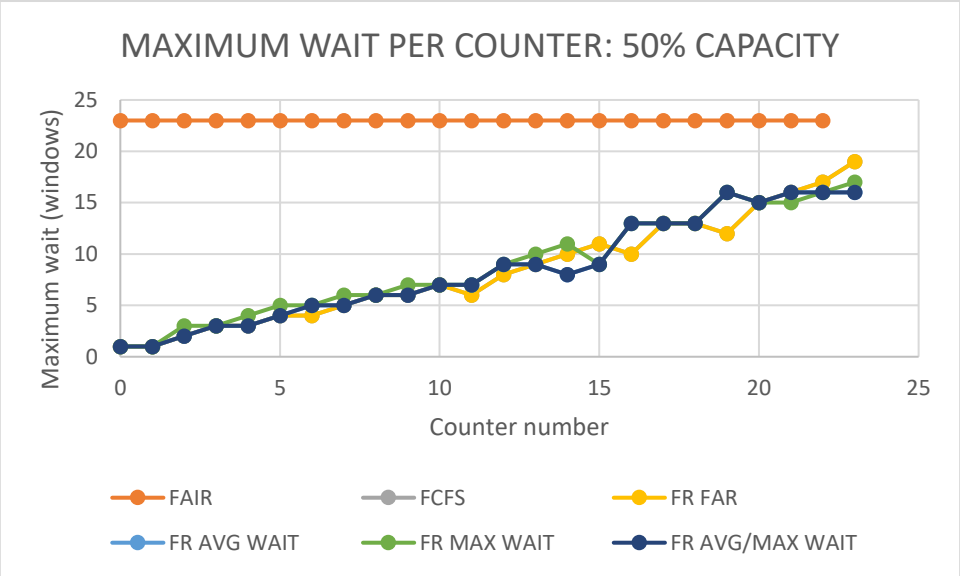
loaded to the take-away band. This is seen as the throughput of all the strategies is the same as with utopia strategy. Small differences found on the values are due to the stochasticity of the bag arrival and are not relevant.

The low arrival rate also impacts the average waiting time of the counters. Excepting the fair strategy that keeps a constant average wait on each counter equal to the number of counters, the rest of strategies keep an average waiting time close to zero. It can be seen in the following figures for 50% of capacity:



It can be concluded that with a low arrival rate, it is not convenient to use the fair strategy, as every counter must wait to get a window every N (where N is the total number of counters) and the arrival rate is low, many windows are left free.

Regarding the maximum waiting time, all strategies except the fair strategy show a similar performance in which the last counters in the direction of the conveyor belt have higher waiting times, still less than the fair strategy.



All in all, it is concluded that the fair strategy is not convenient for low arrival rate of luggage and the rest have a similar good performance.

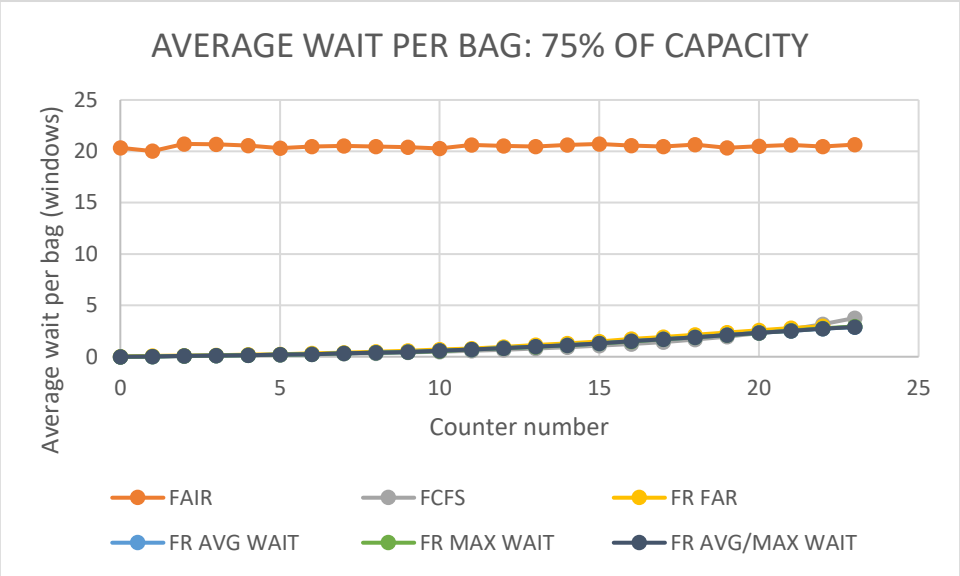
The values for the simulation results at 25% and 50% of system capacity are found on the annexes 9.3.1 and 9.3.2.

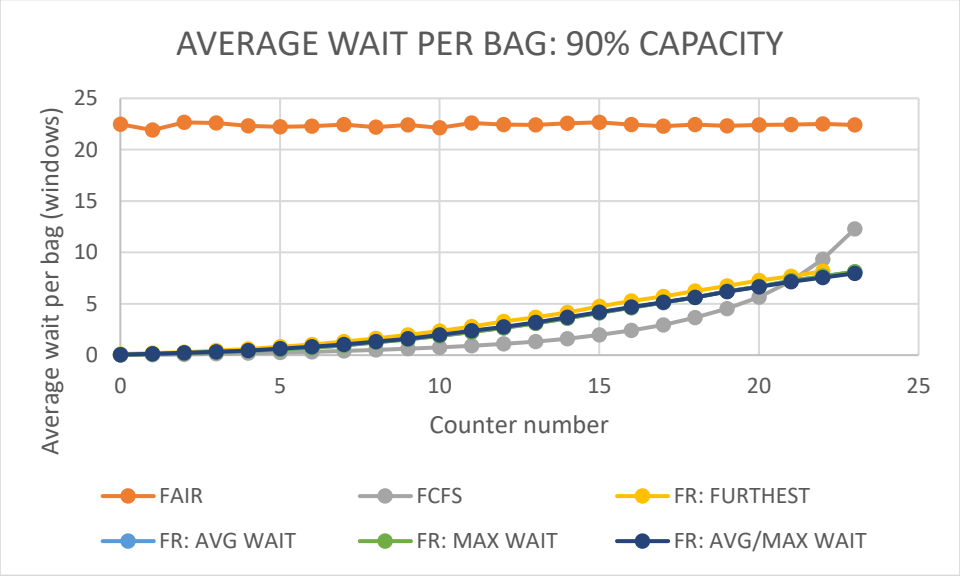
5.6.1.2 Comparison of strategies close to system capacity

When arrival rate increases from 75% and gets closer to the designed system capacity new trends are shown in the performance of the strategies.

The throughput has a fair distribution among the counters, there is not any counters specially affected by the performance of any strategy, which is seen as the throughput has the same profile as the utopia strategy (non-significant differences are due to stochasticity).

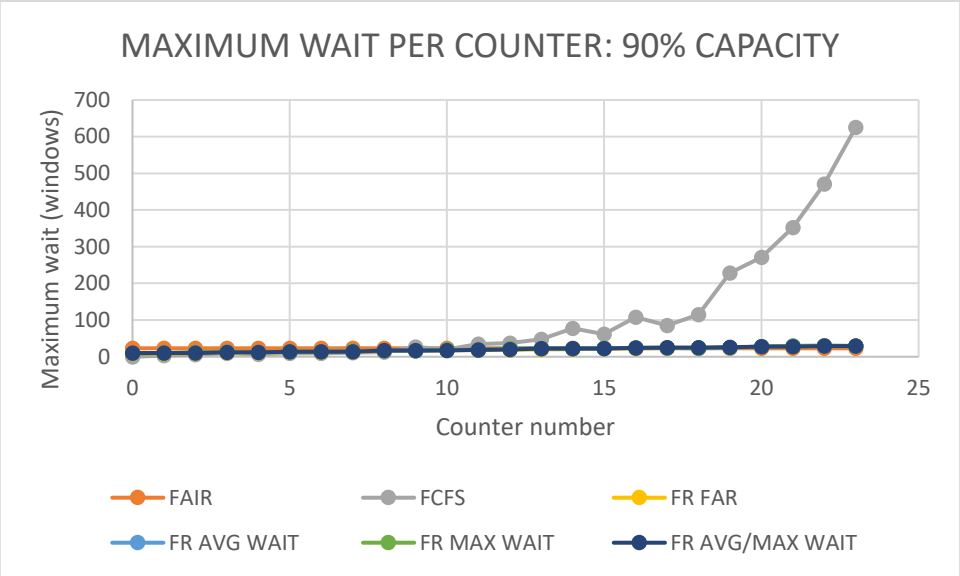
The average waiting time of the counters also presents a similar profile with all strategies (except fair), increasing slightly on the last counters in the belt direction. It is also dependent on the stress of the system, as it can be seen on the 75% capacity and 90% capacity graphs.

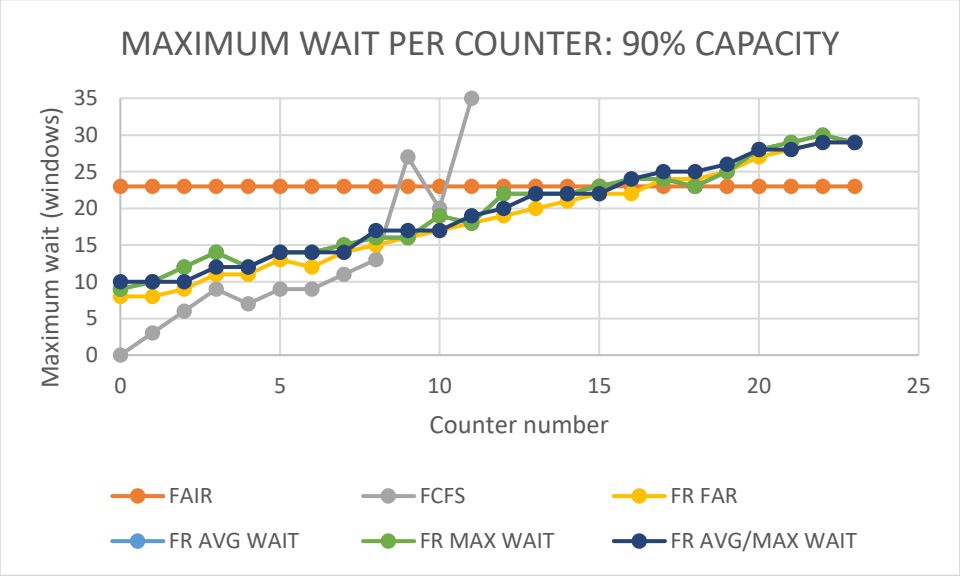




It can also be seen that the FCFS shows higher average waiting times than all the rest (except fair).

In regard to the maximum waiting time, FCFS shows a sharp increase in the waiting time for the last counters in the direction of the conveyor belt. Among the rest of the strategies (except fair), all show a similar behaviour, slightly increasing with the counter number. It can be seen in the following graphs, especially in the second graph, where the scale is enlarged:





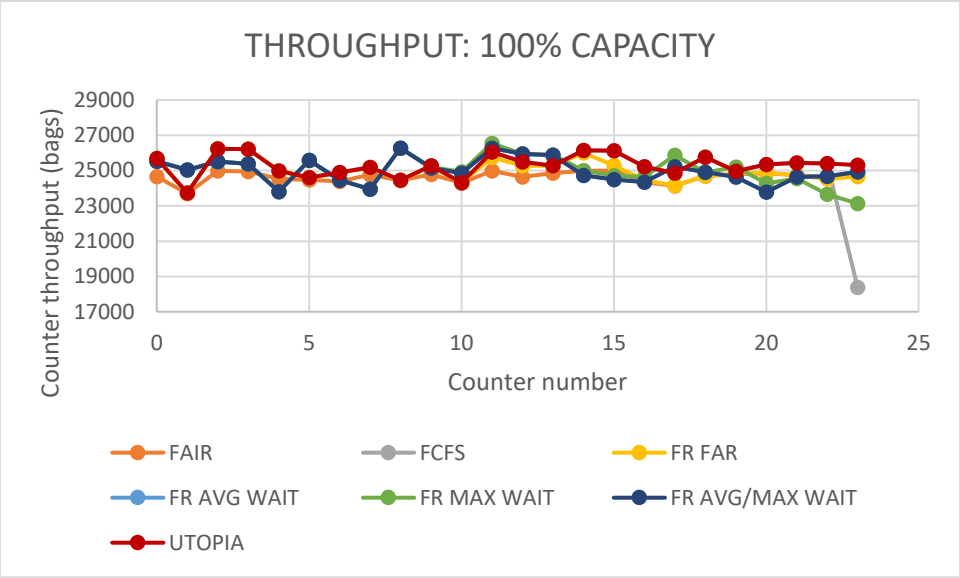
It can be concluded that the FCFS strategy is not convenient when the system stress is close to the designed capacity. Furthermore, among all strategies fair performs with a fair distribution of waiting times among the counters.

The values for the simulation results at 75% and 90% of system capacity are found on the appendices 9.3.3 and 9.3.4.

5.6.1.3 Comparison of strategies at system capacity

When the system works at system capacity, the bad performance of FCFS strategy regarding waiting times is still shown and it can be clearly seen that it has an exponential shape, increasing with the counter number.

Furthermore, the throughput is also affected, and the last counters are loading half of the values of other counters:



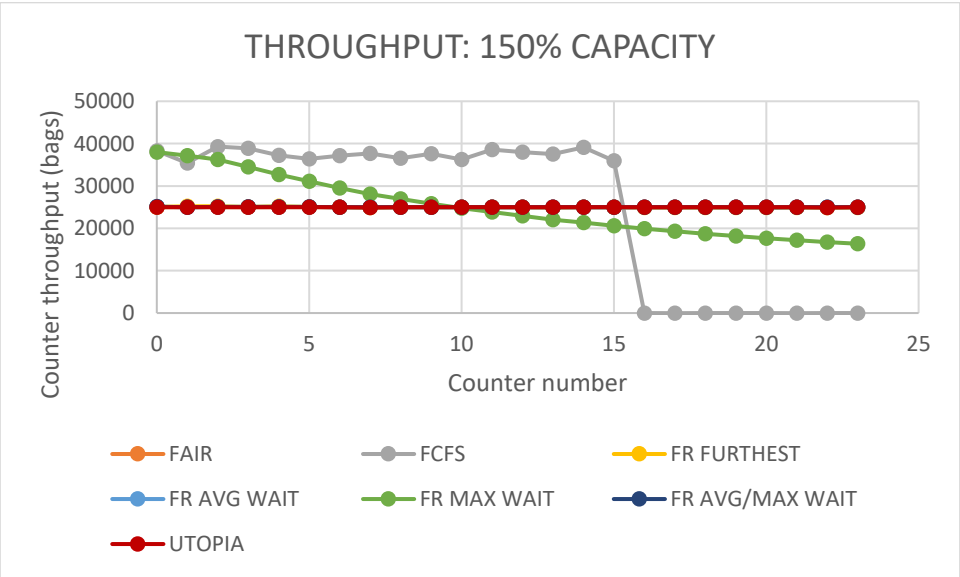
Thus, same conclusion as for close to system capacity, FCFS strategy does not perform at system capacity either.

The graphs for the system at 100% of capacity are found in appendix 9.3.5.

5.6.1.4 Comparison of strategies at higher than system capacity

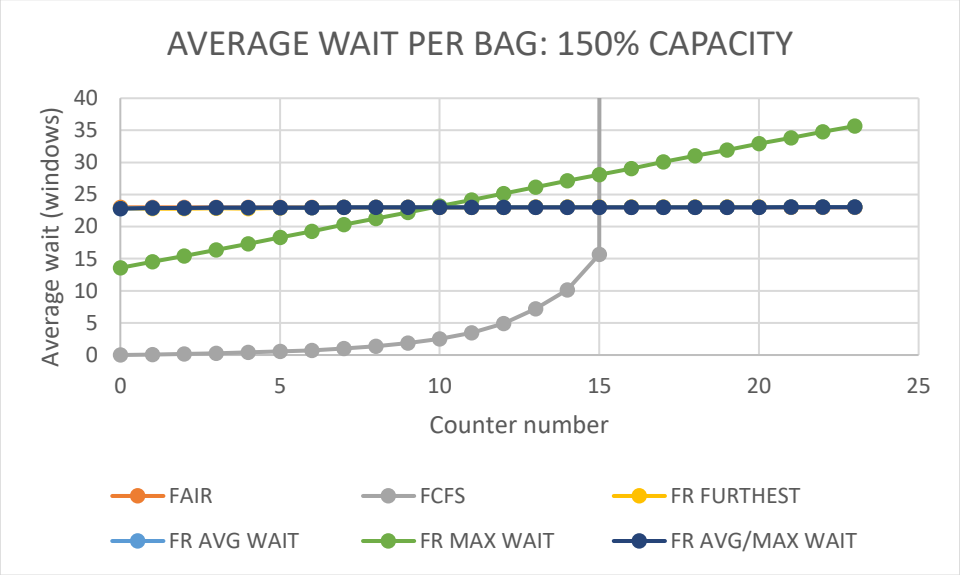
When the stress of the system overpasses its designed capacity, the trends shown are even more accentuated the more it is stressed.

With FCFS the throughput of the last counters is the belt direction is close to zero. The higher it the stress of the system is, the more counters (starting from the furthest in the direction of the conveyor belt) will have a throughput of zero. For example, in the case of 150% of capacity counters from 16 onward have zero throughput. It can be seen in the following graph:

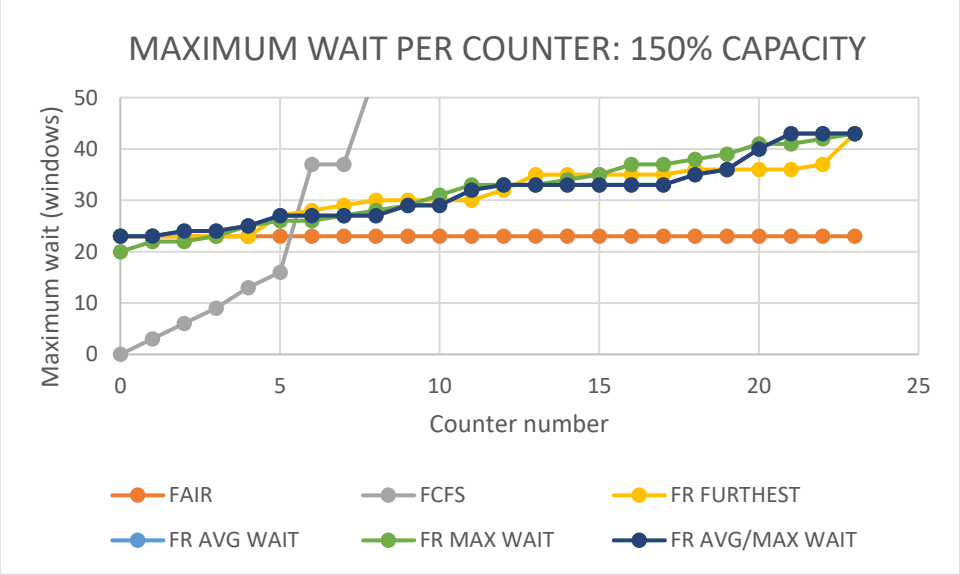


With forward reservation strategies, the throughput diminishes directly dependent on the counter number, but without such a sharp change. And with the fair strategy, all counters have the same throughput.

Regarding the average waiting time, the fair strategy and the forward reservation which combines average wait and maximum wait present a good and fair performance. FCFS as expected for the last counters, which have zero throughput, presents infinite waiting time. For 150% it is shown in the graphs below:



Regarding the maximum waiting time, it also presents a sharp increase on the last counters in the direction of the conveyor belt. The rest of the strategies maintain good values of maximum waiting time. Again, the fair strategy is the best when the stress rate is high. It can be seen in the graph for 150%:



Graphs for the system performance at 110%, 125%, 150% and 200% of capacity are found on the appendixes 9.3.6, 9.3.7, 9.3.8 and 9.3.9, respectively.

5.6.1.5 Conclusion on comparison of strategies for all system capacities

Now that the performance of the strategies has been seen for all stress rates, a conclusion of which of the strategies works best will be concluded.

Firstly, the FCFS strategy can be discarded. It has a considerable worse performance than the rest of the strategies when the system is close to, at or above the designed capacity. It has a bad performance both for throughput and for waiting times. Note that FCFS is the current strategy being used in Copenhagen Airport, and it is the worst, according to the present study.

Secondly, the fair strategy will also be discarded. Although it presents a good performance when the system is overstressed, it does not perform well when stress is low. The usual case of the system is with low stress rates, so it would not perform well because many windows will be free. This would even be worse in the real system than in the simulation, as many counters are known to be closed for long periods of time, resulting in free windows that would never be used.

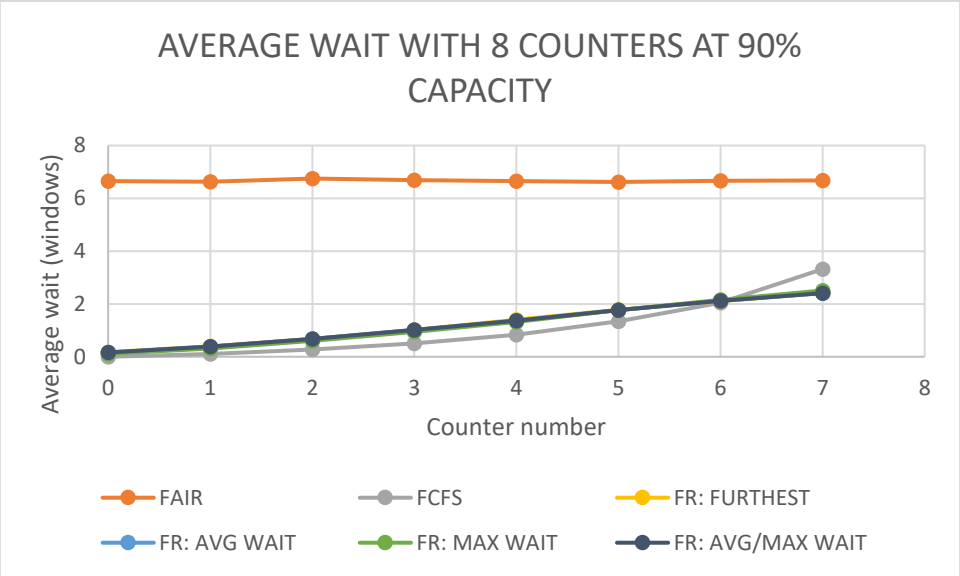
Finally, it can be said that all the forward reservation strategies present a similar good behaviour. Any of them would present a good performance and could be implemented with the current system of the airport. Among them, FR AVG/MAX WAIT would be the best as it gives a balance between average waiting time and maximum waiting time when prioritizing the counter reserving the window.

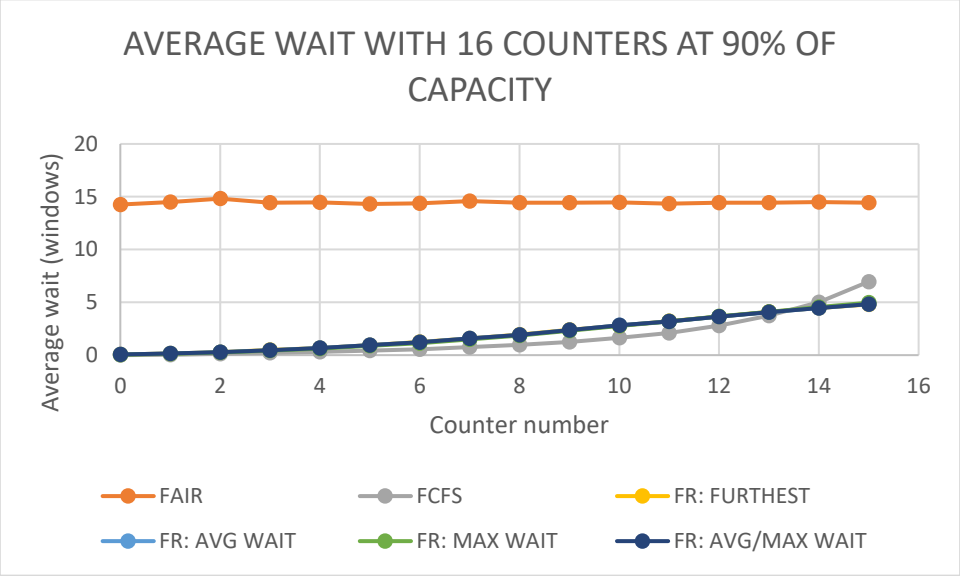
5.6.2 Sensitivity of the strategies to the number of counters

In this section, it will be checked that all strategies worked as shown previously also with 8 and 16 counters. To do it, all the strategies will be proven for the same criteria as before (throughput, average wait per counter and maximum wait) with the system working at 90% of its capacity. The reason for choosing 90% is that it was the critical point for performance previously. As a reference, the study for 24 counters is found on the appendix 9.3.4.

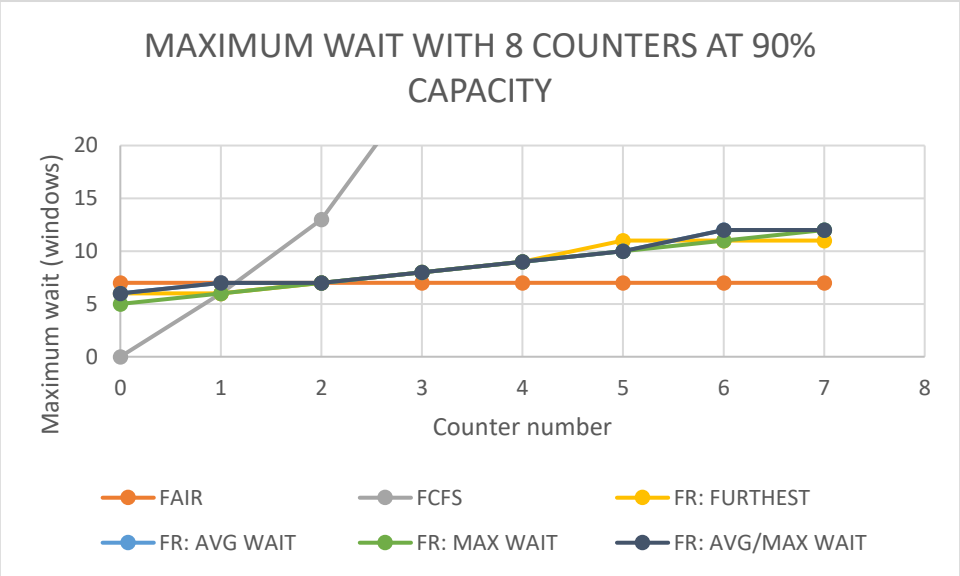
Starting with the throughput, it is proved that the it is not affected by the strategies. Although there is some variability in the throughput among the counters, it is not due to the strategies. It is caused by the stochasticity of the system itself: the less the number of counters, the most probable it is that they have a different bag arrival. The graphs for 8 counters and 16 counters are found in appendices 9.4.1 and 9.4.2, respectively.

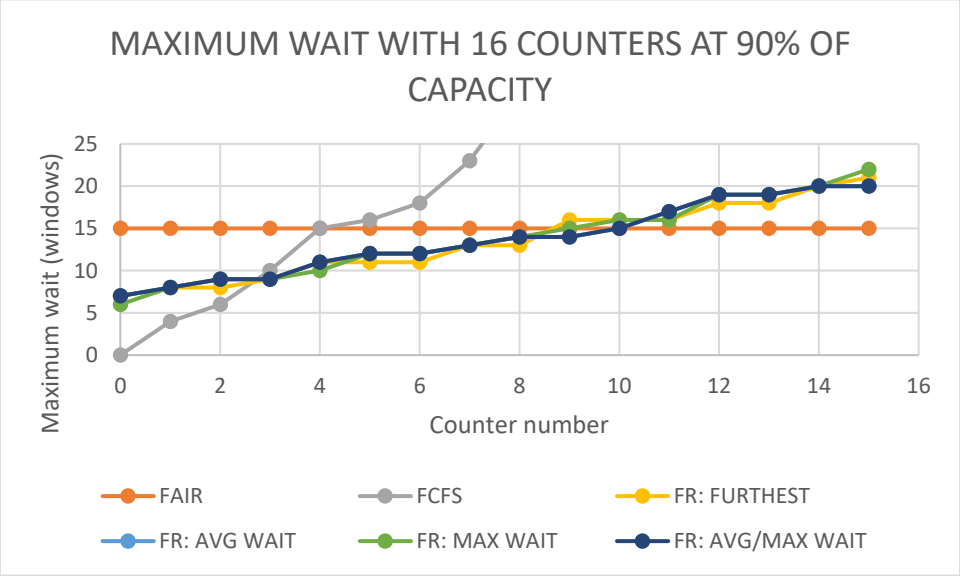
Regarding the average waiting time, the performance of the FCFS strategy is not as bad is it is in the case of 24 counters, but it is still the worst strategy after fair strategy. Fair strategy has the worst performance the less the number of counters, as there are many windows unused. It can be seen in the following graphs:





In regards to the maximum wait, it is clear again that the FCFS strategy does not perform well. The waiting times are exponential growing the further the counter it is in the direction of the take-away belt. It can be observed in the following graphs:





All in all, the forward reservation strategies are the ones that perform best for all stress of the system and for all number of counters. Choosing one, in special the FR AVG/MAX WAIT is the one among the reservation strategies that performs best, as it considers average and maximum wait.

5.6.3 Sensitivity of the forward reservation strategies to the number of windows for forward reservation

Checking the performance of each of the forward reservation strategies with a different number of windows to reserve, it is proven that none of them is affected by the number of windows to reserve. The results (for throughput, average wait and maximum wait) are the same in each strategy. They are included in the appendices 9.4.1 and 9.4.2.

5.6.4 Comparison of strategies with a take-away time of 20 seconds

Now that the performance of the strategies has been studied, it is needed to check if they accomplish Copenhagen Airport performance goals. The performance goal states that 96% of the luggage must be taken within 20 seconds.

To do it, the 20 seconds can be approximated to 7 windows (as each window passes every 3 seconds). The results of the percentages of baggage that are taken within 20 seconds depending on the strategy implemented and the % of system capacity the system is working at can be seen on the following table:

% of system capacity	strategy					
	FAIR	FCFS	FR: FURTHEST	FR: AVG WAIT	FR: MAX WAIT	FR: AVG/MAX WAIT
20%	23,14	99,99	100	100	100	100
30%	19,4	99,93	99,97	99,96	99,96	99,96
40%	15,96	99,73	99,84	99,84	99,83	99,84
50%	12,31	99,22	99,45	99,49	99,45	99,49
70%	5,65	96,51	96,47	96,57	96,46	96,57
80%	3,22	94,22	91,96	92,07	91,96	92,07
90%	1,35	91,72	80,33	80,74	80,9	80,74
100%	0,2	90,3	11,71	11,89	16,78	11,89

Table 1. Percentage of luggage that accomplishes Copenhagen Airport performance goal of being taken away within 20 seconds depending on the stress of the system and the strategy implemented.

The forward reservation strategies, all accomplish with the criteria when the input of luggage is up to 70% of its designed capacity. Therefore, it is checked that they fulfil the criteria given.

Just out of curiosity for the reader, fair strategy shows the worst performance regarding this criterion, with the lowest values for all the % of system capacity. And, on the contrary, FCFS shows a surprisingly good performance.

FCFS shows such a good performance since almost all bags are loaded with low waiting times. Only a few, in the last counters in the direction of the conveyor belt have infinite waiting time and will never be loaded. This situation explains that the % of bags taken away within seven windows is close to 100%.

Even though FCFS strategy shows the best score with this criterion, it will not be recommended to use, as the performance related to all other strategies is worse in waiting time and throughput as it was shown in previous sections.

5.6.5 Selection of the best strategy for check-in counters problem

Summarizing, the forward reservation strategies are the ones that perform best for all stress of the system and for all number of counters. They also accomplish with the criteria established by Copenhagen Airport that 96% of the luggage should be taken within 20 seconds (for the system working up to 70% of its capacity). Choosing one in special the FR AVG/MAX WAIT is the one among the reservation strategies that performs best, as it considers average and maximum wait.

6 EARLY BAGGAGE STORAGE: ANALYSIS AND IMPROVEMENT

6.1 Conceptual model description

To ease the understanding of the Early Baggage Storage problem and functionality, the problem will be divided into different points to make its comprehension easier.

6.1.1 Early Baggage Storage functionality

The *Early Baggage Storage (EBS)* is the area where bags wait until its *chute* is allocated.

The *chute* is the space where luggage is placed until it is carried to the aircraft. Each chute contains the luggage that will be carried in a specific aircraft during a defined period or *chute time window.* The time window is defined by its *opening time* and its *closing time.* The window opening time is 2:30 h before flight departure (3:30 h for overseas flights) and the closing time is 20 min before departure (30 min for overseas flights).

Then, if a bag arrives while its chute is opened, it will go to its allocated chute. But, if it arrives before, it will have to be stored in the EBS until it can be released to go to its chute. Summarizing, the objective of the EBS is to store the bags arriving before its chute opening time and release them before its chute closing time.

6.1.2 Early Baggage Storage in Copenhagen Airport

In Copenhagen Airport the Early Baggage Storage is composed by 14 belt tapes. Each of them has a different length, which means that they have different capacities, each of them can store different number of bags.

The current system works by assigning a *release time* to each of the belts, that is the specified time when the conveyor belt will be totally emptied. Them, all the bags stored in the conveyor will be loaded into the sorting system (there are two sorting systems) for re-sorting.

The system allocates release times to the EBS lanes in 15 minutes intervals, and when a bag arrives tries to allocate it to the belt whose release time is earliest. The system does not consider the capacity of the belts when allocating release times to them.

The storage belts can be divided into two groups, depending on the sorting system they load the bags into. The first group (EBS 1, 2, 9 and 10) releases baggage to sorter 2; the second (EBS 3, 4, 5, 6, 11, 12, 13 and 14) releases baggage to sorter 1. EBS 7 and 8 have the possibility to release to both sorters but usually release to sorter 1. Then, two belts from different groups, for example EBS 1 and 4, could release its baggage at the same time, as they do it in independent sorters.

Currently the possibility of part of the belt being emptied is not performed, but it could be considered in a future implementation since there are controls to manage the length of the EBS conveyor unloaded in the sorting system.

Other control criteria allowed by the system is the allocation of all the bags with the same destination to one belt. However, it is not possible to assign multiple flights to the same belt. It is an option that is performed ad-hoc when a big transfer flight arrives.

The physical device where the EBS control is implemented is a *Programable Logic Controller* (PLC). It is an industrial computer control system that continuously monitors input conditions, runs a program defined by the user and scans output conditions. This means that any program that the user can define can be programmed in a PLC, which includes the decision criteria for allocating the items to the belts and the time-based strategy for emptying the belts.

6.1.3 Definition of good performance in Early Baggage Storage

Now that the reader is familiar with the EBS and its functionality, the following point to introduce is how to define how the performance of the EBS is (in relative terms of good or bad).

It is stated that a good Early Baggage Storage system will accomplish its objectives:

- Assurance of on-time bags to its chutes (release of each bag before its chute closing time).
- Minimization of the stress of the sorting system along time, understood by the multiplication of the number of times the belts are being emptied by its respective number of bags loaded into the sorting system.
- Minimization of the manually handled bags and maximization of the belt storage utilization.
- Even distribution of bags among the belts.
- Tracking and registration of the bag information.

And the strategy implemented for the control of the EBS must accomplish with the system constraints:

- Time constraints
    - Bag must be released so it arrives before its chute closing time.
    - Earliest time bag can be transported to the chute is chute opening time.
    - Sorting the bags and emptying the lanes takes a specific time.
- Physical constraints in Copenhagen Airport
    - All the bags in the belt must be emptied (or emptied until the position of the bag plus safety margin).
    - There can only be two belts from different groups releasing luggage at a given time.

6.1.4 Definition of the EBS process

In this section, the sequence of events a bag follows when arriving into the sorting system of the BHS is explained.

When a bag enters the BHS the check-in time is recorded, and the system conveys the bag to its destination: either the allocated chute or the early baggage storage. In the case that there is not enough space in the EBS the bag would be loaded into a container for manual handling. It is the programmed software in the PLC which makes the decision criteria. In case the bag went to EBS, it will stay in the belt until it is the *belt release time* or until it is the *bag panic time*.

In the Early Baggage Storage, the strategy programmed in the PLC will empty the belts, so that the bags can arrive on time to their respective chutes. However, it is possible that a bag is stored in a belt whose release time is later than the chute closing time allocated for the bag. To avoid that the bag misses the aircraft in this case, the system keeps track of the *panic time* of each bag, that is the latest time the bag must leave the EBS to be in time in its allocated chute. Knowing how long it takes to empty a whole belt (*emptying time*, from now on), the bag panic time will be equal to:

$$bag\ panic\ time = chute\ closing\ time - belt\ emptying\ time$$

In other words, *bag panic time* will be defined as the latest time that a bag must leave the EBS or the latest time it must be checked-in to arrive on time to its allocated chute.

We will define *emptying time* as time it takes to empty a whole belt and allocate a bag to its corresponding location. It includes emptying the bags from the belt, loading them into the sorting system and the process of sorting itself.

The time it takes to move the bag from the check-in point (initial point) or from the loading point of the EBS into the sorting system until the bag arrives to its corresponding location, allocated chute or designated belt in EBS will be defined as the *sorting time*.

Summarizing, the process of sorting and storage (process number 4 in Figure 1. Baggage processes in an airport. Source: Copenhagen Airports) contains the following events:

1. Registration of the check-in time of the bag in the sorting system.
2. Allocation decision: allocated chute or Early Baggage Storage. If EBS is full, bag will go to a manual handling container.
    a. If the bag goes into its allocated chute (it arrives during chute time window), it is transported (during sorting time) to the chute. Process finished.
3. If the bag goes into EBS, it is transported to a belt, process of duration “sorting time”.
4. System decision criteria of which belt the bag must go to.
5. Bag remains in the belt until one of the two following events occur:
    a. Bag panic time: belt can be fully emptied or partially emptied (only until the bag needed).
    b. Belt release time: all belt is emptied.
6. Loading of the belt into the sorting system & resorting of the bag/bags. Process of duration “emptying time”.
7. Restart of the process from the beginning. Repeat until the bag arrives to its allocated chute.

The process can be seen in the activity diagram in the Appendix 9.3. “Early Baggage Storage. Activity diagram”.

6.1.5 Assumptions and considerations

Regarding the behaviour of the model, the following assumptions are made:

- Conveyor speed will not be considered a variable in the study. It will be a parameter considered fixed in the system.
- No changes in the physical dimensions of the system will be considered. The number of belts and their designated capacities will be fixed and out of the scope of study. The present analysis will focus on changing the decision software criteria.
- The processing of the manual container for bags exceeding the designed capacity of the system will be total responsibility of the handling personnel. Once a bag arrives to the manual handling container, it will not re-enter the sorting system, it will be manually loaded into its corresponding chute in time manually. Anyway, in the case that the system utilization diminished, and the staff wanted to load the suitcases in the BHS again, bags would be considered again in the system with the check-in time corresponding when loaded into the sorting system again.
- Strategy for bag allocation to belts is time-based. The option of allocating a whole flight to a belt will be done when transfer and inbound luggage arrival are previously known. Therefore, flight to belt allocation will be out of the scope of the study and left to the baggage handling system personnel for ad-hoc situations.
- The same allocation strategy will be kept along the day and along the year, so it must support all arrival rates of luggage in Copenhagen Airport.
- “Emptying time” is assumed to be constant and equal to 5 minutes. This value is established as the mean value of the difference between the check-in time and the timestamp for entering the EBS.
- “Sorting time” is assumed to be constant and equal to 4 minutes. Based in the difference between the exit EBS timestamp and the chute window closing time.

There will be also some considerations regarding the system availability and record of data:

- Chute allocation solution is a previous requirement to the implementation of the present study. The chute which is allocated to each flight, and therefore, the chute that each bag must go to is a previous information to the resolution of the EBS allocation.
- For each bag, the data registered that will be used is:
    - *Bag ID*: code or number given by the airport to uniquely identify every piece of baggage
    - *Check-in time* is registered at the beginning of the sorting system.
    - Allocated chute: chute that is assigned to the aircraft, where the bag must arrive.
    - *Opening* and *closing* chute window *times*. Established start and end of the chute time window.
- For each bag, it is possible to track down in which belt it is and if it needs to be emptied at a certain point of time.

- The position of the bag in the belt is also known, but there is not a precise measure of the position in the length of the belt where it is, as it is a conveyor belt and bags have different sizes.
- All pieces of baggage have a standard size, and the capacities given can be used for them. No special sizes of items are considered.
- The allocation strategy must be implemented in a PLC and all data used in the software must be available by the system.

6.2 Model building and data collection

The model has been implemented in Python, according to the conceptual model explained and the assumptions and considerations established. The code for the simulator can be found in the appendix 9.7 “Early baggage storage simulator in Python”. The model can be separated into different parts:

1. **Initialization of the problem:** definition of the import packages needed for the simulation, definition of parameters, values and variables that will be used. Among the parameters *sorting time*, *emptying time* and *horizon* are included. The capacity that each belt will have is also established.

   Sorting time, as defined in the assumptions, will be considered equal to 4 min, and it includes the time since the bag is checked-in the sorting system until it arrives to its location: chute or EBS.

   Emptying time, as defined in the assumptions, will be equal to 5 min, including the unloading time from the belt to the sorting system (1 min) and the sorting time itself (4 min).

   The *horizon* will be the total simulation time, that will be 24 hours.

   Belt capacities are (in bags of average size) 73, 86, 95, 95, 120,113, 116, 124, 73, 86, 95, 95, 120 and 113. They can be modified by inlets.

2. **Bag and chutes data input:** bags and its respective data is read from a csv file. In this process the choice of the data to read is made.

3. **Definition of release times:** it is the process itself of defining the release time labels for each of the belts. Each belt will have a release time, and when the belt is emptied in the mentioned release time, another release time label will be assigned to the belt, that will be the next time the belt is planned to be emptied. There will be three different ways of defining the release times, that will be called “way 1”, “way 2” and “way 3”.

4. **Selection of the EBS allocation strategy**. It is the choice of the allocation strategy that will be used when running the simulation. There will be different strategies, that will be explained in the following section.

5. **Simulation**. Reached this point, the process itself is represented by iterations. Each iteration will represent a second, and in every second there will be three processes of which one or more could happen simultaneously. The simulation time will be "horizon", equal to 24 hours.
    a. **Bag check-in**: it is the event of a bag arriving to the sorting system. There is a sequence of events occurring:
        i. A function establishes if it goes to the EBS or not. As mentioned before, a bag will only go to EBS if it arrives earlier than the opening time of the window.
        ii. In the case it goes to the EBS, it will be allocated to a belt according to the strategy selected in the previous part. It will only be allocated to a belt if there is free capacity, otherwise it will be allocated to another belt following the strategy criteria. If none have an empty space, the bag will be placed in a manual handling container.
    b. **Belt release time**: if it is the designated time for a belt to be emptied, all bags will be unloaded from the belt and loaded into the sorting system. Then, baggage will follow the sorting process, if the chute window is open they will be transported to the chute, otherwise they will enter again the EBS, with a "new check-in time".
    c. **Bag panic time**: if it is the panic time of a bag, the last time the bag must leave the EBS to go to its chute, the belt where it is stored will be empty. The option of emptying only until the position of the bag can also be selected. In the software, the belt will be emptied until exactly the position of the bag needed. In a real implementation in the system, an additional length will also be emptied to ensure that the bag is emptied due to the fact that there could be different bag sizes and spaces among them.
    To ease keeping track of the panic times of the bags in a specific belt, a *belt panic time* is introduced, which consists in the earliest panic time of all the bags contained in the belt. Then, the panic time for a specific belt is stored with the corresponding bag IDs that must be released at the given panic time.

6. **Showing status**. It will print into a file the values of utilization of all the belts every 5 minutes.
7. **Statistics**. It prints into a file the values for the number of times each belt is emptied due to a release time label, due to a panic time, and the number of bags that are manually handled along the day.

6.3 Experiment planning and description

Summarizing the Early Baggage Storage Problem can be divided into different and dependent subproblems. The first problems regard the problem definition stated in 2.2.1 “Early Baggage Storage allocation”, while the last corresponds to 2.2.2 “Early Baggage Storage used as a buffer area”. A subsection will be dedicated to each of them.

6.3.1 Early Baggage Storage Allocation

This part of the experiment is focused the problem defined in 2.2.1 “Early Baggage Storage allocation”: to which belt allocate the bags arriving into the EBS and how to empty them, ensuring that the bags arrive on time to its chute and minimizing the stress of the system.

6.3.1.1 Assigning the bag to a belt

It is the strategy that chooses which belt the bag must go to. The concept of *eligible* belt will be introduced and used in most of the strategies. The eligible belts are the ones whose release time ensures that the bag arrives in time to its allocated time chute. Therefore, the belt will be eligible if the belt has a release time contained in the interval:

$$(chute\ window\ opening\ time - sorting\ time, \quad bag\ panic\ time)$$

Note that all belts have a designed capacity, so a bag can only be allocated to a belt if it has enough capacity.

The strategies that will be experimented are the following:

1- TOTALLY RANDOM STRATEGY

With this strategy, the selection of the belt the bag goes to is totally random. A bag could go to any of the belts that have available space, regardless of the release times of the belts. If none have empty space, the bag will go to the manual handling container.

2- RANDOM AMONG ELIGIBLE STRATEGY

With this strategy, the selection of the belt the bag is assigned to is random among the eligible belts. An eligible conveyor with free space will be randomly selected, and if none have empty space, the bag will go to the manual handling container.

3- EARLIEST BELT STRATEGY

With this strategy the item will be assigned to the storage conveyor whose release time is the earliest among the eligible and has enough capacity. In other words, the choice will be among the belts with free capacity and release time as close as possible to the chute window opening time. If none of them have enough capacity, the bag will be manually handled.

The objective of this strategy is to keep the bags the shortest possible time in the storage and transport them to their allocated chute the earliest possible according to the established release times.

4- LATEST BELT STRATEGY

Opposite to the "earliest belt strategy", this strategy assigns the bag to the belt whose release time is as close as possible to the bag panic time. The belt, as in any strategy, will be eligible and have free capacity. If no belt meets the requirements, the bag will be manually handled.

The objective is to store the bags as much time as possible and they will be transported to their respective chutes the latest time with the possible release time labels.

5- LEAST FILLED BELT (LESS BAGS) STRATEGY

This strategy balances baggage storage load in all belts by assigning a bag to the belt that stores the less number of bags at the time the bag is being checked-in. In case two or more belts have the same number of bags stored, the choice will be random among them.

The purpose of this strategy is to balance the load among the belts, so there is a more equal distribution of bags among the belts.

6- MOST FREE SPACE STRATEGY

This strategy balances baggage storage load in all belts by assigning a bag to the belt that has the freest capacity at the time the bag is being checked-in. In case two or more belts have the same number of bags stored, the choice will be random among them.

The objective of this strategy is to balance load on the belts, so all the belts have a similar free capacity. Having similar free space in all belts is useful for the incoming bags, as the strategy tries to guarantee space in any belt and prevents the accumulation of bags in specific belts.

7- CURRENT CPH AIRPORT STRATEGY

It is the strategy currently implemented in Copenhagen Airport. It will be implemented to test the results of the rest of the strategies against the current performance.

With this strategy, the bag is assigned to the earliest belt. In other words, it will go to the belt with the earliest release time that has available space. But there is an exception: if the earliest belt chosen has a release time later than 45 min before the chute closing time for the bag, then the bag will go to another belt. The belt release time will be the release time of the earliest belt possible, regardless of the availability of capacity.

For all the strategies, if there is not space available in the belt indicated for the strategy, a bag will go to the least filled belt (more space) among all the belts, before being manually handled.

6.3.1.2 Establishing release time labels to the belts

Considering the system constraints regarding release time in the previous section, it is clear that the conveyor emptying strategy must be time-based. Then, the question of how to define the time-based labels arises. When establishing the release time labels, the reader must imagine a system as the example shown in the following image:

Belt	Release labels						
1	00:00	01:45	03:30	05:15	07:00	…	…
2	00:15	02:00	03:45	05:30	07:15	…	…
3	00:30	02:15	04:00	05:45	07:30	…	…
4	00:45	02:30	04:15	06:00	…	…	…
5	01:00	02:45	04:30	06:15	…	…	…
6	01:15	03:00	04:45	06:30	…	…	…
7	01:30	03:15	05:00	06:45	…	…	…
8	03:30	13:30	23:30	…	…	…	…
9	05:30	15:30	01:30	…	…	…	…
10	07:30	17:30	…	…	…	…	…
11	09:30	19:30	…	…	…	…	…
12	11:30	21:30	…	…	…	…	…
13	inf						
14	inf						

Figure 17. Representation of release labels "Way 1".

Each of the belts will have planned times when it will be emptied and load its bags on the sorting system. The first release label will be the next time the given belt will be emptied and, once it is emptied the next release label will be the following one.

In general, the belts can be divided into three categories according to the frequency of its planned release times:

- "Frequently released belts": will be the belts that will be emptied more frequently
- "Not frequently released belts": will be the belts that will be emptied regularly but with a lower frequency than the "frequently released belts"
- "Rest belts": will be the belts that will never be emptied on a planned basis. They will only be emptied when a bag stored is needed.

The key points to consider when defining the release time labels are:

- There must be at least one eligible belt at any given time during the day. Thus 22 hours of the day must be covered with the 14 release times for the belts (bags are only accepted in the EBS 22 hours before they flight departure time).
- The strategy must be consistent throughout the day and throughout the year, so no major changes in the strategy can be done along time.
- When a belt is emptied, a new release time label should be already defined to substitute the previous label.
- Chute window time is two hours (three hours for overseas flights).
- One of the objectives of the current study in the EBS is to minimize the total number of times that the belts are emptied, with the objective of not stressing the sorting system.

Thus, there will be three options implemented, called "way 1", "way 2" and "way 3".

- "Way 1": the three types of belts will be used. There will be seven frequently released belts with a frequency of 15 min, five not frequently released belts and two rest belts. Each of the not frequently released belts will cover a time span of 2 hours.
- "Way 2": it is a variation of way 1 in which the not frequently emptied belts will be released with a frequency of 1,5 hours.
- "Way 3": it uses only two types of belts: two rest belts and frequently emptied belts, with a release frequency of 15 minutes (30 min will also be tested).

For "Way 1" and "Way 2" the number of frequently emptied and not frequently emptied belts will be varied during testing to find the trade-off that shows better performance.

6.3.1.3 Matching the capacities of the belts with the release label definition

Depending on the release labels strategy, it may be more convenient to empty the longest belts more frequently or to empty the shorter belts more frequently. This problem will consider different cases that will be combined with the release time labels methods.

Specifically, the capacities will be ordered randomly ("Capacity 1"), from lower to higher values ("Capacity 2") and alternatively higher and lower values ("Capacity 3"). For all the cases the "rest belts" will have the highest values.

6.3.1.4 Choice of emptying all belt or emptying the belt until a given position

As the Early Baggage Storage consists of conveyor lines, it is possible to empty the whole belt or part of it, emptying a certain length of the conveyor.

Emptying part of the belt until a given position (physically a length margin will be emptied as well) is useful when it is only needed to release a specific bag in the belt, because of its panic time. Nevertheless, all the bags that are stored previous in line will be emptied as well.

Then, when simulating, it will also be tested how emptying the whole belt or emptying it until a given position affects the performance of the system.

6.3.2 Use of the Early Baggage Storage as a buffer

It is the second part of the problem statement defined in 2.2.2 "Early Baggage Storage used as a buffer area". The problem consists in finding the minimum chute time window for which the system can perform well.

All possible combinations of strategies will be tested to analyse the system and find the best solution. To analyse the system, different combinations of allocation strategy, release time labels definition and belt capacity definition will be experimented in the simulation.

The best combination will be tested against the different intensities of arrival of bags, real data form Copenhagen airport will be used for three study cases: a normal day (represented by the

data of 31-7-2018), a low intensity (represented by the data of 11-3-2018) and a high peak in bag arrival rate (15-7-2018).

Once the best combination is found, the behaviour of the EBS system being used as a buffer will be tested, obtaining the minimum chute window that allows the system to perform effective and efficiently.

6.3.3 Early baggage storage used as a buffer area

The planning and experiment that will be done addresses the problem 2.2.2 "Early Baggage Storage used as a buffer area": it will be studied if it is possible to reduce the number of chute window opening time and how it would affect the stress of the system. The study will be done with the strategy that performs best on the EBS allocation problem of the subsection before.

To modify the chute window time, either the window opening time, the window closing time, or both must be modified. Since the window closing time is precedent to the flight departure, it will be considered fixed, and the modifications will be done forwarding or delaying the chute opening time.

The study will consider modifications in the chute opening time for all the chutes at the same time of 15 and 30 minutes.

To measure the stress of the sorting system, the times the belts are released will be multiplied by the number of bags loaded every time a belt is emptied.

6.4 Experimental design

6.4.1 Early Baggage Storage Allocation

Due to the complexity of the problem underlaying in the addition of the different subproblems, the total number of experimental combinations possible will be:

$$\begin{aligned} &total\ number\ of experimental\ combinations \\ &= number\ of\ bag\ to\ belt\ allocation\ strategies \\ &* number\ of\ release\ time\ label\ strategies \\ &* number\ of\ capacity\ assignment\ strategies \\ &* number\ of\ arrival\ rate\ intensity\ data\ sets = 7 * 3 * 3 * 3 = 189 \end{aligned}$$

To properly see the influence of one strategy in the overall system behaviour, one strategy per subproblem will be fixed in the experiment runs. The focus will to obtain an idea of how each of the strategies in each of the subproblems affects the global performance of the Early Baggage Storage. Following this line of thought and Rosetti's principles in [27] more strategies and rerun experiments of the system will be made, until the individual influence of a strategy in the system is obtained, or until an identifiable significant correlation is obtained for strategies in different subproblems.

The bag and chutes data input are read from a csv file. Three data files will be used, each one corresponding to the full luggage arrival of a day in the Baggage Handling System:

a. Normal day (represented by 31-7-2018): contained in the file 'bags_normal.csv'.
b. Busy day (represented by 15-7-2018): in the file 'bags_high.csv'.
c. Idle day (represented by 11-3-2018): found under 'bags_low.csv'.

6.4.2 Early Baggage storage used as a buffer area

The model used for the simulation will be the same, fixing the choice of strategy, capacity and release times strategy. In other words, the simulation system will work with the parameters obtained in the EBS allocation problem that performed best.

The experiment will vary the window opening time by 15 and 30 minutes earlier or later to show how the strategy is affected by the window chute opening time.

6.5 Verification and validation

The simulation run time is 24 hours, which corresponds to the data given by Copenhagen Airport. Since the system has a daily pattern, the study would be representative for all days of the year and arrival rates, as the three most representative days are simulated.

To validate the system, the individual behaviour of a bag will be followed through all the discrete events for all the strategies and combinations. To globally validate the model, the utilization of the belts is tracked every 5 minutes, which will be verified with the individual discrete events of each bag. Each combination has also been simulated three times, to ensure the repeatability of the experiment, and the validity of its results.

6.6 Results

6.6.1 Early Baggage Storage Allocation

The process to conclude on a best strategy to use for Early Baggage Storage has been an iterative process including modifications of the allocation strategies. Many simulation runs were tested changing parameters of the strategies to find better performance results.

Therefore, the results will be presented to show the process of modifications. The results will be divided into preliminary results, strategy specific modifications, comparison of the best strategies and testing against different bag arrival rates. One subsection will be dedicated to each of the topics.

6.6.1.1 Preliminary results

In this section preliminary results and observations will be given after a single simulation of the process with all the combinations possible planned in section 6.4.1 "Early Baggage Storage Allocation". The preliminary results for each of the strategies are found in the subsections of the appendix 9.8.1, for each of the strategies under the column "Basic strategy". To compare the number of manually handled bags for all strategies, the data comparing all strategies and a graph are found on appendix 9.8.1.8.

6.6.1.1.1 General observations about bag arrival and allocation strategies

- There is a peak in bag arrival to EBS around midday, which may result, depending on the arrival rate, in manually handled bags since the system capacity may not be enough to store them. Consequently, as the number of manually handled bags is a parameter to minimize, the strategy chosen must show a low number of manually handled bags compared to the rest of strategies.

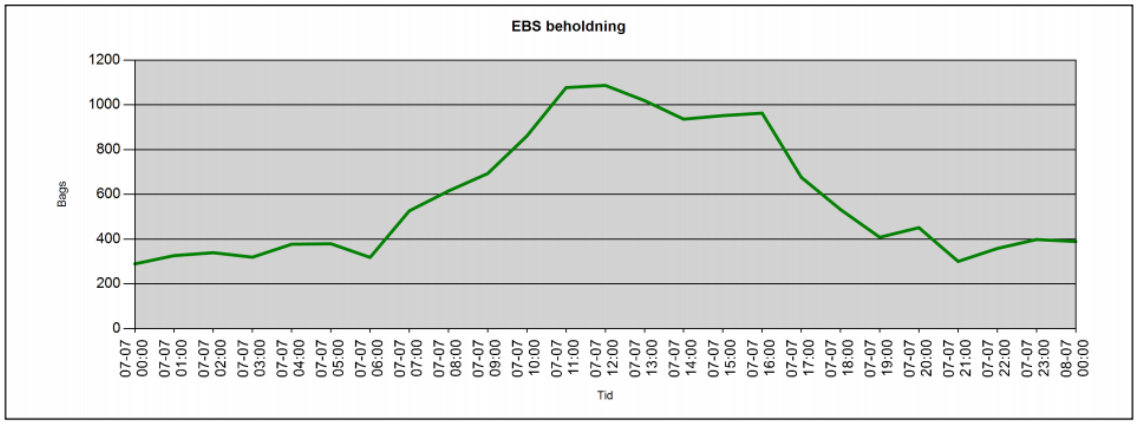


Figure 18. EBS utilization during an average day. Source: Copenhagen Airport

- With a low arrival rate of bags to the EBS, the system performs well according to number of manually handled bags for all strategies, as it can be seen in the following comparison of manually handled bags with all the preliminary results for the strategies.

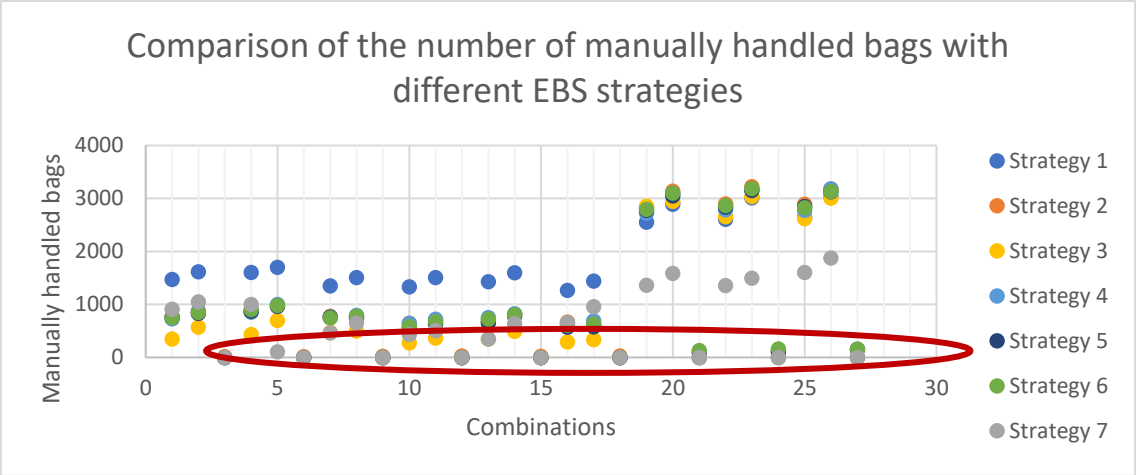


Figure 19. Comparison of the number of manually handled bags with different EBS strategies. Highlighted the results for low bag arrival rate.

The circle in red shows the manually handled bags for the combinations with a low bag arrival. It is seen that all strategies have none manually handled bags for these. Then, it is concluded that the focus of the study will consider all arrival rates with special interest in high arrival rates, as it highlights the difference among strategies.

- The number of manually handled bags is highly dependent on the allocation strategy, release labels for the belts and the matching of the capacities with the release label definition. So, all the parameters to define in the simulation have significant influence on the system.
- The simulation results show the importance of diminishing the manually handled bags. In the case the belts the bag should go to according to the strategy are full, the bag must go to any of the belts before being manually handled. Different modifications will be done in the strategies to cover this usual case (especially in the peak arrival time).

6.6.1.1.2 General observations about release labels definition

- Many bags arrive to the EBS storage about 10 hours before their flight, which results in a high utilization of the belts with the latest release labels. Consequentially, the minimum number of "rest" belts needed is at least two with the highest capacities. Then, the three release labels defined were tested with two rest belts.
- "Way 3" for the planned release labels definition shows the highest number of manually handled bags for all strategies. It was preliminary tested for a frequency of 15 minutes and for 30 min, showing even worse bag handling results for a frequency of 30 min. Basic results are given for the strategies with 15 min.

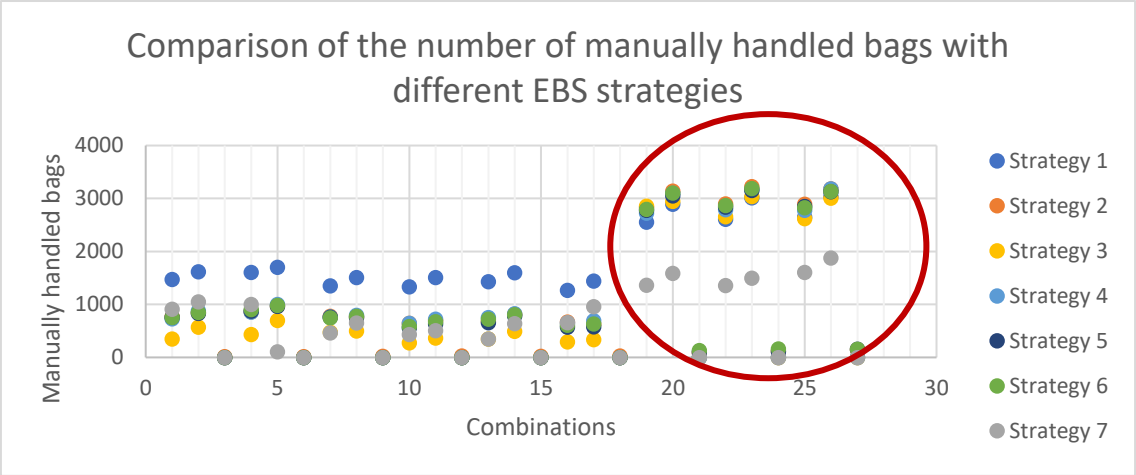


Figure 20. Comparison of the number of manually handled bags with different EBS strategies. Highlighted the results for combination with "way 3".

- "Way 1" and "Way 2" both show a good performance and it has been tested that the trade-off of seven frequently emptied belts and five not frequent emptied belts works well, as it covers well the arrival profile of bags to the EBS for all allocation strategies.

6.6.1.1.3 General observations about the matching of belts capacities with the release labels definition

- As it was previously concluded, the largest capacities will be assigned to the "rest belts", which present the highest utilization among all lanes.
- Larger values of capacities are also needed for the belts that cover longer time periods, since they show the following highest values of utilization: they store many bags and the lanes are not emptied as frequently as the "frequent" belts.
- These previous statements were tested in the strategies and, it was concluded that "Capacity 3" is not a good matching of the belts. The reason for its bad performance is that the matching is totally random, and it is proven that the matching of capacities with the planned release labels definition affects the performance of the system, so any of the other strategies present a better performance.

6.6.1.1.4 General observations on emptying the whole belt or emptying until a given position

- With the defined plan for release labels, the times the system is emptied due to panic time is considerably lower than the times it is being emptied by panic time. Therefore, the influence of emptying the belt until a given position is not direct and clear.
  However, it can be said it is better for balancing storage loads that the whole belt is emptied. The reason is that when a belt is totally emptied the bags whose chute window is not open yet usually go to some of the other belts, balancing loads.
- Therefore, due to its simpler use and the better balance of loads, emptying the whole belt is concluded to perform better.

6.6.1.2 Strategy modifications

Considering the preliminary conclusions and observations, each strategy was modified. The case of each of the strategies will be explained in detail.

6.6.1.2.1 Strategy 1- Totally random

Totally random strategy is used as a reference of the worst-case scenario. It performs worse than any other strategy regarding to manual handled bags as it can be seen in the graph below comparing the number of manually handled bags for all strategies:

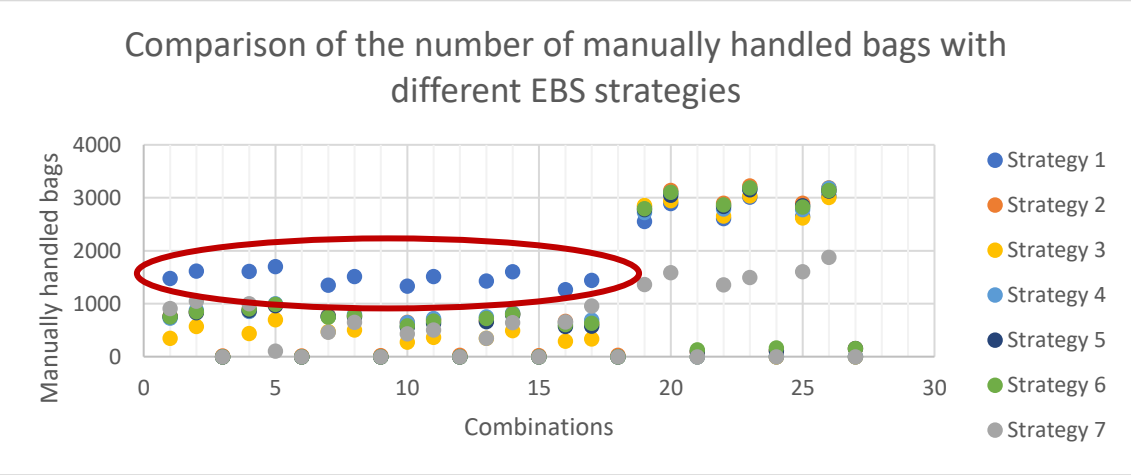


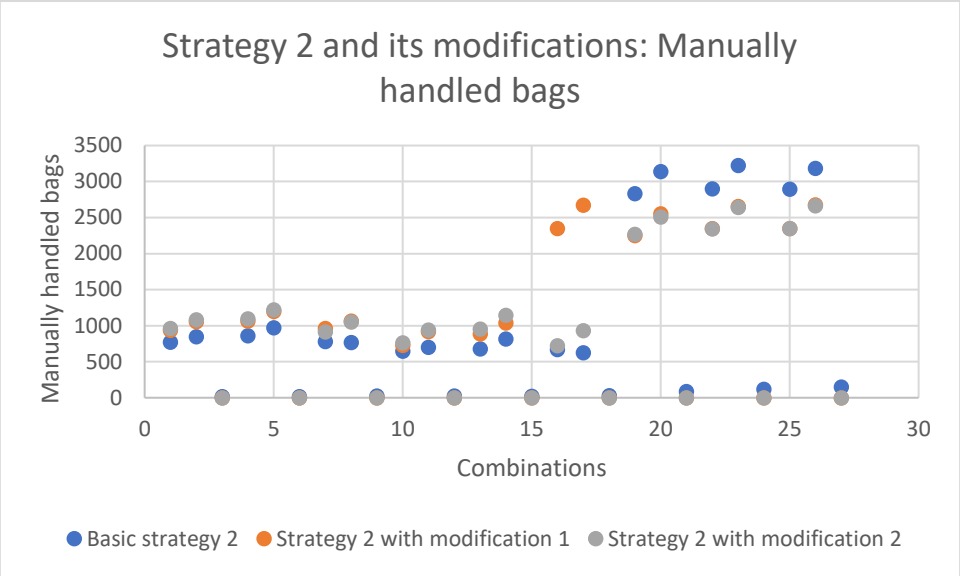
Figure 21. Comparison of the number of manually handled bags with different EBS strategies. Highlighted the results for strategy 1- Totally random

Therefore, it will never be used in the real system. The results for each of the combinations are shown in appendix 9.8.1.1.

6.6.1.2.2 Strategy 2- Random among eligible

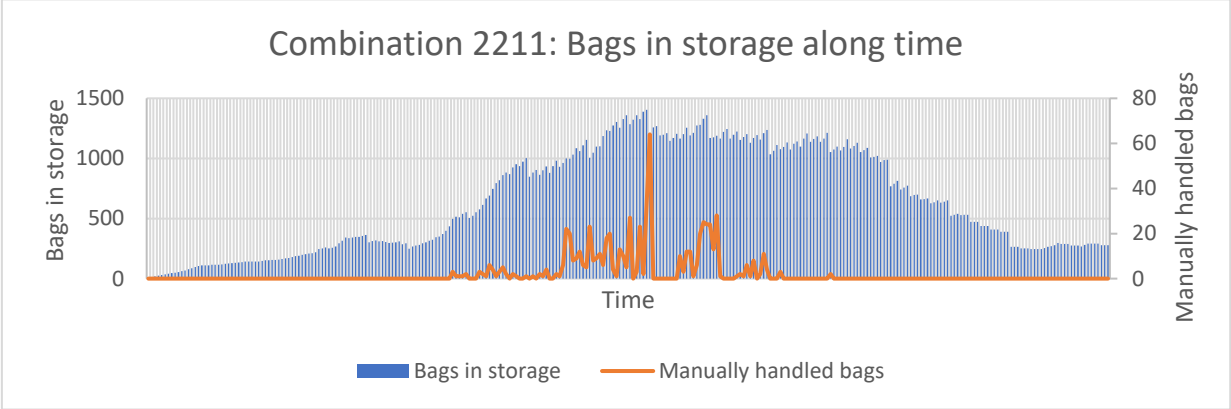
Random among eligible allocates the bag to a belt that has space between the earliest possible belt and the latest possible belt. It shows a lower value of manually handled bags than totally random, and the values vary among the manually handled bags obtained for the earliest and latest belt strategy. Results are found in the appendix 9.8.2.

With this strategy and the purpose of diminishing the manually handled bags, two modifications were tested, consisting in the case when all eligible belts are full. In the case they were full, a second strategy was used. The modifications consisted in using the last belt strategy or the most space belt strategy. The results for the simulation are included in appendix 9.8.1.2. and can be summarized in the following graph:



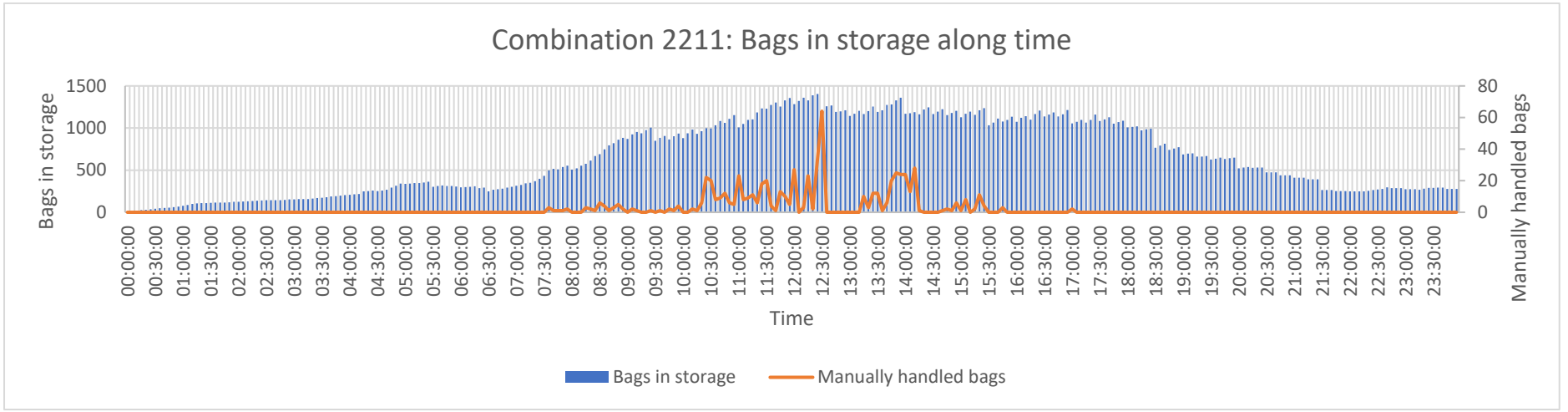
Surprisingly, the basic strategy works better than when added modifications. The reason why the original strategy performs better is that the bag is either in a belt eligible or manually handled. If it is not in an eligible belt, it does not use space in other belts not eligible.

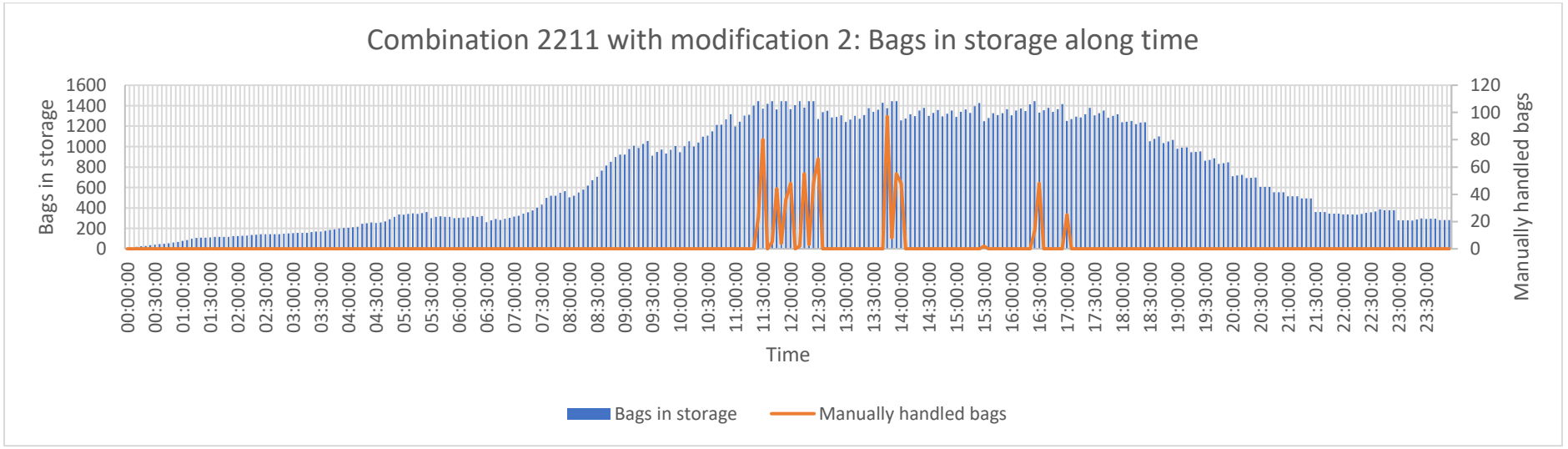
The problem with the basic strategy is that there are manually handled bags when the system still has capacity, as it can be seen in the following representation of the bags along time:



As it is seen, there are manually handled bags when the system does not reach its maximum capacity (1443 bags).

Looking at the performance along time, for the strategy without modifications the total number of stored bags is lower along time, and the peaks in manually handled bags are lower but more frequent. The results will be shown in the graphs in the following page for combination 221X, as it is the one that shows the lowest values for manually handled bags. The results for 2211 (combination with the data for a normal day) are shown.


Combination 2211: Bags in storage along time
Bags in storage
Manually handled bags
Time
Bags in storage
Manually handled bags


Combination 2211 with modification 2: Bags in storage along time
Bags in storage
Manually handled bags
Time
Bags in storage
Manually handled bags

Although random among eligible shows a good performance, it will not be implemented in the real system since it does not use all the capacity. For situations in which the system does not reach its designed capacity, there will be manually handled bags, which is an undesirable situation.

6.6.1.2.3 Strategy 3- Earliest possible belt

The earliest belt strategy shows the lowest number of manual handled bags compared to other strategies, as it can be seen in the graph:

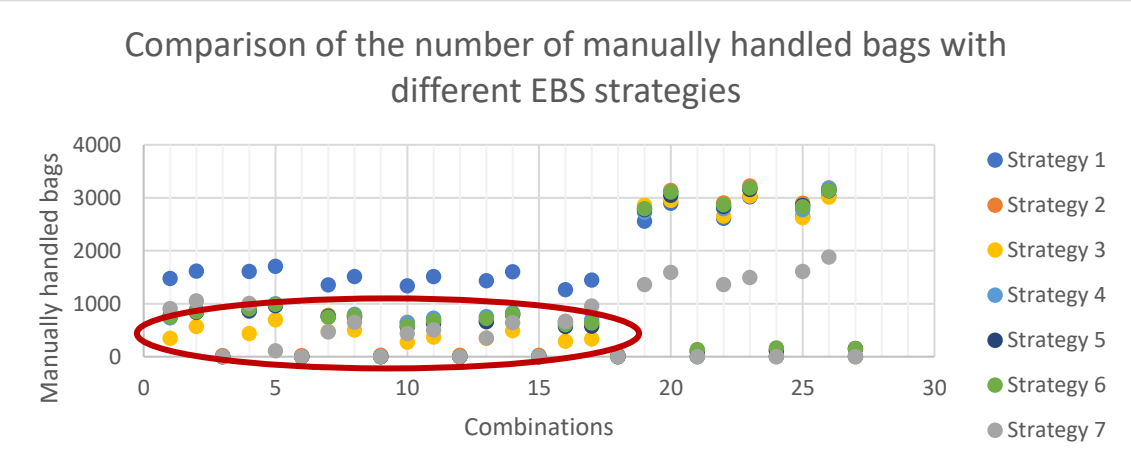
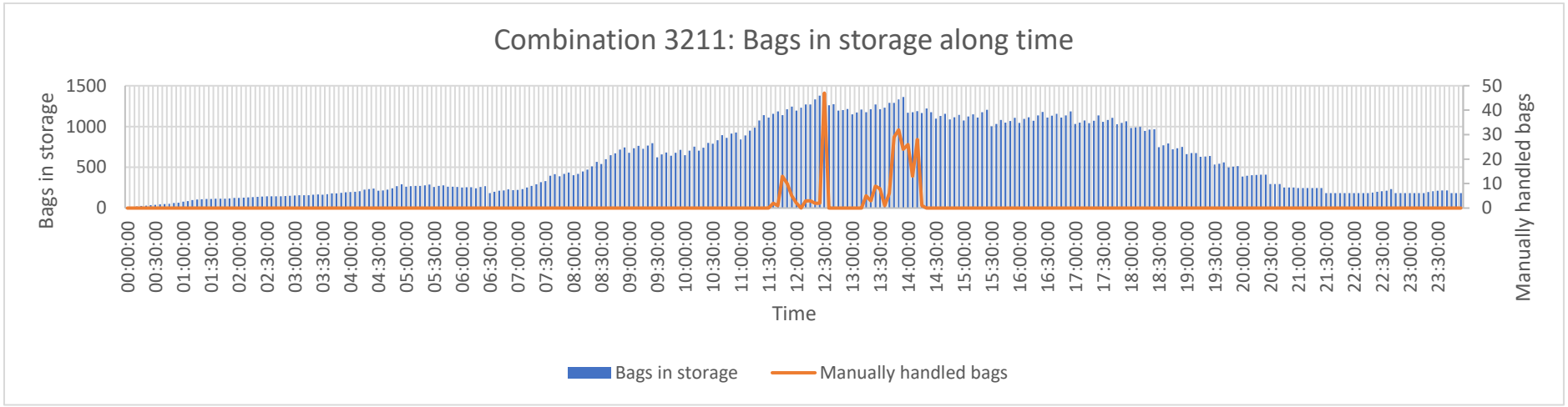
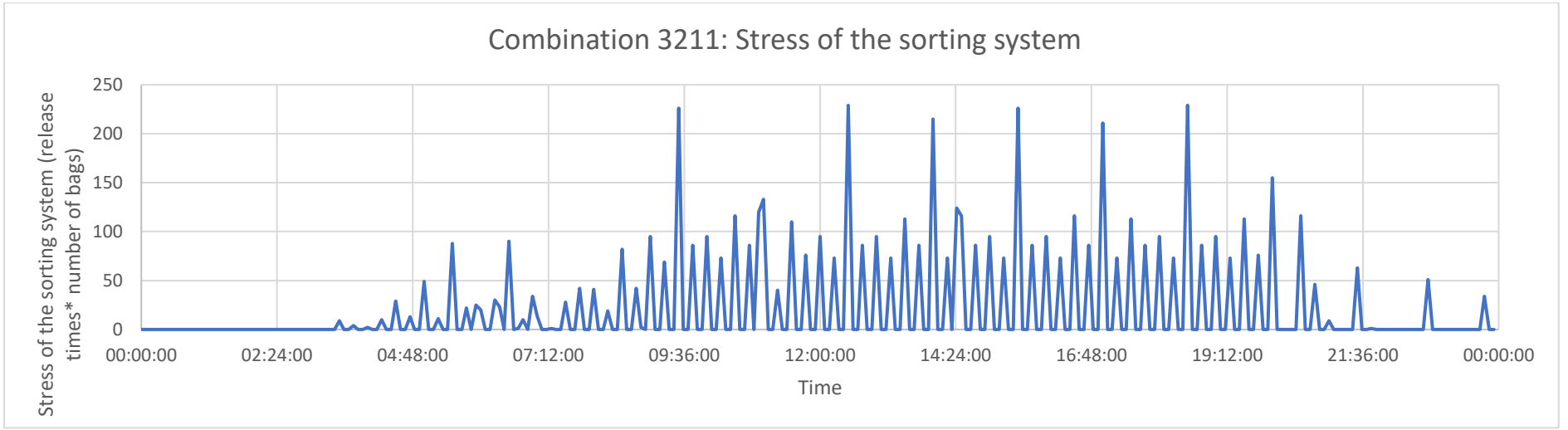


Figure 22. Comparison of the number of manually handled bags with different EBS strategies. Highlighted the results for strategy 3- Earliest possible belt

Results for manually handled bags are found in appendix 9.8.1.3.

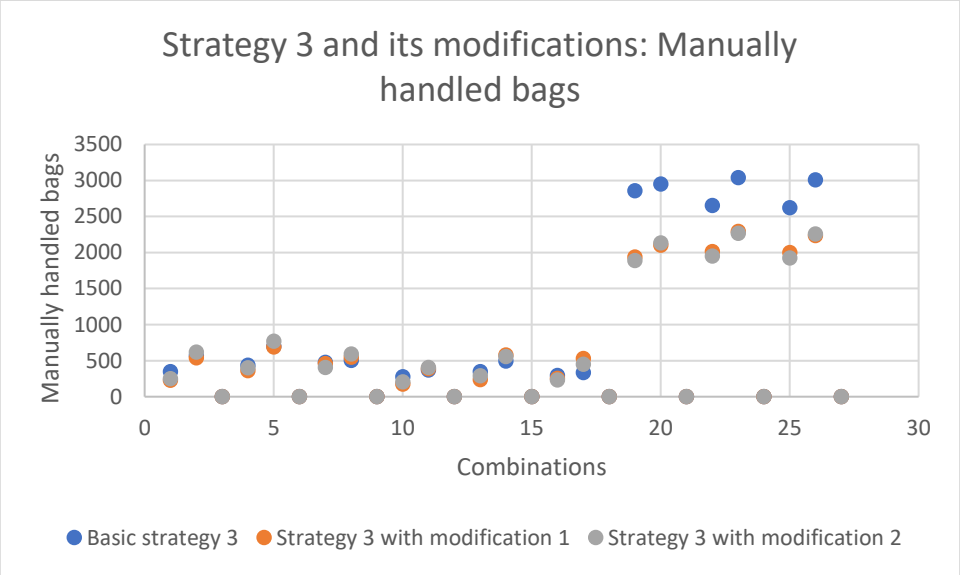
Looking at the detailed performance of the system along time, it also presents a good performance, with a constant level of stress in the sorting system, with a constant height of all the peaks, and with a maximum of two belts releasing at a time. The number of bags in storage along time and the sorting system stress can be seen in the following page:


Combination 3211: Bags in storage along time
Bags in storage
Manually handled bags
Time
Bags in storage
Manually handled bags


Combination 3211: Stress of the sorting system
Stress of the sorting system (release times* number of bags)
Time

This strategy has been modified so it includes the latest belt or the most space strategy in the situation that all eligible belts are full.

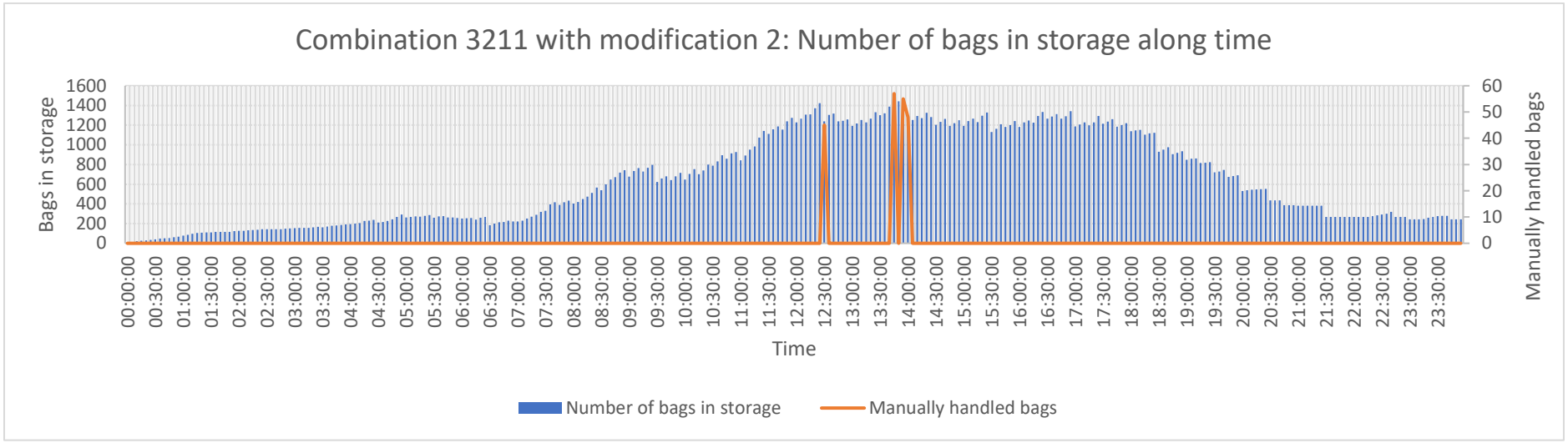
As it can be seen in the following graph, the most space belt works better almost all combinations as a second strategy in case all the eligible belts are full. There are two reasons that explain its better performance. Firstly, the stress of the system is the minimum possible if the just allocated bag is needed before belt release time: if a belt needs to be emptied by panic time, it is better if it is one of the most space belt as it will release the least number of bags possible. Secondly, it contributes to balance the free space for other belts.

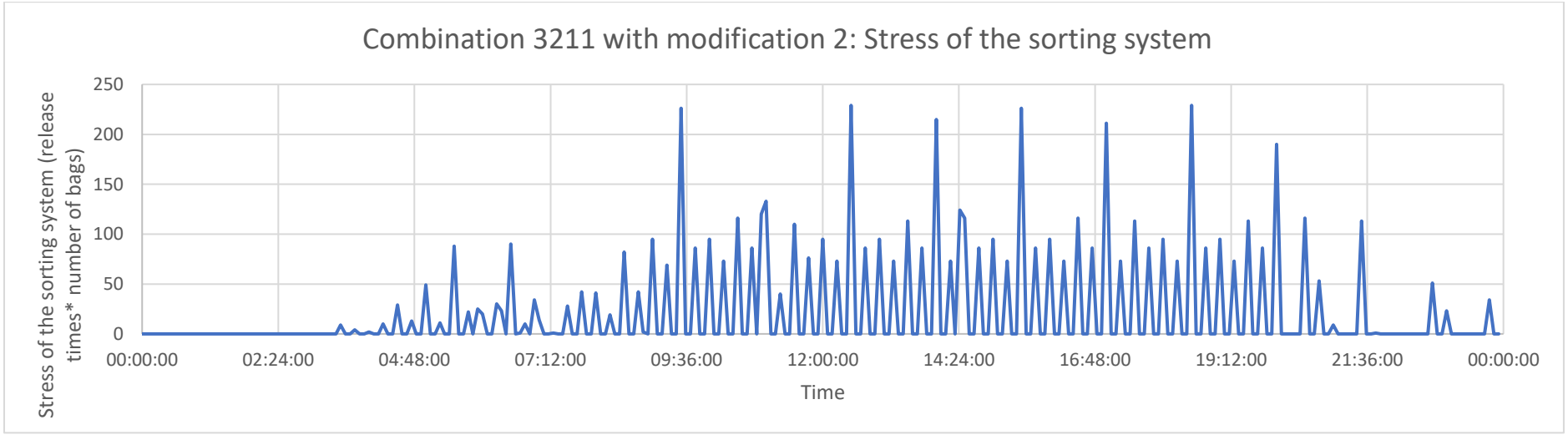


Therefore, using the strategy with the most space belt as the second strategy would be the most convenient regarding to manual handling bags.

Looking at the performance of this second modification along time, it shows similar results to the original strategy. The total number of peaks in manually handled bags is diminished, but the peaks show a higher value. The higher peaks do not represent a disadvantage, as it is more convenient to use the capacity until full, and then the manually manage the rest of the bags.

Furthermore, the stress of the system is also stable and constant, with a similar pattern to the original strategy which is a desirable situation for the EBS. The performance along time can be seen in the graphs in the following page:


Combination 3211 with modification 2: Number of bags in storage along time
Bags in storage
Manually handled bags
Time
Number of bags in storage
Manually handled bags


Combination 3211 with modification 2: Stress of the sorting system
Stress of the sorting system (release times* number of bags)
Time

Summarizing, the earliest belt strategy using the second modification shows a good performance and could be implemented in the system. It is one of the best strategies in comparison with the others. Of all the combinations, the one that performs better is 321X, including "way 2" of release labels and alternates lower and higher capacities in the belts.

6.6.1.2.4 Strategy 4- Latest belt possible

This strategy shows a number of manually handled bags close to the reference random among eligible strategy. The number of manually handled bags is higher for almost all combinations compared to the rest of the strategies as it can be seen in the following graphs, where in the second graph the scale is augmented and focused on the combinations without "way 3":

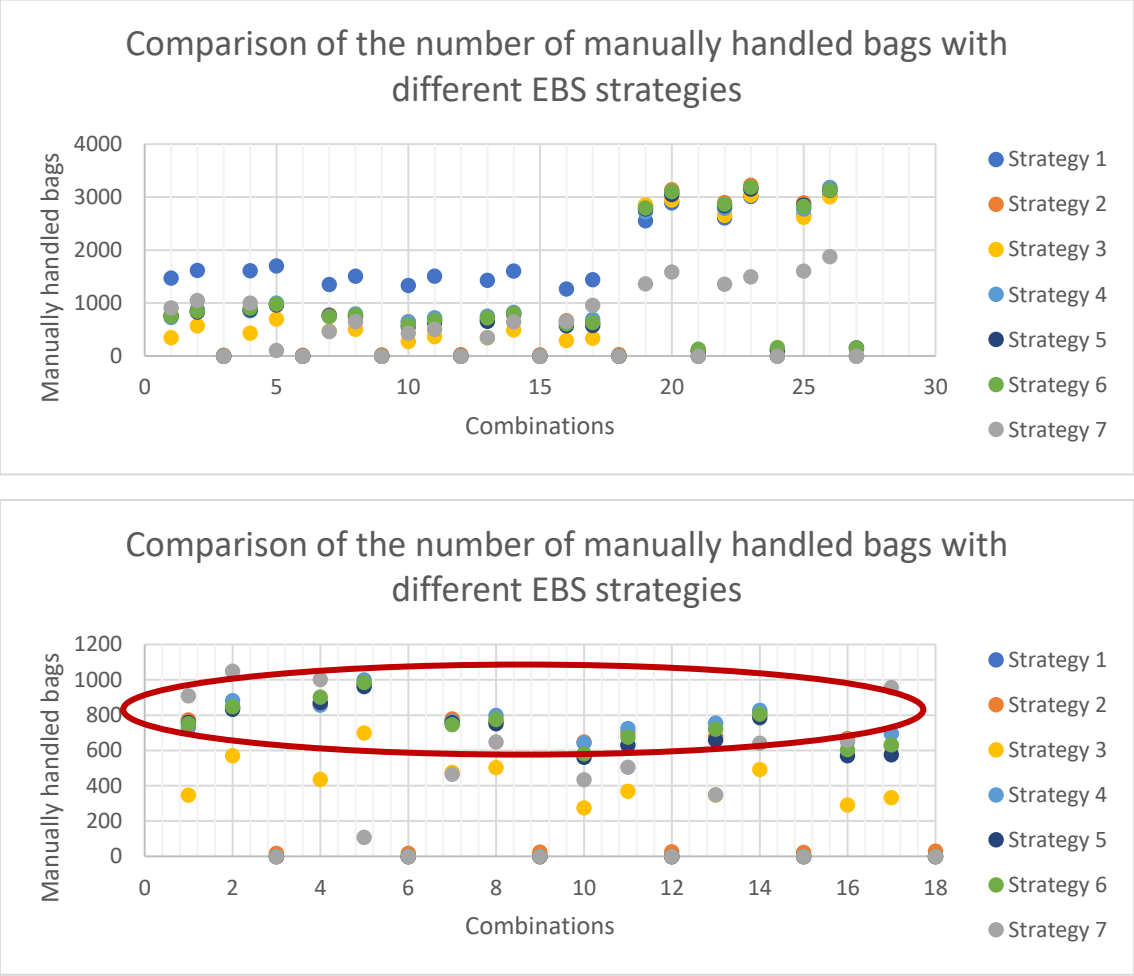


Figure 23. Combination of manually handled bags with different EBS strategies. Combinations excepting "way 3". Highlighted the results for strategy 4- Latest possible belt.

The fact that the latest belt strategy keeps the bags for longer times than other strategies means that the bags are using the storage space longer, and with a high input of bags, the system does not have enough capacity, resulting in a high number of manually handled bags.

With this strategy, using latest belt and most space belt have also been tested, the results are shown in appendix 9.8.1.5. In most of the combinations tested the second modification using

the latest belt if there is not space in the eligible belt works better as it shown in the graph below:

With the second combination, the bag would be allocated to the latest belt eligible if there is capacity, and if there is not capacity in any of the eligible belts, it would go to the latest belt not eligible.

All in all, since the latest belt strategy performs worse compared to other strategies, it will not be implemented.

6.6.1.2.5 Strategy 5- Least filled belt (less bags)

Strategies least filled belt (less bags) and most space belt show a similar number of manually handled bags, showing the last strategy a slightly smaller number, as it can be seen in the graph:

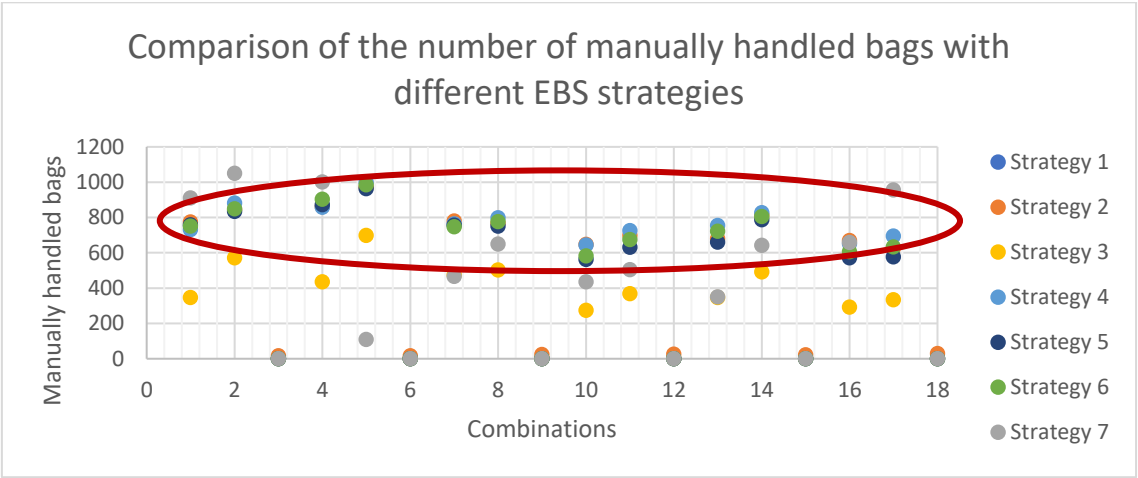
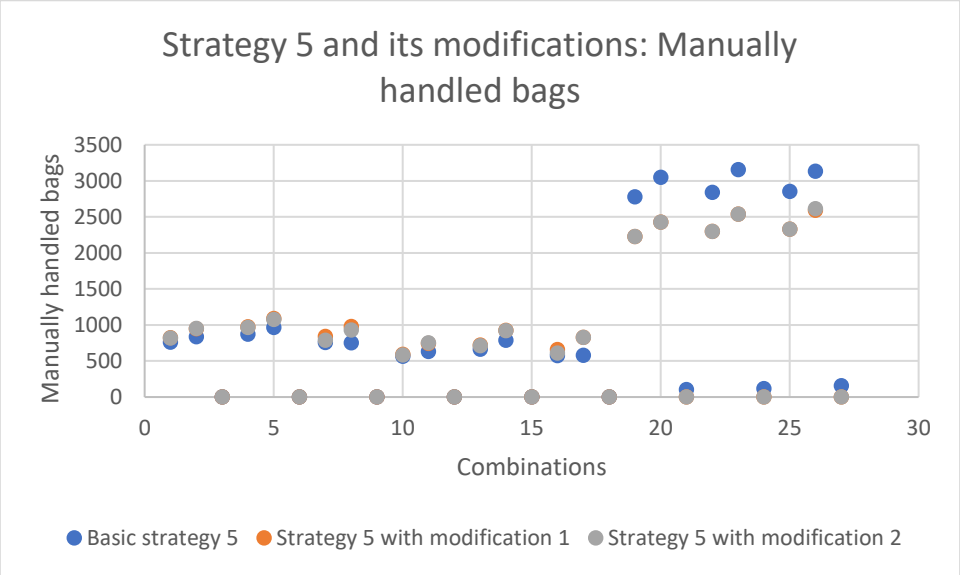
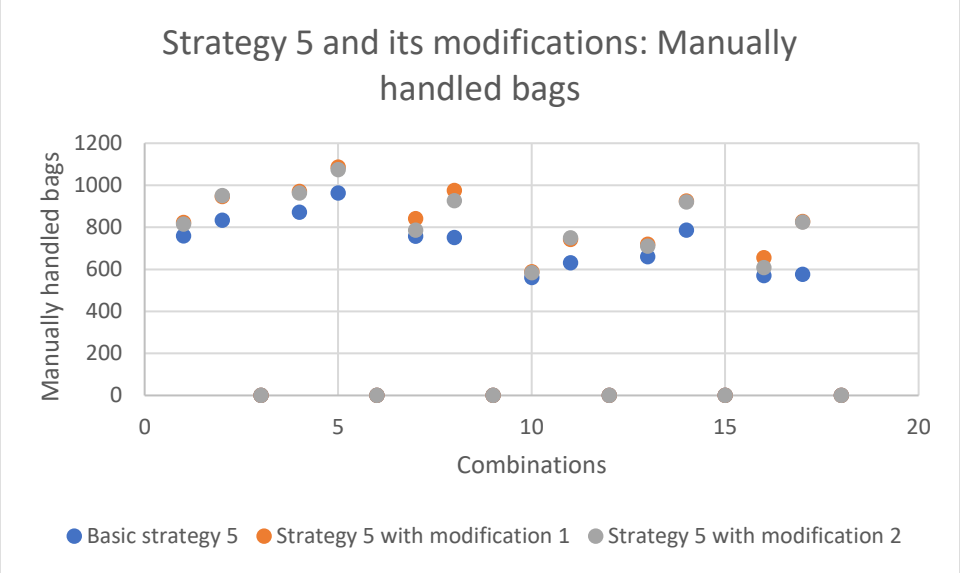


Figure 24. Comparison of the number of manually handled bags with different EBS strategies. Combinations excepting "Way 3". Highlighted the results for strategy 5- Belt least filled (less bags) and strategy 6- Most space belt.

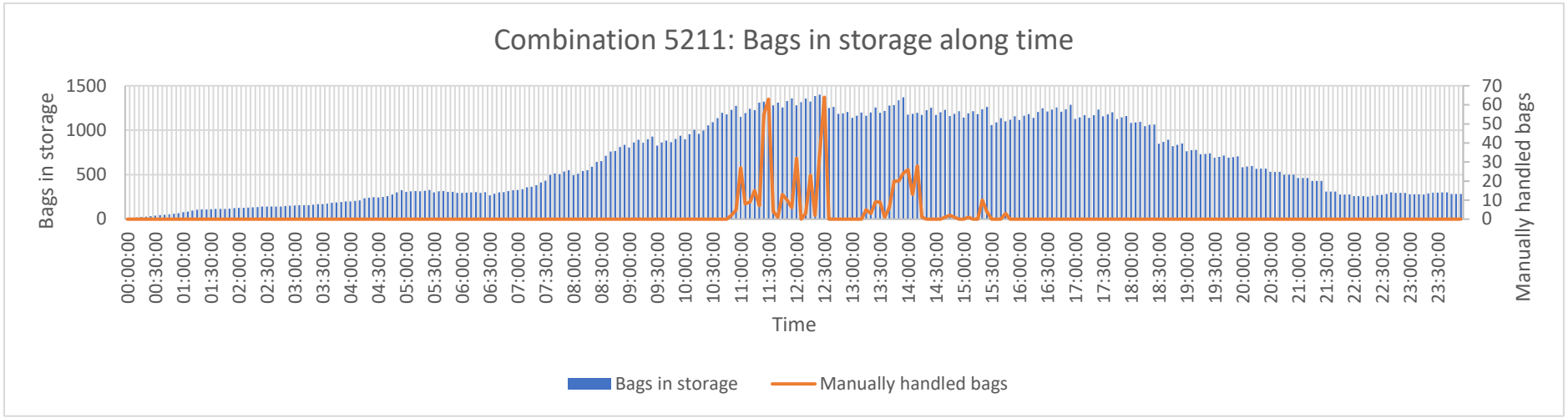
They have also been modified to include latest belt strategy or most space as a second strategy when none of the belts determined by the strategy have free space. The strategy with

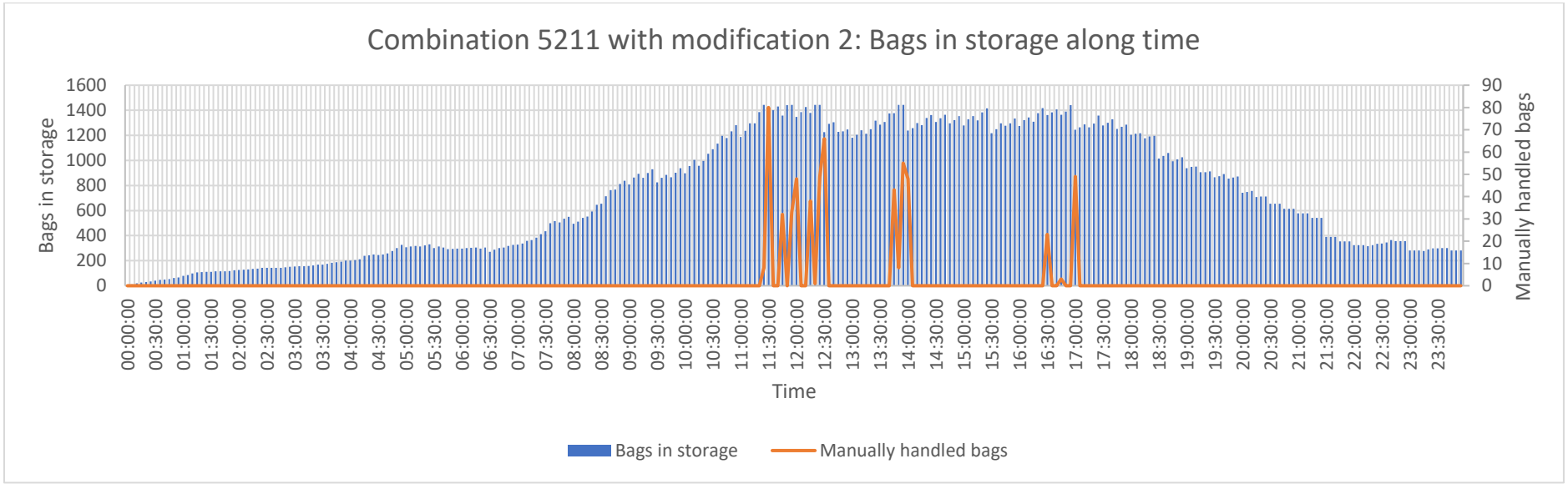
modifications presents a higher number of manually handled bags, which results since it presents more frequent low peaks of manually handled bags rather than high peaks. The difference between the basic strategy and including modifications is not significant, as the difference is approximately of 20 bags. The results for least filled belt are found in appendix 9.8.1.5. and it can be seen in the graphs, where the second has the scale augmented and does not shown the values for combinations with "way 3":

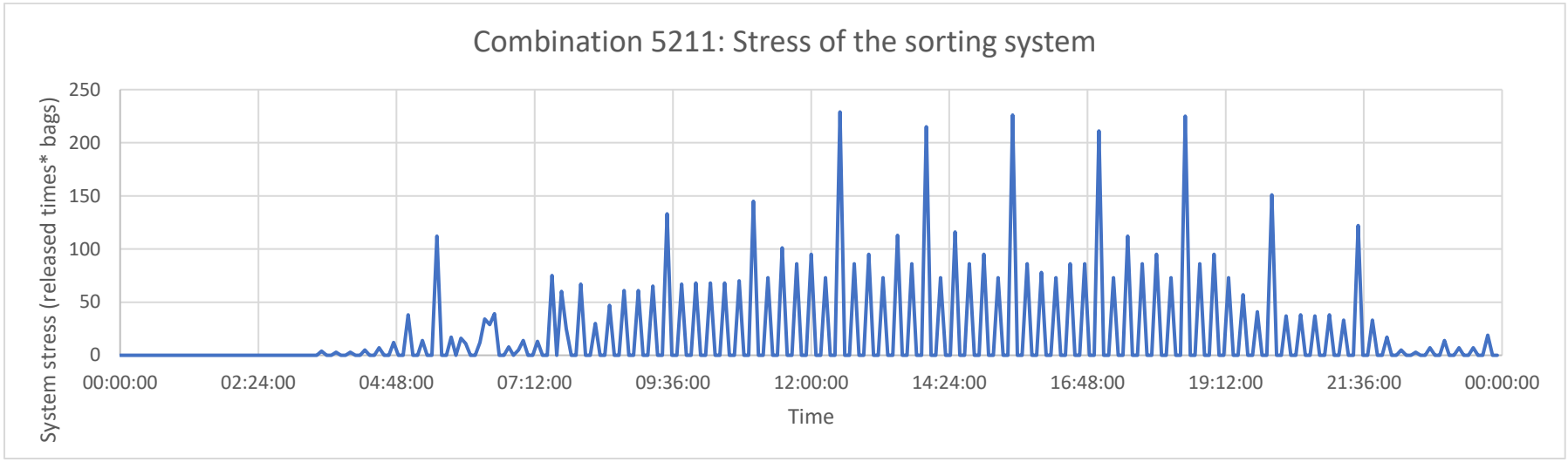


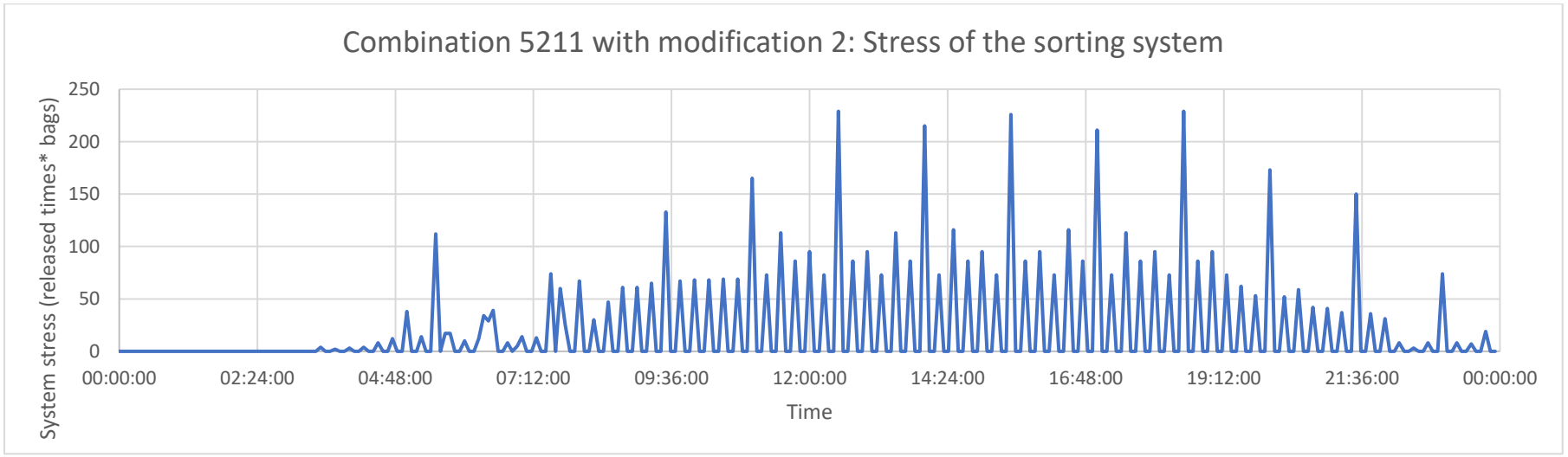


Looking at the behaviour along time and as expected, least filled belt strategy accomplishes its purpose of balancing the belts and maintaining a constant level of stress of the sorting system, as every time one belt is released, a similar number of bags is loaded into the sorting system. The behaviour of the system along time can be seen in the graphs in the following page.


Combination 5211: Bags in storage along time
Bags in storage
Manually handled bags
Time
Bags in storage
Manually handled bags


Combination 5211 with modification 2: Bags in storage along time
Bags in storage
Manually handled bags
Time
Bags in storage
Manually handled bags


Combination 5211: Stress of the sorting system
System stress (released times* bags)
250
200
150
100
50
0
00:00:00
02:24:00
04:48:00
07:12:00
09:36:00
12:00:00
14:24:00
16:48:00
19:12:00
21:36:00
00:00:00
Time


Combination 5211 with modification 2: Stress of the sorting system
System stress (released times* bags)
250
200
150
100
50
0
00:00:00
02:24:00
04:48:00
07:12:00
09:36:00
12:00:00
14:24:00
16:48:00
19:12:00
21:36:00
00:00:00
Time

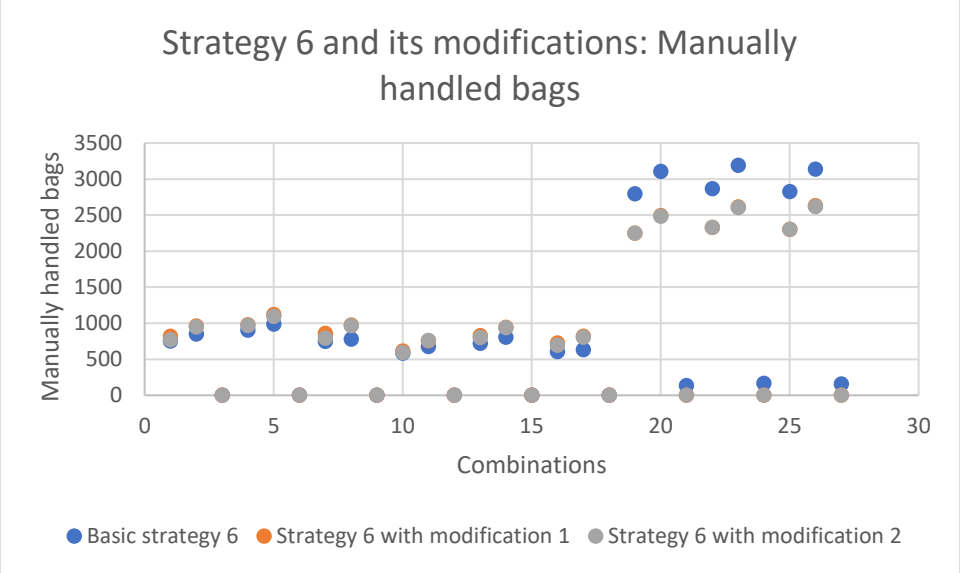
It is seen that with the second modification the total number of peaks in manually handled bags is lower, but the number of manually handled bags in each peak is higher.

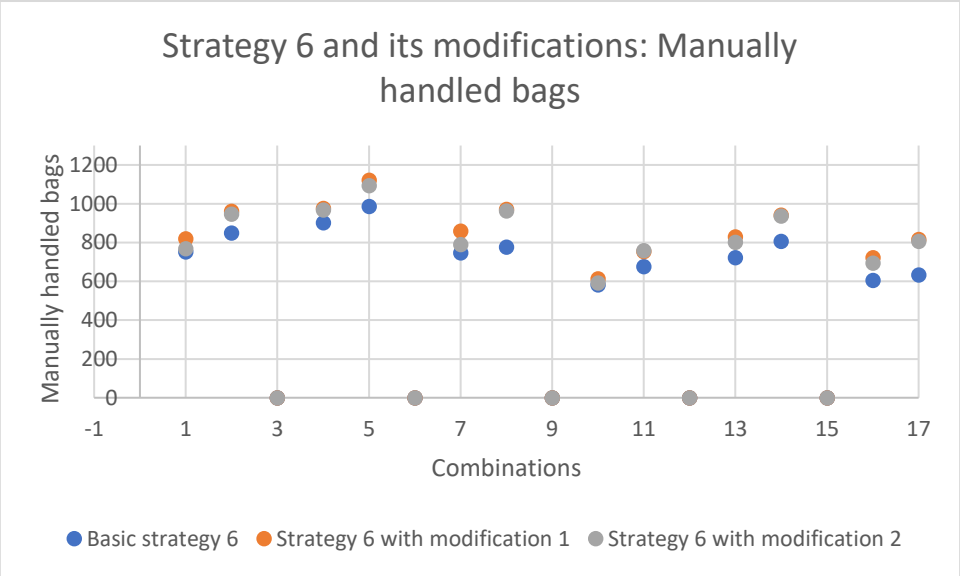
Summarizing, the combination of having a strategy that balances the belts, with a second one in the case all eligible belts are full that assigns it to the most space belt results in a low number of belts releasing at the same time. The system is overall balanced: the main strategy balances load among the belts and, when there is not space in the eligible belts, the bags allocated to the same chute generally go to the same belt. The overall balance of the system is a significant advantage, but it is necessary to remember that there are other strategies which perform better in manually handled bags.

6.6.1.2.6 Strategy 6- Most space belt

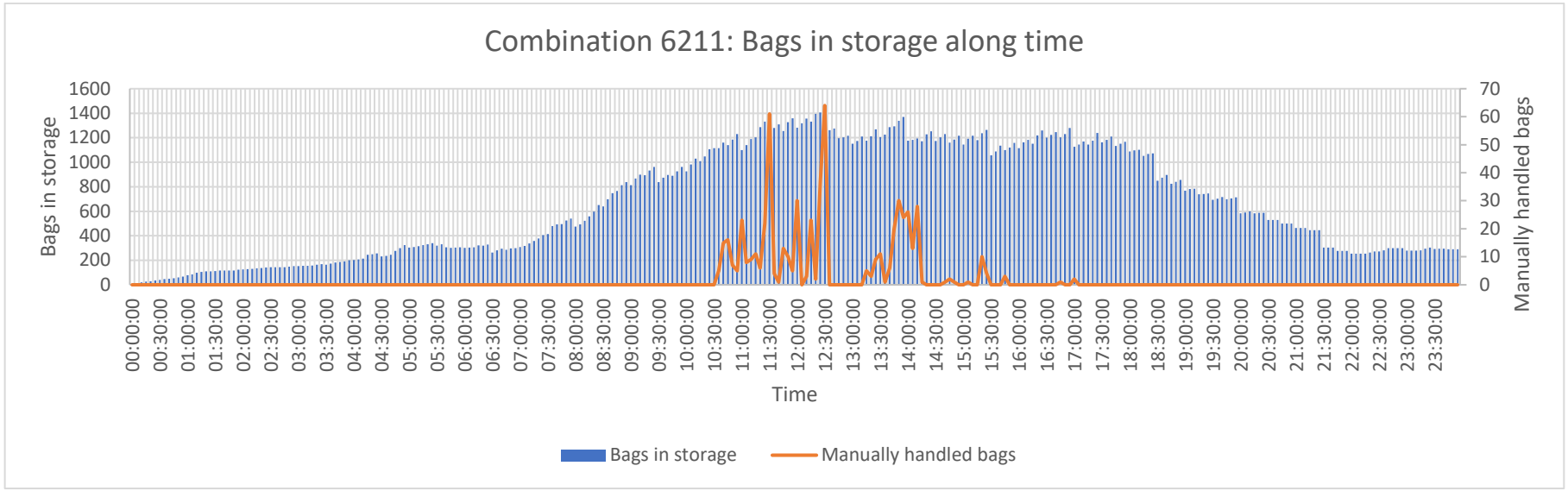
This strategy shows slightly better results regarding to manually handled bags than least filled strategy. These are found in appendix 9.8.1.6.

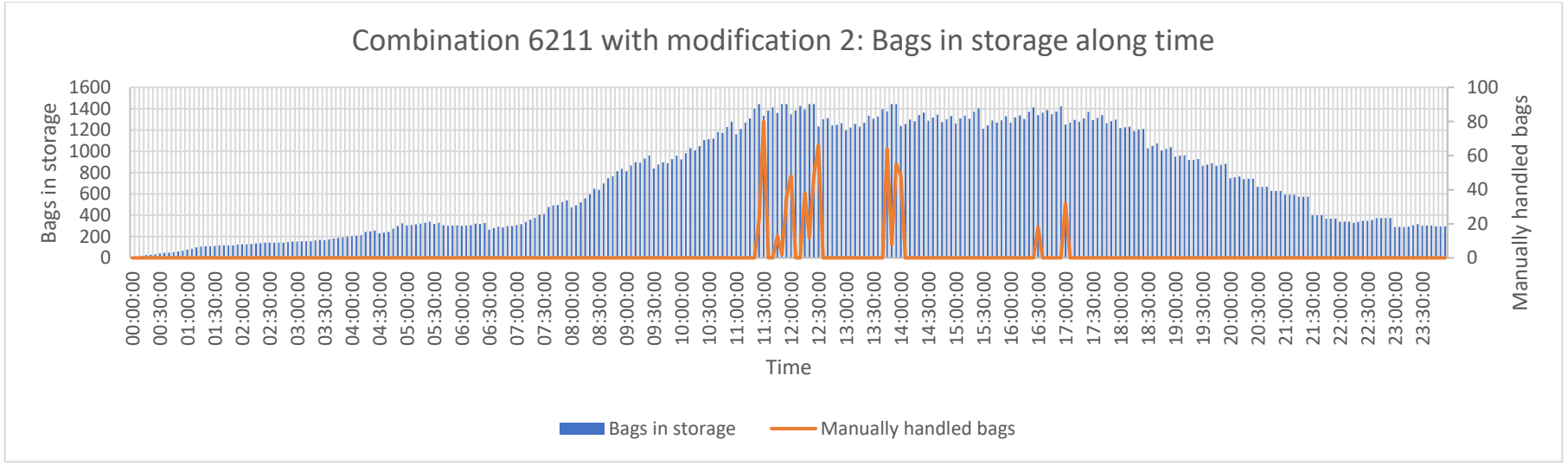
Looking at the performance along time, it does not present any peaks when loading to the sorting system and only one belt releases at a time. Modifications including the last belt strategy and the most space belt strategy as a second strategy, when there is no space in the eligible belt have been included. It can be seen in the following graphs, where in the second the scale is augmented and strategies with “way 3” are not shown:

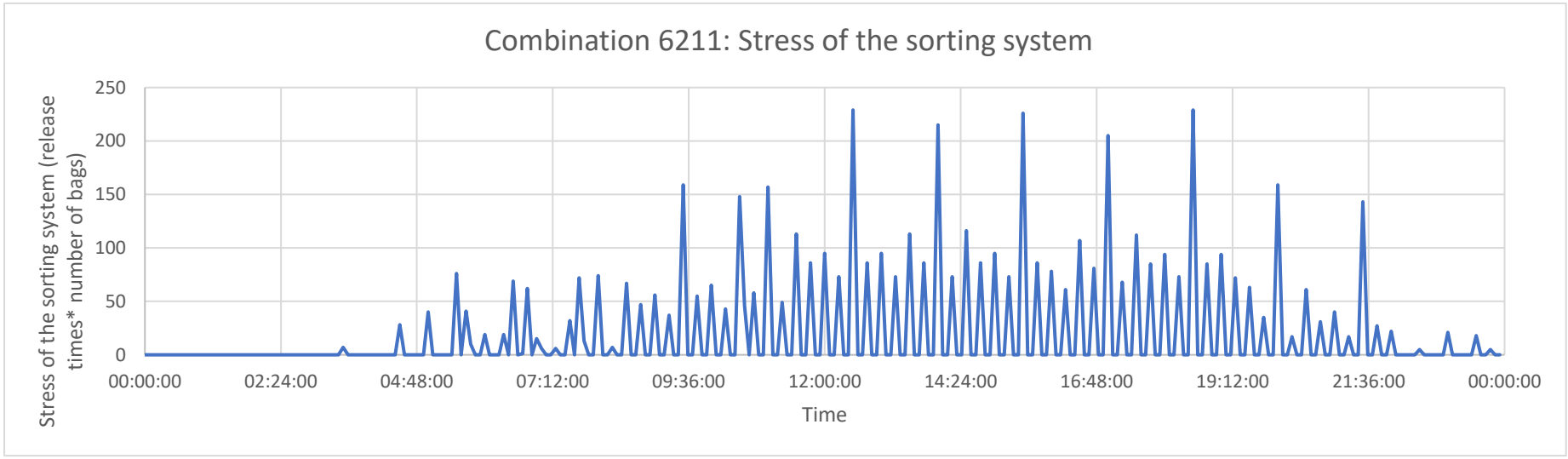


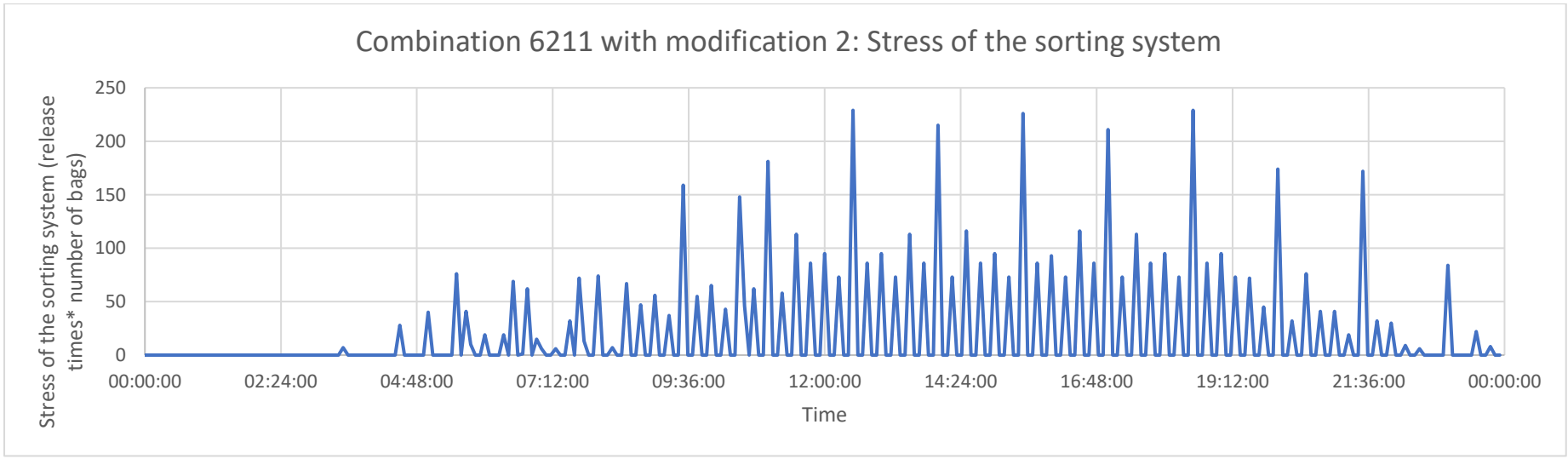


The strategies with modifications generally show a worse performance of manually handled bags than the original strategy, again, for the fact that it shows higher peaks of manually handled bags. It can be seen in the performance along time that with the modification (adding the most space belt strategy if none of the eligible belts has free space) the peaks in manually handled bags are higher but they are lower in number. It can be seen in the following page:


Combination 6211: Bags in storage along time
Bags in storage
Manually handled bags
Time
Bags in storage
Manually handled bags


Combination 6211 with modification 2: Bags in storage along time
Bags in storage
Manually handled bags
Time
Bags in storage
Manually handled bags


Combination 6211: Stress of the sorting system
Stress of the sorting system (release times* number of bags)
250
200
150
100
50
0
00:00:00
02:24:00
04:48:00
07:12:00
09:36:00
12:00:00
14:24:00
16:48:00
19:12:00
21:36:00
00:00:00
Time


Combination 6211 with modification 2: Stress of the sorting system
Stress of the sorting system (release times* number of bags)
250
200
150
100
50
0
00:00:00
02:24:00
04:48:00
07:12:00
09:36:00
12:00:00
14:24:00
16:48:00
19:12:00
21:36:00
00:00:00
Time

6.6.1.2.7 Strategy 7- Current CPH Airport strategy

The current strategy shows the best performance regarding to manually handled bags, results included in appendix 9.8.1.7. The results of the current Copenhagen Airport Strategy compared to the rest of strategies are seen in the following graph:

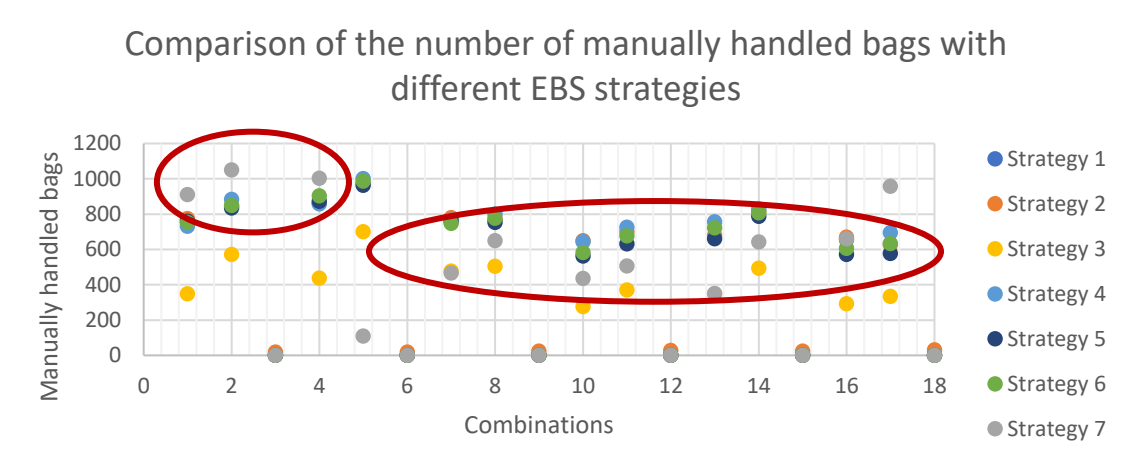


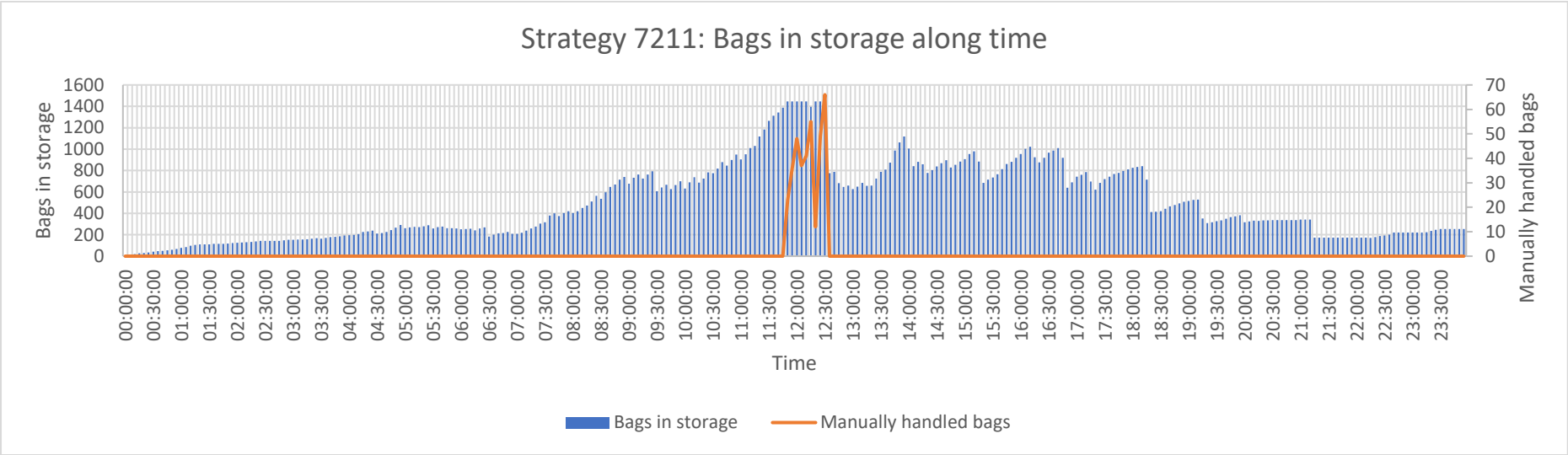
Figure 25. Comparison of the number of manually handled bags with different EBS strategies. Highlighted the results for the strategy currently implemented in CPH Airport

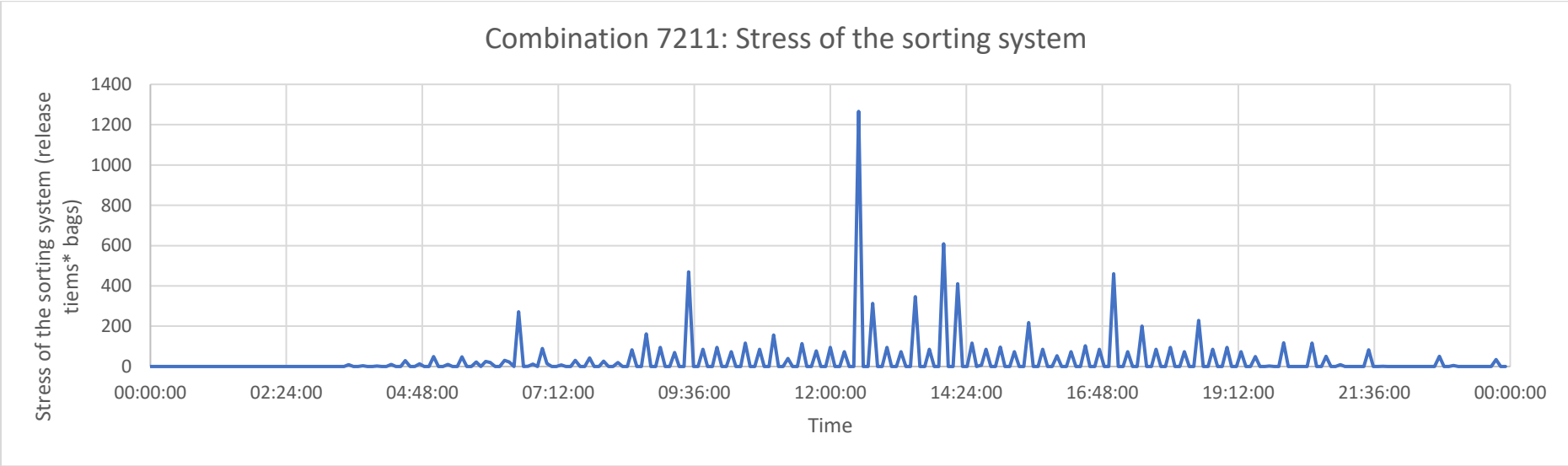
The strategy is one of the bests in manually handled bags because as it is based in the earliest belt principle, bags stay for a short time in the EBS. The longer a bag stays in the EBS is since it arrives until 45 minutes before its chute closing time.

However, there is a main drawback when analysing the behaviour of the strategy along time: several belts can release bags at the same time. As the strategy changes the release labels of the belts when a bag goes to a belt whose release time is later than 45 min before chute closing time, several belts can have the same release label, and then, will release at the same time.

This behaviour is seen in the graphs in the following page. In the graph representing the bags in storage along time it is seen that there are sharps drops in the number of bags in storage, which indicates that many belts are releasing at the same time.

The number of bags that are released at a given point in time are also shown in the second graph representing the stress of the system. It is seen that there is one high peak of stress in the system, where more than 1200 bags are loaded in the sorting system at a time. This situation would be infeasible in the real system and it would require manual intervention to solve it, as only two belts can be emptied at the same time.


Strategy 7211: Bags in storage along time
Bags in storage
Manually handled bags
Time
Bags in storage
Manually handled bags


Combination 7211: Stress of the sorting system
Stress of the sorting system (release tiems* bags)
Time

Among all the combinations tested, the one that performs better is 721X, with “way 2” of release labels and alternating lower and higher capacities.

Although this strategy is the one that keeps the bags for the least time allowing more free space in the EBS, it needs manual management to solve the situations where many belts have the same release time. Therefore, the idea of using dynamic release labels will be considered, but the current strategy by itself, without manual changes, is not feasible in the real system.

6.6.1.3 Comparison of the best strategies

Firstly, the criteria to discard strategies is the number of manually handled bags. This criterion indicates the number of bags that had not any space in the EBS when arriving, so when minimizing it, the utilization of the EBS lanes and of the whole system is being maximized.

Taking the number of manually handled bags as the criteria to discard strategies, it can be said that strategies totally random, random among eligible and latest possible belt are directly discarded due to its significantly higher values.

Among the rest of strategies, least filled belt and most space, show a good and similar performance. The number of manually handled bags is lower for the earliest strategy and current CPH Airport strategy. Among these, the last two show the best results regarding manual handled bags.

The reason why the last two are the best is the short time bags are stored, compared to other strategies. Both are based on the principle of releasing the bags the earliest possible, with the first possible release label once the chute is opened, minimizing the time the bag spends in storage. This performance is very useful in the peak of bag arrival at midday. However, the current strategy has the big disadvantage of assigning the same release label to many belts. This disadvantage results in manual interaction of the EBS managers to redefine the release labels of the belts.

Therefore, according to the manually handled bags criterion, it can be concluded that the earliest belt strategy and the current strategy with manual interaction of the EBS managers will be the best strategies.

Secondly, with the performance along time as the second criterion to discard strategies, the same conclusion is obtained. In other words, the number of times the belts have planned releases and panic time releases also shows that earliest belt-based strategies are the best.

Random strategies (totally random and random among eligible) are discarded because releases by panic time are frequent and uncontrolled. Latest belt strategy is also discarded because of the high utilization of the rest belts: as bags are allocated to the last possible belt, several are allocated to the belts covering longer timespans, which results in higher utilizations of these, and many times being derived to the rest belts when there is not space in the longer timespan belts. Balancing strategies (least filled belt and most space belt) show a good performance along time, with a similar number of planned release times and panic times than the earliest belt-based strategies.

Therefore, summarizing, the earliest possible belt and the current CPH Airport strategy are the strategies that perform best.

Among all the combinations possible for these two strategies, the best combinations are 321X and 721X. These combinations include “way 2” and “capacity 1”.

“Way 2” is the definition of release labels where there are seven frequently released belts with a frequency of 15 min, five not frequently released belts covering a time span of 1,5 hour each and two rest belts.

“Capacity 1” is the matching of capacities with alternatively high and low values, and with the highest values of capacity for the rest belts. It shows a good performance as the sum of two consecutive belts capacity will be similar for all belts. The longer timespan belts also have higher capacities than the frequently released belts.

All in all, both earliest belt strategy and the current CPH Airport strategy with manual interaction have proven to perform well in the EBS system with “way 2” of release labels and alternatively low and high capacities.

6.6.1.4 Testing of the strategies against arrival rate

It was checked that the best strategies (earliest belt and current strategy) work for all arrival rates, modifying the arrival rate of the bags arriving to EBS by 10% and by 20%. The number of manually handled bags for the combinations given when the arrival rate changes are given in the following tables:

For Earliest possible belt strategy:

Combination	Normal day	Arrival rate decreased by 10%	Arrival rate decreased by 20%	Arrival rate increased by 10%	Arrival rate increased by 20%
3211	205	205	0	549	1056
3212	404	0	0	1047	1593
3213	0	0	0	0	0

Table 2. Number of manually handled bags for Earliest possible belts strategy when changing the arrival rate.

And for the current CPH Airport strategy:

Combination	Normal day	Arrival rate decreased by 10%	Arrival rate decreased by 20%	Arrival rate increased by 10%	Arrival rate increased by 20%
7211	435	37	0	688	1119
7212	505	0	0	1047	1593
7213	0	0	0	0	0

Table 3.Number of manually handled bags for the Current CPH Airport strategy when changing the arrival rate.

As expected, the number of manually handled bags increases with the arrival rate and diminishes with the reduction of the arrival rate. It is seen that the increase in the number of manually handled bags is not directly related to the increase in the bags arriving to the EBS.

For the case when the system is overloaded, the number of bags is more than doubled for both strategies when there is an increase of 10% in the bag arrival to the EBS. However, when the increase is of 20%, the number of manually handled bags does not double again, but slightly increases 500 bags approximately. This is shown on the combinations 3212 and 7212.

When the system is not excessively overloaded, which occurs in 3211 and 7211, the number of manually handled bags is directly related to the bag arrival increase.

6.6.1.5 Conclusion on the best EBS strategy

When comparing all strategies tested, both earliest belt strategy and the strategy Copenhagen Airport is currently using show the best performance in the EBS regarding manual handling bags, utilization and release times.

The only inconvenient of the strategy currently implemented in the airport is that it requires manual intervention when there are more than two belts with the same release time. The solution to it to automatically avoid having more than two belts with the same release label when dynamically allocating the labels. This would mean to include a check before allocating a new release time so that the number of belts with that release time is lower or equal than two. In the case that there are two belts with the release time selected, the new belt would have the release time plus five minutes. Then, it would be emptied five minutes later.

This check before allocating the release label would not considerably affect the system as it is based in the earliest belt possible principle and it would solve the problem of too many belts releasing at the same time. On the contrary, it would ease the manual intervention of the EBS management.

6.6.2 Early Baggage storage used as a buffer

In this section the results for the second problem will be presented. It will be showed how it performs depending on the chute opening time. The test was done for all the bags arriving to the luggage handling system, and not only to the arriving to the EBS, as it is expected that a higher number of bags will go to EBS if there is a reduction of chute opening time.

To ease the understanding of the implications of changing the chute opening time, the section will be divided into manual handled bags, EBS storage along time and stress of the system.

6.6.2.1 Effect on the number of manually handled bags

The number of manually handled bags for both early possible belt and current CPH Airport strategy are included in the following tables:

For earliest possible belt strategy:

Combination	Normal day	Chute opening time extended 15 min	Chute opening time extended 30 min	Chute opening time reduced 15 min	Chute opening time reduced 30 min
3211	205	0	0	1464	3208
3212	404	0	0	2106	3879
3213	0	0	0	0	0

For current CPH Airport strategy:

Combination	Normal day	Chute opening time extended 15 min	Chute opening time extended 30 min	Chute opening time reduced 15 min	Chute opening time reduced 30 min
7211	435	0	0	1440	3208
7212	505	0	0	1705	3879
7213	0	0	0	0	0

Regarding the results for manually handled bags when changing the chute opening times, it is clear that there is a high dependency on the opening time.

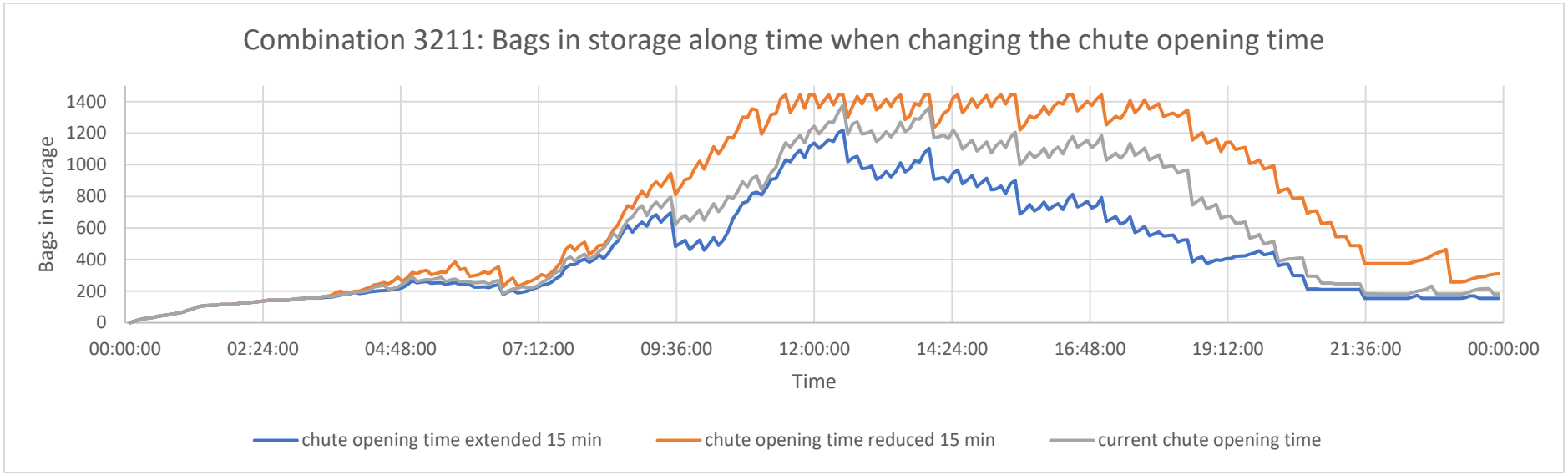
When increasing chute opening time at least 15 min, there are no manually handled bags. It is due since many bags go directly to its chute and do not go to the EBS. Therefore, the total number of bags stored in the EBS is lower and the system is not overloaded.

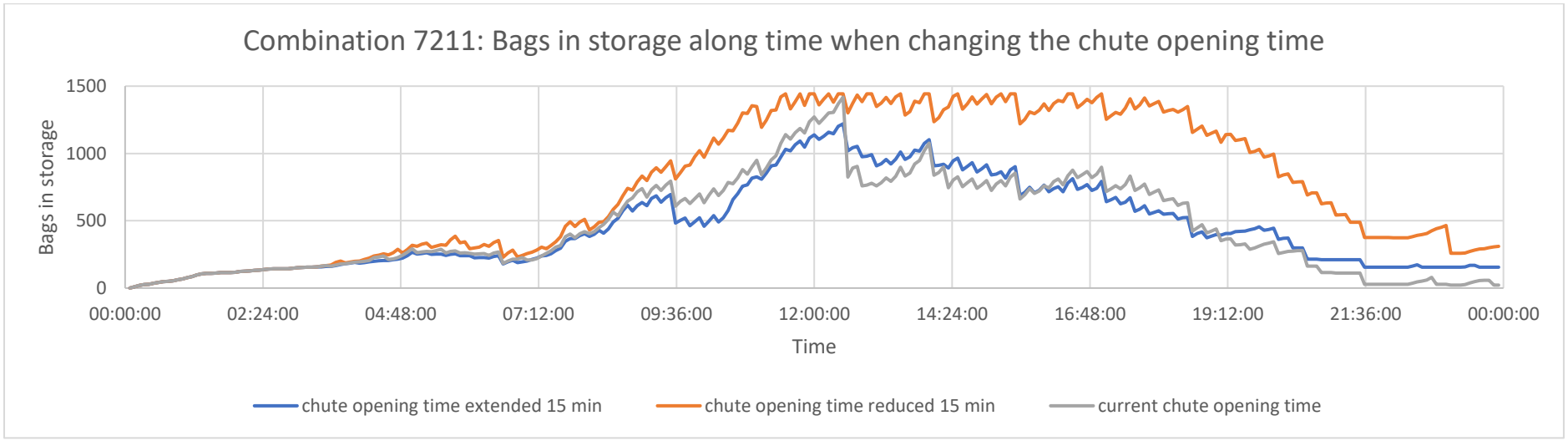
On the contrary, when the chute opening time is reduced, the number of manually handled bags shows a sharp increase because the system stores a higher number of bags and does not have enough capacity for all the incoming bags.

Thus, it can be concluded that it is not convenient to reduce chute opening times when bag arrival is high. If done, the airport luggage handling staff should be prepared for a high load of manually handled bags.

6.6.2.2 Effect on the number of bags in storage in EBS along time

Regarding to the number of bags in storage in the EBS along time, it can be seen that the number of bags along time is increased with the reduction of the chute opening time. The number of bags along time and the planned releases follow almost the same pattern for both the earliest belt strategy and the current CPH Airport strategy. It can be seen in the following page:


Combination 3211: Bags in storage along time when changing the chute opening time
Bags in storage
1400
1200
1000
800
600
400
200
0
00:00:00
02:24:00
04:48:00
07:12:00
09:36:00
12:00:00
14:24:00
16:48:00
19:12:00
21:36:00
00:00:00
Time
chute opening time extended 15 min
chute opening time reduced 15 min
current chute opening time


Combination 7211: Bags in storage along time when changing the chute opening time
Bags in storage
1500
1000
500
0
00:00:00
02:24:00
04:48:00
07:12:00
09:36:00
12:00:00
14:24:00
16:48:00
19:12:00
21:36:00
00:00:00
Time
chute opening time extended 15 min
chute opening time reduced 15 min
current chute opening time

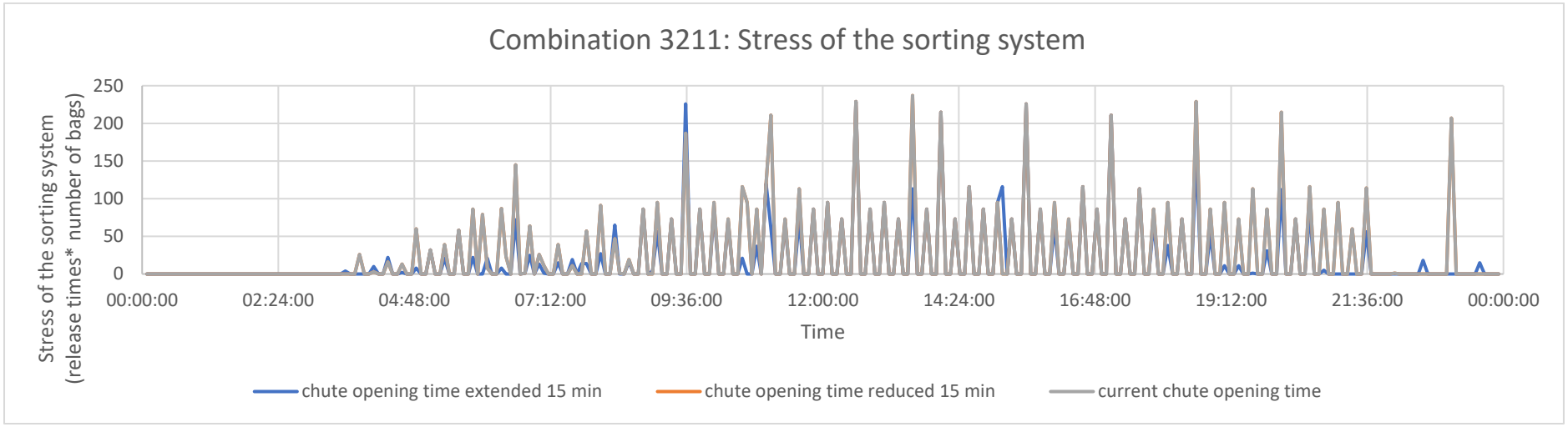
6.6.2.3 Effect on the stress of the sorting system along time

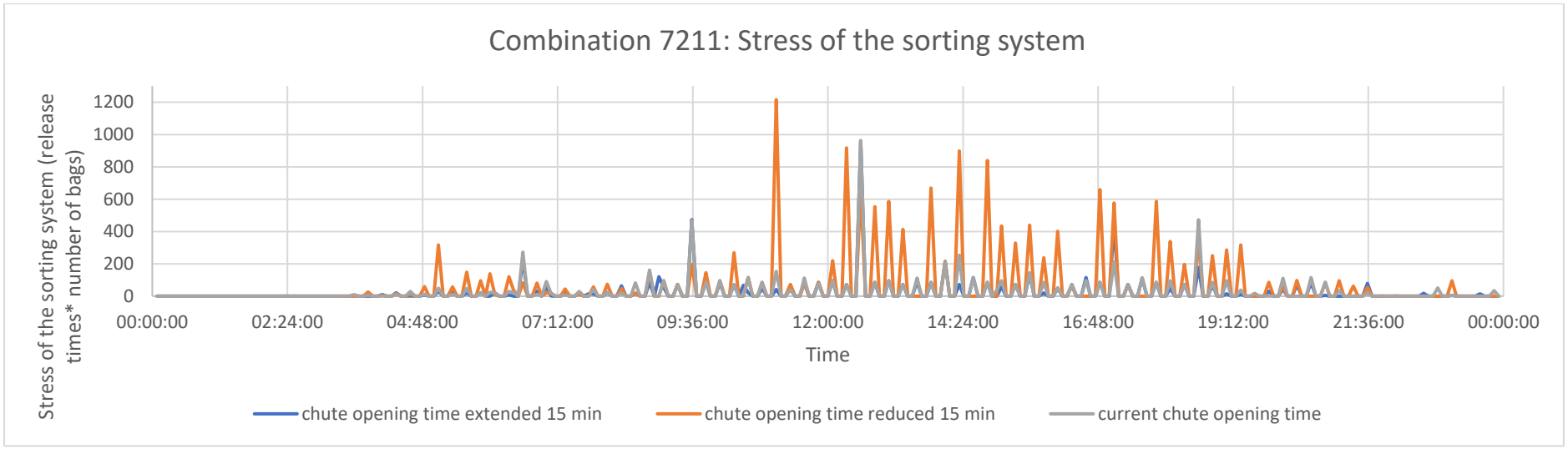
Regarding to the stress of the system, there is a high difference among how each of the strategies are affected.

The earliest belt strategy is not highly affected by the change of the chute opening time. When increasing or reducing the chute window times, the pattern the stress of the sorting system is the same, with a small increase of the peaks of stress.

However, the strategy currently implemented in Copenhagen Airport is highly affects the sorting system when changing the chute opening times. When the chute opening time is reduced by 15 minutes, the peaks are increased 5 times. This indicates that in the system there would be more than two belts releasing at the same time, which as explained before, is an infeasible situation. It would require manual interaction to allocate the release labels to avoid several belts releasing at the same time.

The graphs for both strategies can be seen in the following page.


Combination 3211: Stress of the sorting system
Stress of the sorting system (release times* number of bags)
250
200
150
100
50
0
00:00:00
02:24:00
04:48:00
07:12:00
09:36:00
12:00:00
14:24:00
16:48:00
19:12:00
21:36:00
00:00:00
Time
chute opening time extended 15 min
chute opening time reduced 15 min
current chute opening time


Combination 7211: Stress of the sorting system
Stress of the sorting system (release times* number of bags)
1200
1000
800
600
400
200
0
00:00:00
02:24:00
04:48:00
07:12:00
09:36:00
12:00:00
14:24:00
16:48:00
19:12:00
21:36:00
00:00:00
Time
chute opening time extended 15 min
chute opening time reduced 15 min
current chute opening time

6.6.2.4 Conclusion of Early Baggage Storage used as a buffer

As it has been shown in the results, the possibility of using the Early Baggage Storage as a buffer is feasible. As the utilization of the EBS is not 100%, there is some margin that can be used as a buffer. Using the EBS as a buffer would increase the efficiency of the chutes, as less chutes are needed.

It is possible to reduce the chute opening times and keep more bags in the EBS, but it would also carry some consequences: high increase in the number of manually handled bags if the system is overloaded, high stress of the sorting system and high utilization of the EBS.

These consequences are not presented as a disadvantage, but it is needed to note that the current EBS strategy implemented in Copenhagen Airport is highly affected by the reduction of the chute opening times. Manual interaction to allocate the release labels so no more than two belts release at the same time would be needed, especially around midday and when the bag arrival is high.

On the one hand, the early baggage storage works better with high arrival of bags and does not require manual management of the release labels, which may be a possible solution.

On the other hand, it is possible to continue using the current strategy and to add the change where the system controls the number of belts that are releasing at the same time. If this change explained in the EBS strategies is included, no more than two belts would be emptied at the same time, and the strategy is expected to not be so highly affected by the chute opening times.

Either the earliest belt strategy or the current with the explained modification are presented as the best solutions for using the early baggage storage as a buffer.

7 IMPROVEMENTS, RECOMMENDATIONS AND CONCLUSION

Baggage handling is an important process among airport operations. It is one of the processes that is associated with customer satisfaction and, its bad performance is associated with high dissatisfaction.

In the Baggage Handling System many processes are carried, starting with the check-in of the luggage. In Copenhagen Airport terminal 3, the introduction of self-bag drop counters has resulted in an increase in the waiting times in the last counters in the direction of the conveyor belt. It is expected that more self-bag drop counters are included in the future, which would accentuate even more the issue. The present study has shown that the current performance, First Come First Served, has the most unfair distribution of waiting times among counters, especially when the bag arrival is close to the system capacity.

Different techniques have been tested in the present study to solve this issue using simulation tools. Among the strategies tested, those including forward reservation of windows showed the fairest waiting times among counters. These strategies are based on the idea that the counters can reserve an incoming free window according to a priority criterion. Among the strategies tested, forward reservation with a criterion with equally maximum wait and average wait weight showed the best performance regarding waiting times and throughput.

It is important to note that when using simulation tools, no optimal solution is obtained for a problem. The real behaviour of the system is analysed and only the performance of the techniques tested is obtained. As a consequence, there is no certainty that the strategy giving the best performance among the tested is indeed the best strategy for the given system.

Therefore, the forward reservation strategy weighting equally average wait and maximum wait it the one that performs best among the tested. Although there is no certainty that it is indeed the best window allocation strategy, its performance is much better than the current First Come First Served strategy.

All strategies tested have covered the main variables and goals possible: fair strategy with the intention to make a fair distribution of waiting times and forward reservation strategies prioritizing the distance in the direction of the conveyor belt, average wait and maximum wait. They have been compared against the benchmark of the current FCFS strategy and against the maximum utopic throughput.

All in all, in the check-in counters problem all main strategies have been tested and it has been proven that the need of change of strategy is clear. In further studies, a combination of forward reservation strategies could be tested, prioritizing a specific counter when it has a value of waiting time over a threshold, reorganizing the reservations.

Another process implicated in the Baggage Handling System is the Early Baggage Storage. It stores the bags that arrive before the chute allocated to its flight is open.

In Copenhagen Airport the EBS is composed by 14 conveyor belts of different capacities. This physical distribution of the storage and the time constraints of the bags form a complex system where it is critical to ensure the on-time arrival of the bags to its chutes while minimizing the stress of the sorting system when emptying the belts.

It results in a problem where it is needed to increase the efficiency of the EBS creating a belt allocation strategy and emptying strategy for the same.

The present thesis has included a deep study in the topic, testing many different belt allocation strategies, definition of release times for the belts and matching of the release labels with the capacities of the belts.

It was concluded that allocating an arriving bag to the belt with the earliest emptying time was the best strategy in addition to the current strategy implemented in CPH Airport, that is based in the same principle but requires manual interaction when redefining the emptying times. A recommended improvement for the present strategy in order to reduce the manual management with belt release labels is to add five minutes to the label when the number of belts that can release at a time is already covered. With this recommendation, the current system will improve reducing its manual management.

The results for this study are based on an iterative process covering the main scenarios possible, and then, it can be said that the obtained are the best results possible regarding the constraints of the system.

Along with the improvement of the efficiency of the EBS comes the possibility of using it as a buffer, so it can store bags if the chute opening window times are reduced. It was shown that it is a feasible possibility and how the system would be affected.

When reducing the chute opening times, the efficiency of the chutes is increased: the number of chutes needed to cover all flights would be reduced. Given the results of the present study on how the EBS is affected by the reduction of the chute opening time, further studies could cover the optimization of the chute opening time and the chute allocation problem together. The possibility to optimize these areas together would increase even more the efficiency of the system, which is needed as Copenhagen Airport has a limited space and more resources cannot be added to the same.

Summarizing, the present thesis includes a profound study in two areas of the baggage handling system: check-in counters and Early Baggage Storage. Check-in counters study consists of a window allocation algorithm to maximize throughput and a fair distribution of waiting time among counters while the Early Baggage Storage study consists of the bag allocation problem and emptying strategy, on the top of including the possibility of using the storage as a buffer to reduce the chute opening time.

The results obtained show that there is margin for improvement in the areas, and following the recommendations explained, the efficiency of the baggage handling system will increase. Further studies on the topics have also been presented so that even more improvements can be found in the future.

8 BIBLIOGRAPHY

[1] CPH Airport, “CPH Airport stadistics webpage,” 23 08 2018. [Online]. Available: https://www.cph.dk/en/about-cph/investor/traffic-statistics/2018.

[2] ACI Europe, “ACI Europe Stadistics,” [Online]. Available: https://www.aci-europe.org. [Accessed 23 8 2018].

[3] A. Valentin and E. Verbraeck, “Simulation building blocks for airport terminal modeling,” *Proceedings of the Winter Simulation Conference, San Diego, CA, USA, 2002,* vol. 2, pp. 1199-1206.

[4] N. Doshi and R. Moriyama, “Application of simulation models in airport facility design,” *Proceedings of the Winter Simulation Conference, San Diego, CA, USA, 2002,* vol. 2, pp. 1725-1730.

[5] S. Appelt, R. Batta, Li Lin and C. Drury, “Simulation of passenger check-in at a medium-sized US airport,” *2007 Winter Simulation Conference, Washington, DC, 2007,* pp. 1252-1260, 2007.

[6] A. Ascó, J. Atkin and E. Burke, “The airport sorting allocation problem,” *Proceedings of the 5th Multidisciplinary International Scheduling Conference, MISTA,* 2011.

[7] A. Abdelghany, K. Abdelghany and R. Narasimha, “Scheduling baggage-handling facilities in congesting airports,” *J. Air Transport Management,* no. 12, pp. 76-81, 2006.

[8] M. Savrasovs, A. Medvedev, E. Sincova and L. D. Pedrera, “Riga airport baggage handling system simulation,” *J. Otamendi, A. Bargiela, J. Montes (Eds) Proceedings 23rd European Conference on Modelling and Simulation ECMS,* 2009.

[9] J. P. Cavada, C. E. Cortés and P. A. Rey, “A simulation approach to modelling baggage handling systems at an international airport,” *Simulation Practice and Theory,* no. 75, pp. 146-164, 2017.

[10] T. L. Shell, B. G. Chaw and C. Grammich, “Designing airports for security: an anlysis of proposed changes at LAX,” *Rand Issue Paper,* pp. 1-8, 2003.

[11] S. Hafizogullari, G. Bender and C. Tunasar, “Simulation's role in baggage screening at the airports: a case study,” *Proceedings of the 2003 Winter Simulation Conference,* pp. 1833-1837, 2003.

[12] D. Li and H. Chen, “Review of security inspection networking system development in china ariport and its trend in the future,” *CCST 2005, 39th International Carnahan Annual Conference on Security Technology,* pp. 178-181, 2005.

[13] M. Johnstone, D. Creighton and S. Nahavandi, “Status-based routing in baggage handling systems: searching verses learning,” *IEEE Transaction on Systems, Man and Cybernetics,* no. 40 (2), pp. 189-200, 2010.

[14] T. Zhang, Y. Ouyang and Y. He, "Traceable Air Handling System Based on RFID Tags in the Airport," *Journal of Theoretical and Applied Electronic Commerce Research,* vol. 3, pp. 106-115, 2008.

[15] S. Meshram, T. Gujar, P. R. Wankhede and A. Singh, "Baggage Tracing and Handling System using RFID and IoT for Airports," *2016 International Conference on Computing, Analytics and Security Trends (CAST),* 2016.

[16] Y. Zeinaly, B. D. Schutter and H. Helledoom, "A Model Predictive Approach for Baggage Handling Systems," *Proceedings of the 16th International IEEE Annual Conference on Intelligent Transportation Systems, Hague, Netherlands,* pp. 687-693, 2013.

[17] M. Johnstone, V. T. Le, J. Zhang, S. Nahavandi and D. Creighton, "A generalised data analysis approach for baggage handling systems simulation," *2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Seoul, 2012,* pp. 1681-1687.

[18] G. Aguilera-Venegas, J. L. Galán-García, E. Mérida-Casermeiro and P. Rodríguez-Cielos, "An accelerated-time simulation of baggage traffic in an airport terminal," *Mathematics and Computers in Simulation,* no. 104, pp. 58-66, 2014.

[19] V. T. Le, D. Creighton and S. Nahavandi, "Simulation-based input loading condition optimisation of airport baggage handling systems," *Proceedings of the IEEE Intelligent Transportation Systems Conference,* pp. 574-579, 2007.

[20] P. E. Joines and N. M. Van Dijk, "Simulation of Check-In at Airports," *Proceedings of the 2001 Winter Simulation Conference, Piscataway,* pp. 1023-1026, 2001.

[21] R. Xu and R. Piratesh, "Airline-catering plant material handling system analysis with simulation and scaled animation," *Proceedings of the 1991 Winter Simulation Conference,* pp. 402-410, 1991.

[22] G. G. Jing, W. D. Kelton and J. C. Arantes, "Modeling a controlled conveyor network with merging configuration," *Proceedings of the 1998 Winter Simulation Conference,* pp. 1041-1048, 1998.

[23] H. W. Chun and R. W. T. Mak, "Intelligent Resource Simulation for an Airport Check-In Counter Allocation System," *IEEEE Transactions On Systems, Man and Cybernetics, 29 (3),* pp. 325-335, 1999.

[24] M. Johnstone, D. Creighton and S. Nahavandi, "Simulation-based baggage handling sytem merge analysis," *Simulation Modelling Practice and Theory,* no. 53, pp. 45-59, 2015.

[25] J. C. Arantes and S. Deng, "Modeling and solution methods for the design and control of conveyor systems with merge configuration," *Progress in Material Handling Research ed R. J. Graves, L. F. McGinnis, D. J. Medeiros, R. Ward and M. R. Wilhelm Material Handling Institute,* pp. 35-50, 1996.

[26] J. T. Lin, Ping-Hsi Shish, E. Huang and Chun-Chih Chiu, "Airport baggage handling system simulation modeling using SysML," *2015 International Conference on Industrial Engineering and Operations Management (IEOM), Dubai, 2015,* pp. 1-10.

[27] M. D. Rossetti, Simulation modeling and Arena, Wiley, 2010.

[28] A. M. Law and W. D. Kelton, Simulation modeling and analysis, McGraw-Hill, 2000.

[29] J. Banks, J. S. Carson and B. L. Nelson, Discrete Event System Simulation, Prentice Hall, 1996.

[30] Anylogic, "Discrete Event Simulation in Anylogic," [Online]. Available: https://www.anylogic.com/use-of-simulation/discrete-event-simulation/. [Accessed 27 08 2018].

[31] Stadista, "Airline industry passenger traffic," [Online]. Available: https://stadista.com/stadistics. [Accessed 23 8 2018].

[32] V. S. Siromiatnikov, M. Ortega, L. N. Oleinikova, J. M. Garcia and F. Sandoval, "Using simulation to support optimization of conveyor system," *Second IAESTED International Multi-Conference, Novosibirsk, Russia, 2005,* pp. 248-253.

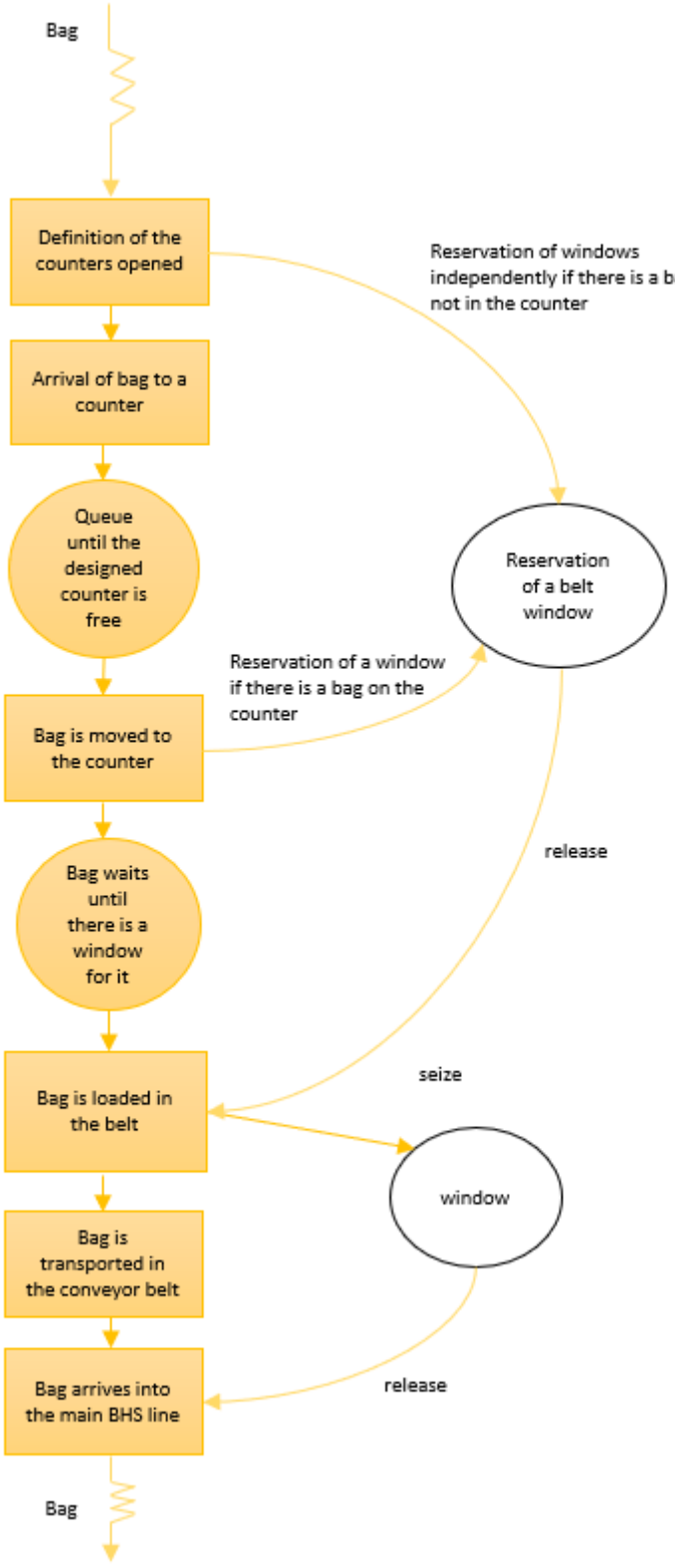
[33] M. Mujica Mota, "Check-in allocation improvements through the use of simulation-optimization approach," *Transportation Research Part A,* no. 77, pp. 320-335, 2015.

[34] I. Grigoryev, Anylogic 7 in three days, Anylogic, 2016.

[35] Anylogic, "Anylogic books," [Online]. Available: https://www.anylogic.com/resources/books/. [Accessed 27 8 2018].

9 APPENDICES

9.1 Check-in counters: window algorithm problem. Activity diagram



9.2 Check-in counters simulator in C++

```
/* ============================================================================
CHECK-IN COUNTERS SIMULATOR IN C++
============================================================================ */

/* ----------------------------------------------------------------------------
inclusion of library files
---------------------------------------------------------------------------- */

#define _CRT_SECURE_NO_WARNINGS
#include <cstdlib>
#include <cstdio>
#include <ctime>
#include <cstdarg>
#include <cstring>
#include <cmath>
#include <algorithm>
#include <list>
#include <vector>
using namespace std;

/* ----------------------------------------------------------------------------
parameters
---------------------------------------------------------------------------- */

#define COUNTERS        24    /* number of counters */
#define HORIZON       600000    /* how many steps do we simulate */ //500
hours* 3600s/h* iteration/3s = 600 000
#define NONE       -1
#define TRUE        1
#define FALSE       0
#define UTOPIA                  1
#define FAIR                    2
#define FCFS                    3
#define FRFAR                   4
#define FRAVGWAIT               5
#define FRMAXWAIT               6
#define FRAVGWAITMAXWAIT        7
#define fwindows    COUNTERS        //windows to reserve a part of the
reserved[]

/* ----------------------------------------------------------------------------
variables
----------------------------------------------------------------------------*/

FILE *out;
int belt[COUNTERS];             //contains if the window is free (0),
contains 1 bag (1) or more (2,3,4...)
```

```
int reserved[COUNTERS];         //contains the counter number for which
each window is reserved
int freserved[fwindows];        //contains the windows that can be used
for forward reservation, a part of the reserved[]
int bag[COUNTERS];              //contains if there is a bag in the
counter or not (defined by arrivals)
int queue[COUNTERS];            //contains the number of bags that are in
queue after the bag being checked-in
int wait[COUNTERS];             //contains the number of bags that are
waiting at a specific counter
int maxwait[COUNTERS];          //contains the maximum waiting time in
each of the counters
int totwait[COUNTERS];          //contains the total number of bags taken
away from the counter
int totbags[COUNTERS];          //contains the total number of bags that
waited at the counter (same bag waits two times or three...)
int prioritycounter[COUNTERS];  //contains the counters associated with
the priority values in priorityvaluesordered[COUNTERS]. The counter with
the highest criteria will be at the end
double priorityvalues[COUNTERS];//contains the values for priority
reservation. They are not ordered
double priorityvaluesordered[COUNTERS];//contains the values for priority
reservation. They are ordered from small to highest criteria of priority
double arrivalrate[COUNTERS];   //defines the arrival rate at each of the
counters (probability up to 100%)
double sumarrivalrate;          //contains the sum of all the arrivalrate
values in arrivalrate[COUNTERS]
int hasreserved[COUNTERS];      //contains if the counter reserved or not
in the last iteration (1= counter reserved a window)

/* ------------------------------------------------------------------
Struct
------------------------------------------------------------------ */

struct prioStruct {
int counter;
double prioValue;
prioStruct(int k, double s) : counter(k), prioValue(s) {}
bool operator < (const prioStruct& str) const {
return (prioValue < str.prioValue);
}
};


/* ------------------------------------------------------------------
macro functions
------------------------------------------------------------------ */

#define srand(x)     srand48(x)
```

```
#define randm(x)    (lrand48() % (long) (x))


/*-------------------------------------------------------------------------
sort the counters according to a priority criteria
-------------------------------------------------------------------------*/

void prioritize() {
vector<prioStruct> vec;
for (int i = 0; i < COUNTERS; i++) {
vec.push_back(prioStruct(i, priorityvalues[i]));
}
// Shuffle the structure so that the order of the elements is random
random_shuffle(vec.begin(), vec.end());
// Sort the structure based on priorityvalues. Stable sort makes sure
that the relative position between equal values is kept
stable_sort(vec.begin(), vec.end());
// Take out the counter number for from the structure back into the
prioritycounter array.
for (int i = 0; i < COUNTERS; i++) {
prioritycounter[i] = vec[i].counter;
}
}

/* ------------------------------------------------------------------------
init
-------------------------------------------------------------------------*/

void init(void) /* initialize all arrays */
{
for (int i = 0; i < COUNTERS; i++) {
belt[i] = FALSE;
reserved[i] = NONE;
hasreserved[i] = FALSE;
bag[i] = FALSE;
wait[i] = 0;
queue[i] = 0;
maxwait[i] = 0;
totwait[i] = 0;
totbags[i] = 0;
}
for (int i = 0; i < fwindows; i++) {
freserved[i] = NONE;
}
}

/* ------------------------------------------------------------------------
arrivalrate
------------------------------------------------------------------------ */
```

```
void arrivaldef() {     //changes the arrival rate for every counter
for (int i = 0; i < COUNTERS; i++) {
arrivalrate[i] = rand() % 101;
}
sumarrivalrate = 0;
for (int i = 0; i < COUNTERS; i++) {
sumarrivalrate = sumarrivalrate + arrivalrate[i];
}
}

/* ------------------------------------------------------------------------
arrivals
------------------------------------------------------------------------*/

void arrivals(int k, int STRESSRATE)
/* arrival of new bags */
{
double r;
for (int i = 0; i < COUNTERS; i++) {
if (bag[i] == TRUE) {   //if there is a bag in the counter, bags to the
queue
r = rand() % 101;
if (arrivalrate[i] == 0) {
/*nothing happens*/
}
else {
if (r < ((arrivalrate[i] * STRESSRATE) / (sumarrivalrate * 100))) {
queue[i] = queue[i] + 1;
}
}
}
if (bag[i] == FALSE) {  //there is no bag in the counter, bag to check-in
counter
r = rand() % 101;

if (arrivalrate[i] == 0) {
bag[i] = FALSE;
wait[i] = 0;
}
else  {
if ((r < ((arrivalrate[i] * STRESSRATE) / (sumarrivalrate * 100)))) {
bag[i] = TRUE;
hasreserved[i] = FALSE;
wait[i] = 0;
}
}
```

```
//after thhis point bag may be created and arrived into the counter. if
no bag has arrived into the counter, one of the queue has to go into the
counter
if ((bag[i] == FALSE) && (queue[i] != 0)) {
bag[i] = TRUE;
queue[i] = queue[i] - 1;
wait[i] = 0;
hasreserved[i] = FALSE;
}

}


}
}
/* ------------------------------------------------------------------------
--
assignwindows
------------------------------------------------------------------------
*/

void assignwindows(int strategy, int iteration) {
int counterreserving;
if (strategy == UTOPIA) {
for (int i = 0; i < COUNTERS; i++) {
reserved[i] = i;  //no real allocation is done
}
}
else if (strategy == FAIR) {
for (int i = 0; i < COUNTERS; i++) {
reserved[i] = (i + iteration) % COUNTERS;
}
}
else if (strategy == FCFS) {
for (int i = 0; i < COUNTERS; i++) {
reserved[i] = i;
}
}
else if (strategy == FRFAR) {
for (int i = COUNTERS - 1; i >= 0; i--) {
//the furthest away reserve first, so from counters-1 til counter=0 (the
for loop does that)
if ((bag[i] == TRUE) && (hasreserved[i] == FALSE)) { //if there is a bag
and it has not reserved yet a window, reserve a window
//first reserve in reserve[]
int DONE = 0;
for (int k = i; k >= 0; k--) {      //can only reserve windows that are
coming, have not passed yet
if ((reserved[k] == NONE) && (belt[k] == FALSE)) {
```

```
reserved[k] = i;
hasreserved[i] = TRUE;
DONE = 1;
break;
}
}
//secondly, only if reserved[] is all allocated, then reserve in
freserve[]
if (DONE != 1) {
for (int k = fwindows - 1; k >= 0; k--) {
if ((freserved[k] == NONE)) {
freserved[k] = i;
hasreserved[i] = TRUE;
break;
}

}
}
}
}
}
else if (strategy == FRAVGWAIT) {
for (int i = 0; i < COUNTERS; i++) {
if (totbags[i] == 0) {
priorityvalues[i] = 0;
}
else {
priorityvalues[i] = totwait[i] / (double)totbags[i];
}
}
prioritize();
for (int i = COUNTERS - 1; i >= 0; i--) {
//the higher average waiting time will reserve first
counterreserving = prioritycounter[i];

int DONE = 0;
if ((bag[counterreserving] == TRUE) && (hasreserved[counterreserving] ==
FALSE)) { //if there is a bag and it has not reserved yet a window,
reserve a window
//first reserve in reserve[]
int DONE = 0;
for (int k = counterreserving; k >= 0; k--) {       //can only reserve
windows that are coming, have not passed yet
if ((reserved[k] == NONE) && (belt[k] == FALSE)) {
reserved[k] = counterreserving;
hasreserved[counterreserving] = TRUE;
DONE = 1;
break;
}
```

```
}
//secondly, only if reserved[] is all allocated, then reserve in
freserve[]
if (DONE != 1) {
for (int k = fwindows - 1; k >= 0; k--) {
if ((freserved[k] == NONE)) {
freserved[k] = counterreserving;
hasreserved[counterreserving] = TRUE;
break;
}

}
}
}
}
}
else if (strategy == FRMAXWAIT) {
for (int i = 0; i < COUNTERS; i++) {
priorityvalues[i] = (double)wait[i];
}
prioritize();
for (int i = COUNTERS - 1; i >= 0; i--) {  // the higher average waiting
time will reserve first
counterreserving = prioritycounter[i];
int DONE = 0;
if ((bag[counterreserving] == TRUE) && (hasreserved[counterreserving] ==
FALSE)) { //if there is a bag and it has not reserved yet a window,
reserve a window
//first reserve in reserve[]
int DONE = 0;
for (int k = counterreserving; k >= 0; k--) {       //can only reserve
windows that are coming, have not passed yet
if ((reserved[k] == NONE) && (belt[k] == FALSE)) {
reserved[k] = counterreserving;
hasreserved[counterreserving] = TRUE;
DONE = 1;
break;
}
}
//secondly, only if reserved[] is all allocated, then reserve in
freserve[]
if (DONE != 1) {
for (int k = fwindows - 1; k >= 0; k--) {
if ((freserved[k] == NONE)) {
freserved[k] = counterreserving;
hasreserved[counterreserving] = TRUE;
break;
}
```

```
}
}
}
}
}
else if (strategy == FRAVGWAITMAXWAIT) {
for (int i = 0; i < COUNTERS; i++) {
if (totbags[i] == 0 && wait[i] == 0) {
priorityvalues[i] = 0;
}
else {
priorityvalues[i] = 0.5 * (double)wait[i] + 0.5 * (totwait[i] /
(double)totbags[i]);
}
}
//the counter whose combination of average waiting time per bag and max
waiting time is bigger, reserves first
prioritize();
for (int i = COUNTERS - 1; i >= 0; i--) {

counterreserving = prioritycounter[i];
int DONE = 0;
if ((bag[counterreserving] == TRUE) && (hasreserved[counterreserving] ==
FALSE)) { //if there is a bag and it has not reserved yet a window,
reserve a window
//first reserve in reserve[]
int DONE = 0;
for (int k = counterreserving; k >= 0; k--) {       //can only reserve
windows that are coming, have not passed yet
if ((reserved[k] == NONE) && (belt[k] == FALSE)) {
reserved[k] = counterreserving;
hasreserved[counterreserving] = TRUE;
DONE = 1;
break;
}
}
//secondly, only if reserved[] is all allocated, then reserve in
freserve[]
if (DONE != 1) {
for (int k = fwindows - 1; k >= 0; k--) {
if ((freserved[k] == NONE)) {
freserved[k] = counterreserving;
hasreserved[counterreserving] = TRUE;
break;
}

}
}
}
```

```
}

}
}
/* ---------------------------------------------------------------------
showstatus
--------------------------------------------------------------------- */

void showstatus(int iteration)
{
int i;
printf("iteration %d\n", iteration);
printf("counter :");
for (i = 0; i < COUNTERS; i++) printf("%2d ", i);
printf("\n");
printf("freservd:");
for (i = 0; i < COUNTERS; i++) printf("%2d ", freserved[i]);
printf("\n");
printf("reserved:");
for (i = 0; i < COUNTERS; i++) printf("%2d ", reserved[i]);
printf("\n");
printf("bag     :");
for (i = 0; i < COUNTERS; i++) printf("%2d ", bag[i]);
printf("\n");
printf("belt    :");
for (i = 0; i < COUNTERS; i++) printf("%2d ", belt[i]);
printf("\n");
printf("hasreser:");
for (i = 0; i < COUNTERS; i++) printf("%2d ", hasreserved[i]);
printf("\n");
}



/* ---------------------------------------------------------------------
movebags
---------------------------------------------------------------------*/

void movebags( int strategy){ /* move bags from counter to convoyer belt
*/
int i;
if (strategy == UTOPIA) {  //move all the bags in, on top of the rest
that is in the take-away band
for (i = 0; i < COUNTERS; i++) {
if (bag[i] == TRUE) {
belt[i] = belt[i] + bag[i];
if (wait[i] > maxwait[i]) {
maxwait[i] = wait[i];
}
```

```
wait[i] = 0;
bag[i] = FALSE;
//totwait[i] = totwait[i] + wait[i];
totbags[i] = totbags[i] + 1;
}
}
}
else if ((strategy == FAIR) || (strategy == FCFS)) { //move in the take-
away band if the window is free
for (i = 0; i < COUNTERS; i++) {
if((bag[i] == TRUE) && (reserved[i] == i) && (belt[i] == FALSE)) {
/* bag is moved to convoyer belt */
belt[i] = TRUE;
bag[i] = FALSE;
//totwait[i] = totwait[i] + wait[i];
if (wait[i] > maxwait[i]) {
maxwait[i] = wait[i];
}
wait[i] = 0;    // reseting wait to zero
totbags[i] = totbags[i] + 1;
}
}
}
else if (strategy == FRFAR || strategy == FRAVGWAIT || strategy ==
FRMAXWAIT || strategy == FRAVGWAITMAXWAIT) { //move in the take-away band
if the window is free
for (i = 0; i < COUNTERS; i++) {
if ((bag[i] == TRUE) && (reserved[i] == i) && (belt[i] == FALSE) ) {
//reserved[i] = -1; change the value of reserve, so if the window is
taken, it is taken forever (before the bag is put into, it is waiting
into the bag, and after that, there is a bag in, so it should not be able
to put another bag in)
belt[i] = TRUE;
bag[i] = FALSE;
if (wait[i] > maxwait[i]) {
maxwait[i] = wait[i];
}
wait[i] = 0;    // reseting wait to zero
totbags[i] = totbags[i] + 1;
hasreserved[i] = FALSE;
}
}

}

}


/* ------------------------------------------------------------------
```

```
moveright
------------------------------------------------------------------------*/

void moveright(int strategy) /* move convoyer belt one step right */
{
int i;
for (i = COUNTERS - 1; i > 0; i--) {
belt[i] = belt[i - 1];
reserved[i] = reserved[i - 1];
}
if (strategy == FRFAR || strategy == FRAVGWAIT || strategy ==
FRAVGWAITMAXWAIT || strategy == FRMAXWAIT) {
reserved[0] = freserved[fwindows - 1];
for (i = fwindows - 1; i > 0; i--) {
freserved[i] = freserved[i - 1];
}
freserved[0] = NONE;
}
belt[0] = FALSE;
if (strategy == UTOPIA || strategy == FCFS || strategy == FAIR) {
reserved[0] = NONE;
}
//for bags that are not moved into the conveyor
for (i = 0; i < COUNTERS; i++) {
if (bag[i] == TRUE) {
wait[i] = wait[i] + 1;
totwait[i] = totwait[i] + 1;
}
}
}
/* ------------------------------------------------------------------------
statistics
------------------------------------------------------------------------*/

void statistics(void) /* print statistics */
{
int i;
fprintf(out, "Counter;Total Bags;Total Wait;Wait pr. bag; Max
wait;Queue\n");
for (i = 0; i < COUNTERS; i++) {
fprintf(out, "%d;%d;%d;%.2f;%d;%d\n",
i, totbags[i], totwait[i], totwait[i] / (double)totbags[i], maxwait[i],
queue[i]);
printf("%d;%d;%d;%.2f;%d;%d\n",
i, totbags[i], totwait[i], totwait[i] / (double)totbags[i], maxwait[i],
queue[i]);
}
}
/* ------------------------------------------------------------------------
```

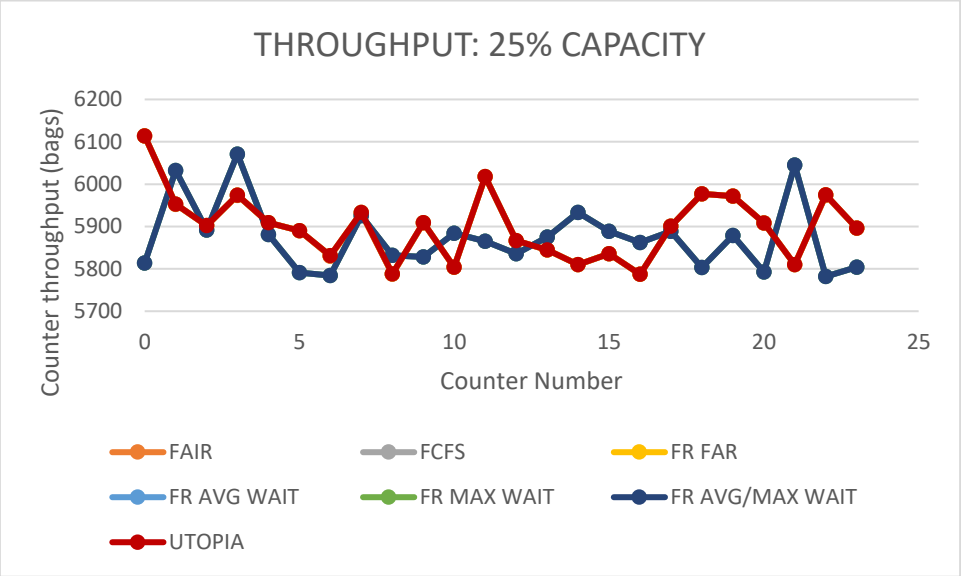
```
main
------------------------------------------------------------------ */

int main(int argc, char* argv[]) {
int mainLoopCounter, strategy;
int STRESSRATE;
init();
if (argc == 4) {
strategy = atoi(argv[1]);
STRESSRATE = atoi(argv[2]);
out = fopen(argv[3], "w");
}
else {
do {
printf("Select the window allocation algorithm:\n1-UTOPIA\n2-FAIR\n3-
FCFS\n4-FORWARD RESERVATION FAR\n5-FORWARD RESERVATION AVERAGE WAIT\n6-
FORWARD RESERVATION MAX WAIT\n7-FORWARD RESERVATION AVG & MAX WAIT\n");
scanf("%d", &strategy);
if (strategy < 1 || strategy > 7) {
printf("It is not a valid number\n\n");
}
printf("Insert a stressrate \n");
scanf("%d", &STRESSRATE);
} while (strategy < 1 || strategy > 7);
out = fopen("results.csv", "w");
}
for (mainLoopCounter = 0; mainLoopCounter < HORIZON; mainLoopCounter++) {
if (mainLoopCounter == 0 || mainLoopCounter % 1200 == 0) {      //every
hour (1200 iterations) it gets into the loop and the arrival rate changes
(1 iteration = 3 seconds,
arrivaldef();
}
arrivals(mainLoopCounter,STRESSRATE);
assignwindows(strategy, mainLoopCounter);
//showstatus(mainLoopCounter);
movebags(strategy);
moveright(strategy);
if (mainLoopCounter % 5000 == 0) {
printf("%d out of %d iterations completed \n", mainLoopCounter, HORIZON);
}
}
statistics();
fprintf(out, "Strategy; %d\n", strategy);
fclose(out);
return 0;
}
```

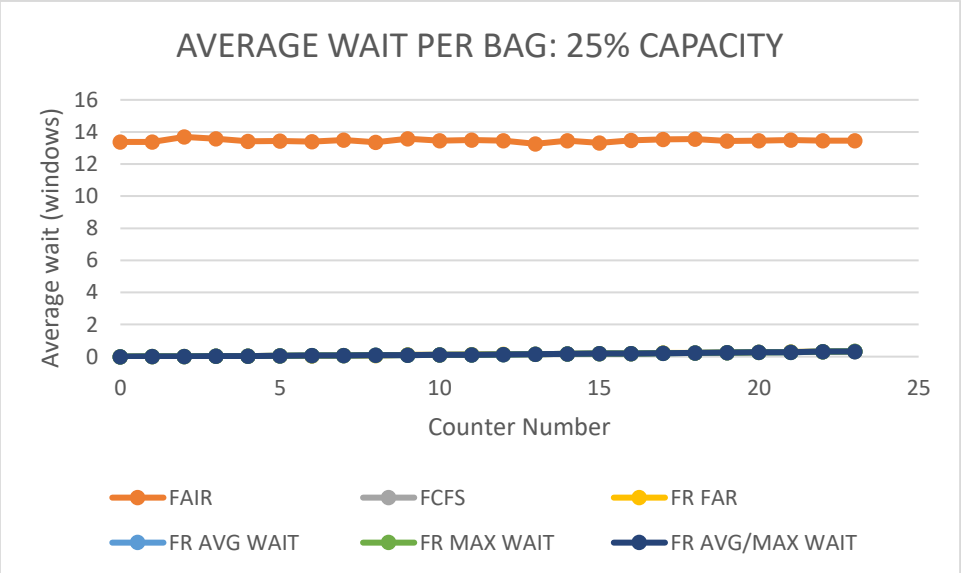
9.3 Check-in counters. Comparison of strategies depending on the stress of the system

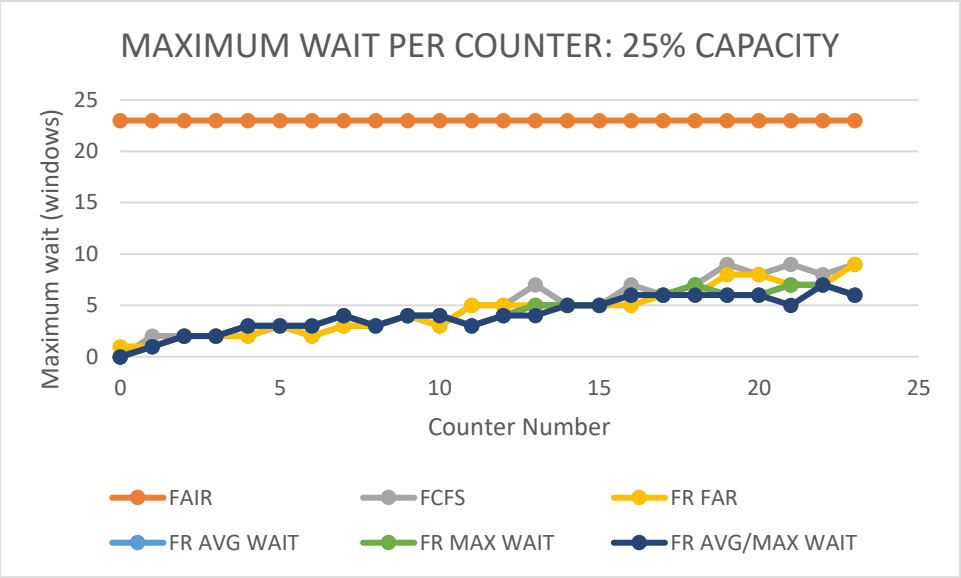
The present appendix includes all simulation results comparing how the different window allocation algorithms perform depending on the stress of the system.

Each subsection of the appendix corresponds to the results for a level of stress of the system, expressed in a percentage of its designed capacity. In each subsection, the performance of the strategies regarding throughput, average wait of the counter and maximum wait is given.

9.3.1 Comparison of strategies at 25 % of system capacity






MAXIMUM WAIT PER COUNTER: 25% CAPACITY
Maximum wait (windows)
25
20
15
10
5
0
0
5
10
15
20
25
Counter Number
FAIR
FCFS
FR FAR
FR AVG WAIT
FR MAX WAIT
FR AVG/MAX WAIT

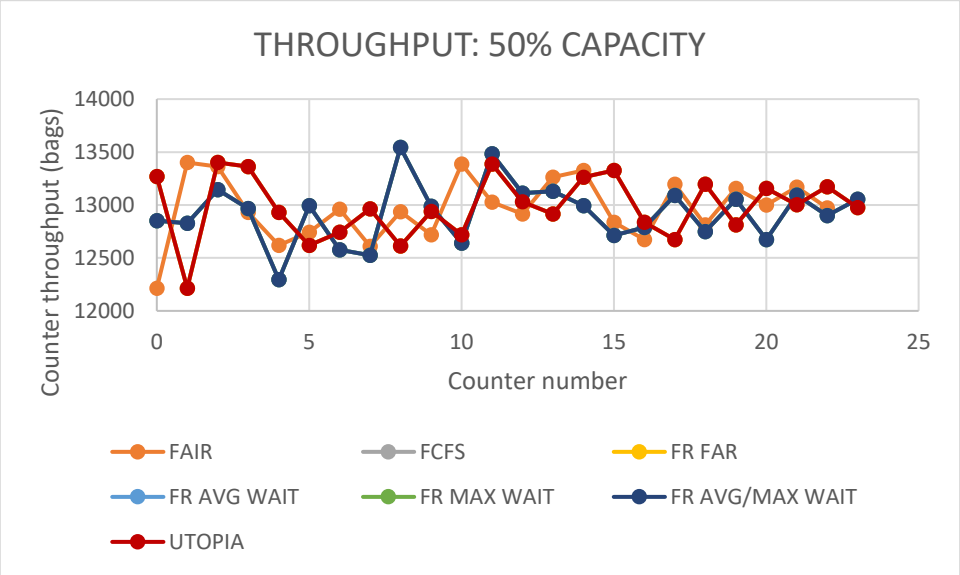
STRATEGY	COUNTER	THROUGHPUT (bags)	AVERAGE WAIT (windows)	MAXIMUM WAIT (windows)
UTOPIA	0	6114	0	0
	1	5953	0	0
	2	5902	0	0
	3	5974	0	0
	4	5909	0	0
	5	5890	0	0
	6	5831	0	0
	7	5933	0	0
	8	5788	0	0
	9	5909	0	0
	10	5804	0	0
	11	6018	0	0
	12	5867	0	0
	13	5845	0	0
	14	5810	0	0
	15	5836	0	0
	16	5787	0	0
	17	5901	0	0
	18	5977	0	0
	19	5972	0	0
	20	5908	0	0
	21	5810	0	0
	22	5975	0	0
	23	5896	0	0
FAIR	0	6114	13,37	23
	1	5952	13,38	23
	2	5902	13,69	23
	3	5973	13,57	23
	4	5909	13,42	23
	5	5890	13,44	23
	6	5830	13,39	23
	7	5931	13,5	23
	8	5787	13,36	23
	9	5908	13,57	23
	10	5804	13,46	23
	11	6018	13,5	23
	12	5866	13,45	23
	13	5845	13,26	23
	14	5810	13,46	23
	15	5836	13,32	23
	16	5787	13,47	23
	17	5901	13,53	23

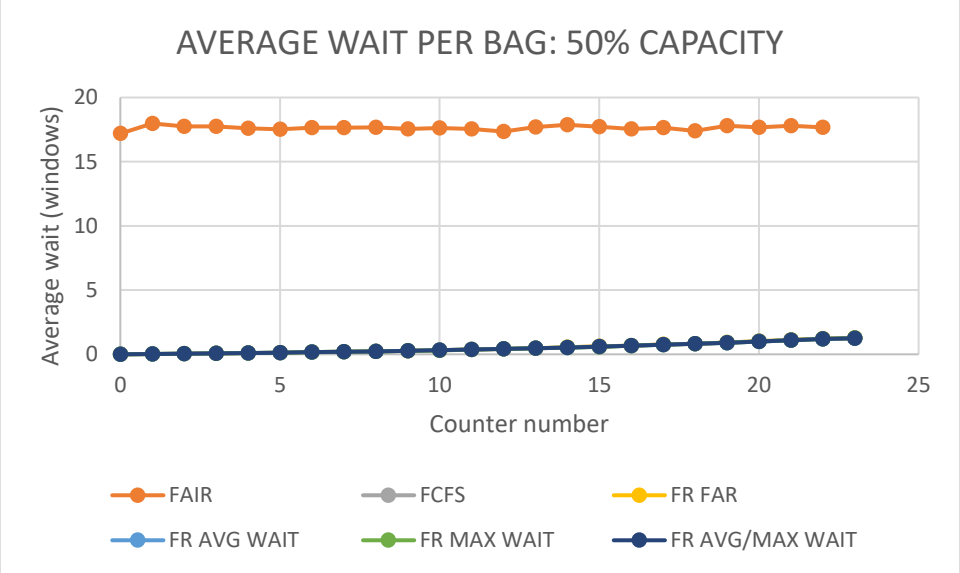
	18	5977	13,55	23
	19	5972	13,44	23
	20	5908	13,45	23
	21	5810	13,5	23
	22	5973	13,45	23
	23	5896	13,46	23
FCFS	0	6114	0	0
	1	5953	0,01	2
	2	5902	0,02	2
	3	5974	0,03	2
	4	5909	0,04	2
	5	5890	0,05	3
	6	5831	0,06	2
	7	5933	0,08	3
	8	5788	0,08	3
	9	5909	0,11	4
	10	5804	0,12	3
	11	6018	0,13	5
	12	5867	0,14	5
	13	5845	0,15	7
	14	5810	0,18	5
	15	5836	0,19	5
	16	5787	0,2	7
	17	5901	0,22	6
	18	5977	0,23	7
	19	5972	0,25	9
	20	5908	0,27	8
	21	5810	0,29	9
	22	5975	0,31	8
	23	5896	0,33	9
FR: FURTHEST	0	6114	0	1
	1	5953	0,01	1
	2	5902	0,02	2
	3	5974	0,03	2
	4	5909	0,04	2
	5	5890	0,05	3
	6	5831	0,06	2
	7	5933	0,08	3
	8	5788	0,08	3
	9	5909	0,11	4
	10	5804	0,12	3
	11	6018	0,13	5
	12	5867	0,15	5
	13	5845	0,15	5
	14	5810	0,18	5

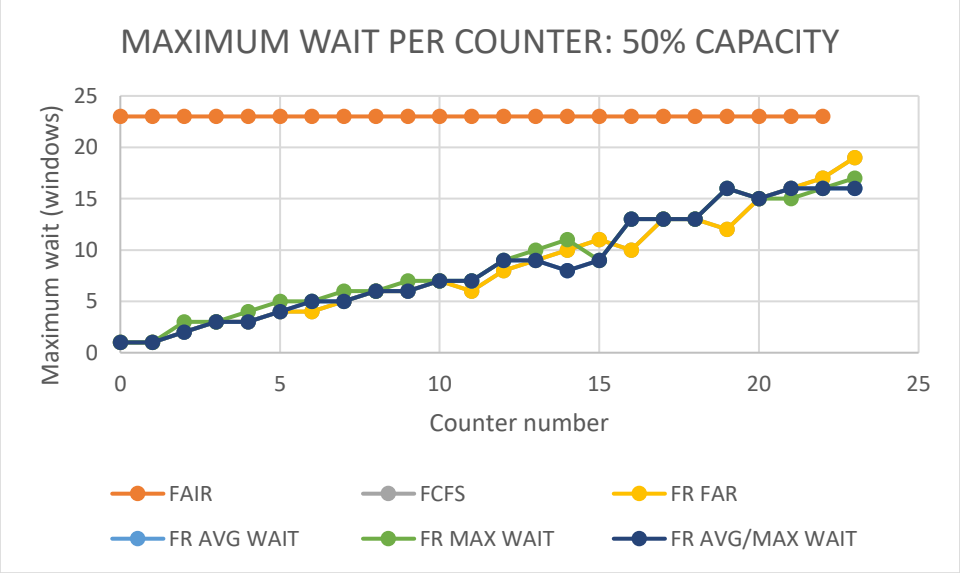
	15	5836	0,2	5
	16	5787	0,19	5
	17	5901	0,23	6
	18	5977	0,23	6
	19	5972	0,25	8
	20	5908	0,27	8
	21	5810	0,29	7
	22	5975	0,31	7
	23	5896	0,33	9
FR: AVERAGE WAIT	0	5814	0	0
	1	6032	0,01	1
	2	5892	0,02	2
	3	6071	0,03	2
	4	5881	0,04	3
	5	5791	0,06	3
	6	5784	0,07	3
	7	5926	0,08	4
	8	5832	0,09	3
	9	5828	0,1	4
	10	5884	0,11	4
	11	5865	0,12	3
	12	5836	0,14	4
	13	5875	0,16	4
	14	5933	0,17	5
	15	5889	0,19	5
	16	5862	0,2	6
	17	5889	0,21	6
	18	5803	0,23	6
	19	5879	0,25	6
	20	5793	0,27	6
	21	6045	0,28	5
	22	5782	0,32	7
	23	5804	0,32	6
FR: MAX WAIT	0	5814	0	0
	1	6032	0,01	1
	2	5892	0,02	2
	3	6071	0,03	2
	4	5881	0,04	3
	5	5791	0,06	3
	6	5784	0,07	3
	7	5926	0,08	4
	8	5832	0,09	3
	9	5828	0,1	4
	10	5884	0,11	4
	11	5865	0,12	3

	12	5836	0,14	4
	13	5875	0,15	5
	14	5933	0,17	5
	15	5889	0,2	5
	16	5862	0,2	6
	17	5889	0,21	6
	18	5803	0,23	7
	19	5879	0,25	6
	20	5793	0,27	6
	21	6045	0,28	7
	22	5782	0,32	7
	23	5804	0,33	6
FR: AVG/MAX WAIT	0	5814	0	0
	1	6032	0,01	1
	2	5892	0,02	2
	3	6071	0,03	2
	4	5881	0,04	3
	5	5791	0,06	3
	6	5784	0,07	3
	7	5926	0,08	4
	8	5832	0,09	3
	9	5828	0,1	4
	10	5884	0,11	4
	11	5865	0,12	3
	12	5836	0,14	4
	13	5875	0,16	4
	14	5933	0,17	5
	15	5889	0,19	5
	16	5862	0,2	6
	17	5889	0,21	6
	18	5803	0,23	6
	19	5879	0,25	6
	20	5793	0,27	6
	21	6045	0,28	5
	22	5782	0,32	7
	23	5804	0,32	6

9.3.2 Comparison of strategies at 50% of system capacity







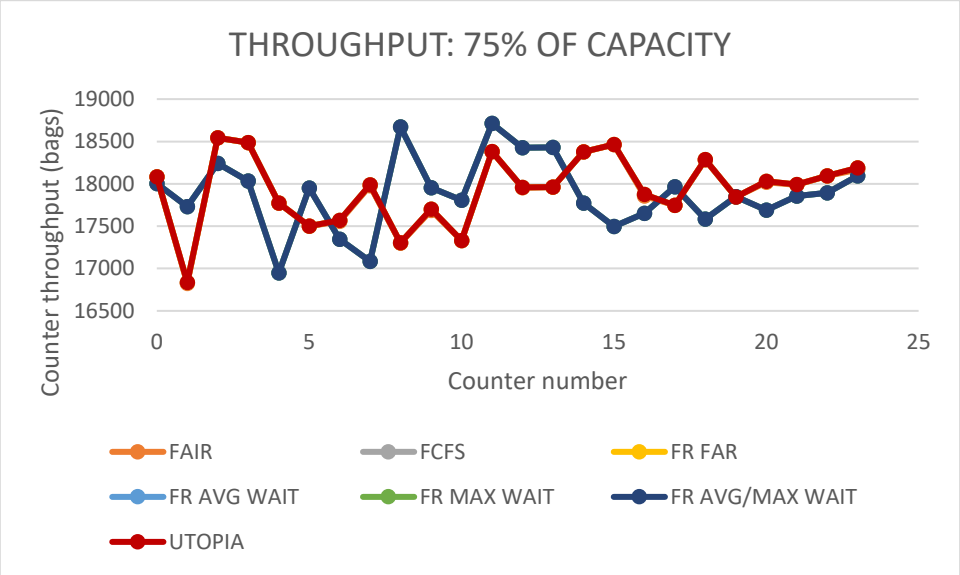
STRATEGY	COUNTER	THROUGHPUT (bags)	AVERAGE WAIT (windows)	MAXIMUM WAIT (windows)
UTOPIA	0	13268	0	0
	1	12215	0	0
	2	13401	0	0
	3	13364	0	0
	4	12930	0	0
	5	12619	0	0
	6	12744	0	0
	7	12963	0	0
	8	12612	0	0
	9	12939	0	0
	10	12718	0	0
	11	13388	0	0
	12	13029	0	0
	13	12915	0	0
	14	13262	0	0
	15	13328	0	0
	16	12837	0	0
	17	12673	0	0
	18	13196	0	0
	19	12813	0	0
	20	13156	0	0
	21	13003	0	0
	22	13171	0	0
	23	12975	0	0
FAIR	1	12213	17,19	23
	2	13401	17,98	23
	3	13363	17,74	23
	4	12930	17,76	23
	5	12619	17,59	23
	6	12742	17,53	23
	7	12961	17,65	23
	8	12611	17,66	23
	9	12937	17,67	23
	10	12718	17,56	23
	11	13386	17,63	23
	12	13028	17,54	23
	13	12915	17,34	23
	14	13262	17,71	23
	15	13328	17,87	23
	16	12836	17,73	23
	17	12673	17,55	23
	18	13196	17,65	23

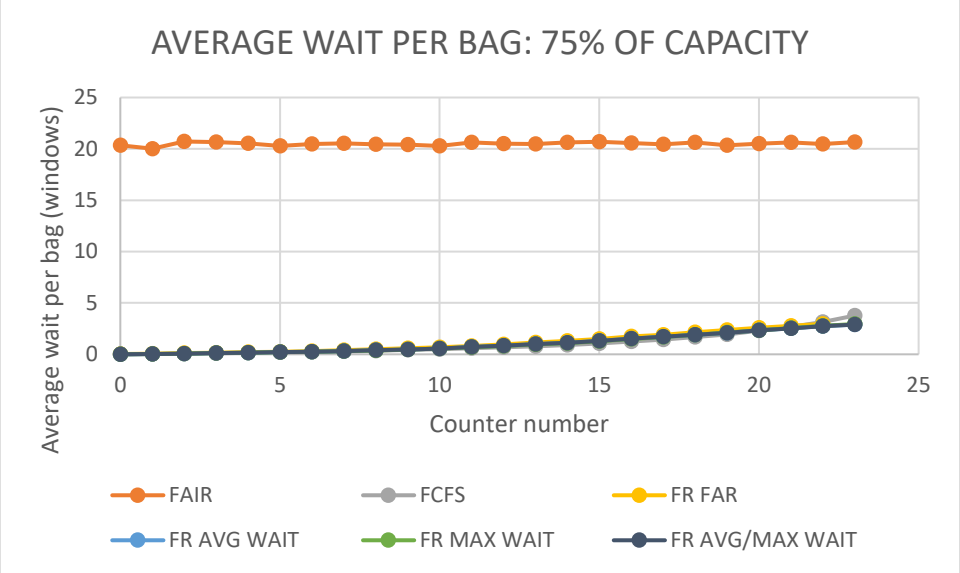
	19	12813	17,4	23
	20	13156	17,81	23
	21	13001	17,68	23
	22	13169	17,79	23
	23	12974	17,67	23
FCFS	0	13268	0	1
	1	12215	0,03	1
	2	13401	0,05	2
	3	13364	0,07	3
	4	12930	0,1	3
	5	12619	0,13	4
	6	12744	0,16	4
	7	12963	0,19	5
	8	12612	0,22	6
	9	12939	0,28	6
	10	12718	0,31	7
	11	13388	0,37	6
	12	13029	0,42	8
	13	12915	0,48	9
	14	13262	0,54	10
	15	13328	0,62	11
	16	12837	0,67	10
	17	12673	0,76	13
	18	13196	0,83	13
	19	12813	0,92	12
	20	13156	1,03	15
	21	13003	1,11	16
	22	13171	1,21	17
	23	12975	1,27	19
FR: FURTHEST	0	13268	0	1
	1	12215	0,03	1
	2	13401	0,05	2
	3	13364	0,07	3
	4	12930	0,1	3
	5	12619	0,13	4
	6	12744	0,16	4
	7	12963	0,19	5
	8	12612	0,22	6
	9	12939	0,28	6
	10	12718	0,31	7
	11	13388	0,37	6
	12	13029	0,42	8
	13	12915	0,48	9
	14	13262	0,54	10
	15	13328	0,62	11

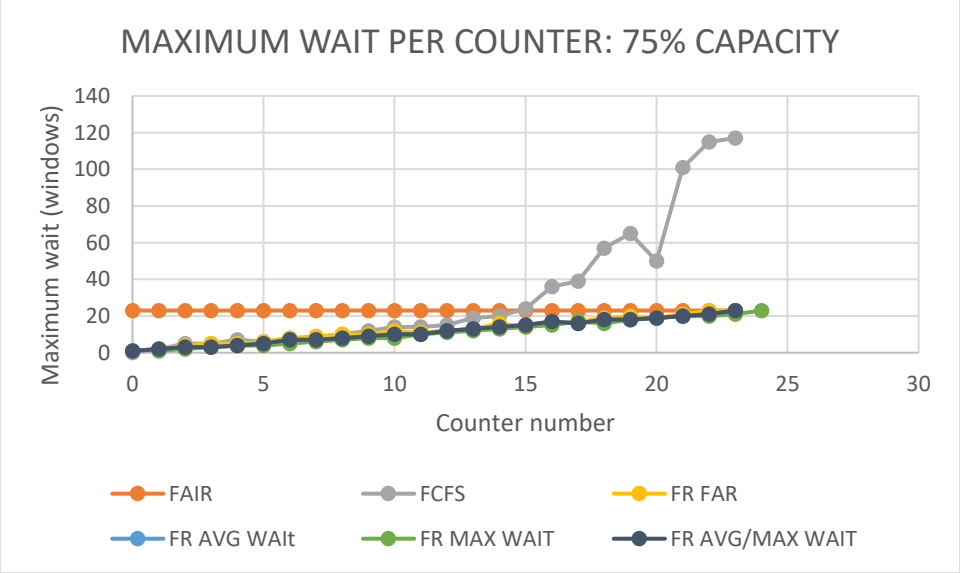
	16	12837	0,67	10
	17	12673	0,76	13
	18	13196	0,83	13
	19	12813	0,92	12
	20	13156	1,03	15
	21	13003	1,11	16
	22	13171	1,21	17
	23	12975	1,27	19
FR: AVG WAIT	0	12853	0	1
	1	12829	0,02	1
	2	13145	0,05	2
	3	12966	0,08	3
	4	12295	0,1	3
	5	12994	0,13	4
	6	12576	0,16	5
	7	12524	0,19	5
	8	13544	0,23	6
	9	12987	0,26	6
	10	12640	0,31	7
	11	13485	0,36	7
	12	13112	0,42	9
	13	13131	0,46	9
	14	12994	0,53	8
	15	12713	0,59	9
	16	12789	0,67	13
	17	13091	0,74	13
	18	12750	0,83	13
	19	13054	0,9	16
	20	12674	0,99	15
	21	13093	1,09	16
	22	12900	1,19	16
	23	13054	1,25	16
FR: MAX WAIT	0	12853	0	1
	1	12829	0,02	1
	2	13145	0,05	3
	3	12966	0,08	3
	4	12295	0,1	4
	5	12994	0,13	5
	6	12576	0,16	5
	7	12524	0,19	6
	8	13544	0,23	6
	9	12987	0,26	7
	10	12640	0,3	7
	11	13485	0,36	7
	12	13112	0,41	9

	13	13131	0,46	10
	14	12994	0,53	11
	15	12713	0,58	9
	16	12789	0,66	13
	17	13091	0,74	13
	18	12750	0,83	13
	19	13054	0,9	16
	20	12674	0,99	15
	21	13093	1,09	15
	22	12900	1,2	16
	23	13054	1,26	17
FR: AVG/ MAX WAIT	0	12853	0	1
	1	12829	0,02	1
	2	13145	0,05	2
	3	12966	0,08	3
	4	12295	0,1	3
	5	12994	0,13	4
	6	12576	0,16	5
	7	12524	0,19	5
	8	13544	0,23	6
	9	12987	0,26	6
	10	12640	0,31	7
	11	13485	0,36	7
	12	13112	0,42	9
	13	13131	0,46	9
	14	12994	0,53	8
	15	12713	0,59	9
	16	12789	0,67	13
	17	13091	0,74	13
	18	12750	0,83	13
	19	13054	0,9	16
	20	12674	0,99	15
	21	13093	1,09	16
	22	12900	1,19	16
	23	13054	1,25	16

9.3.3 Comparison of strategies at 75% of system capacity







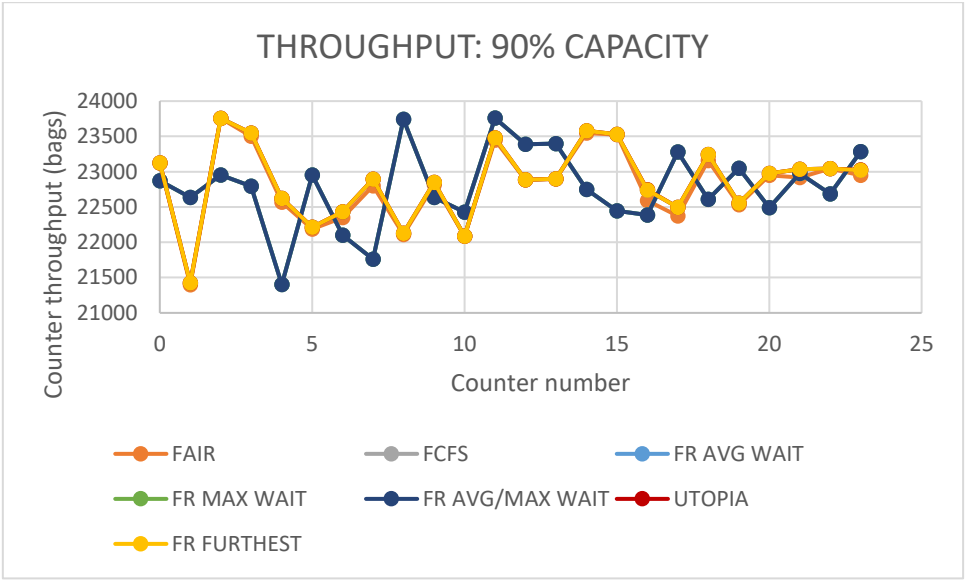
STRATEGY	COUNTER	THROUGHPUT (bags)	AVERAGE WAIT (windows)	MAXIMUM WAIT (windows)
UTOPIA	0	18081	0	0
	1	16837	0	0
	2	18543	0	0
	3	18488	0	0
	4	17774	0	0
	5	17500	0	0
	6	17569	0	0
	7	17987	0	0
	8	17303	0	0
	9	17702	0	0
	10	17331	0	0
	11	18382	0	0
	12	17956	0	0
	13	17960	0	0
	14	18379	0	0
	15	18464	0	0
	16	17875	0	0
	17	17746	0	0
	18	18287	0	0
	19	17845	0	0
	20	18029	0	0
	21	17991	0	0
	22	18094	0	0
	23	18188	0	0
FAIR	0	18081	20,34	23
	1	16825	20,02	23
	2	18543	20,72	23
	3	18487	20,67	23
	4	17774	20,55	23
	5	17499	20,3	23
	6	17556	20,47	23
	7	17975	20,54	23
	8	17302	20,45	23
	9	17687	20,41	23
	10	17331	20,29	23
	11	18380	20,63	23
	12	17955	20,52	23
	13	17960	20,46	23
	14	18379	20,62	23
	15	18463	20,7	23
	16	17854	20,56	23
	17	17745	20,45	23

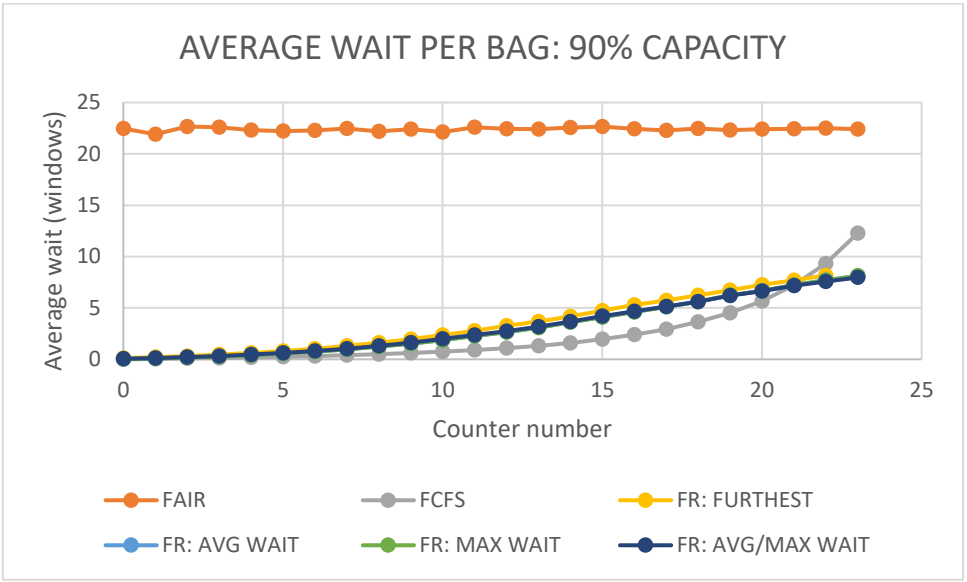
	18	18284	20,64	23
	19	17845	20,34	23
	20	18020	20,51	23
	21	17989	20,62	23
	22	18092	20,48	23
	23	18164	20,66	23
FCFS	0	18081	0	0
	1	16837	0,03	2
	2	18543	0,06	5
	3	18488	0,1	5
	4	17774	0,14	7
	5	17500	0,19	6
	6	17569	0,23	8
	7	17987	0,29	8
	8	17303	0,34	10
	9	17702	0,43	12
	10	17331	0,48	14
	11	18382	0,58	14
	12	17956	0,68	15
	13	17960	0,79	19
	14	18379	0,91	20
	15	18464	1,07	24
	16	17874	1,24	36
	17	17746	1,43	39
	18	18287	1,67	57
	19	17845	1,94	65
	20	18029	2,3	50
	21	17991	2,67	101
	22	18094	3,16	115
	23	18187	3,77	117
FR: FURTHEST	0	18081	0	1
	1	16837	0,03	2
	2	18543	0,07	4
	3	18488	0,11	5
	4	17774	0,15	5
	5	17500	0,2	6
	6	17569	0,25	8
	7	17987	0,32	9
	8	17303	0,39	10
	9	17702	0,49	10
	10	17331	0,58	12
	11	18382	0,69	11
	12	17956	0,81	12
	13	17960	0,97	13
	14	18379	1,14	16

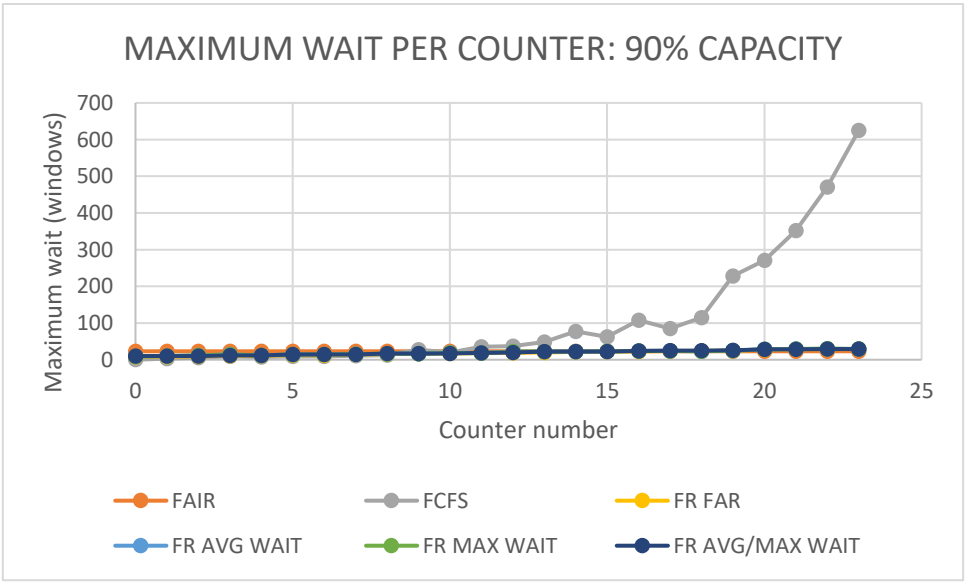
	15	18464	1,3	14
	16	17874	1,5	16
	17	17746	1,73	17
	18	18287	1,91	19
	19	17845	2,15	20
	20	18029	2,37	19
	21	17991	2,59	21
	22	18094	2,78	23
	23	18187	2,97	23
FR: AVG WAIT	0	18003	0	1
	1	17730	0,03	2
	2	18243	0,07	3
	3	18035	0,11	3
	4	16949	0,15	4
	5	17951	0,2	5
	6	17347	0,25	7
	7	17086	0,32	7
	8	18671	0,39	8
	9	17953	0,47	9
	10	17807	0,57	10
	11	18712	0,7	10
	12	18426	0,83	12
	13	18429	0,98	13
	14	17772	1,12	14
	15	17498	1,31	15
	16	17650	1,51	17
	17	17964	1,7	16
	18	17584	1,9	18
	19	17847	2,1	18
	20	17691	2,33	19
	21	17855	2,52	20
	22	17895	2,73	21
	23	18095	2,89	23
FR: MAX WAIT	0	18003	0	1
	1	17730	0,03	2
	2	18243	0,07	3
	3	18035	0,11	4
	4	16949	0,15	4
	5	17951	0,2	5
	6	17347	0,24	6
	7	17086	0,32	7
	8	18671	0,38	8
	9	17953	0,46	8
	10	17807	0,55	10
	11	18712	0,69	11

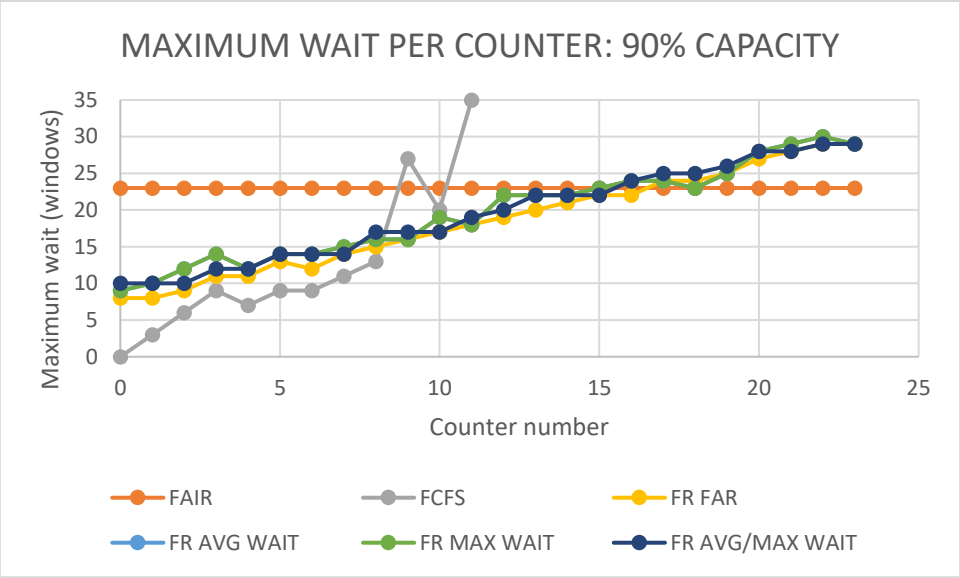
	12	18426	0,82	12
	13	18429	0,96	13
	14	17772	1,1	14
	15	17498	1,29	15
	16	17650	1,49	17
	17	17964	1,69	16
	18	17584	1,89	18
	19	17847	2,11	19
	20	17691	2,34	20
	21	17855	2,54	20
	22	17895	2,77	21
	23	18095	2,95	23
FR: AVG/ MAX WAIT	0	18003	0	1
	1	17730	0,03	2
	2	18243	0,07	3
	3	18035	0,11	3
	4	16949	0,15	4
	5	17951	0,2	5
	6	17347	0,25	7
	7	17086	0,32	7
	8	18671	0,39	8
	9	17953	0,47	9
	10	17807	0,57	10
	11	18712	0,7	10
	12	18426	0,83	12
	13	18429	0,98	13
	14	17772	1,12	14
	15	17498	1,31	15
	16	17650	1,51	17
	17	17964	1,7	16
	18	17584	1,9	18
	19	17847	2,1	18
	20	17691	2,33	19
	21	17855	2,52	20
	22	17895	2,73	21
	23	18095	2,89	23

9.3.4 Comparison of strategies at 90% of system capacity








MAXIMUM WAIT PER COUNTER: 90% CAPACITY
Maximum wait (windows)
35
30
25
20
15
10
5
0
0
5
10
15
20
25
Counter number
FAIR
FCFS
FR FAR
FR AVG WAIT
FR MAX WAIT
FR AVG/MAX WAIT

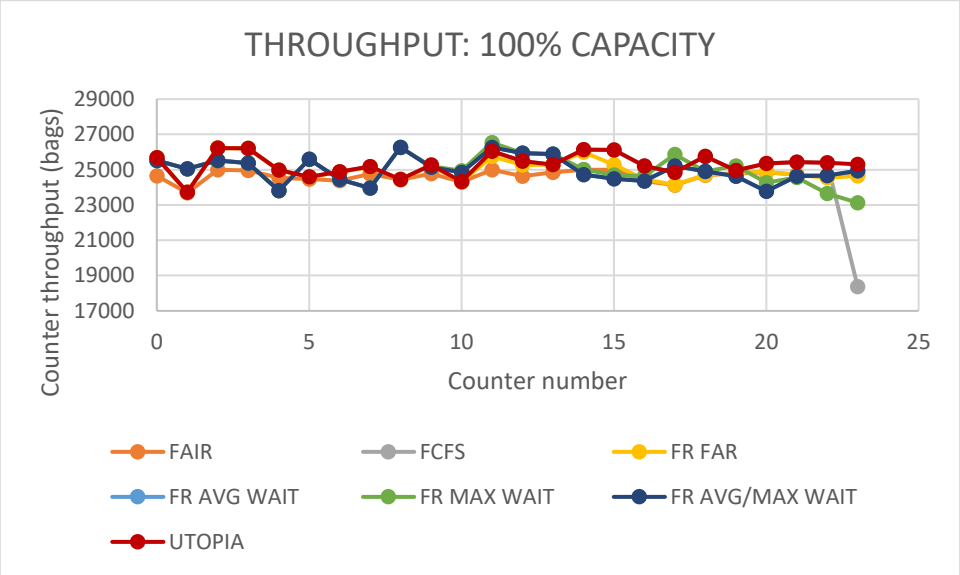
STRATEGY	COUNTER	THROUGHPUT (bags)	AVERAGE WAIT (windows)	MAXIMUM WAIT (windows)
UTOPIA	0	23128	0	0
	1	21424	0	0
	2	23757	0	0
	3	23548	0	0
	4	22621	0	0
	5	22214	0	0
	6	22433	0	0
	7	22894	0	0
	8	22126	0	0
	9	22849	0	0
	10	22089	0	0
	11	23479	0	0
	12	22887	0	0
	13	22900	0	0
	14	23580	0	0
	15	23530	0	0
	16	22744	0	0
	17	22497	0	0
	18	23242	0	0
	19	22553	0	0
	20	22977	0	0
	21	23035	0	0
	22	23044	0	0
	23	23024	0	0
FAIR	0	23121	22,47	23
	1	21400	21,91	23
	2	23757	22,66	23
	3	23500	22,6	23
	4	22567	22,31	23
	5	22186	22,22	23
	6	22351	22,29	23
	7	22799	22,46	23
	8	22108	22,2	23
	9	22819	22,4	23
	10	22089	22,14	23
	11	23448	22,59	23
	12	22886	22,44	23
	13	22900	22,41	23
	14	23546	22,57	23
	15	23526	22,65	23
	16	22589	22,43	23
	17	22375	22,28	23

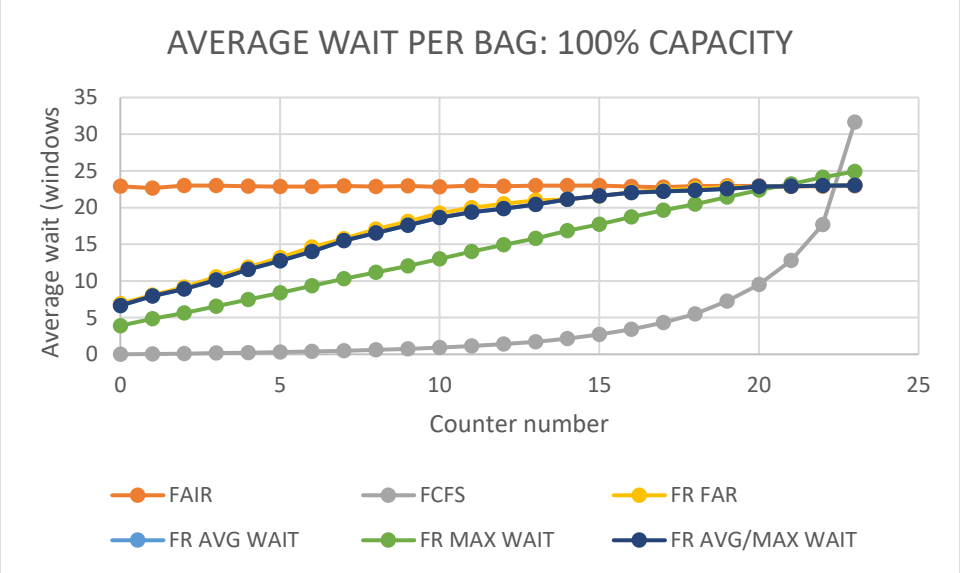
	18	23157	22,46	23
	19	22532	22,31	23
	20	22949	22,42	23
	21	22916	22,44	23
	22	23042	22,51	23
	23	22947	22,41	23
FCFS	0	23128	0	0
	1	21424	0,04	3
	2	23757	0,08	6
	3	23548	0,13	9
	4	22621	0,19	7
	5	22214	0,25	9
	6	22433	0,32	9
	7	22894	0,4	11
	8	22126	0,51	13
	9	22849	0,63	27
	10	22089	0,75	20
	11	23479	0,9	35
	12	22887	1,1	37
	13	22900	1,32	48
	14	23580	1,6	77
	15	23530	1,97	62
	16	22742	2,41	108
	17	22496	2,94	85
	18	23242	3,64	115
	19	22553	4,51	228
	20	22977	5,66	271
	21	23035	7,22	352
	22	23044	9,31	471
	23	23023	12,28	625
FR: FURTHEST	0	23128	0,04	8
	1	21424	0,11	8
	2	23757	0,2	9
	3	23548	0,3	11
	4	22621	0,45	11
	5	22214	0,61	13
	6	22433	0,8	12
	7	22894	1,04	14
	8	22126	1,3	15
	9	22849	1,62	16
	10	22089	1,97	17
	11	23479	2,35	18
	12	22887	2,78	19
	13	22900	3,27	20
	14	23580	3,69	21

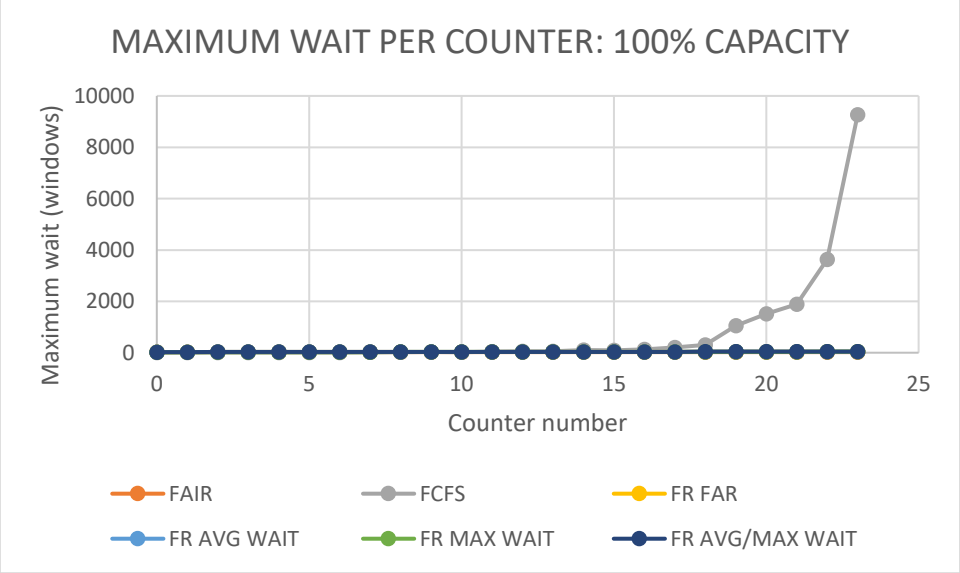
	15	23530	4,16	22
	16	22742	4,74	22
	17	22497	5,28	24
	18	23242	5,72	24
	19	22553	6,24	25
	20	22977	6,74	27
	21	23035	7,27	28
	22	23044	7,68	29
	23	23023	8,17	29
FR: AVG WAIT	0	22873	0,03	9
	1	22635	0,1	10
	2	22955	0,19	12
	3	22796	0,29	14
	4	21403	0,41	12
	5	22952	0,57	14
	6	22100	0,73	14
	7	21762	0,95	15
	8	23741	1,24	16
	9	22635	1,52	16
	10	22426	1,88	19
	11	23759	2,26	18
	12	23390	2,65	22
	13	23396	3,09	22
	14	22750	3,6	22
	15	22444	4,12	23
	16	22385	4,62	24
	17	23279	5,12	24
	18	22608	5,61	23
	19	23049	6,22	25
	20	22489	6,66	28
	21	22977	7,23	29
	22	22684	7,68	30
	23	23285	8,12	29
FR: MAX WAIT	0	22873	0,03	9
	1	22635	0,1	10
	2	22955	0,19	12
	3	22796	0,29	14
	4	21403	0,41	12
	5	22952	0,57	14
	6	22100	0,73	14
	7	21762	0,95	15
	8	23741	1,24	16
	9	22635	1,52	16
	10	22426	1,88	19
	11	23759	2,26	18

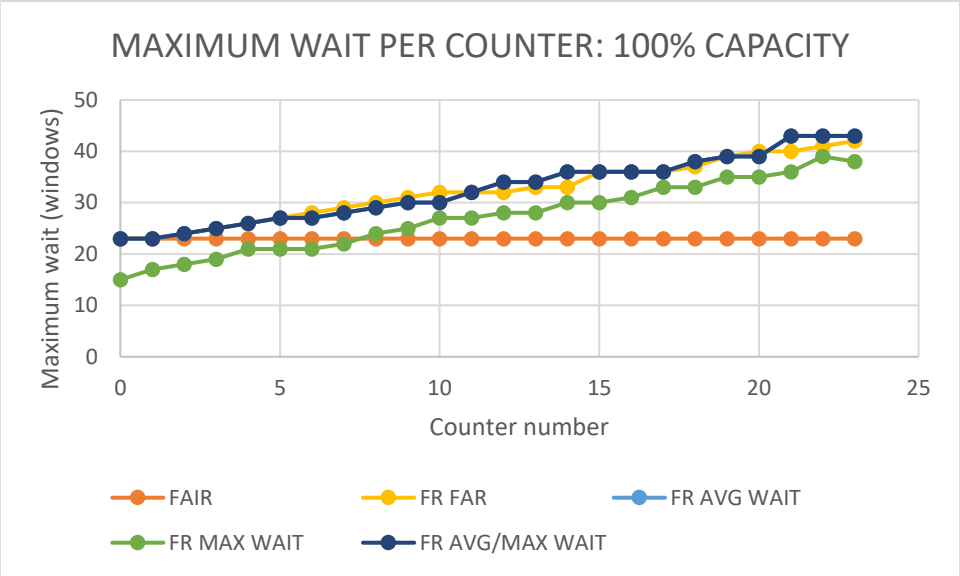
	12	23390	2,65	22
	13	23396	3,09	22
	14	22750	3,6	22
	15	22444	4,12	23
	16	22385	4,62	24
	17	23279	5,12	24
	18	22608	5,61	23
	19	23049	6,22	25
	20	22489	6,66	28
	21	22977	7,23	29
	22	22684	7,68	30
	23	23285	8,12	29
FR: AVG/ MAX WAIT	0	22873	0,05	10
	1	22635	0,12	10
	2	22955	0,22	10
	3	22796	0,32	12
	4	21403	0,45	12
	5	22952	0,62	14
	6	22100	0,8	14
	7	21762	1,03	14
	8	23741	1,32	17
	9	22635	1,61	17
	10	22426	1,98	17
	11	23759	2,36	19
	12	23390	2,75	20
	13	23396	3,18	22
	14	22750	3,69	22
	15	22444	4,19	22
	16	22385	4,67	24
	17	23279	5,15	25
	18	22608	5,62	25
	19	23049	6,19	26
	20	22489	6,63	28
	21	22977	7,15	28
	22	22684	7,56	29
	23	23285	7,96	29

9.3.5 Comparison of strategies at 100% of system capacity








MAXIMUM WAIT PER COUNTER: 100% CAPACITY
Maximum wait (windows)
50
40
30
20
10
0
0
5
10
15
20
25
Counter number
FAIR
FR FAR
FR AVG WAIT
FR MAX WAIT
FR AVG/MAX WAIT

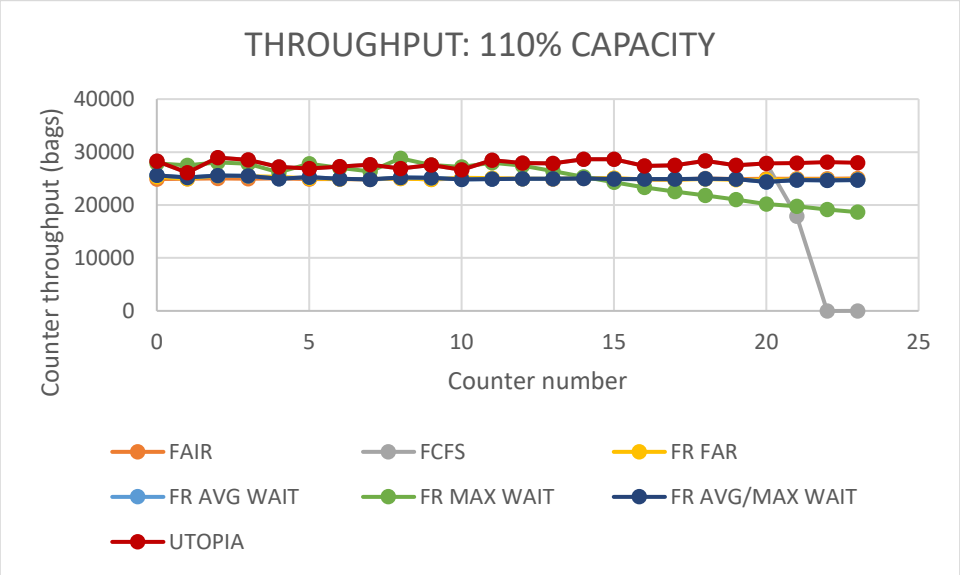
STRATEGY	COUNTER	THROUGHPUT (bags)	AVERAGE WAIT (windows)	MAXIMUM WAIT (windows)
UTOPIA	0	25687	0	0
	1	23728	0	0
	2	26228	0	0
	3	26212	0	0
	4	24986	0	0
	5	24603	0	0
	6	24890	0	0
	7	25180	0	0
	8	24458	0	0
	9	25270	0	0
	10	24317	0	0
	11	26053	0	0
	12	25493	0	0
	13	25280	0	0
	14	26136	0	0
	15	26128	0	0
	16	25218	0	0
	17	24848	0	0
	18	25760	0	0
	19	24963	0	0
	20	25350	0	0
	21	25427	0	0
	22	25397	0	0
	23	25312	0	0
FAIR	0	24659	22,89	23
	1	23693	22,65	23
	2	24990	23	23
	3	24954	22,99	23
	4	24534	22,89	23
	5	24469	22,84	23
	6	24385	22,84	23
	7	24757	22,96	23
	8	24433	22,87	23
	9	24785	22,94	23
	10	24301	22,82	23
	11	24988	23	23
	12	24629	22,92	23
	13	24857	22,97	23
	14	24988	22,99	23
	15	24999	23	23
	16	24356	22,87	23
	17	24127	22,76	23

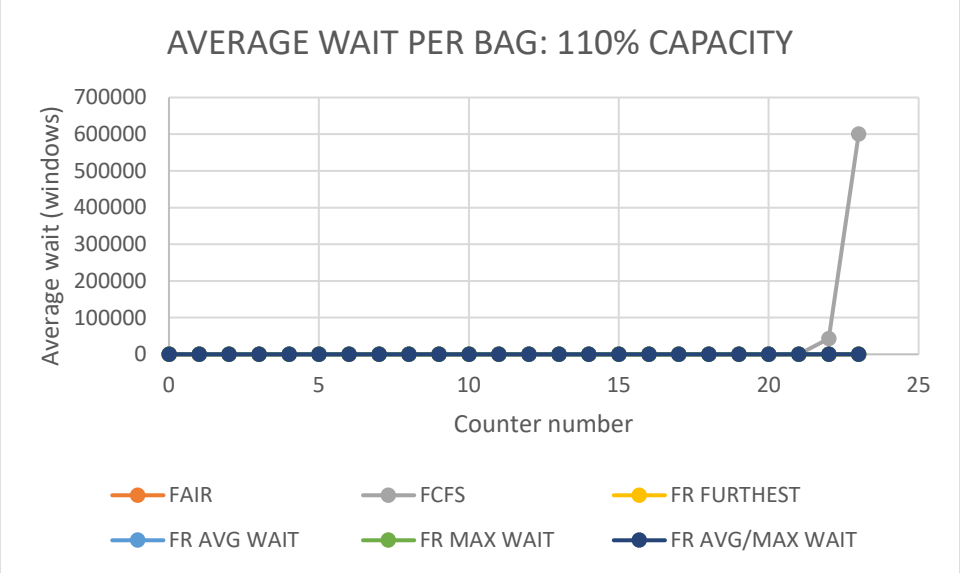
	18	24694	22,95	23
	19	24791	22,96	23
	20	24846	22,94	23
	21	24724	22,92	23
	22	24638	22,92	23
	23	24766	22,95	23
FCFS	0	25687	0	0
	1	23728	0,04	3
	2	26228	0,09	6
	3	26212	0,15	9
	4	24986	0,22	7
	5	24603	0,29	10
	6	24890	0,37	9
	7	25180	0,47	13
	8	24458	0,6	14
	9	25270	0,74	32
	10	24317	0,92	22
	11	26053	1,12	45
	12	25493	1,39	51
	13	25280	1,7	49
	14	26136	2,12	107
	15	26127	2,7	94
	16	25215	3,4	126
	17	24847	4,31	202
	18	25760	5,51	303
	19	24963	7,26	1048
	20	25349	9,52	1511
	21	25427	12,77	1887
	22	25397	17,66	3637
	23	18372	31,65	9268
FR: FURTHEST	0	25687	6,91	23
	1	23728	8,07	23
	2	26228	9,14	24
	3	26211	10,57	25
	4	24986	11,88	26
	5	24597	13,18	27
	6	24877	14,6	28
	7	25166	15,75	29
	8	24457	17,04	30
	9	25236	18,11	31
	10	24315	19,22	32
	11	25751	19,97	32
	12	25264	20,49	32
	13	25280	21	33
	14	26000	21,13	33

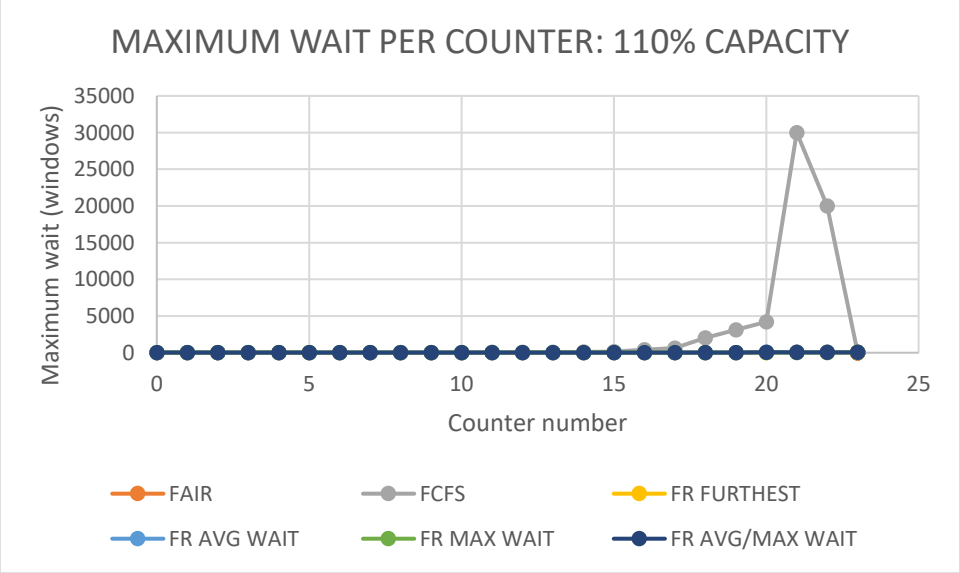
	15	25285	21,57	36
	16	24458	22,1	36
	17	24142	22,38	36
	18	24698	22,58	37
	19	24814	22,76	39
	20	24889	22,81	40
	21	24714	22,88	40
	22	24535	23,01	41
	23	24658	23,08	42
FR: AVG WAIT	0	25520	6,64	23
	1	25045	7,95	23
	2	25515	8,91	24
	3	25384	10,11	25
	4	23810	11,56	26
	5	25587	12,73	27
	6	24460	14,02	27
	7	23953	15,49	28
	8	26271	16,55	29
	9	25140	17,59	30
	10	24851	18,63	30
	11	26263	19,37	32
	12	25941	19,84	34
	13	25879	20,4	34
	14	24730	21,1	36
	15	24488	21,6	36
	16	24356	22,04	36
	17	25215	22,21	36
	18	24906	22,32	38
	19	24640	22,55	39
	20	23777	22,85	39
	21	24634	22,92	43
	22	24675	23,01	43
	23	24935	23,02	43
FR: MAX WAIT	0	25520	3,9	15
	1	25045	4,85	17
	2	25515	5,64	18
	3	25384	6,52	19
	4	23816	7,46	21
	5	25586	8,38	21
	6	24460	9,33	21
	7	23953	10,29	22
	8	26271	11,16	24
	9	25196	12,02	25
	10	24936	12,98	27
	11	26537	13,99	27

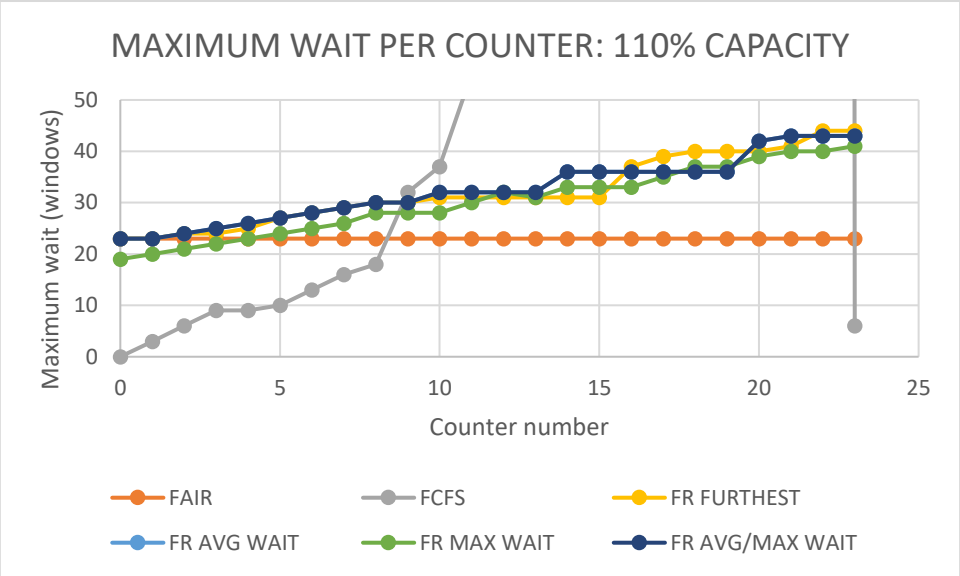
	12	25943	14,91	28
	13	25884	15,77	28
	14	25006	16,85	30
	15	24707	17,73	30
	16	24633	18,73	31
	17	25870	19,63	33
	18	24899	20,48	33
	19	25205	21,41	35
	20	24271	22,4	35
	21	24551	23,2	36
	22	23655	24,11	39
	23	23128	24,92	38
FR: AVG/ MAX WAIT	0	25520	6,64	23
	1	25045	7,95	23
	2	25515	8,91	24
	3	25384	10,11	25
	4	23810	11,56	26
	5	25587	12,73	27
	6	24460	14,02	27
	7	23953	15,49	28
	8	26271	16,55	29
	9	25140	17,59	30
	10	24851	18,63	30
	11	26263	19,37	32
	12	25941	19,84	34
	13	25879	20,4	34
	14	24730	21,1	36
	15	24488	21,6	36
	16	24356	22,04	36
	17	25215	22,21	36
	18	24906	22,32	38
	19	24640	22,55	39
	20	23777	22,85	39
	21	24634	22,92	43
	22	24675	23,01	43
	23	24935	23,02	43

9.3.6 Comparison of strategies at 110% of system capacity








MAXIMUM WAIT PER COUNTER: 110% CAPACITY
Maximum wait (windows)
50
40
30
20
10
0
0
5
10
15
20
25
Counter number
FAIR
FCFS
FR FURTHEST
FR AVG WAIT
FR MAX WAIT
FR AVG/MAX WAIT

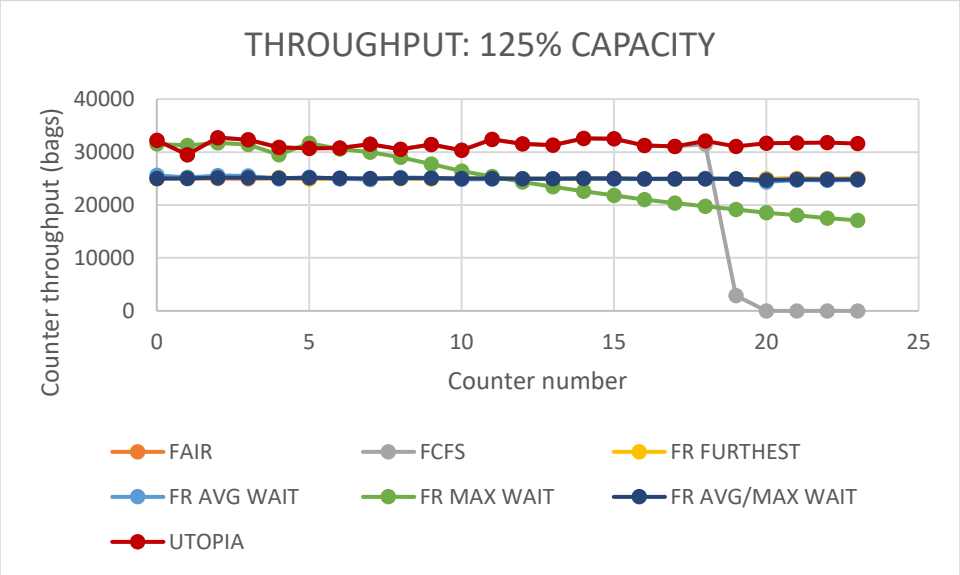
STRATEGY	COUNTER	THROUGHPUT (bags)	AVERAGE WAIT (windows)	MAXIMUM WAIT (windows)
UTOPIA	0	28309	0	0
	1	26117	0	0
	2	28954	0	0
	3	28547	0	0
	4	27185	0	0
	5	26903	0	0
	6	27255	0	0
	7	27636	0	0
	8	26905	0	0
	9	27558	0	0
	10	26645	0	0
	11	28474	0	0
	12	27943	0	0
	13	27842	0	0
	14	28649	0	0
	15	28667	0	0
	16	27353	0	0
	17	27479	0	0
	18	28321	0	0
	19	27481	0	0
	20	27858	0	0
	21	27944	0	0
	22	28091	0	0
	23	27984	0	0
FAIR	0	24927	22,97	23
	1	24976	22,99	23
	2	24998	23	23
	3	24954	22,99	23
	4	24997	23	23
	5	24886	22,96	23
	6	24889	22,97	23
	7	24819	22,97	23
	8	24997	23	23
	9	24841	22,96	23
	10	24999	23	23
	11	24988	23	23
	12	24951	22,99	23
	13	24918	22,98	23
	14	24998	23	23
	15	24999	23	23
	16	24814	22,97	23
	17	24796	22,95	23

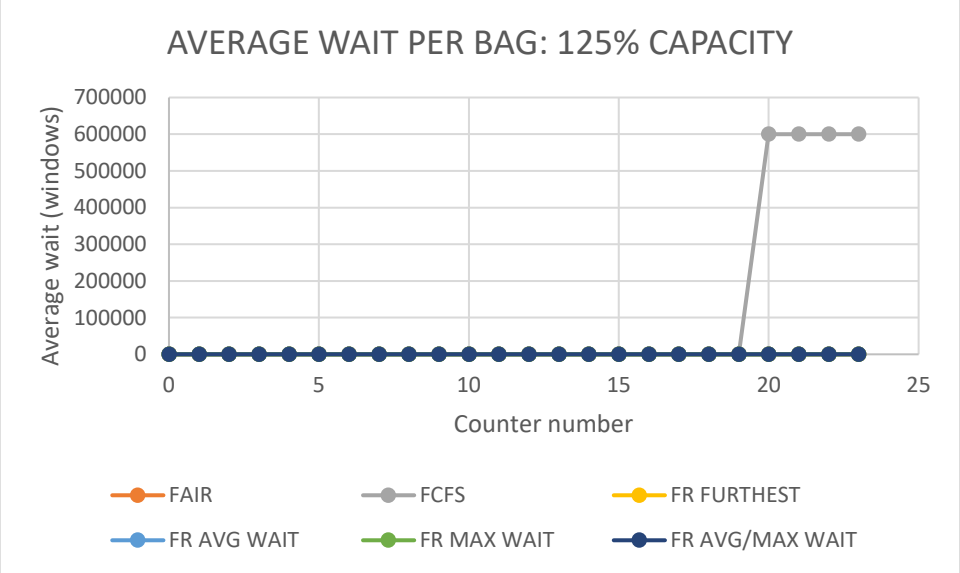
	18	24993	23	23
	19	24873	22,98	23
	20	24948	22,98	23
	21	24937	22,98	23
	22	24933	22,99	23
	23	24998	23	23
FCFS	0	28309	0	0
	1	26117	0,05	3
	2	28954	0,1	6
	3	28547	0,17	9
	4	27185	0,25	9
	5	26903	0,33	10
	6	27255	0,43	13
	7	27636	0,56	16
	8	26905	0,71	18
	9	27558	0,9	32
	10	26645	1,11	37
	11	28474	1,39	55
	12	27943	1,75	56
	13	27842	2,21	70
	14	28649	2,81	113
	15	28666	3,72	168
	16	27349	4,9	395
	17	27478	6,44	628
	18	28321	8,62	2036
	19	27481	12,01	3112
	20	27857	16,52	4214
	21	17908	32,47	30010
	22	14	42852	19995
	23	1	599970	6
FR: FURTHEST	0	25316	21,77	23
	1	24981	22,1	23
	2	25519	22,14	24
	3	25416	22,25	24
	4	25335	22,33	25
	5	25007	22,52	27
	6	24889	22,63	28
	7	24874	22,78	29
	8	25034	22,76	30
	9	24851	22,83	30
	10	25117	22,85	31
	11	25064	22,87	31
	12	24983	22,91	31
	13	24954	22,94	31
	14	25061	22,92	31

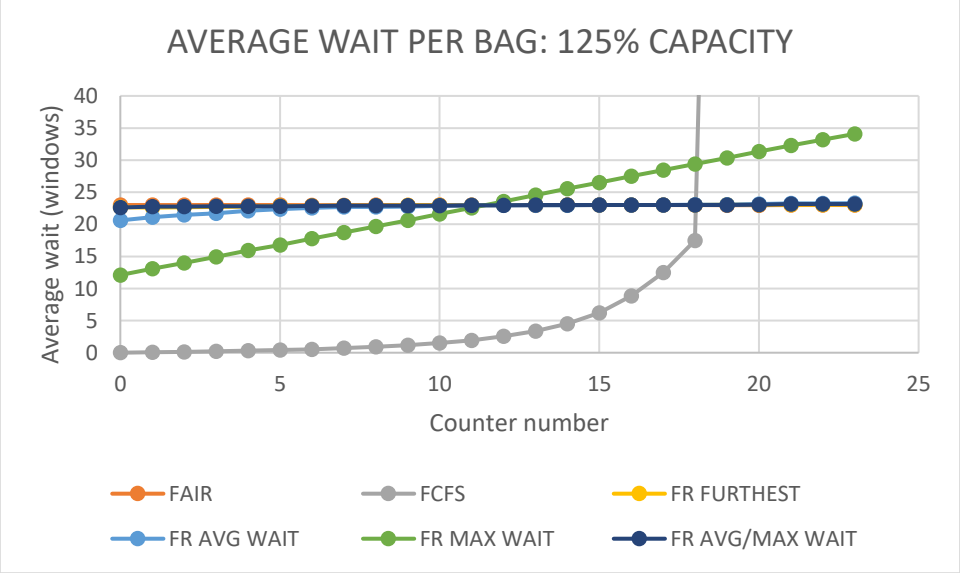
	15	25085	22,92	31
	16	24811	22,99	37
	17	24768	23,05	39
	18	24912	23,08	40
	19	24754	23,11	40
	20	24804	23,14	40
	21	24843	23,14	41
	22	24772	23,16	44
	23	24836	23,16	44
FR: AVG WAIT	0	25598	20,61	23
	1	25218	21,12	23
	2	25549	21,44	24
	3	25532	21,72	25
	4	24959	22,13	26
	5	25248	22,35	27
	6	24940	22,57	28
	7	24848	22,73	29
	8	25208	22,73	30
	9	25121	22,76	30
	10	24866	22,88	32
	11	24917	22,94	32
	12	24940	22,96	32
	13	24983	22,98	32
	14	25014	22,98	36
	15	24917	23,01	36
	16	24922	23,03	36
	17	24923	23,03	36
	18	24962	23,04	36
	19	24906	23,05	36
	20	24357	23,14	42
	21	24722	23,27	43
	22	24651	23,28	43
	23	24700	23,29	43
FR: MAX WAIT	0	27871	9,3	19
	1	27503	10,37	20
	2	28042	11,23	21
	3	27802	12,17	22
	4	26166	13,16	23
	5	27825	14,07	24
	6	26998	15,05	25
	7	26325	16,08	26
	8	28820	16,95	28
	9	27541	17,95	28
	10	27222	18,92	28
	11	27988	19,8	30

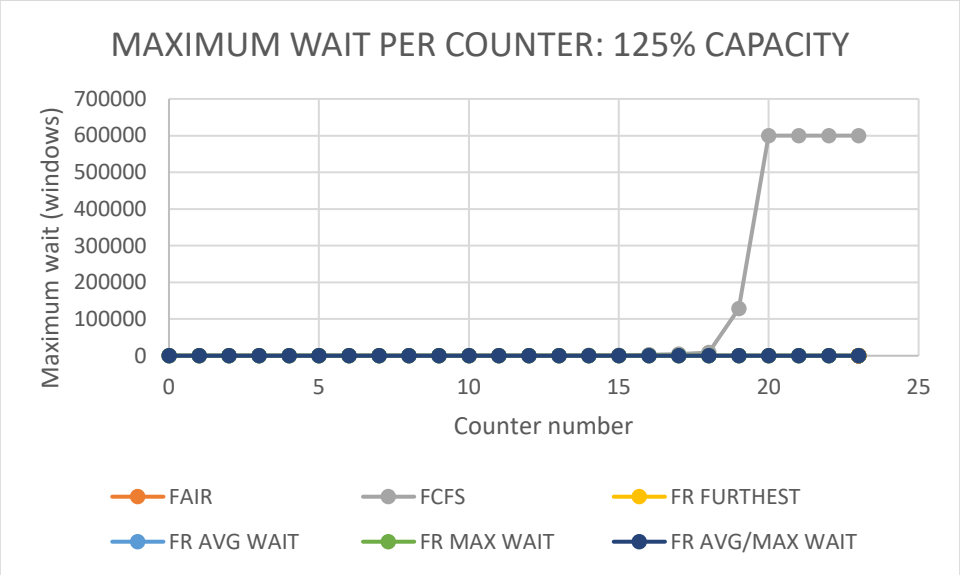
	12	27427	20,77	32
	13	26381	21,7	31
	14	25343	22,65	33
	15	24263	23,62	33
	16	23344	24,66	33
	17	22551	25,56	35
	18	21790	26,53	37
	19	21044	27,48	37
	20	20210	28,44	39
	21	19743	29,39	40
	22	19150	30,3	40
	23	18647	31,17	41
FR: AVG/ MAX WAIT	0	25598	20,61	23
	1	25218	21,12	23
	2	25549	21,44	24
	3	25532	21,72	25
	4	24959	22,13	26
	5	25248	22,35	27
	6	24940	22,57	28
	7	24848	22,73	29
	8	25208	22,73	30
	9	25121	22,76	30
	10	24866	22,88	32
	11	24917	22,94	32
	12	24940	22,96	32
	13	24983	22,98	32
	14	25014	22,98	36
	15	24917	23,01	36
	16	24922	23,03	36
	17	24923	23,03	36
	18	24962	23,04	36
	19	24906	23,05	36
	20	24357	23,14	42
	21	24722	23,27	43
	22	24651	23,28	43
	23	24700	23,29	43

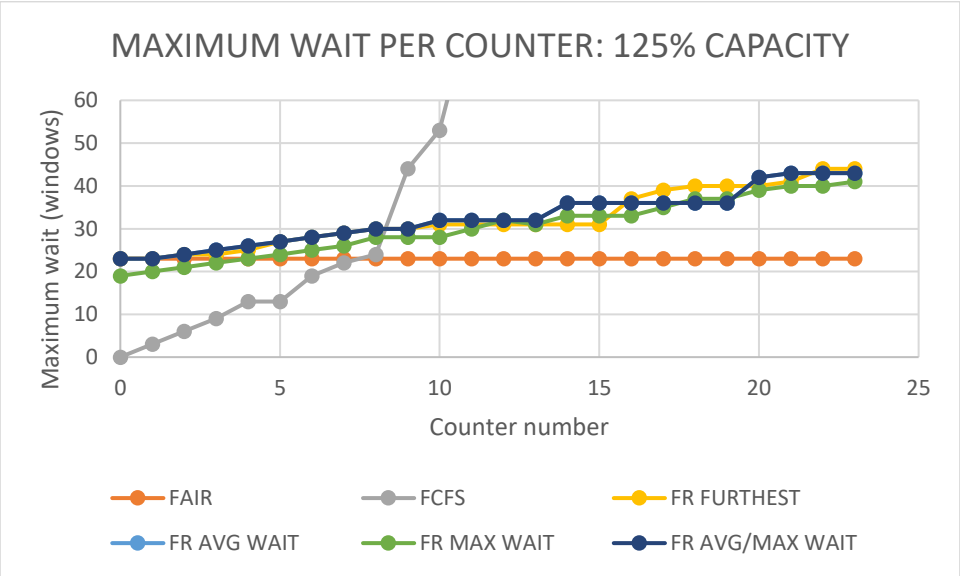
9.3.7 Comparison of strategies at 125% of system capacity








MAXIMUM WAIT PER COUNTER: 125% CAPACITY
Maximum wait (windows)
700000
600000
500000
400000
300000
200000
100000
0
0
5
10
15
20
25
Counter number
FAIR
FCFS
FR FURTHEST
FR AVG WAIT
FR MAX WAIT
FR AVG/MAX WAIT


MAXIMUM WAIT PER COUNTER: 125% CAPACITY
Maximum wait (windows)
60
50
40
30
20
10
0
0
5
10
15
20
25
Counter number
FAIR
FCFS
FR FURTHEST
FR AVG WAIT
FR MAX WAIT
FR AVG/MAX WAIT

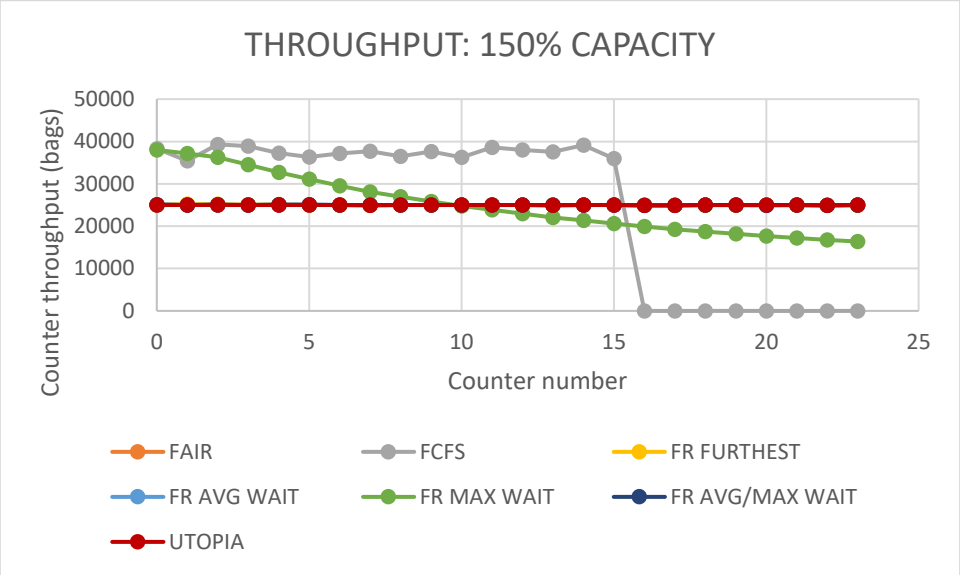
STRATEGY	COUNTER	THROUGHPUT (bags)	AVERAGE WAIT (windows)	MAXIMUM WAIT (windows)
UTOPIA	0	32186	0	0
	1	29515	0	0
	2	32722	0	0
	3	32315	0	0
	4	30896	0	0
	5	30686	0	0
	6	30781	0	0
	7	31499	0	0
	8	30535	0	0
	9	31450	0	0
	10	30352	0	0
	11	32380	0	0
	12	31575	0	0
	13	31299	0	0
	14	32586	0	0
	15	32510	0	0
	16	31261	0	0
	17	31083	0	0
	18	32110	0	0
	19	31079	0	0
	20	31685	0	0
	21	31759	0	0
	22	31816	0	0
	23	31611	0	0
FAIR	0	24969	22,99	23
	1	24999	23	23
	2	24999	23	23
	3	24965	22,99	23
	4	24997	23	23
	5	24952	22,99	23
	6	24935	22,98	23
	7	24851	22,98	23
	8	24998	23	23
	9	24955	22,98	23
	10	24999	23	23
	11	24999	23	23
	12	24966	23	23
	13	24941	22,98	23
	14	24998	23	23
	15	24999	23	23
	16	24906	22,99	23
	17	24957	23	23

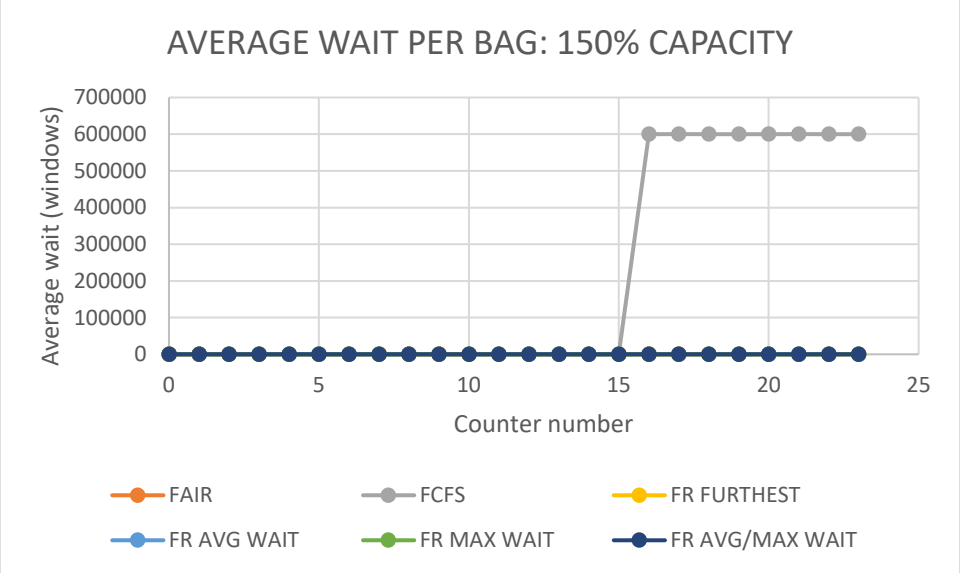
	18	24999	23	23
	19	24907	22,98	23
	20	24968	22,98	23
	21	24998	23	23
	22	24944	22,99	23
	23	24998	23	23
FCFS	0	32186	0	0
	1	29515	0,06	3
	2	32722	0,12	6
	3	32315	0,2	9
	4	30896	0,3	13
	5	30686	0,41	13
	6	30781	0,53	19
	7	31499	0,7	22
	8	30535	0,91	24
	9	31450	1,18	44
	10	30352	1,5	53
	11	32378	1,94	82
	12	31574	2,55	150
	13	31299	3,38	372
	14	32586	4,5	510
	15	32509	6,19	1061
	16	31257	8,84	2566
	17	31082	12,48	4360
	18	31501	17,48	9004
	19	2875	207,68	127946
	20	0	600000	600000
	21	0	600000	600000
	22	0	600000	600000
	23	0	600000	600000
FR: FURTHEST	0	25048	22,68	23
	1	25270	22,6	23
	2	25191	22,67	24
	3	25184	22,72	25
	4	25189	22,74	26
	5	24972	22,85	27
	6	25027	22,88	28
	7	24856	22,92	29
	8	25069	22,91	29
	9	24955	22,94	30
	10	25029	22,96	32
	11	25011	22,98	32
	12	24951	22,99	32
	13	24932	23,01	32
	14	24978	23,01	32

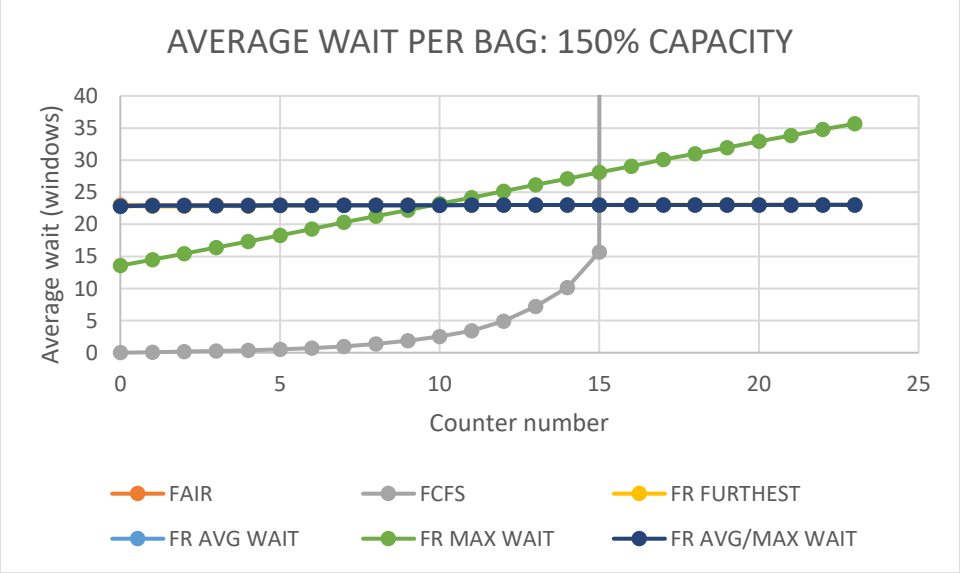
	15	24990	23,01	32
	16	24893	23,03	32
	17	24923	23,03	32
	18	24967	23,03	32
	19	24901	23,04	39
	20	24915	23,06	41
	21	24936	23,06	41
	22	24890	23,06	41
	23	24928	23,07	41
FR: AVG WAIT	0	25598	20,61	23
	1	25218	21,12	23
	2	25549	21,44	24
	3	25532	21,72	25
	4	24959	22,13	26
	5	25248	22,35	27
	6	24940	22,57	28
	7	24848	22,73	29
	8	25208	22,73	30
	9	25121	22,76	30
	10	24866	22,88	32
	11	24917	22,94	32
	12	24940	22,96	32
	13	24983	22,98	32
	14	25014	22,98	36
	15	24917	23,01	36
	16	24922	23,03	36
	17	24923	23,03	36
	18	24962	23,04	36
	19	24906	23,05	36
	20	24357	23,14	42
	21	24722	23,27	43
	22	24651	23,28	43
	23	24700	23,29	43
FR: MAX WAIT	0	31551	12,1	21
	1	31270	13,08	21
	2	31736	14	22
	3	31444	14,91	23
	4	29522	15,95	24
	5	31685	16,8	26
	6	30606	17,78	26
	7	29991	18,72	27
	8	29036	19,65	28
	9	27722	20,64	29
	10	26423	21,63	30
	11	25400	22,58	30

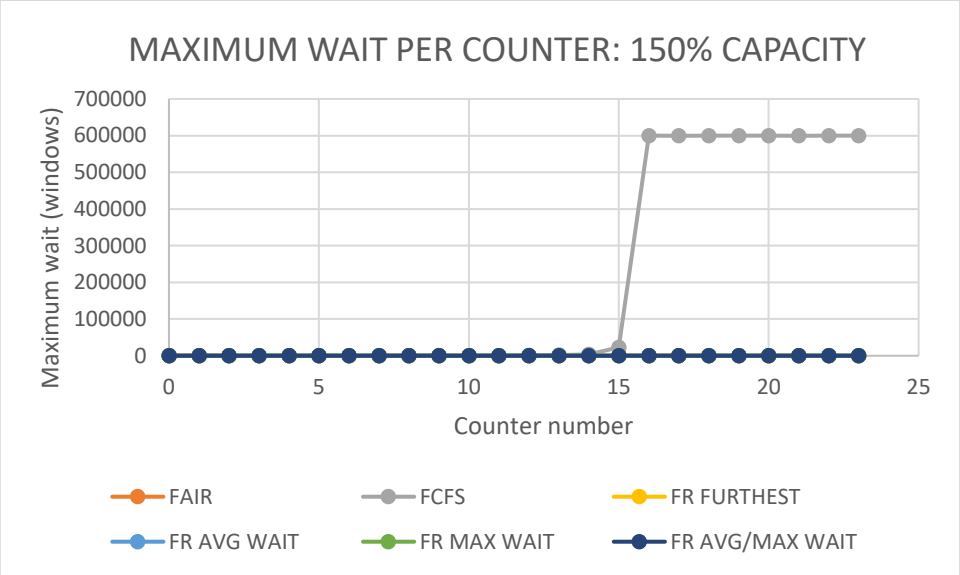
	12	24334	23,58	32
	13	23443	24,57	33
	14	22612	25,53	34
	15	21804	26,52	34
	16	21016	27,5	35
	17	20344	28,45	36
	18	19730	29,41	38
	19	19127	30,36	40
	20	18527	31,34	40
	21	18038	32,26	40
	22	17535	33,2	42
	23	17103	34,08	42
FR: AVG/ MAX WAIT	0	25045	22,62	23
	1	25030	22,74	23
	2	25169	22,74	24
	3	25131	22,79	25
	4	25098	22,81	25
	5	25152	22,81	27
	6	25067	22,87	27
	7	25031	22,9	27
	8	25095	22,9	30
	9	25106	22,9	31
	10	24993	22,93	31
	11	25023	22,95	31
	12	24961	22,97	32
	13	24982	22,99	34
	14	25006	22,99	34
	15	25005	22,99	34
	16	24954	23,01	34
	17	24954	23,01	34
	18	24988	23,01	34
	19	24977	23,01	35
	20	24724	23,07	42
	21	24839	23,15	43
	22	24827	23,15	43
	23	24835	23,16	43

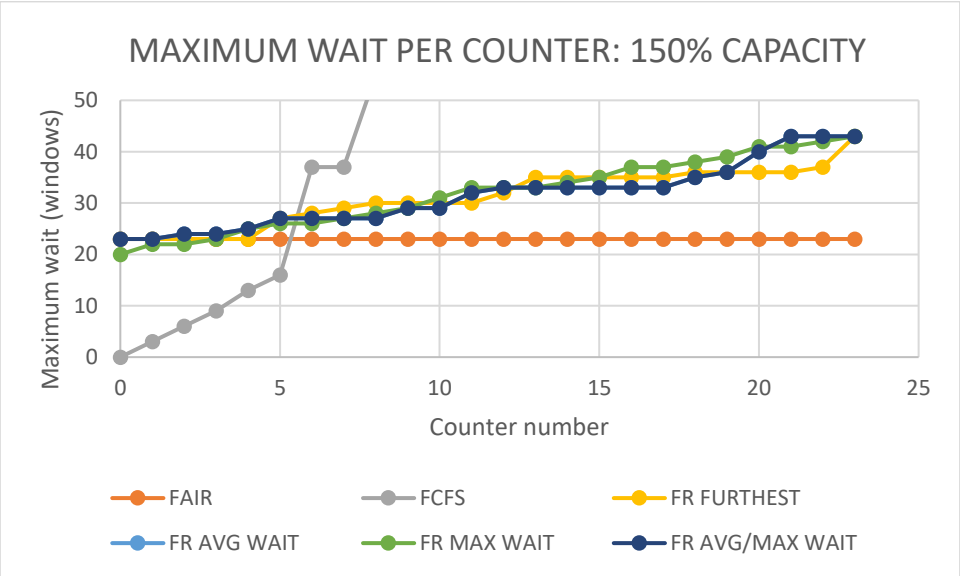
9.3.8 Comparison of strategies at 150% of system capacity








MAXIMUM WAIT PER COUNTER: 150% CAPACITY
Maximum wait (windows)
700000
600000
500000
400000
300000
200000
100000
0
0
5
10
15
20
25
Counter number
FAIR
FCFS
FR FURTHEST
FR AVG WAIT
FR MAX WAIT
FR AVG/MAX WAIT


MAXIMUM WAIT PER COUNTER: 150% CAPACITY
Maximum wait (windows)
50
40
30
20
10
0
0
5
10
15
20
25
Counter number
FAIR
FCFS
FR FURTHEST
FR AVG WAIT
FR MAX WAIT
FR AVG/MAX WAIT

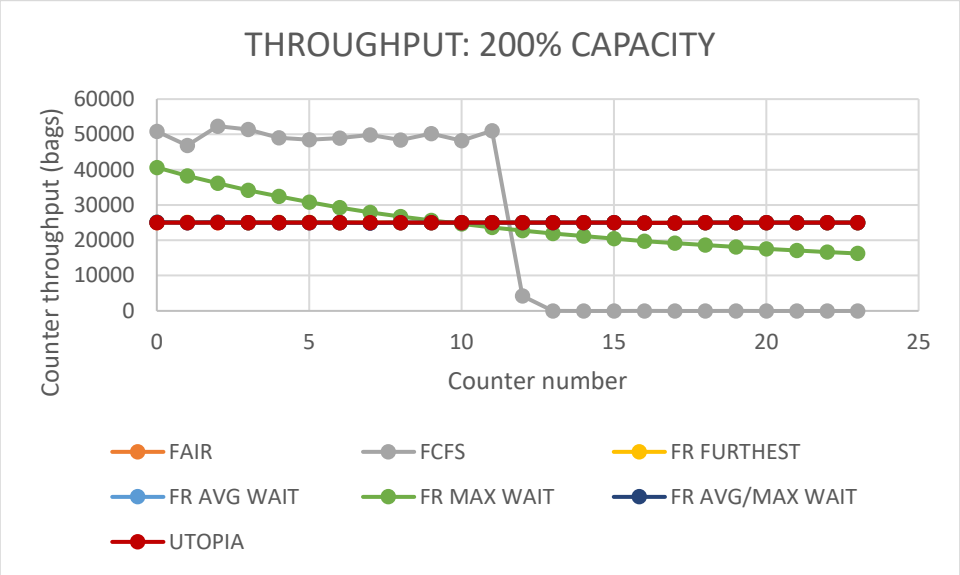
STRATEGY	COUNTER	THROUGHPUT (bags)	AVERAGE WAIT (windows)	MAXIMUM WAIT (windows)
FAIR	0	24994	23	23
	1	24999	23	23
	2	24999	23	23
	3	24965	22,99	23
	4	24999	23	23
	5	24999	23	23
	6	24963	22,98	23
	7	24923	22,99	23
	8	24998	23	23
	9	24999	23	23
	10	24999	23	23
	11	24999	23	23
	12	24998	23	23
	13	24959	22,99	23
	14	24999	23	23
	15	24999	23	23
	16	24961	23	23
	17	24957	23	23
	18	24999	23	23
	19	24966	22,99	23
	20	24987	23	23
	21	24999	23	23
	22	24944	22,99	23
	23	24999	23	23
FCFS	0	38299	0	0
	1	35396	0,07	3
	2	39308	0,15	6
	3	38905	0,25	9
	4	37214	0,39	13
	5	36368	0,54	16
	6	37133	0,73	37
	7	37719	0,99	37
	8	36525	1,35	55
	9	37601	1,86	142
	10	36247	2,52	219
	11	38628	3,43	377
	12	38006	4,92	644
	13	37523	7,19	1824
	14	39151	10,12	3870
	15	35977	15,68	22405
	16	0	600000	600000
	17	1	599968	599968

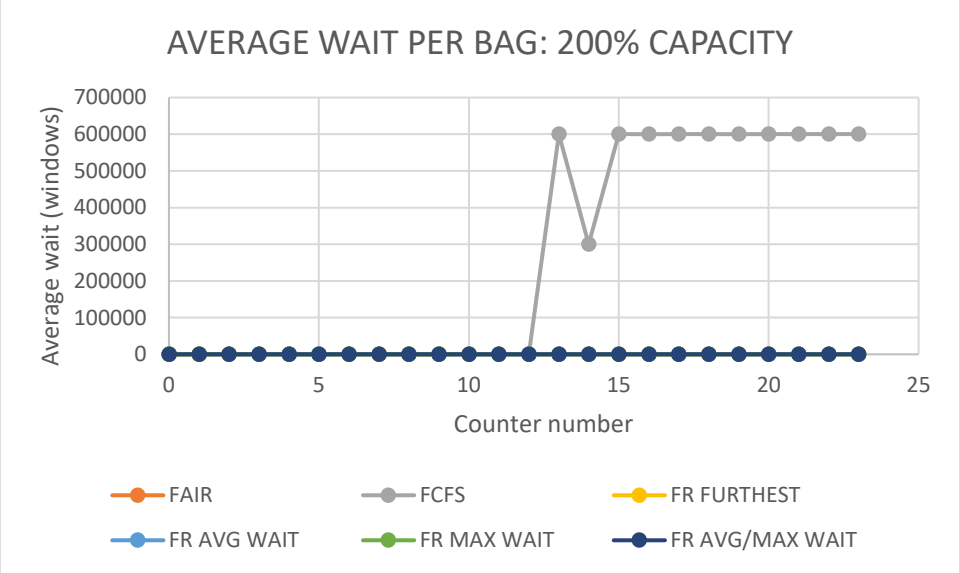
	18	0	600000	600000
	19	0	600000	600000
	20	0	600000	600000
	21	1	599977	599977
	22	0	600000	600000
	23	0	600000	600000
FR: FURTHEST	0	25060	22,85	23
	1	25204	22,79	23
	2	25219	22,79	23
	3	25096	22,85	23
	4	25183	22,82	23
	5	25029	22,89	27
	6	25001	22,94	28
	7	24923	22,98	29
	8	24991	23,01	30
	9	24997	23	30
	10	24995	23	30
	11	24992	23,01	30
	12	24973	23,01	32
	13	24925	23,03	35
	14	24963	23,03	35
	15	24963	23,03	35
	16	24922	23,04	35
	17	24917	23,04	35
	18	24951	23,05	36
	19	24950	23,05	36
	20	24946	23,05	36
	21	24950	23,05	36
	22	24909	23,05	37
	23	24941	23,06	43
FR: AVG WAIT	0	25103	22,81	23
	1	24962	22,91	23
	2	25059	22,91	24
	3	25037	22,93	24
	4	25042	22,93	25
	5	25059	22,94	27
	6	24999	22,96	27
	7	24963	22,98	27
	8	25022	22,98	27
	9	25022	22,98	29
	10	25013	22,98	29
	11	25013	22,99	32
	12	24998	22,99	33
	13	24978	23	33
	14	24999	23	33

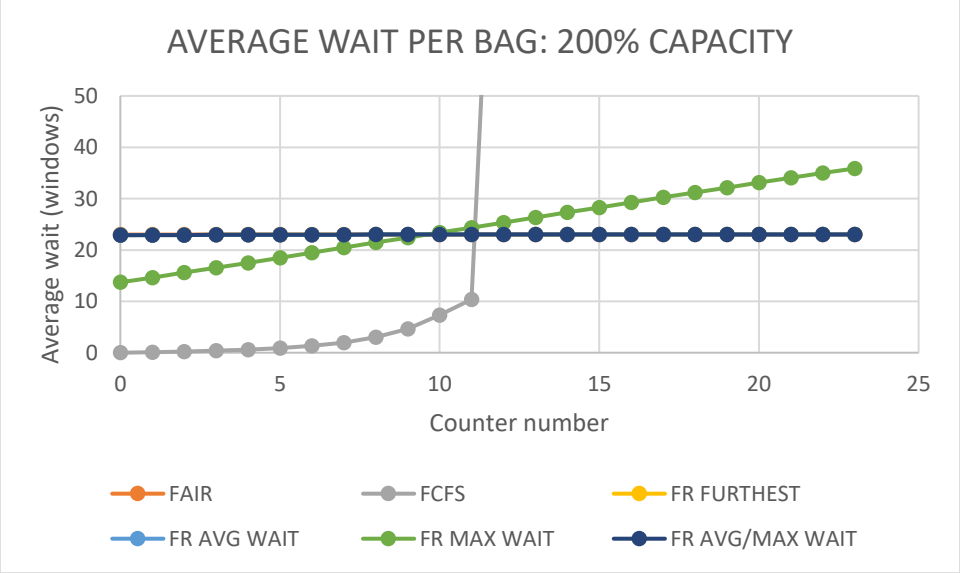
	15	24998	23	33
	16	24957	23,01	33
	17	24957	23,01	33
	18	24985	23,01	35
	19	24982	23,01	36
	20	24955	23,02	40
	21	24968	23,03	43
	22	24960	23,03	43
	23	24966	23,03	43
FR: MAX WAIT	0	38009	13,58	20
	1	37175	14,5	22
	2	36285	15,41	22
	3	34491	16,37	23
	4	32693	17,33	25
	5	31094	18,29	26
	6	29534	19,28	26
	7	28099	20,3	27
	8	26961	21,25	28
	9	25848	22,21	29
	10	24788	23,19	31
	11	23848	24,16	33
	12	22952	25,13	33
	13	22071	26,16	33
	14	21334	27,12	34
	15	20616	28,1	35
	16	19929	29,06	37
	17	19284	30,07	37
	18	18741	31,01	38
	19	18215	31,94	39
	20	17683	32,93	41
	21	17224	33,83	41
	22	16760	34,79	42
	23	16367	35,66	43
FR: AVG/MAX WAIT	0	25103	22,81	23
	1	24962	22,91	23
	2	25059	22,91	24
	3	25037	22,93	24
	4	25042	22,93	25
	5	25059	22,94	27
	6	24999	22,96	27
	7	24963	22,98	27
	8	25022	22,98	27
	9	25022	22,98	29
	10	25013	22,98	29
	11	25013	22,99	32

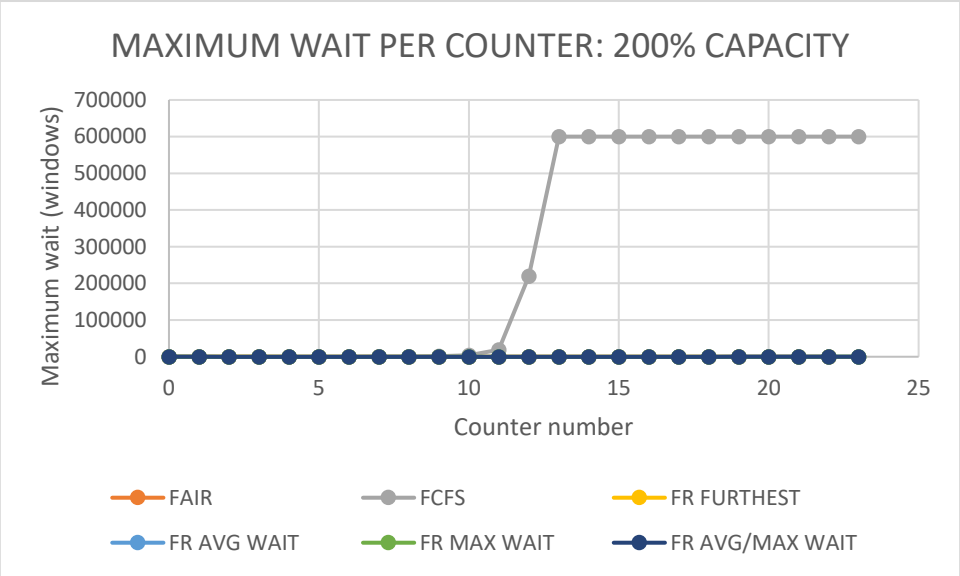
	12	24998	22,99	33
	13	24978	23	33
	14	24999	23	33
	15	24998	23	33
	16	24957	23,01	33
	17	24957	23,01	33
	18	24985	23,01	35
	19	24982	23,01	36
	20	24955	23,02	40
	21	24968	23,03	43
	22	24960	23,03	43
	23	24966	23,03	43

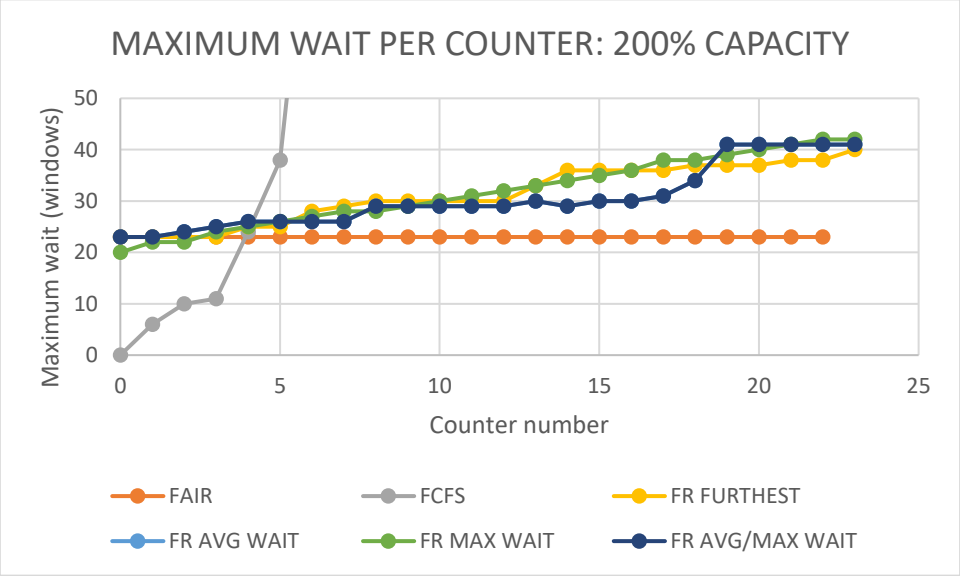
9.3.9 Comparison of strategies at 200% of system capacity








MAXIMUM WAIT PER COUNTER: 200% CAPACITY
Maximum wait (windows)
700000
600000
500000
400000
300000
200000
100000
0
0
5
10
15
20
25
Counter number
FAIR
FCFS
FR FURTHEST
FR AVG WAIT
FR MAX WAIT
FR AVG/MAX WAIT


MAXIMUM WAIT PER COUNTER: 200% CAPACITY
Maximum wait (windows)
50
40
30
20
10
0
0
5
10
15
20
25
Counter number
FAIR
FCFS
FR FURTHEST
FR AVG WAIT
FR MAX WAIT
FR AVG/MAX WAIT

STRATEGY	COUNTER	THROUGHPUT (bags)	AVERAGE WAIT (windows)	MAXIMUM WAIT (windows)
FAIR	0	24999	23	23
	1	24999	23	23
	2	24999	23	23
	3	24985	22,99	23
	4	24999	23	23
	5	25000	23	23
	6	24977	22,99	23
	7	24992	23	23
	8	24999	23	23
	9	24999	23	23
	10	24999	23	23
	11	24999	23	23
	12	24999	23	23
	13	24969	22,99	23
	14	24999	23	23
	15	24999	23	23
	16	24961	23	23
	17	24957	23	23
	18	24999	23	23
	19	24998	23	23
	20	24997	23	23
	21	24999	23	23
	22	24980	22,99	23
	23	24999	23	23
FCFS	0	50837	0	0
	1	46868	0,09	6
	2	52340	0,2	10
	3	51410	0,38	11
	4	49037	0,61	24
	5	48525	0,91	38
	6	48928	1,34	95
	7	49871	1,98	157
	8	48438	3	966
	9	50199	4,61	1549
	10	48212	7,35	4459
	11	51066	10,37	18884
	12	4268	139,58	219326
	13	0	600000	600000
	14	2	299989,5	600000
	15	1	599990	600000
	16	0	600000	600000
	17	1	599968	600000

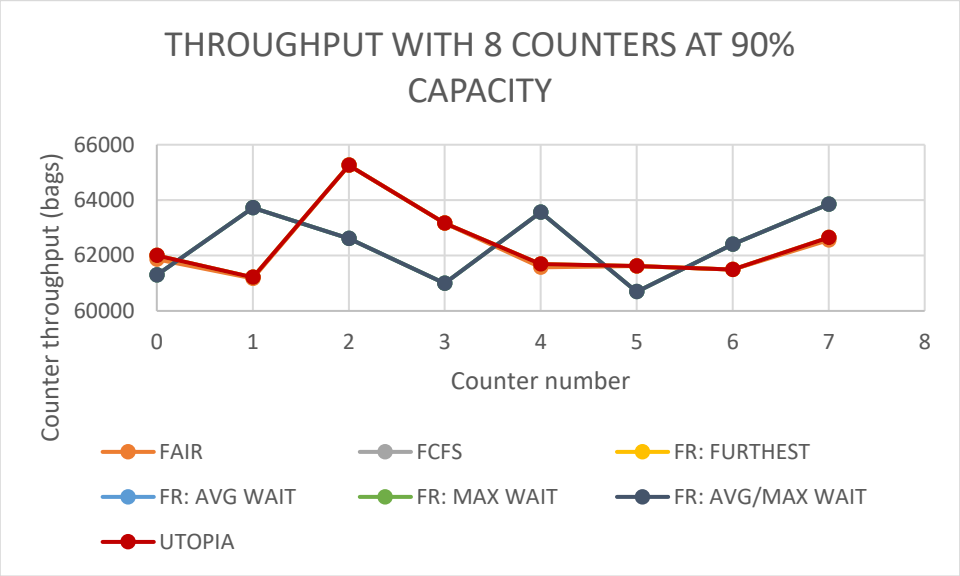
	18	1	599986	600000
	19	0	600000	600000
	20	1	599928	600000
	21	1	599977	600000
	22	1	599984	600000
	23	1	599978	600000
FR: FURTHEST	0	25056	22,92	23
	1	25092	22,91	23
	2	25090	22,91	23
	3	25013	22,95	23
	4	25058	22,94	25
	5	25057	22,94	25
	6	24990	22,97	28
	7	24992	22,99	29
	8	25004	23	30
	9	25005	22,99	30
	10	25004	23	30
	11	25003	23	30
	12	25002	23	30
	13	24960	23,01	33
	14	24984	23,01	36
	15	24984	23,02	36
	16	24944	23,02	36
	17	24939	23,02	36
	18	24973	23,03	37
	19	24970	23,03	37
	20	24969	23,03	37
	21	24970	23,03	38
	22	24968	23,03	38
	23	24968	23,03	40
FR: AVG WAIT	0	25103	22,88	23
	1	25010	22,93	23
	2	25069	22,93	24
	3	25024	22,96	25
	4	25045	22,96	26
	5	25045	22,96	26
	6	24989	22,98	26
	7	24964	22,99	26
	8	25003	23	29
	9	25003	23	29
	10	25000	23	29
	11	25000	23	29
	12	25000	23	29
	13	24982	23,01	30
	14	24992	23,01	29

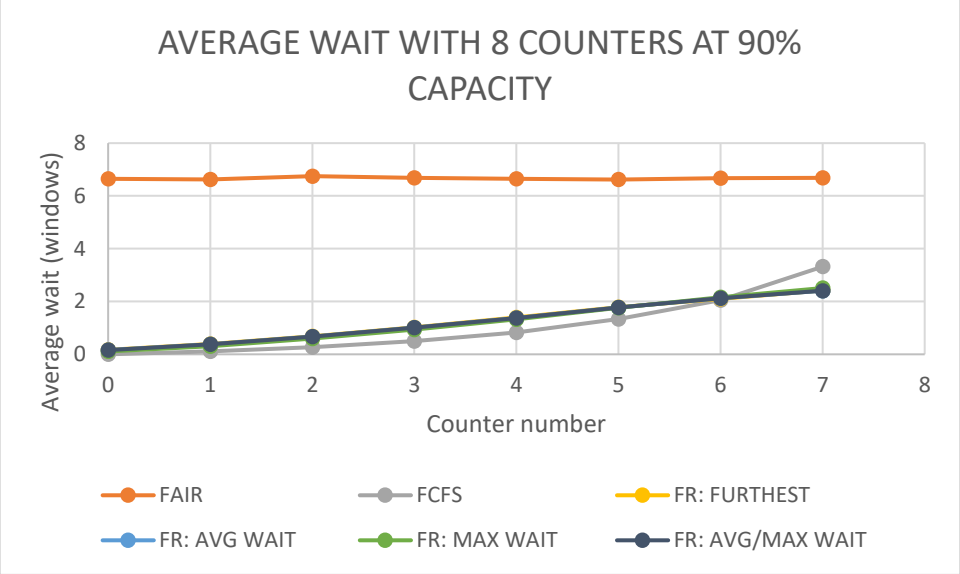
	15	24991	23,01	30
	16	24955	23,01	30
	17	24956	23,01	31
	18	24981	23,02	34
	19	24979	23,02	41
	20	24978	23,02	41
	21	24978	23,02	41
	22	24972	23,02	41
	23	24975	23,02	41
FR: MAX WAIT	0	40641	13,71	20
	1	38217	14,63	22
	2	36193	15,58	22
	3	34160	16,55	24
	4	32414	17,51	25
	5	30822	18,47	26
	6	29252	19,48	27
	7	27886	20,47	28
	8	26738	21,44	28
	9	25608	22,43	29
	10	24604	23,39	30
	11	23661	24,36	31
	12	22775	25,34	32
	13	21949	26,32	33
	14	21192	27,31	34
	15	20510	28,25	35
	16	19790	29,28	36
	17	19185	30,23	38
	18	18631	31,2	38
	19	18106	32,14	39
	20	17596	33,1	40
	21	17124	34,04	41
	22	16677	34,97	42
	23	16274	35,87	42
FR: AVG/ MAX WAIT	0	25103	22,88	23
	1	25010	22,93	23
	2	25069	22,93	24
	3	25024	22,96	25
	4	25045	22,96	26
	5	25045	22,96	26
	6	24989	22,98	26
	7	24964	22,99	26
	8	25003	23	29
	9	25003	23	29
	10	25000	23	29
	11	25000	23	29

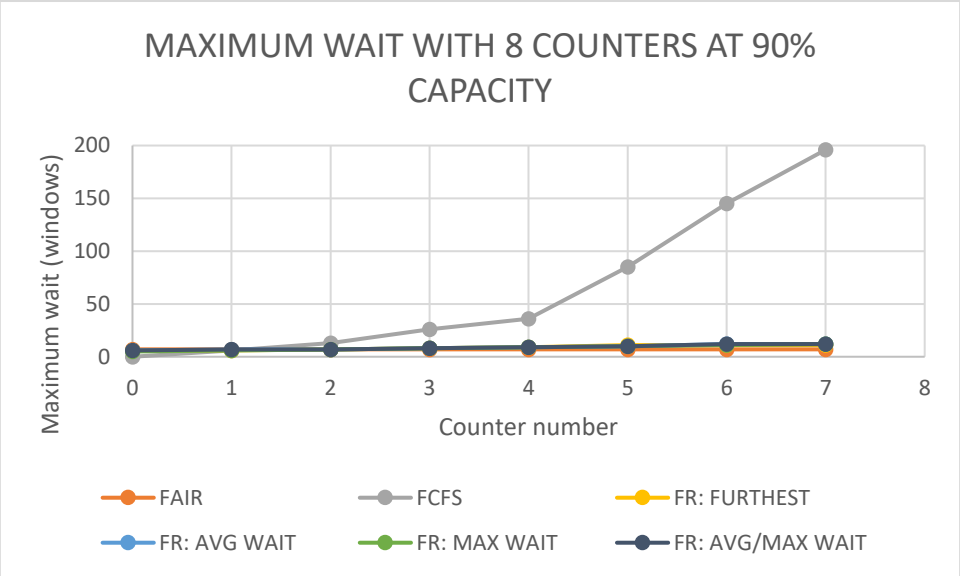
	12	25000	23	29
	13	24982	23,01	30
	14	24992	23,01	29
	15	24991	23,01	30
	16	24955	23,01	30
	17	24956	23,01	31
	18	24981	23,02	34
	19	24979	23,02	41
	20	24978	23,02	41
	21	24978	23,02	41
	22	24972	23,02	41
	23	24975	23,02	41

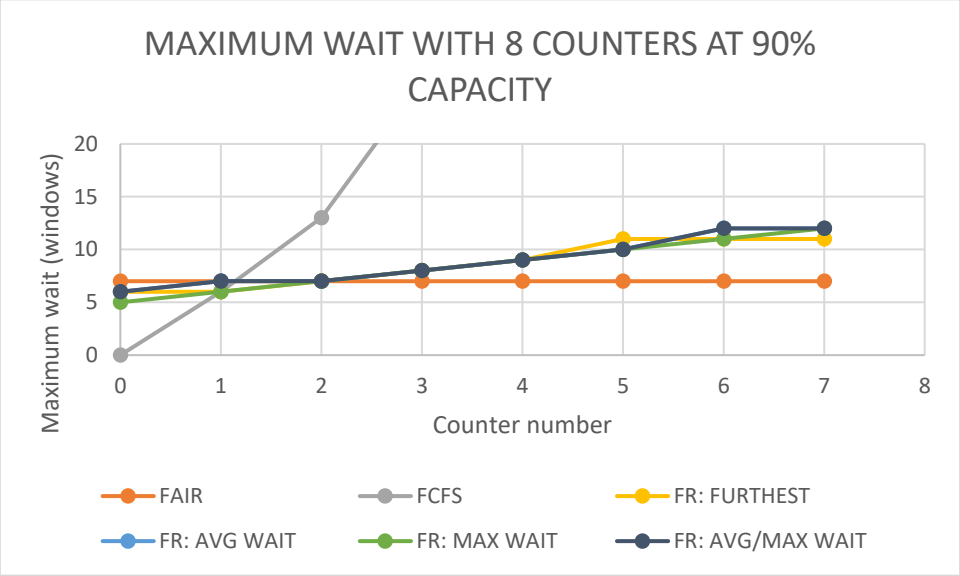
9.4 Check-in counters. Sensitivity of the strategies to the number of counters

9.4.1 Comparison of strategies with 8 counters

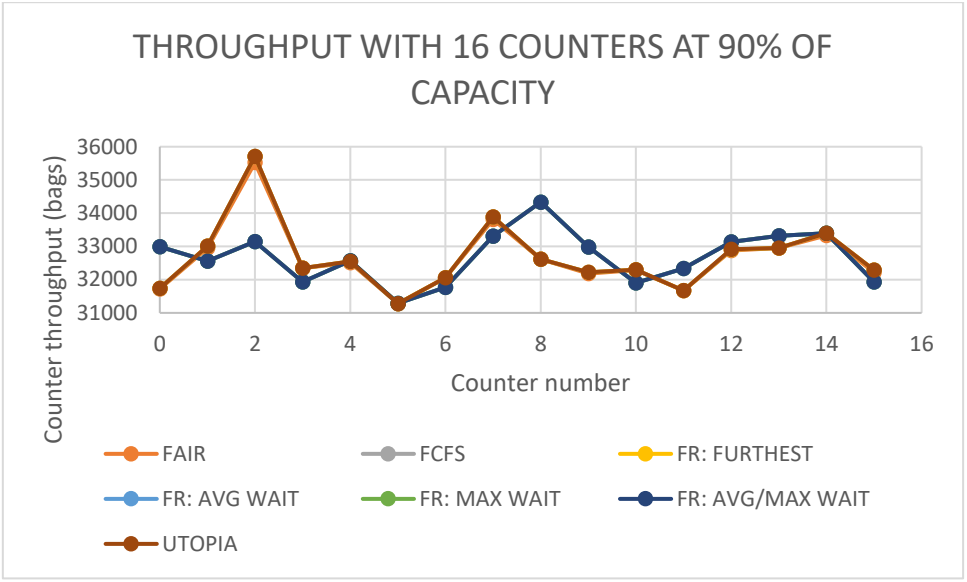


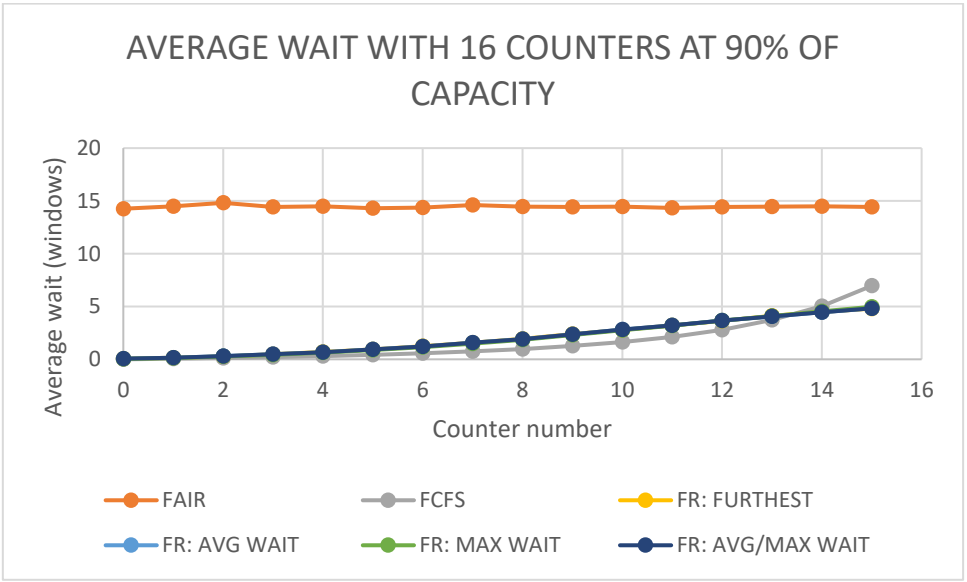


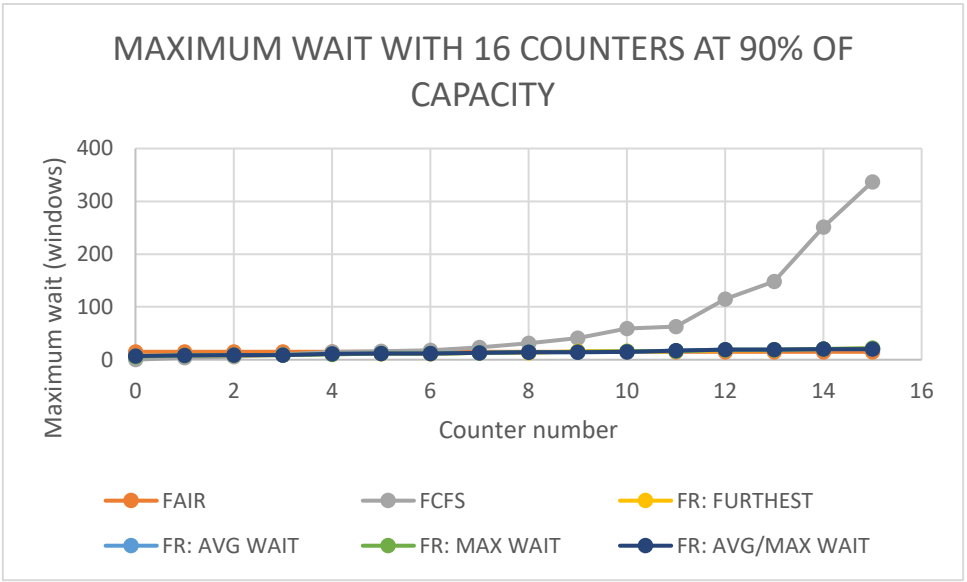

MAXIMUM WAIT WITH 8 COUNTERS AT 90% CAPACITY
Maximum wait (windows)
200
150
100
50
0
0
1
2
3
4
5
6
7
8
Counter number
FAIR
FCFS
FR: FURTHEST
FR: AVG WAIT
FR: MAX WAIT
FR: AVG/MAX WAIT

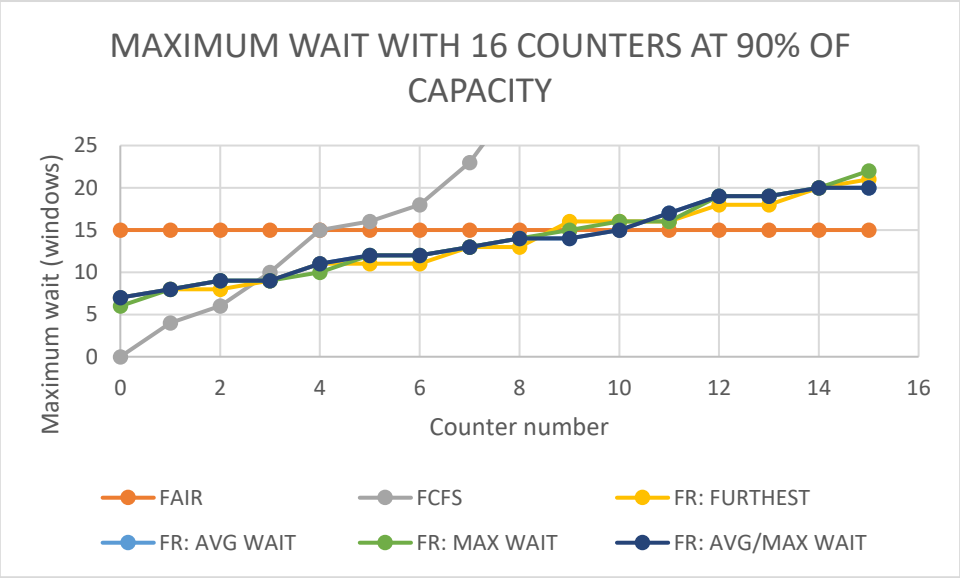

MAXIMUM WAIT WITH 8 COUNTERS AT 90% CAPACITY
Maximum wait (windows)
20
15
10
5
0
0
1
2
3
4
5
6
7
8
Counter number
FAIR
FCFS
FR: FURTHEST
FR: AVG WAIT
FR: MAX WAIT
FR: AVG/MAX WAIT

9.4.2 Comparison of strategies with 16 counters



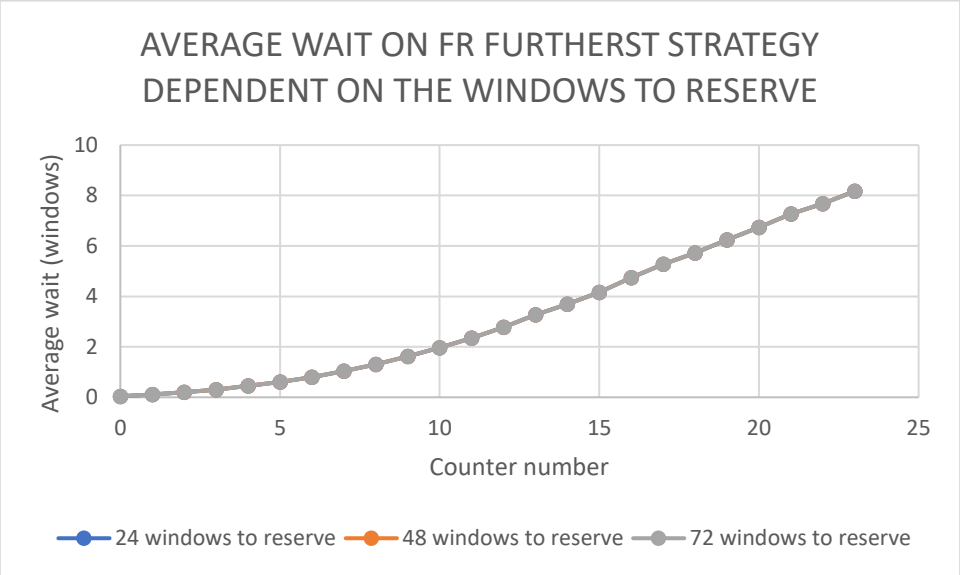


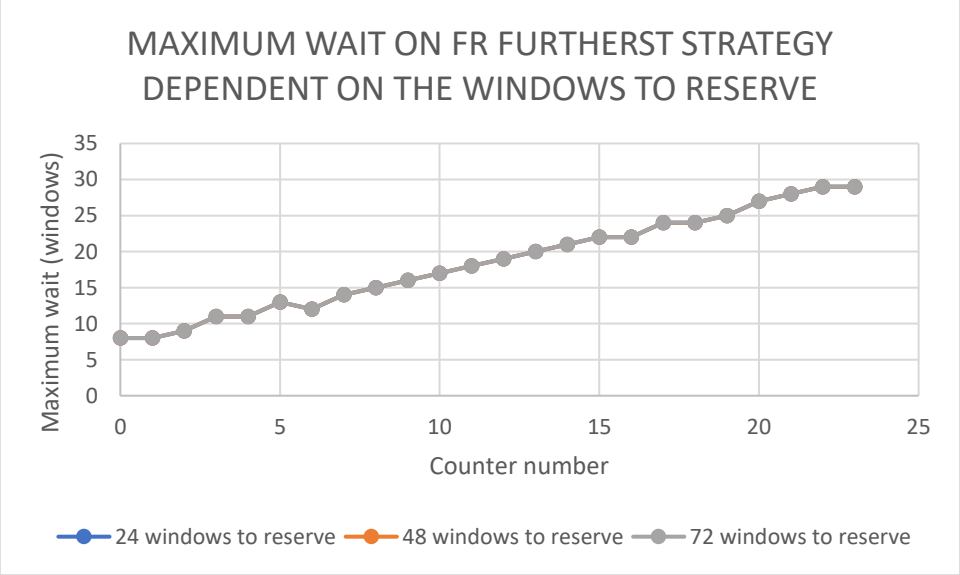


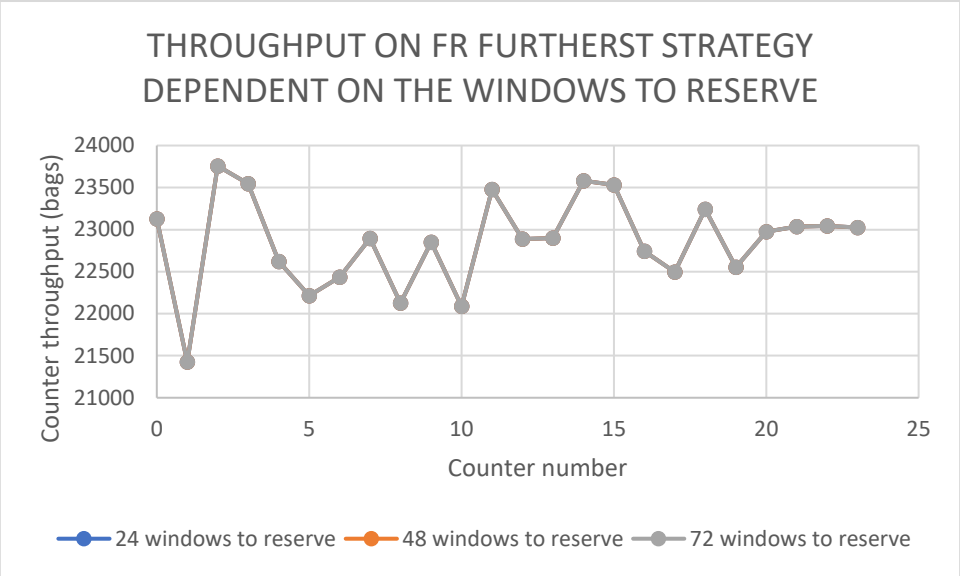

MAXIMUM WAIT WITH 16 COUNTERS AT 90% OF CAPACITY
Maximum wait (windows)
25
20
15
10
5
0
0
2
4
6
8
10
12
14
16
Counter number
FAIR
FCFS
FR: FURTHEST
FR: AVG WAIT
FR: MAX WAIT
FR: AVG/MAX WAIT

9.5 Check-in counters. Sensitivity of the forward reservation strategies to the number of windows to reserve

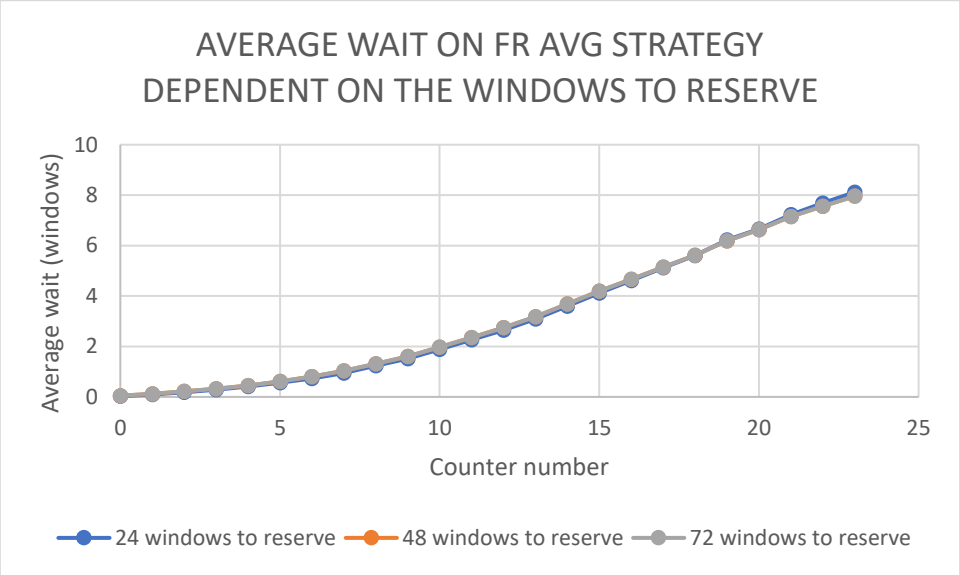
9.5.1 FR: THE FURTHEST strategy. Sensitivity to the number of windows to reserve

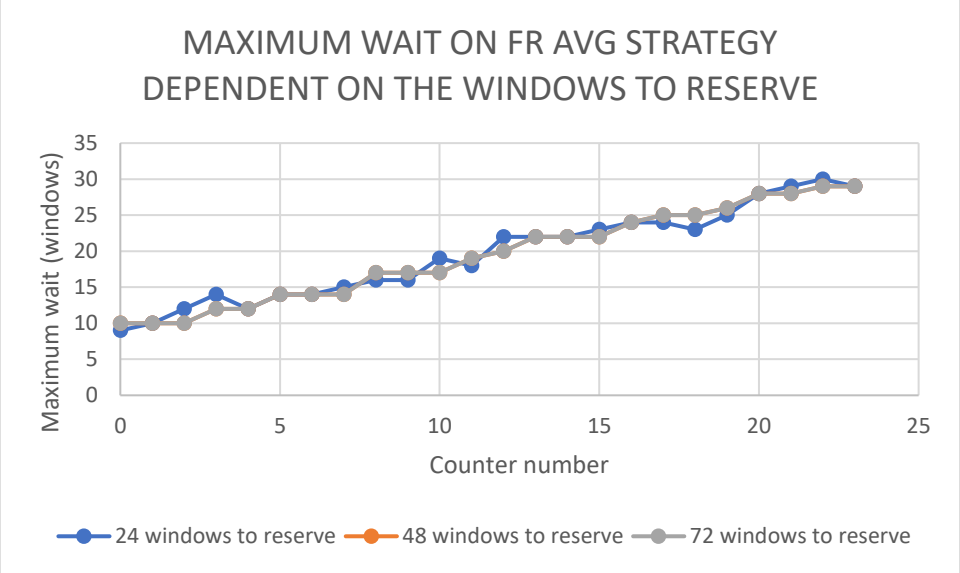


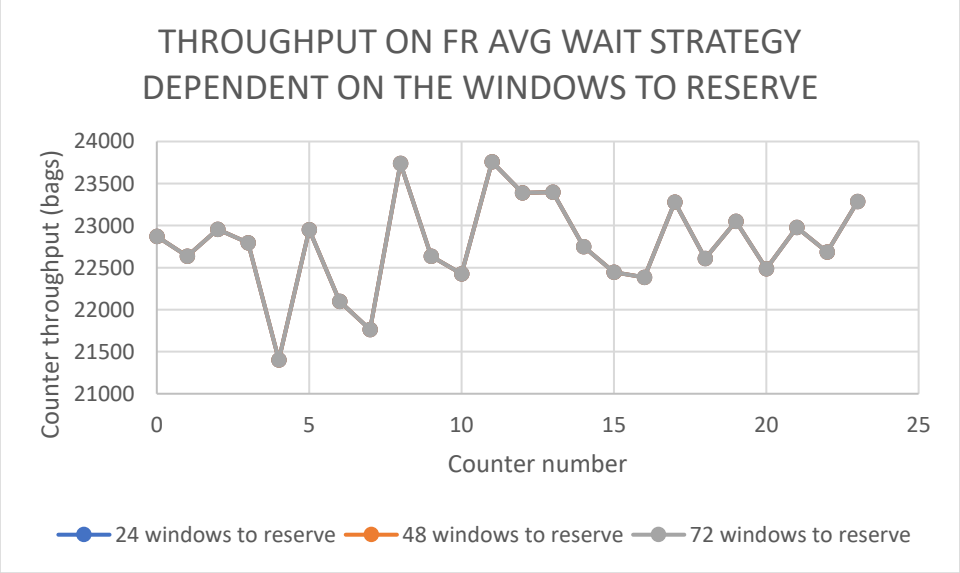



THROUGHPUT ON FR FURTHERST STRATEGY DEPENDENT ON THE WINDOWS TO RESERVE
Counter throughput (bags)
24000
23500
23000
22500
22000
21500
21000
0
5
10
15
20
25
Counter number
24 windows to reserve
48 windows to reserve
72 windows to reserve

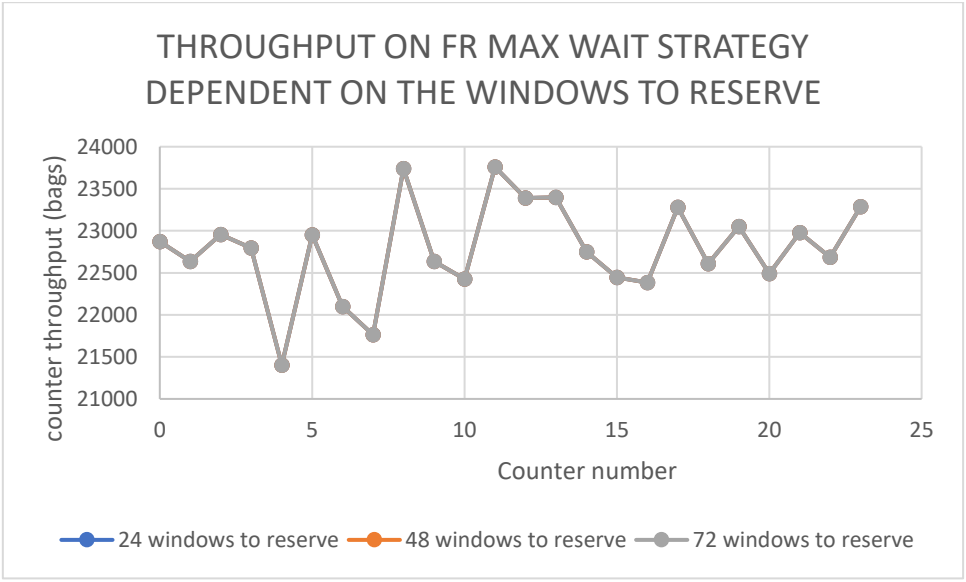
9.5.2 FR: AVG WAIT strategy. Sensitivity to the number of windows to reserve

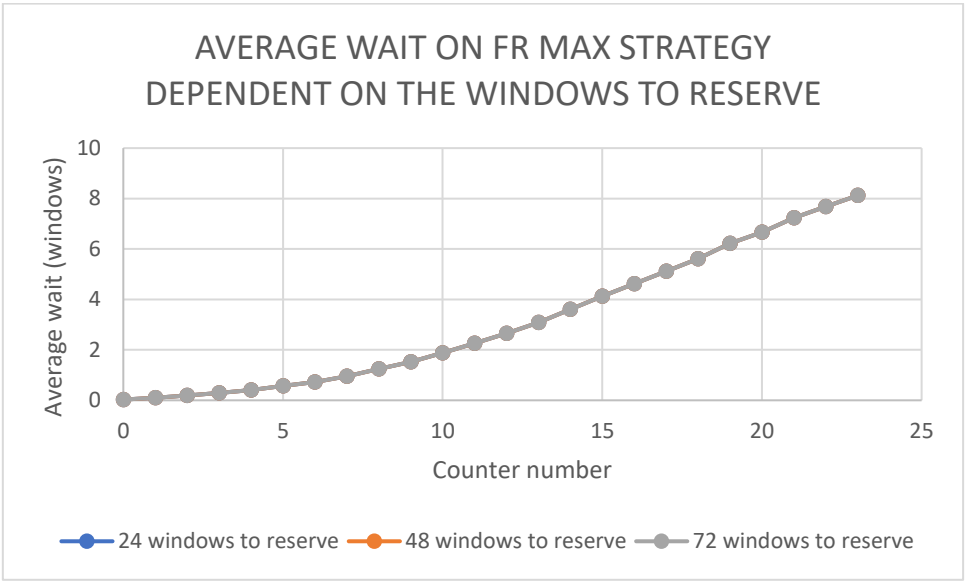


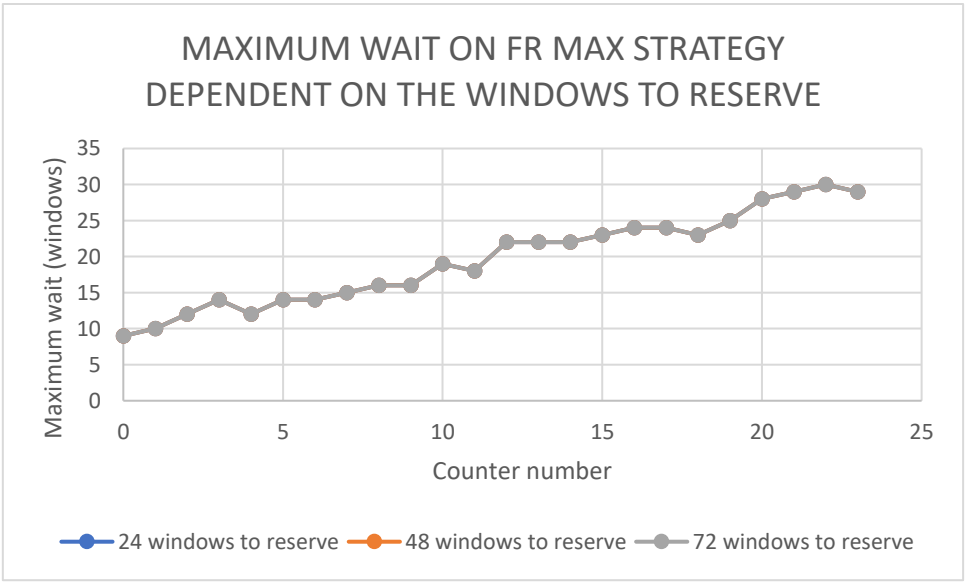




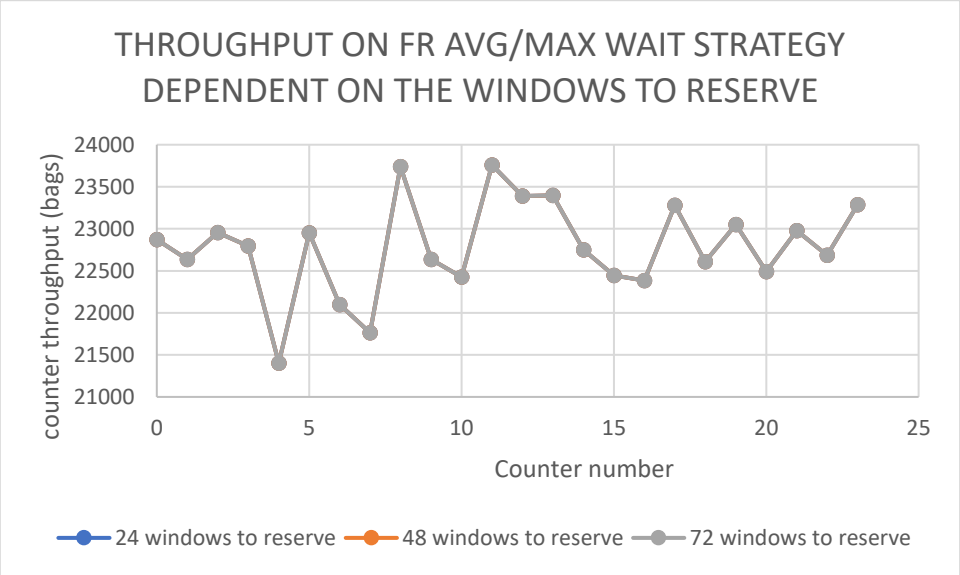
9.5.3 FR: MAX WAIT strategy. Sensitivity to the number of windows to reserve

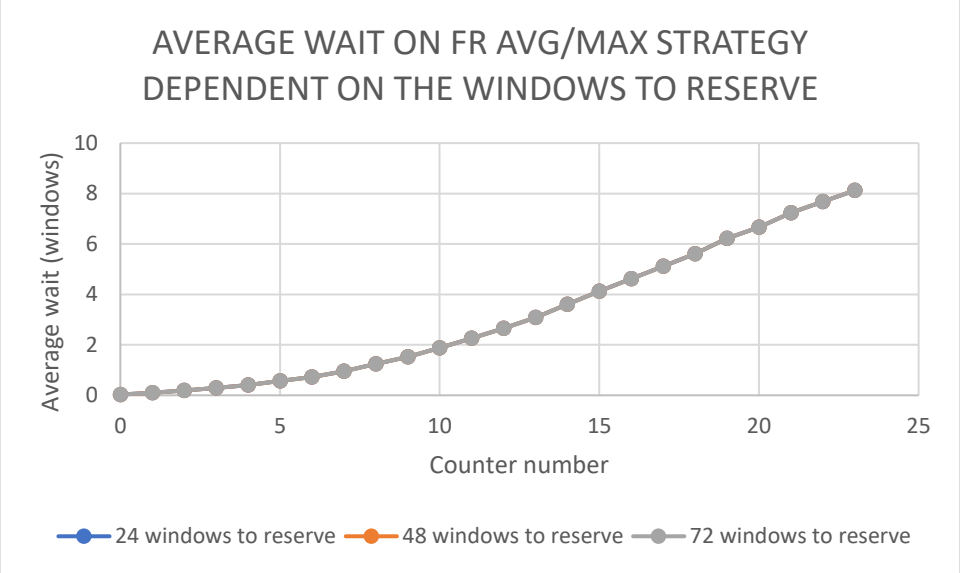


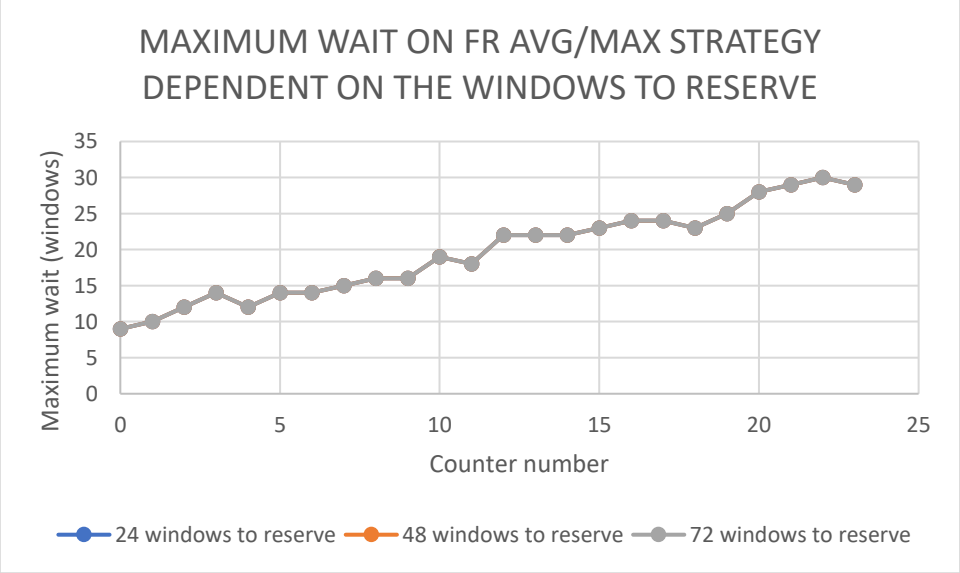




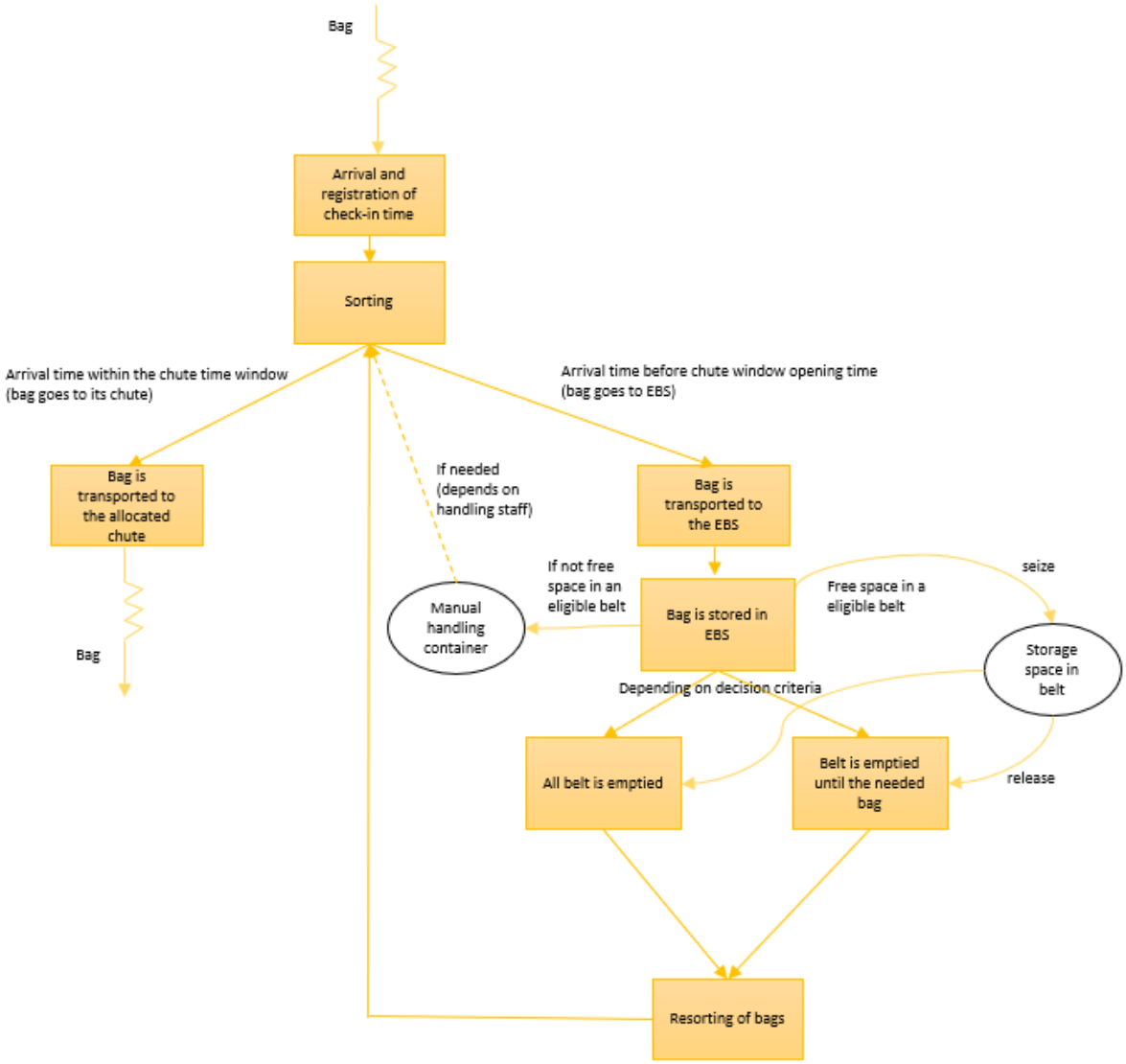
9.5.4 FR: AVG/MAX WAIT strategy. Sensitivity to the number of windows to reserve







9.6 Early Baggage Storage. Activity diagram



9.7 Early baggage storage simulator in Python

```
# INCLUDES
#-------------------------------------------------------------------------
import random
import csv
import sys

# DEFINITION OF INDEXES FOR BAGS AND BELTS MATRIXES
#-------------------------------------------------------------------------
# for belts matrix
nBagsC = 0
capacityC = 1
releaseTimeC = 2
listBagsC = 3
bagIDPanicTimeC = 4
panicTimeC = 5
eligibleC = 6
countEmptiedPlannedReleaseC = 7
labelsC = 8
countEmptiedPanicC = 9
restC = 10

#for bags matrix
bagIDC = 0
flightC = 1
checkInTimeC = 2
openingTimeC = 3
closingTimeC = 4

#partial count
partCountManualHandledBags = 0
partCountEmptiedPlannedReleaseC = 1
partCountEmptiedPanicC = 2
partCountStressC = 3

# FUNCTIONS
#-------------------------------------------------------------------------

def getData(fileName):
dataBags = []
fn = open(fileName,'r')

for line in fn:
dataBags.append(line.replace('\n',''))
return dataBags
```

```
def splitData(data):
#n = len(data)
outData = []
delimiter = ';'
for x in data:
element = x.split(delimiter)
element = list(map(int, element))
outData.append(element)
return outData


def addBag (bags, bagID, beltchosen, panictime):
belts[beltchosen][listBagsC].append(bagID)
belts[beltchosen][nBagsC] = len(belts[beltchosen][listBagsC])
#in case the bag has the same panic time than the belt, include the bagID
with the list of critical bags (bags which need to be released for the
panic time)
if (panictime == belts[beltchosen][panicTimeC]):
belts[beltchosen][bagIDPanicTimeC].append(bagID)

elif (panictime < belts[beltchosen][panicTimeC] ):
belts[beltchosen][bagIDPanicTimeC] = [bagID] #this assigns a list that
only contains the value bagID
belts[beltchosen][panicTimeC] = panictime

def goToEBS (openingtime, iteration):
#decide if it goes to EBS or not
if (iteration < openingtime):
EBS = 1 #bag is "early" or "too early" ("too early" bags will be
accepted)
else:
EBS = 0 #bag is "on time" or "too late"
return EBS

def eligible (openingtime,  sortingtime, panictime, way):
for k in range(nbelts):
belts[k][eligibleC] = 0
if ( (openingtime-sortingtime) <= belts[k][releaseTimeC] <= panictime ):
belts[k][eligibleC] = 1
if(way == 1): #belt 13 is "rest"
belts[13][eligibleC] = 1
belts[12][eligibleC] = 1
elif (way == 2): #belt 13 is "rest"
belts[13][eligibleC] = 1
belts[12][eligibleC] = 1
elif (way == 3):  #belts 12 and 13 are "rest"
belts[12][eligibleC] = 1
belts[13][eligibleC] = 1
```

```
def emptyAllBelt (beltToEmpty, iteration):
partialCount[partCountStressC] = partialCount[partCountStressC] +
len(belts[beltToEmpty][listBagsC])
while(len(belts[beltToEmpty][listBagsC]) != 0):
bagIDLoading = belts[beltToEmpty][listBagsC].pop(0) #this is the bagID
that is going to be loaded again the sorting system
for j in range(len(bags)):
if(bagIDLoading == bags[j][bagIDC]):
bags[j][checkInTimeC] = iteration
break
belts[beltToEmpty][nBagsC] = 0


def emptyUntilPosition (beltToEmpty):
bagID = belts[beltToEmpty][listBagsC][-1]
position = belts[beltToEmpty][listBagsC].index(bagID)
for k in range(position):
belts[beltToEmpty][listBagsC].pop(0)
belts[beltToEmpty][nBagsC] = belts[beltToEmpty][nBagsC] - 1
k=k #just to kill unsused notification
#bags should also be checked in again in the system

def chooseBeltLeastFilledLessBags():
beltLeastFilled = -1
minbagsinbelt = 100000
pool = []
for i in range(nbelts):
if ( (belts[i][eligibleC] == 1) and (belts[i][nBagsC] <
belts[i][capacityC]) and (belts[i][nBagsC] <= minbagsinbelt) ):
if (belts[i][nBagsC] < minbagsinbelt):
minbagsinbelt = belts[i][nBagsC]
beltLeastFilled = i
pool.clear()
pool.append(beltLeastFilled)
elif (belts[i][nBagsC] == minbagsinbelt):
pool.append(i)
if (len(pool) != 0):
beltLeastFilled = random.choice(pool)
return beltLeastFilled

def chooseBeltLeastFilledMoreFreeSpace():
beltLeastFilled = -1
maxFreeSpace = 0
pool = []
for i in range(nbelts):
freeSpace = belts[i][capacityC] - belts[i][nBagsC]
if ( (belts[i][eligibleC] == 1) and (belts[i][nBagsC] <
belts[i][capacityC]) and (freeSpace > maxFreeSpace) ):
```

```
if (freeSpace > maxFreeSpace):
maxFreeSpace = freeSpace
beltLeastFilled = i
pool.clear()
pool.append(beltLeastFilled)
elif (freeSpace == maxFreeSpace):
pool.append(i)

if (len(pool) != 0):
beltLeastFilled = random.choice(pool)
return beltLeastFilled

def planB(): #this function is used when there is no space in the belt
the bag should go to. It goes to the most space belt
beltLeastFilled = -1
maxFreeSpace = 0
pool = []
for i in range(nbelts):
freeSpace = belts[i][capacityC] - belts[i][nBagsC]
if ( (belts[i][nBagsC] < belts[i][capacityC]) and (freeSpace >
maxFreeSpace) ):
if (freeSpace > maxFreeSpace):
maxFreeSpace = freeSpace
beltLeastFilled = i
pool.clear()
pool.append(beltLeastFilled)
elif (freeSpace == maxFreeSpace):
pool.append(i)
if (len(pool) != 0):
beltLeastFilled = random.choice(pool)
return beltLeastFilled

def chooseBeltCurrent(closingtime):
beltchosen = -1
releaseT = horizon*4
for i in range(nbelts):
if ( (belts[i][eligibleC] == 1) and (belts[i][releaseTimeC] < releaseT)
and (belts[i][nBagsC] < belts[i][capacityC])):
releaseT = belts[i][releaseTimeC]
beltchosen = i
if (beltchosen == -1): manual handling
return beltchosen
if (releaseT < (closingtime - 45*60)):
return beltchosen
releaseT = horizon*4
for i in range(nbelts): earliest eligible belt (regardless of the
capacity)
if ((belts[i][eligibleC] == 1) and (belts[i][releaseTimeC] < releaseT)):
releaseT = belts[i][releaseTimeC]
```

```
if ( belts[13][nBagsC] < belts[i][capacityC] ):
beltchosen = 13
belts[13][releaseTimeC] = releaseT
belts[13][labelsC] = [horizon*3 + 1]
return beltchosen
elif( belts[12][nBagsC] < belts[i][capacityC] ):
beltchosen = 12
belts[12][releaseTimeC] = releaseT
belts[12][labelsC] = [horizon*3]
return beltchosen
else:  #(other with enough capacity and change its release time)
done = 0
beltnumbers = [0,1,2,3,4,5,6,7,8,9,10,11]
while (len(beltnumbers) != 0):
beltchosen = int (random.choice(beltnumbers))
beltnumbers.remove(beltchosen)
if ( ( belts[beltchosen][nBagsC] < belts[beltchosen][capacityC] ) ):
done = 1
savingcurrentreleaseT = belts[beltchosen][releaseTimeC]
if (savingcurrentreleaseT < releaseT):
belts[beltchosen][labelsC].insert(0,releaseT)
if (savingcurrentreleaseT > releaseT):
belts[beltchosen][releaseTimeC] = releaseT
belts[beltchosen][labelsC].insert(0,savingcurrentreleaseT)
belts[beltchosen][labelsC].sort()
if (done == 0): #not space in any belt
beltchosen = -1
return beltchosen


def earliestbelt ():
beltchosen = -1
releaseT = horizon*4
for i in range(nbelts):
print('(belts[{0}][eligibleC] == {1}) and (belts[i][releaseTimeC]{2} <
releaseT {3}) and (belts[i][nBagsC] {4}< belts[i][capacityC] {5})'
.format(i, belts[i][eligibleC], belts[i][releaseTimeC],
releaseT,belts[i][nBagsC], belts[i][capacityC]  ))
if ( (belts[i][eligibleC] == 1) and (belts[i][releaseTimeC] < releaseT)
and (belts[i][nBagsC] < belts[i][capacityC])):
releaseT = belts[i][releaseTimeC]
beltchosen = i
return beltchosen


def latestbelt (): #should try to avoid using the "rest belt" if there is
any other
beltchosen = -1
releaseT = -1
```

```
for i in range(nbelts):
if ( (belts[i][eligibleC] == 1) and (belts[i][releaseTimeC] > releaseT)
and (belts[i][nBagsC] < belts[i][capacityC]) and (belts[i][restC] == 0)):
releaseT = belts[i][releaseTimeC]
beltchosen = i
if beltchosen != -1:
return beltchosen
releaseT = -1
for i in range(nbelts): #if it needs to use the rest
if ( (belts[i][restC] == 1) and (belts[i][eligibleC] == 1) and
(belts[i][releaseTimeC] > releaseT) and (belts[i][nBagsC] <
belts[i][capacityC])):
releaseT = belts[i][releaseTimeC]
beltchosen = i
return beltchosen


def deleteFile(FILENAME):
file = open(FILENAME,'w')
file.close


def writeUtilization(file, iteration):
delimiter = ';'
if iteration == 0:
file.write('Iteration' + delimiter)
for i in range(nbelts):
file.write('{0}'.format(belts[i][capacityC]) + delimiter)
file.write('Manually handled bags' + delimiter)
file.write('Planned emptying' + delimiter)
file.write('Panic Emptying' + delimiter)
file.write('Stress' + delimiter)
file.write('\n')

file.write('{0}' .format(iteration) + delimiter)
for i in range(nbelts):
file.write('{0}'.format(belts[i][nBagsC]) + delimiter)
file.write('{0}' .format(partialCount[partCountManualHandledBags]) +
delimiter)
file.write('{0}' .format(partialCount[partCountEmptiedPlannedReleaseC]) +
delimiter)
file.write('{0}' .format(partialCount[partCountEmptiedPanicC]) +
delimiter)
file.write('{0}' .format(partialCount[partCountStressC]) + delimiter)
file.write('\n')


def statistics():
delimiter = ';'
```

```
file = open('statistics.csv', 'a')
file.write('{0}' .format(strategy) )
file.write('{0}' .format(way) )
file.write('{0}' .format(cap) )
file.write('{0}' .format(arrival) + delimiter)
file.write('{0}'.format(len(manualhandling))+ delimiter)
file.write('\n')
file.close


def writeslabels(way):
if (way == 1): #there are 9 belts with a label t, then 4 belts with 2
hours interval and then the rest
value = 0
t = 15*60 #empty every 15 min
finish = horizon
while (value <=  finish):
for i in range (0, 8):
value = value + t
belts[i][labelsC].append(value)
value = belts[7][labelsC][0]
while (value <= finish):
for i in range (8, 12):
value = value + 2*60*60
belts[i][labelsC].append(value)
belts[12][labelsC].append(horizon*3)
belts[13][labelsC].append(horizon*3+1)
belts[12][restC] = 1
belts[13][restC] = 1

elif(way == 2):
value = 0
t = 15*60 #empty every 15 min
finish = horizon
while (value <  finish):
for i in range (0, 8):
value = value + t
belts[i][labelsC].append(value)
value = belts[7][labelsC][0]
while (value < finish):
for i in range (8, 12):
value = value + 6*t
belts[i][labelsC].append(value)
belts[12][labelsC].append(horizon*3)
belts[13][labelsC].append(horizon*3+1)
belts[12][restC] = 1
belts[13][restC] = 1

elif (way == 3):
```

```
value = 0
t = 30*60
finish = horizon
while (value <  finish):
for i in range (0, 12):
value = value + t
belts[i][labelsC].append(value)
value = belts[11][labelsC][0]
belts[12][labelsC].append(horizon*3)
belts[13][labelsC].append(horizon*3+1)
belts[12][restC] = 1
belts[13][restC] = 1

for i in range(nbelts):
belts[i][labelsC].append(horizon*4)

def updatereleasetime( belttoupdate):
belts[belttoupdate][releaseTimeC] = belts[belttoupdate][labelsC][0]
belts[belttoupdate][labelsC].pop(0)

# SIMULATION LOOP FUNCTION
#-------------------------------------------------------------------------
--

def Simulate(horizon):
for iteration in range(horizon):

# IF IT IS BELT RELEASE TIME
for j in range(nbelts):
if (iteration == belts[j][releaseTimeC]):
beltToEmpty = j
emptyAllBelt(beltToEmpty, iteration)
belts[j][countEmptiedPlannedReleaseC] =
belts[j][countEmptiedPlannedReleaseC] + 1
partialCount[partCountEmptiedPlannedReleaseC] =
partialCount[partCountEmptiedPlannedReleaseC] + 1
belttoupdate = j
if (len(belts[belttoupdate][labelsC]) != 0):
updatereleasetime(belttoupdate)

# IF IT IS BELT PANIC TIME
if ( iteration == belts[j][panicTimeC] ):
beltToEmpty = j
#empty all belt
emptyAllBelt(beltToEmpty, iteration)
partialCount[partCountEmptiedPanicC] =
partialCount[partCountEmptiedPanicC] + 1
belts[j][countEmptiedPanicC] = belts[j][countEmptiedPanicC] + 1
```

```
# IF A BAG ARRIVES (ALLOCATION STRATEGY)
for j in range(len(bags)):
if (iteration == bags[j][checkInTimeC]):
bagID = bags[j][bagIDC]
openingtime = bags[j][openingTimeC]
closingtime = bags[j][closingTimeC]
panictime = bags[j][panicTimeC]

#decide if the bag goes to EBS or not
EBS = goToEBS (openingtime, iteration)
if (EBS == 1): #bag goes to EBS
eligible (openingtime, sortingtime, panictime, way)

if (strategy == 1): # TOTALLY RANDOM
done = 0
beltnumbers = [0,1,2,3,4,5,6,7,8,9,10,11,12,13]
while (len(beltnumbers) != 0):
beltchosen = int (random.choice(beltnumbers))
beltnumbers.remove(beltchosen)
if ( ( belts[beltchosen][nBagsC] < belts[beltchosen][capacityC] ) ):
done = 1
addBag(bags, bagID, beltchosen, panictime)
break
if (done == 0): not space in any belt
manualhandling.append(bagID)
partialCount[partCountManualHandledBags] =
partialCount[partCountManualHandledBags] + 1

elif (strategy == 2): #RANDOM AMONG ELIGIBLE
done = 0
beltnumbers = [0,1,2,3,4,5,6,7,8,9,10,11,12,13]
while (len(beltnumbers) != 0):
beltchosen = int (random.choice(beltnumbers))
beltnumbers.remove(beltchosen)
if ( (belts[beltchosen][eligibleC] == 1) and ( belts[beltchosen][nBagsC]
< belts[beltchosen][capacityC] ) ):
done = 1
addBag(bags, bagID, beltchosen, panictime)
break
if (done == 0):
beltchosen = planB()
if beltchosen != -1:
done = 1
addBag(bags, bagID, beltchosen, panictime)
if (done == 0): #not space in any belt
manualhandling.append(bagID)
partialCount[partCountManualHandledBags] =
partialCount[partCountManualHandledBags] + 1
```

```
elif (strategy == 3): #EARLIEST BELT
beltchosen = earliestbelt()
if (beltchosen != -1):
addBag(bags, bagID, beltchosen, panictime)
if (beltchosen == -1):
beltchosen = planB()
if (beltchosen != -1):
addBag(bags, bagID, beltchosen, panictime)
if (beltchosen == -1):
manualhandling.append(bagID)
partialCount[partCountManualHandledBags] =
partialCount[partCountManualHandledBags] + 1

elif (strategy == 4): #LATEST BELT
beltchosen = latestbelt()
if (beltchosen != -1):
addBag(bags, bagID, beltchosen, panictime)
if (beltchosen == -1):
beltchosen = planB()
if beltchosen != -1:
addBag(bags, bagID, beltchosen, panictime)
if (beltchosen == -1):
manualhandling.append(bagID)
partialCount[partCountManualHandledBags] =
partialCount[partCountManualHandledBags]

elif (strategy == 5): #LEAST FILLED (LESS BAGS)
beltchosen = chooseBeltLeastFilledLessBags()
if (beltchosen != -1):
addBag( bags, bagID, beltchosen, panictime)
if (beltchosen == -1):
beltchosen = planB()
if beltchosen != -1:
addBag(bags, bagID, beltchosen, panictime)
if (beltchosen == -1):
manualhandling.append(bagID)
partialCount[partCountManualHandledBags] =
partialCount[partCountManualHandledBags] + 1

elif (strategy == 6): #MOST SPACE (BALANCE BELTS)
beltchosen = chooseBeltLeastFilledMoreFreeSpace()
if (beltchosen != -1):
addBag( bags, bagID, beltchosen, panictime)
if (beltchosen == -1):
beltchosen = planB()
if beltchosen != -1:
addBag(bags, bagID, beltchosen, panictime)
if (beltchosen == -1):
manualhandling.append(bagID)
```

```
partialCount[partCountManualHandledBags] =
partialCount[partCountManualHandledBags] + 1

elif (strategy == 7): #CURRENT: ASSIGNS EVERY 15 MIN. WHEN 45 MIN BEFORE
CLOSING TIME, A NEW BELT COMES WITH THE SAME RELEASE TIME THAN THE FIRST
15 MIN BELT
beltchosen = chooseBeltCurrent(closingtime)
if (beltchosen != -1):
addBag( bags, bagID, beltchosen, panictime)
if (beltchosen == -1):
beltchosen = planB()
if beltchosen != -1:
addBag(bags, bagID, beltchosen, panictime)
if (beltchosen == -1):
manualhandling.append(bagID)
#print('Manually handled\n')
partialCount[partCountManualHandledBags] =
partialCount[partCountManualHandledBags] + 1

elif (strategy == 8): #CURRENT: ASSIGNS EVERY 15 MIN. WHEN 45 MIN BEFORE
CLOSING TIME, A NEW BELT COMES WITH THE SAME RELEASE TIME THAN THE FIRST
15 MIN BELT
beltchosen = chooseHybrid(closingtime)
#print('Beltchosen {0}\n' .format(beltchosen))
if (beltchosen != -1):
addBag( bags, bagID, beltchosen, panictime)
if (beltchosen == -1):
beltchosen = planB()
if beltchosen != -1:
addBag(bags, bagID, beltchosen, panictime)
if (beltchosen == -1):
manualhandling.append(bagID)
partialCount[partCountManualHandledBags] =
partialCount[partCountManualHandledBags] + 1

if (iteration == 0 or iteration % 300 == 0):        #every 5 min
(5*60=300 iterations) it gets into the loop and shows the status)
writeUtilization(file_status, iteration)
print('Iteration {0}
({1})\n'.format(iteration,round(100*iteration/horizon ,2)))
#reset to 0 the values for partial count
partialCount[partCountManualHandledBags] = 0
partialCount[partCountEmptiedPlannedReleaseC] = 0
partialCount[partCountEmptiedPanicC] = 0
partialCount[partCountStressC] = 0


#################################################################
##### MAIN ######
```

```
############################################################



# CHOICE OF THE PARAMETERS USED IN THE SIMULATION
#-------------------------------------------------------------------------
# This is just to make it easier to run. It will run with a combination
of parameters that will be included from a shell script

if(len(sys.argv) == 6): #4 INPUT ARGUMENTS

strategy = int(sys.argv[1])

way = int(sys.argv[2])

cap = sys.argv[3]
if(cap == '1'):
capacitybelts = [73, 116, 86, 95, 73, 113, 86, 95, 113, 116, 113, 120,
120, 124] #alternate: alternatively higher and lower values (with highest
to the "rest belts")
elif(cap == '2'):
capacitybelts  = [73, 73, 86, 86, 95, 95, 95, 95, 113, 113, 116, 120,
120, 124] #from lower to higher
elif (cap == '3'):
capacitybelts  = [73, 86, 95, 95, 120, 124, 116, 113, 73, 86, 95, 95,
120, 113] #(original values) (just tused as a reference, to see if the
capacity has any impact)

arrival = sys.argv[4]
if(arrival == '1'):
fileNameBags = 'bags_normal.csv'
elif(arrival == '2'):
fileNameBags = 'bags_high.csv'
elif (arrival == '3'):
fileNameBags = 'bags_low.csv'

fileName = sys.argv[5]

else:
print('Select manually the combination of parameters for the
simulation:\n')
print('\nSelect the EBS allocation strategy:\n1-TOTALLY RANDOM\n2-RANDOM
AMONG ELIGIBLE\n3-EARLIEST POSSIBLE\n4-LATEST POSSIBLE\n5-LEAST FILLED
(LESS BAGS)\n6-LEAST FILLED (MORE FREE SPACE)\n7-CURRENT')
strategy = int(input())

print('\nSelect the release time strategy:\n1-way 1\n2-way 2\n3-way 3')
way = int(input())
```

```
print('\nSelect the capacity of the belts:\n1-ALTERNATE\n2-LOWER TO
HIGHER\n3-ORIGINAL VALUES (AS A REF)')
cap = int(input())
if cap ==1:
capacitybelts = [73, 116, 86, 95, 73, 113, 86, 95, 113, 116, 113, 120,
120, 124] #alternate: alternatively higher and lower values (with highest
to the "rest belts")
if cap == 2:
capacitybelts  = [73, 73, 86, 86, 95, 95, 95, 95, 113, 113, 116, 120,
120, 124] #from lower to higher
if cap == 3:
capacitybelts  = [73, 86, 95, 95, 120, 124, 116, 113, 73, 86, 95, 95,
120, 113] #(original values) (just tused as a reference, to see if the
capacity has any impact)

print('\nSelect the bag arrival:\n1-NORMAL\n2-HIGH\n3-LOW')
arrival = int(input())
if arrival == 1:
fileNameBags = 'bags_normal.csv'
if arrival == 2:
fileNameBags = 'bags_high.csv'
if arrival == 3:
fileNameBags = 'bags_low.csv'

fileName = 'utilization.csv'
fileName2 = 'stress.csv'

# READ BAGS AND CHUTES INFO FROM ITS FILES
#-------------------------------------------------------------------------
--
dataBags = getData(fileNameBags)
bags = splitData(dataBags)



# INITIALIZE VALUES
#-------------------------------------------------------------------------
horizon = 24*60*60
nbelts = 14

emptyingtime =  5*60 #5 min to empty the belt (1 min of emptying belt + 4
min of sorting time)
sortingtime = 4*60 #4 min to sort the bag to its corresponding location

belts = []
manualhandling = []
partialCount = [0,0,0,0] #contains the partial count of manual handled
bags, emptied by release time, empty by panic time, stress (number of
```

```
bags that are being released). Will be changed every 5 min (for
utilization file)

for i in range(nbelts):
belts.append([0, capacitybelts[i], horizon*3, [], [], horizon*3, 0, 0,
[], 0, 0]) #nbags, capacity, releasetime, bagID, bagIDPanicTime,
panictime, eligible, emptied times, labels, countPanic, rest

# ADD THE PANIC TIME TO THE BAG
#-------------------------------------------------------------------------
for i in range(len(bags)):
bagpanictime =  bags[i][closingTimeC]-emptyingtime
bags[i].append(bagpanictime)


# WRITES THE RELEASE TIME LABELS (and if the bag is rest or not)
#-------------------------------------------------------------------------
--
writeslabels(way)

# ADD THE FIRST RELEASE TIME FOR THE BELTS
# -------------------------------------------------------------------------
for i in range(nbelts):
belttoupdate = i
updatereleasetime(belttoupdate)

deleteFile(fileName)

### Open file ###

file_status = open(fileName,'a')

Simulate(horizon)

### Close file ###

file_status.close

# FINAL STATISTICS
#-------------------------------------------------------------------------

statistics()
print('Finished')
```

9.8 Simulation results for Early Baggage Storage allocation

The ease the understanding of the results, each combination of strategies will be given a number composed by 4 digits, that will represent the combination of:

- 1st digit: EBS allocation strategy
    1. Totally random
    2. Random among eligible
    3. Earliest possible belt
    4. Latest possible belt
    5. Least filled belt (less bags)
    6. Most space belt
    7. Current
- 2nd digit: Release labels definition
    1. "Way 1": seven frequently released belts with a frequency of 15 min, five not frequently released belts and two rest belts.
    2. "Way 2": it is a variation of way 1 in which the not frequently emptied belts will be released with a frequency of 1,5 hours.
    3. "Way 3": it uses only two types of belts: rest belts and frequently emptied belts, with a release frequency of 15 minutes.
- 3rd digit: Matching of capacities to release label definition
    1. "Capacity 1": alternate low and high values of capacities of the belts with the largest capacities to the rest belts
    2. "Capacity 2": increasing capacities of the belts (from lower to higher) with the largest capacities to the rest belts
    3. "Capacity 3": random capacities for the belts with the largest capacities to the rest belts
- 4th digit: Bag arrival dataset
    1. "Normal": normal day of arrival of bags, represented by the day 31-7-2018.
    2. "High": high arrival rate of bags, represented by the day 15-7-2018.
    3. "Low": low intensity of arrival of bags, represented by the day 11-3-2018.

For example, the combination 1233 will show the combination of the totally random strategy, "Way 2" of release labels definition, "Capacity 3" and "Low" bag arrival dataset.

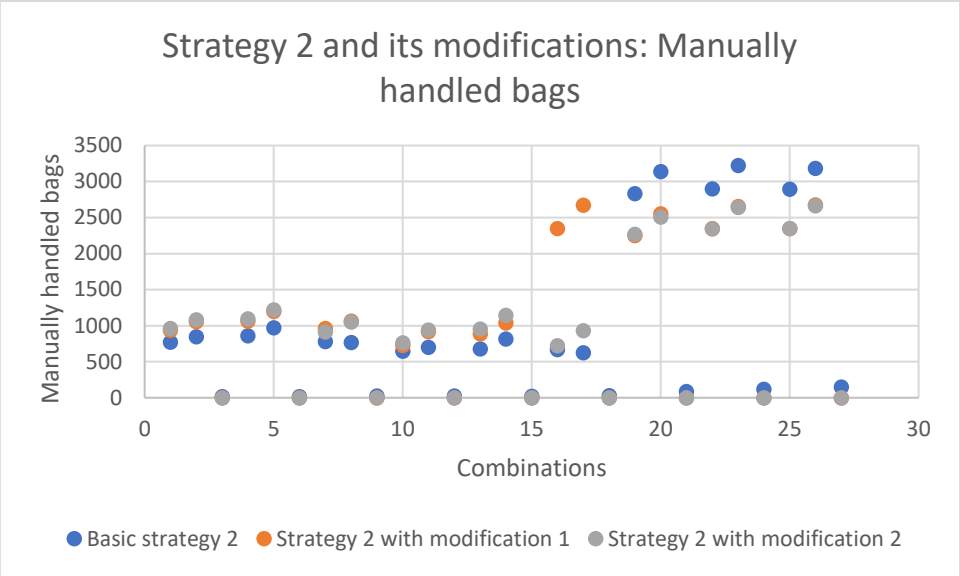
9.8.1 Manually handled bags on EBS strategies and its modifications

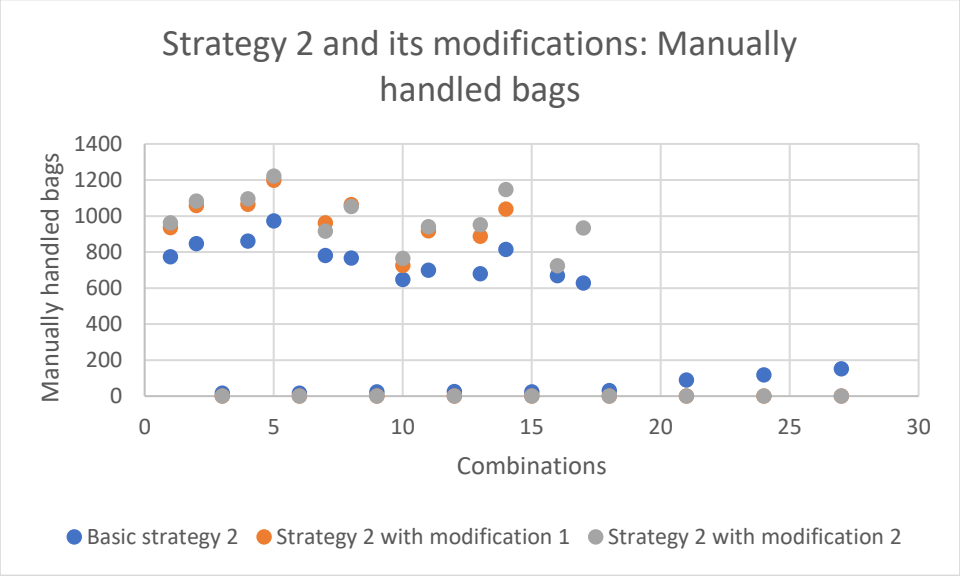
9.8.1.1 Strategy 1- Totally random

Combination	Basic strategy
1111	1476
1112	1617
1113	0
1121	1609
1122	1704
1123	0
1131	1354
1132	1512
1133	0
1211	1335
1212	1512
1213	0
1221	1432
1222	1603
1223	0
1231	1267
1232	1445
1233	0
1311	2556
1312	2894
1313	0
1321	2611
1322	3015
1323	0
1331	2649
1332	3037
1333	0

9.8.1.2 Strategy 2- Random among eligible

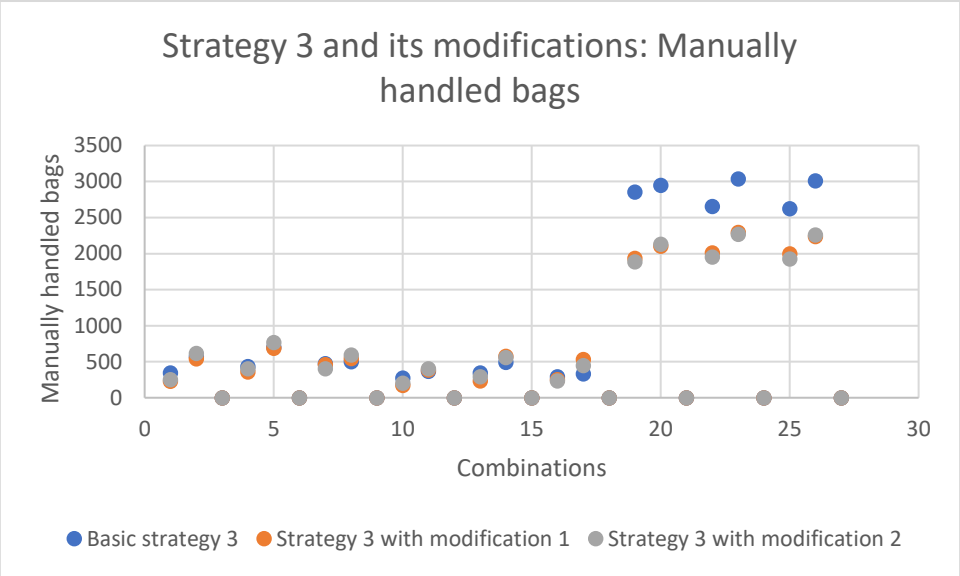
Combination	Basic strategy	Modification 1: using last belt as a second strategy	Modification 2: using most space belt as a second strategy
2111	774	936	961
2112	846	1057	1083
2113	17	0	0
2121	861	1065	1096
2122	973	1198	1221
2123	17	0	0
2131	780	962	916
2132	767	1064	1052
2133	24	0	0
2211	648	726	765
2212	699	917	940
2213	26	0	0
2221	680	887	952
2222	815	1038	1146
2223	23	0	0
2231	669	2347	724
2232	627	2675	934
2233	30	0	0
2311	2834	2253	2268
2312	3139	2552	2510
2313	90	0	0
2321	2901	2350	2344
2322	3225	2657	2644
2323	117	0	0
2331	2896	2347	2349
2332	3186	2677	2665
2333	152	0	0

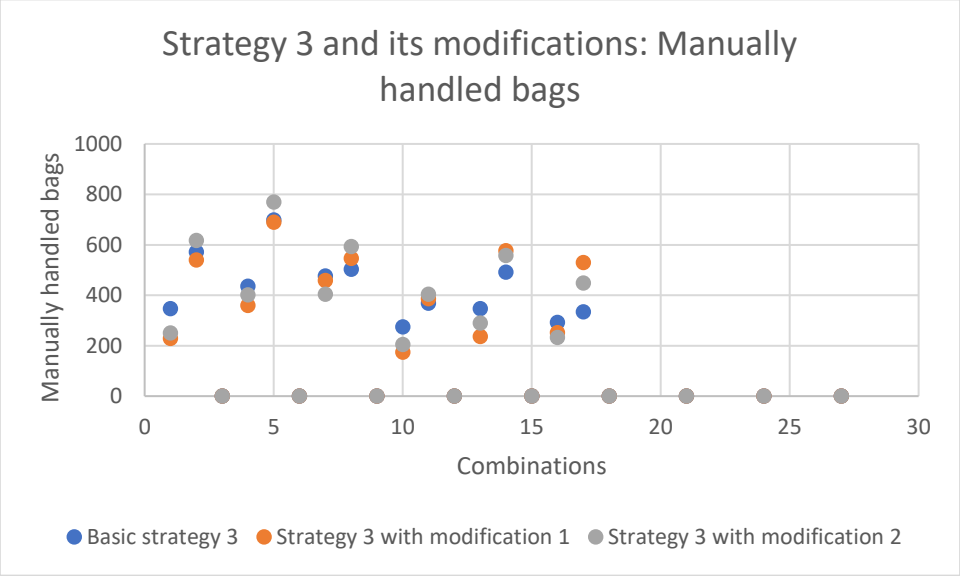

Strategy 2 and its modifications: Manually handled bags
Manually handled bags
3500
3000
2500
2000
1500
1000
500
0
0
5
10
15
20
25
30
Combinations
Basic strategy 2
Strategy 2 with modification 1
Strategy 2 with modification 2


Strategy 2 and its modifications: Manually handled bags
Manually handled bags
1400
1200
1000
800
600
400
200
0
0
5
10
15
20
25
30
Combinations
Basic strategy 2
Strategy 2 with modification 1
Strategy 2 with modification 2

9.8.1.3 Strategy 3- Earliest possible belt

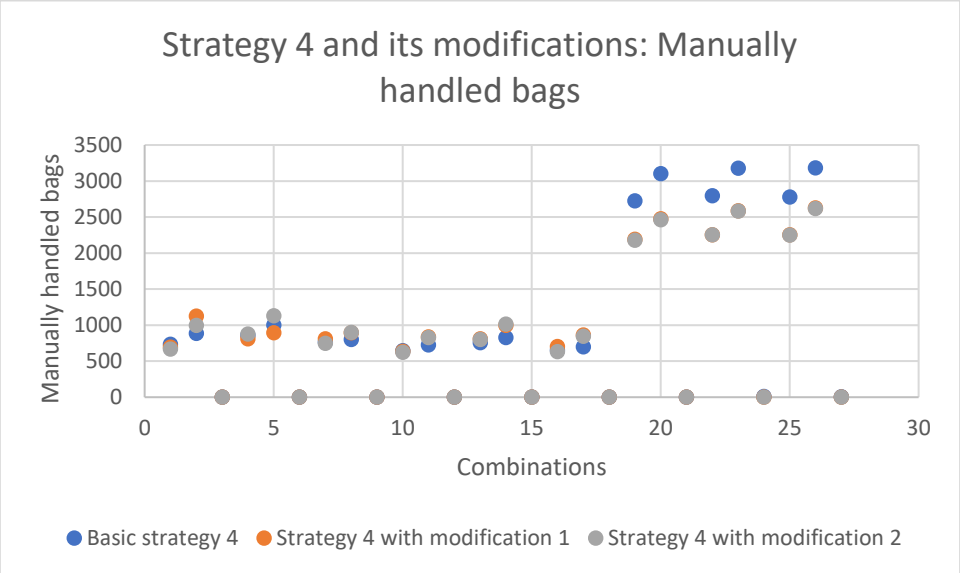
Combination	Basic strategy	Modification 1: using last belt as a second strategy	Modification 2: using most space belt as a second strategy
3111	347	229	250
3112	571	540	617
3113	0	0	0
3121	436	360	402
3122	699	689	769
3123	0	0	0
3131	476	459	404
3132	503	546	593
3133	0	0	0
3211	275	174	205
3212	369	386	404
3213	0	0	0
3221	347	236	290
3222	492	577	558
3223	0	0	0
3231	292	252	233
3232	334	530	448
3233	0	0	0
3311	2857	1936	1888
3312	2949	2103	2132
3313	0	0	0
3321	2655	2011	1952
3322	3038	2295	2267
3323	0	0	0
3331	2623	1998	1925
3332	3009	2236	2258
3333	0	0	0


Strategy 3 and its modifications: Manually handled bags
Manually handled bags
3500
3000
2500
2000
1500
1000
500
0
0
5
10
15
20
25
30
Combinations
Basic strategy 3
Strategy 3 with modification 1
Strategy 3 with modification 2


Strategy 3 and its modifications: Manually handled bags
Manually handled bags
1000
800
600
400
200
0
0
5
10
15
20
25
30
Combinations
Basic strategy 3
Strategy 3 with modification 1
Strategy 3 with modification 2

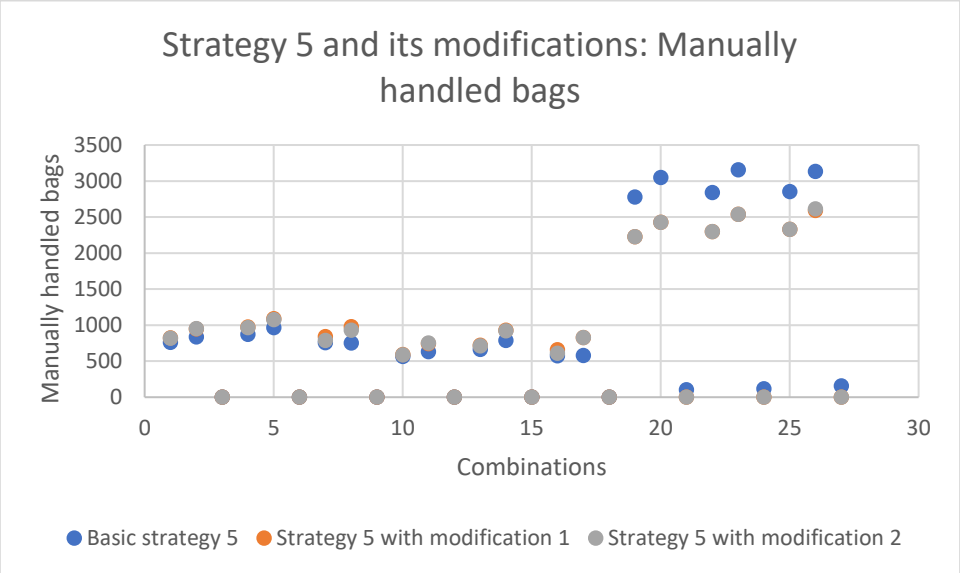
9.8.1.4 Strategy 4- Latest possible belt

Combination	Basic strategy	Modification 1: using last belt as a second strategy	Modification 2: using most space belt as a second strategy
4111	731	692	668
4112	882	1125	996
4113	0	0	0
4121	858	807	877
4122	1000	892	1128
4123	0	0	0
4131	764	807	746
4132	799	892	895
4133	0	0	0
4211	644	636	620
4212	725	833	826
4213	0	0	0
4221	755	808	800
4222	828	996	1012
4223	0	0	0
4231	652	700	629
4232	695	861	845
4233	0	0	0
4311	2725	2192	2175
4312	3100	2473	2463
4313	0	0	0
4321	2792	2254	2254
4322	3178	2584	2583
4323	7	0	0
4331	2777	2254	2249
4332	3181	2624	2617
4333	5	0	0



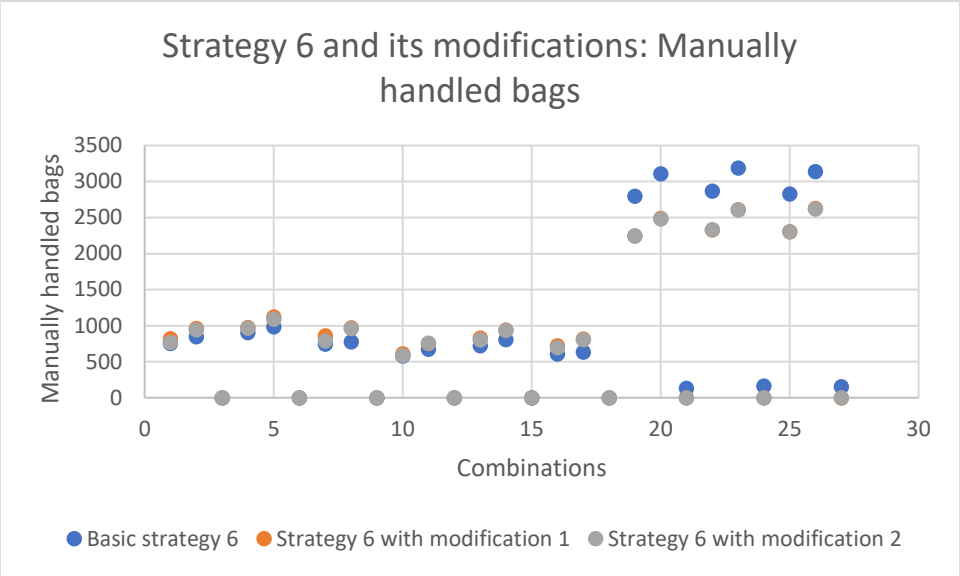
9.8.1.5 Strategy 5- Least filled belt (less bags)

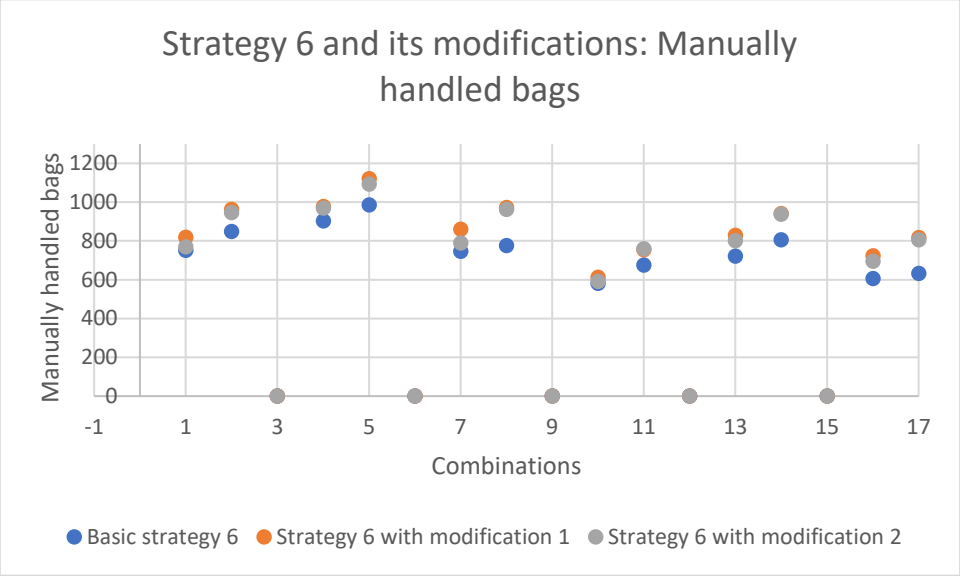
Combination	Basic strategy	Modification 1: using last belt as a second strategy	Modification 2: using most space belt as a second strategy
5111	759	823	816
5112	834	946	951
5113	0	0	0
5121	872	971	964
5122	963	1087	1075
5123	0	0	0
5131	757	841	787
5132	752	976	927
5133	0	0	0
5211	562	589	584
5212	631	743	750
5213	0	0	0
5221	660	719	710
5222	787	926	920
5223	0	0	0
5231	571	656	608
5232	577	828	825
5233	0	0	0
5311	2776	2227	2226
5312	3048	2425	2424
5313	103	0	0
5321	2838	2298	2298
5322	3154	2538	2535
5323	114	0	0
5331	2851	2327	2327
5332	3130	2589	2611
5333	154	0	0



9.8.1.6 Strategy 6- Most space belt

Combination	Basic strategy	Modification 1: using last belt as a second strategy	Modification 2: using most space belt as a second strategy
6111	752	819	769
6112	849	962	946
6113	0	0	0
6121	903	977	969
6122	986	1121	1093
6123	0	0	0
6131	747	860	790
6132	776	972	963
6133	0	0	0
6211	582	613	592
6212	676	754	759
6213	0	0	0
6221	722	829	801
6222	806	941	937
6223	0	0	0
6231	606	723	695
6232	633	817	806
6233	0	0	0
6311	2796	2248	2248
6312	3107	2491	2484
6313	132	0	0
6321	2868	2330	2330
6322	3189	2611	2606
6323	164	0	0
6331	2828	2303	2303
6332	3138	2629	2618
6333	154	0	0


Strategy 6 and its modifications: Manually handled bags
Manually handled bags
3500
3000
2500
2000
1500
1000
500
0
0
5
10
15
20
25
30
Combinations
Basic strategy 6
Strategy 6 with modification 1
Strategy 6 with modification 2

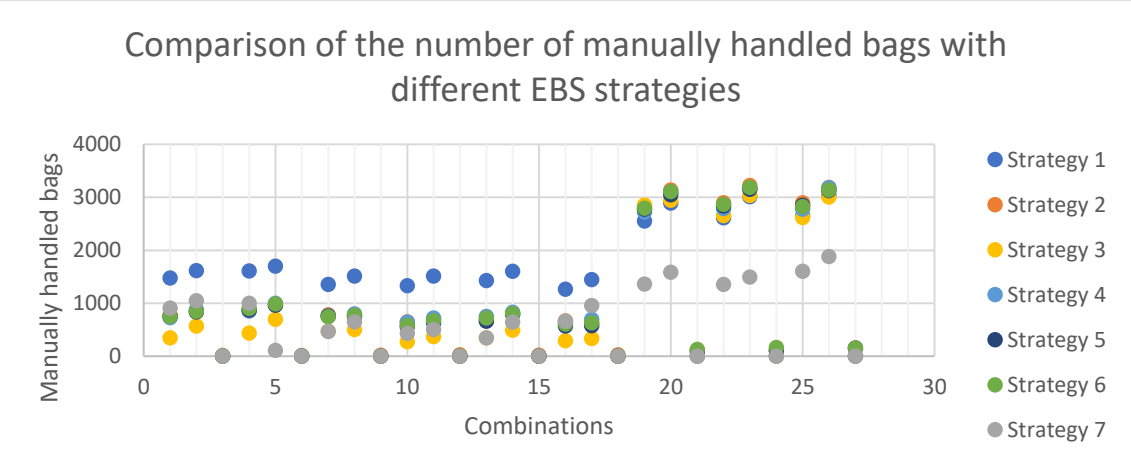

Strategy 6 and its modifications: Manually handled bags
Manually handled bags
1200
1000
800
600
400
200
0
-1
1
3
5
7
9
11
13
15
17
Combinations
Basic strategy 6
Strategy 6 with modification 1
Strategy 6 with modification 2

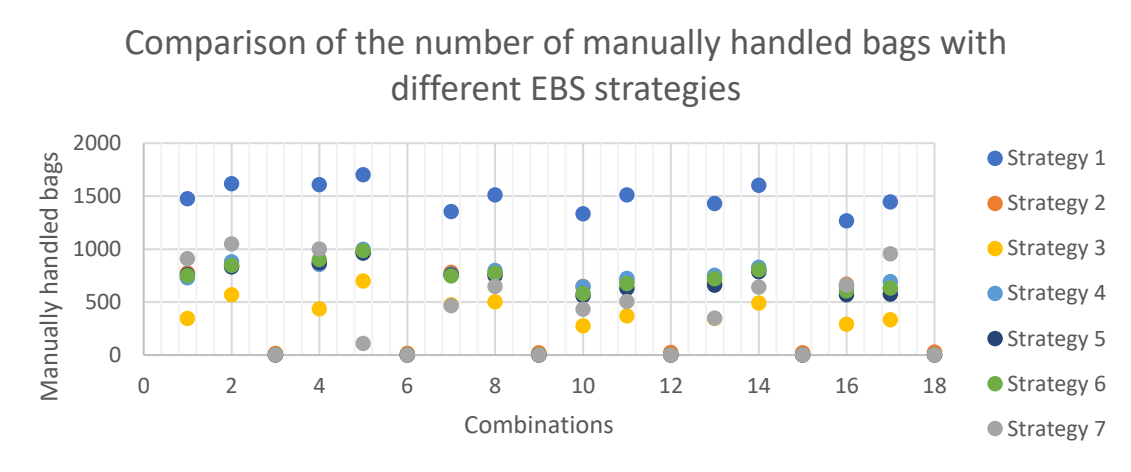
9.8.1.7 Strategy 7- Current CPH Airport strategy

Combination	Basic strategy
7111	910
7112	1050
7113	0
7121	1002
7122	109
7123	0
7131	466
7132	649
7133	0
7211	435
7212	505
7213	0
7221	350
7222	642
7223	0
7231	659
7232	957
7233	0
7311	1364
7312	1588
7313	0
7321	1359
7322	1495
7323	0
7331	1607
7332	1880
7333	0

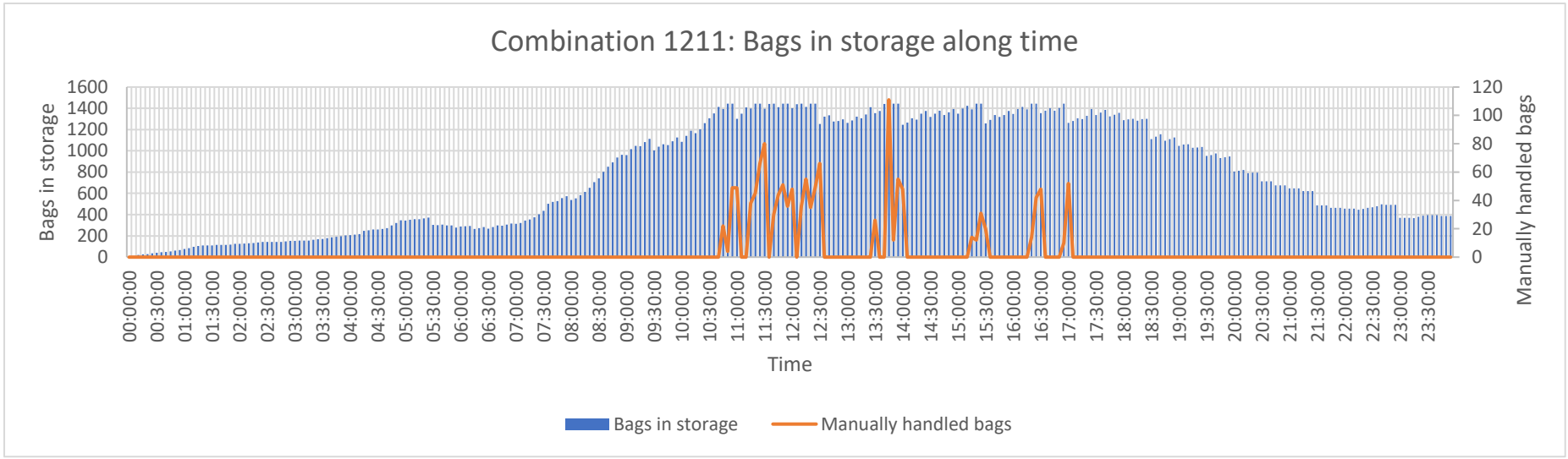
9.8.1.8 Comparison of the number of manually handled bags with different strategies

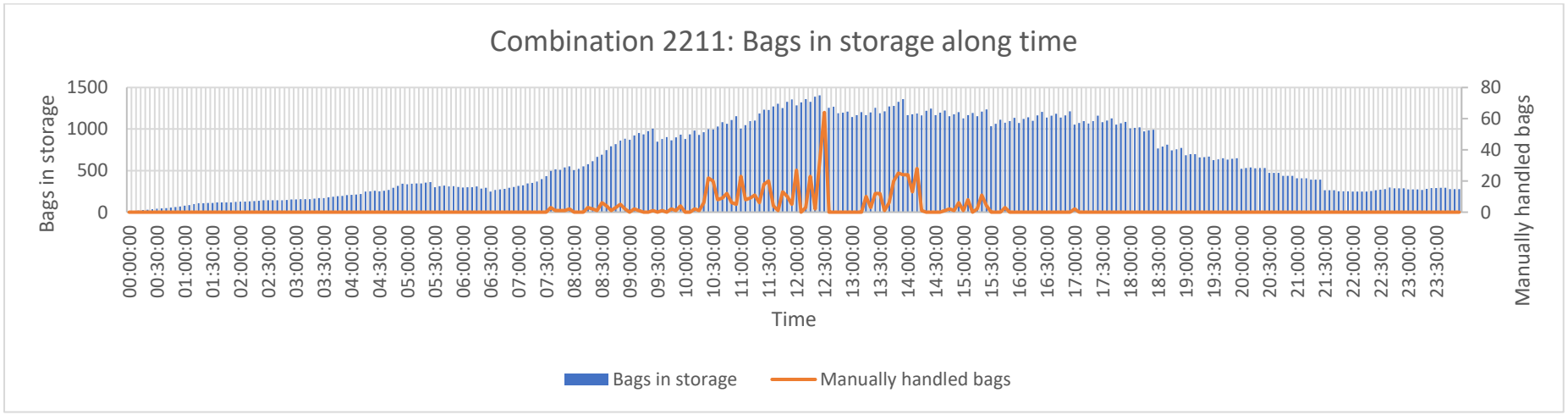
Comb. Nr in graph	Combination	Strategy						
		1	2	3	4	5	6	7
1	X111	1476	774	347	731	759	752	910
2	X112	1617	846	571	882	834	849	1050
3	X113	0	17	0	0	0	0	0
4	X121	1609	861	436	858	872	903	1002
5	X122	1704	973	699	1000	963	986	109
6	X123	0	17	0	0	0	0	0
7	X131	1354	780	476	764	757	747	466
8	X132	1512	767	503	799	752	776	649
9	X133	0	24	0	0	0	0	0
10	X211	1335	648	275	644	562	582	435
11	X212	1512	699	369	725	631	676	505
12	X213	0	26	0	0	0	0	0
13	X221	1432	680	347	755	660	722	350
14	X222	1603	815	492	828	787	806	642
15	X223	0	23	0	0	0	0	0
16	X231	1267	669	292	652	571	606	659
17	X232	1445	627	334	695	577	633	957
18	X233	0	30	0	0	0	0	0
19	X311	2556	2834	2857	2725	2776	2796	1364
20	X312	2894	3139	2949	3100	3048	3107	1588
21	X313	0	90	0	0	103	132	0
22	X321	2611	2901	2655	2792	2838	2868	1359
23	X322	3015	3225	3038	3178	3154	3189	1495
24	X323	0	117	0	7	114	164	0
25	X331	2649	2896	2623	2777	2851	2828	1607
26	X332	3037	3186	3009	3181	3130	3138	1880
27	X333	0	152	0	5	154	154	0

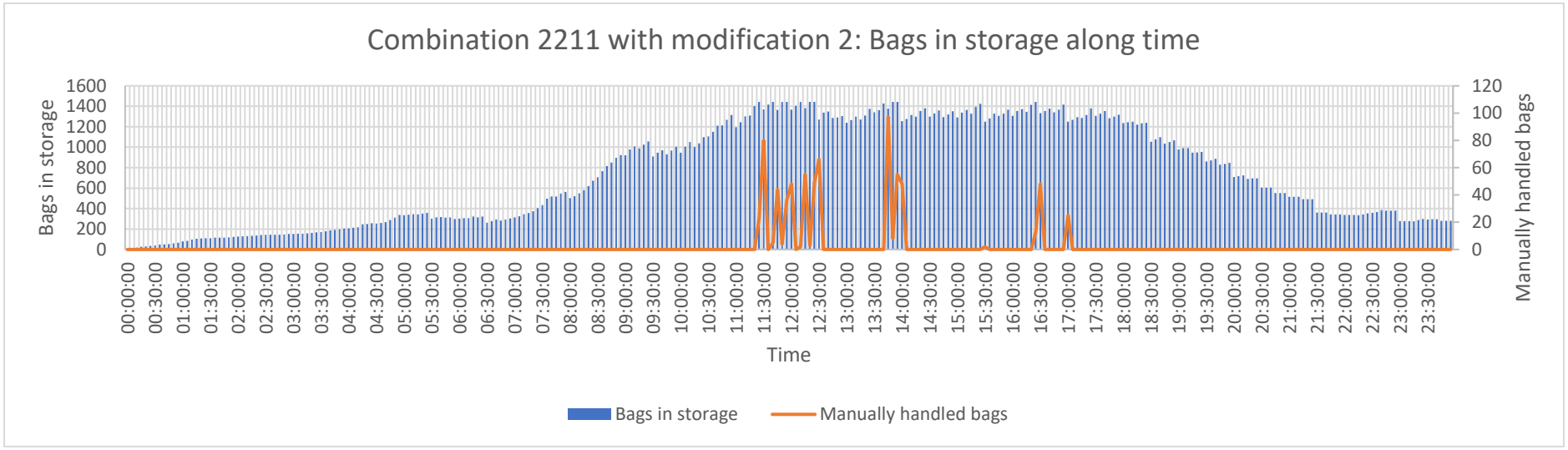


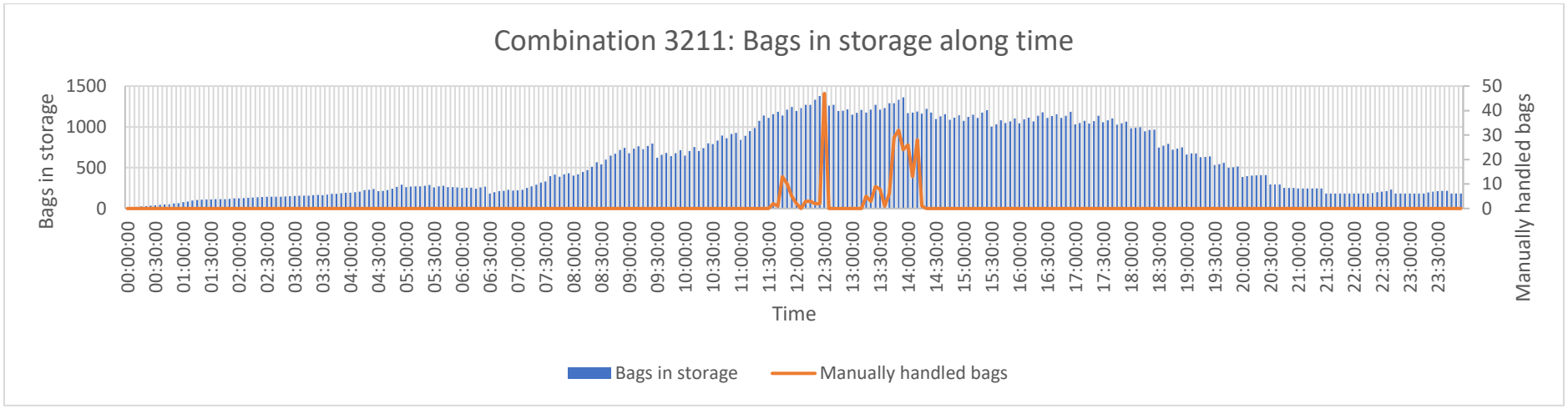

Comparison of the number of manually handled bags with different EBS strategies
Manually handled bags
2000
1500
1000
500
0
0
2
4
6
8
10
12
14
16
18
Combinations
Strategy 1
Strategy 2
Strategy 3
Strategy 4
Strategy 5
Strategy 6
Strategy 7

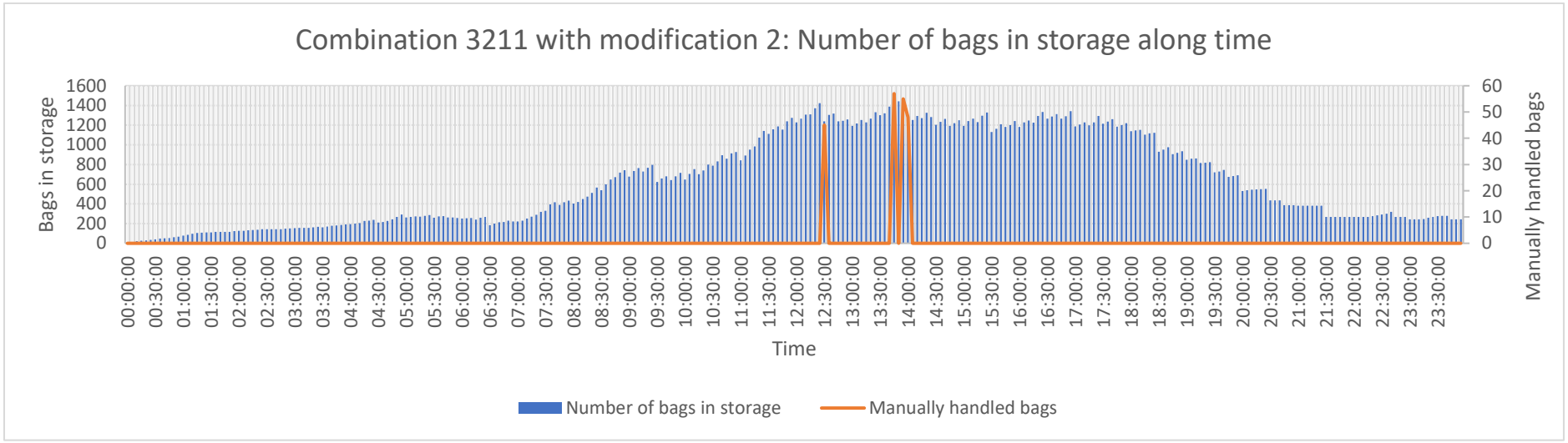
9.8.2 Bags on storage along time on EBS strategies

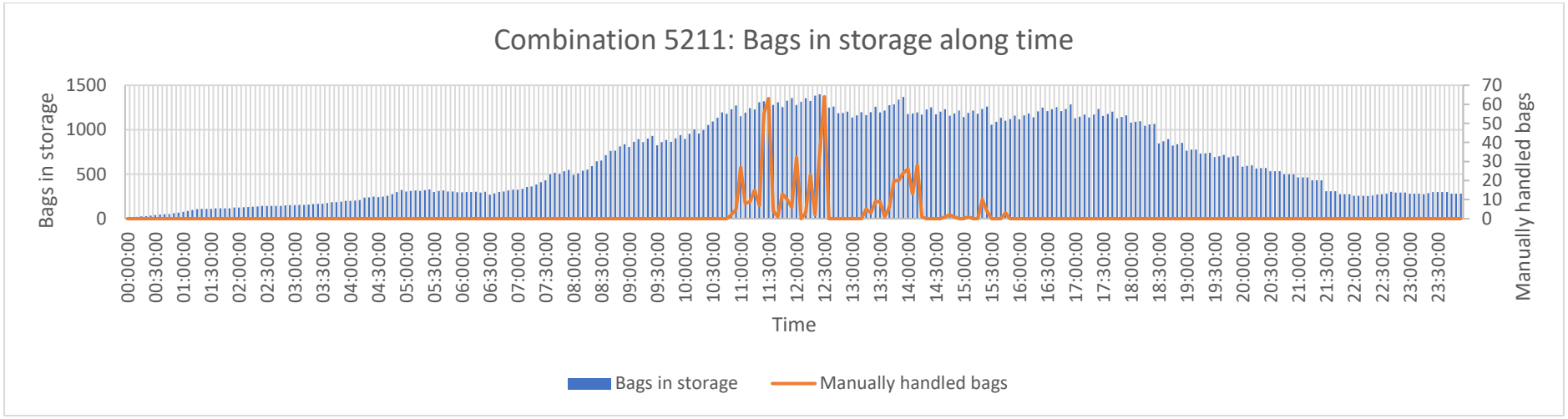


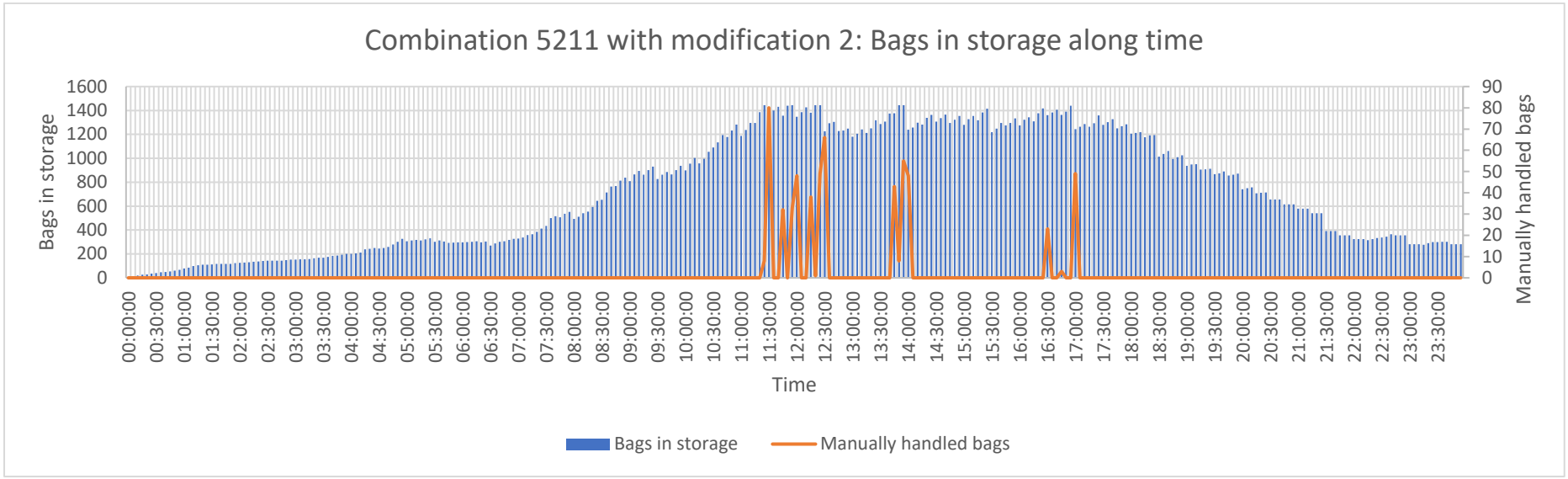


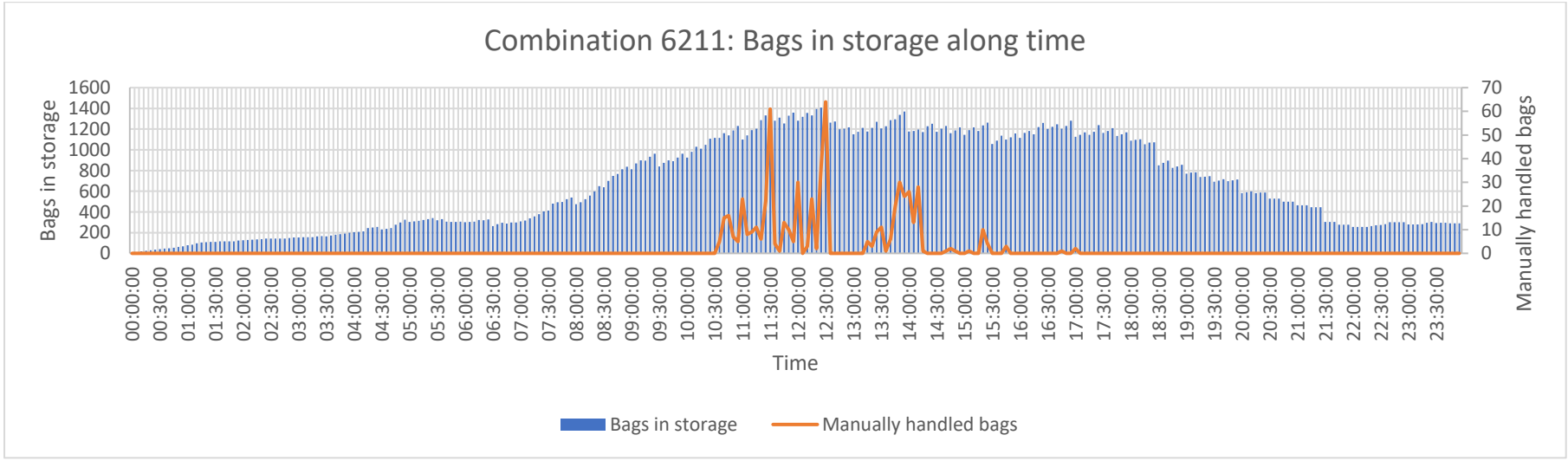

Combination 2211 with modification 2: Bags in storage along time
Bags in storage
Manually handled bags
Time
Bags in storage
Manually handled bags

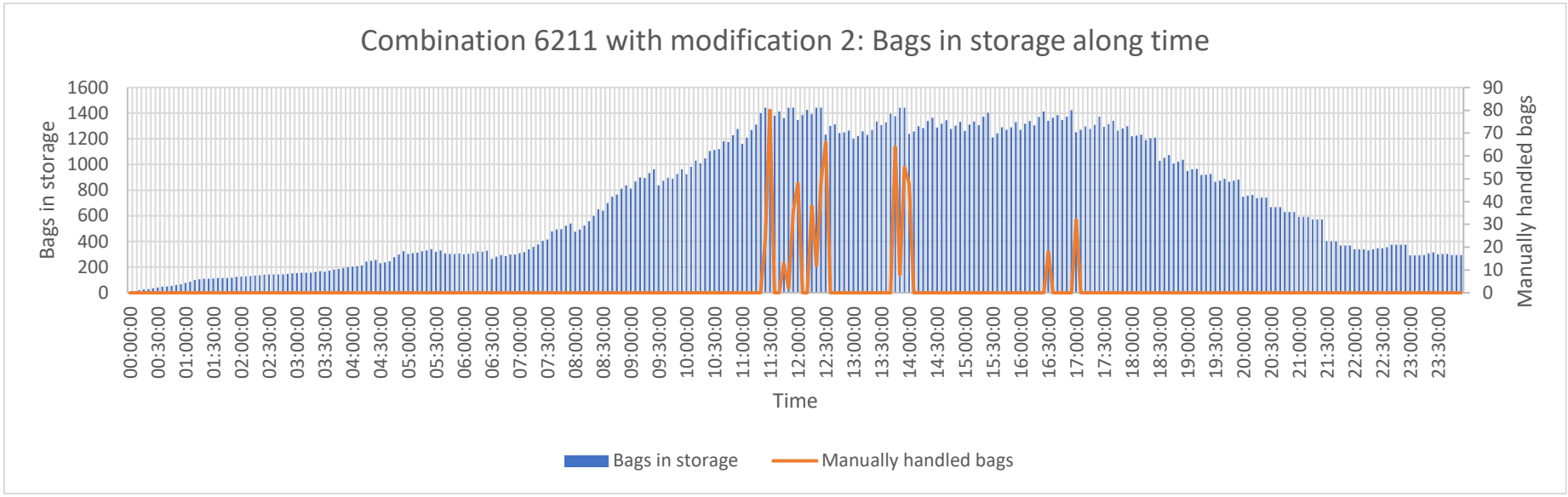

Combination 3211: Bags in storage along time
Bags in storage
Manually handled bags
Time
Bags in storage
Manually handled bags

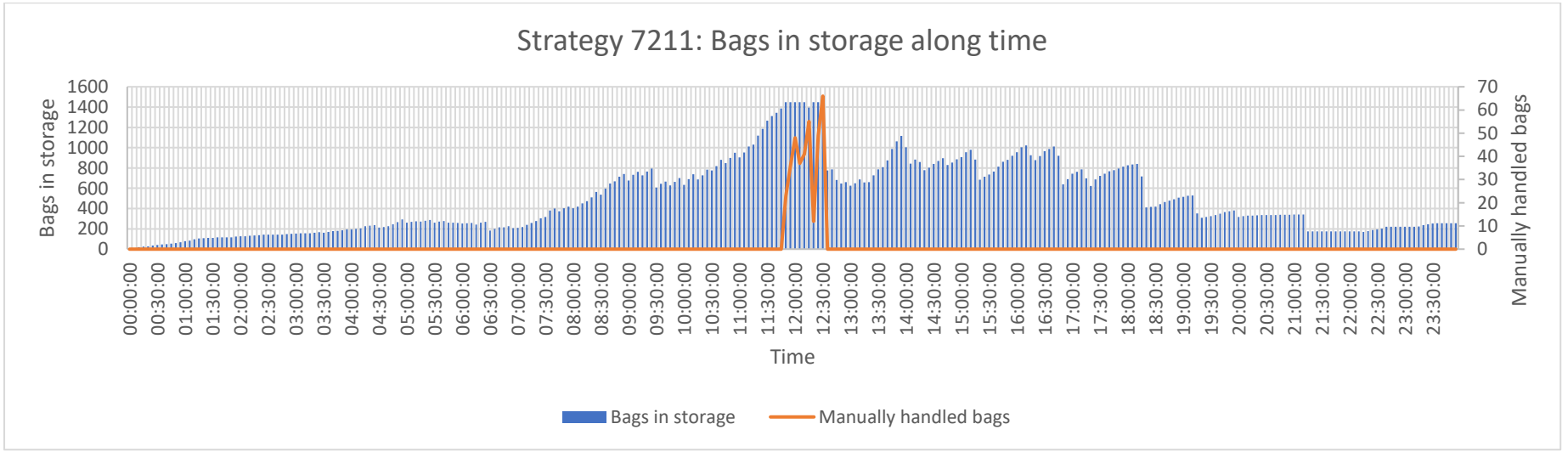

Combination 3211 with modification 2: Number of bags in storage along time
Bags in storage
Manually handled bags
Time
Number of bags in storage
Manually handled bags


Combination 5211: Bags in storage along time
Bags in storage
Manually handled bags
Time
Bags in storage
Manually handled bags

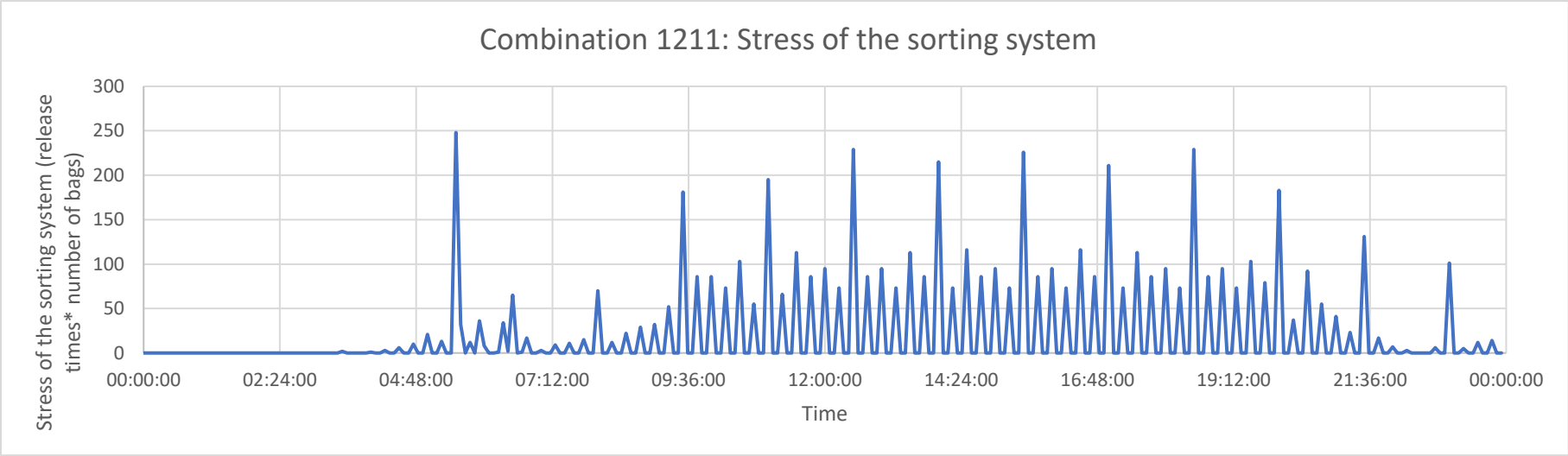

Combination 5211 with modification 2: Bags in storage along time
Bags in storage
Manually handled bags
Time
Bags in storage
Manually handled bags

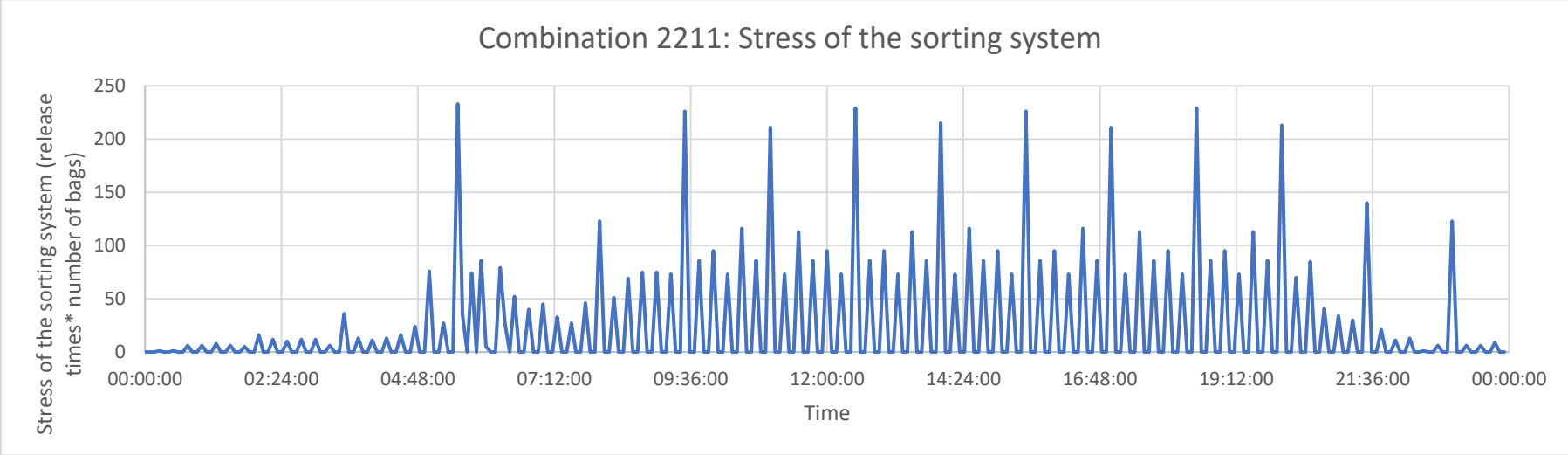

Combination 6211: Bags in storage along time
Bags in storage
Manually handled bags
Time
Bags in storage
Manually handled bags

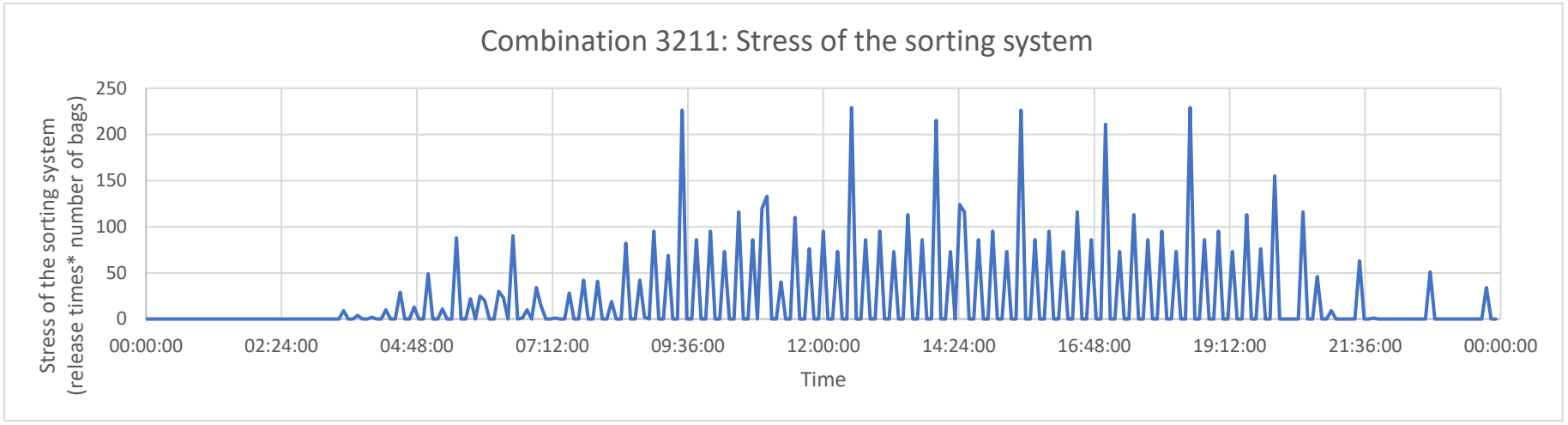

Combination 6211 with modification 2: Bags in storage along time
Bags in storage
Manually handled bags
Time
Bags in storage
Manually handled bags

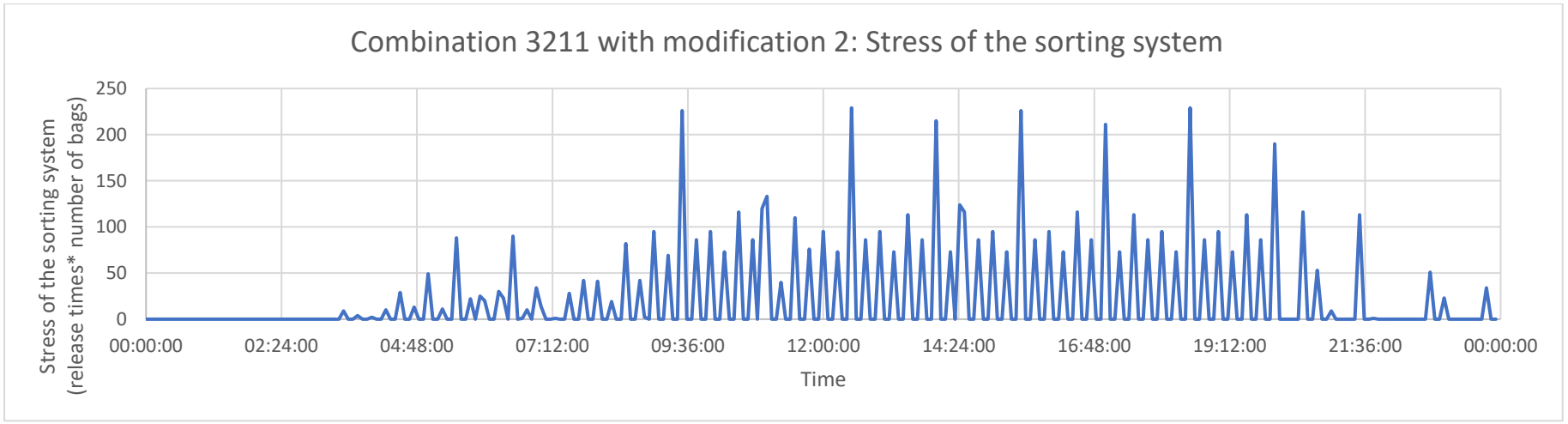

Strategy 7211: Bags in storage along time
Bags in storage
Manually handled bags
Time
Bags in storage
Manually handled bags

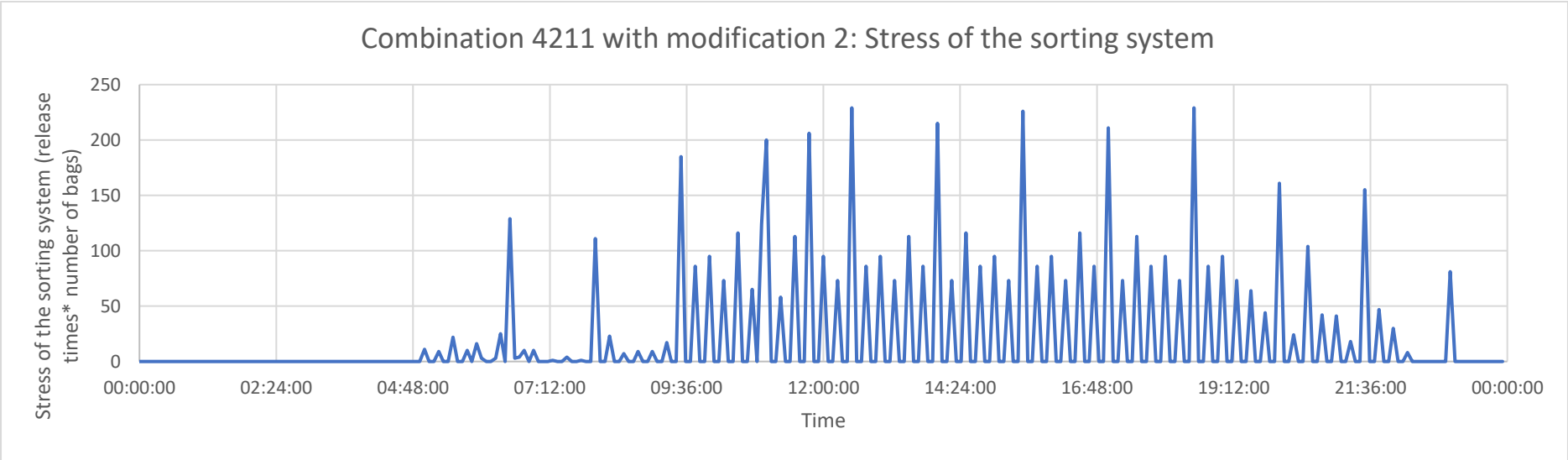
9.8.3 Stress of the system along time with the EBS strategies

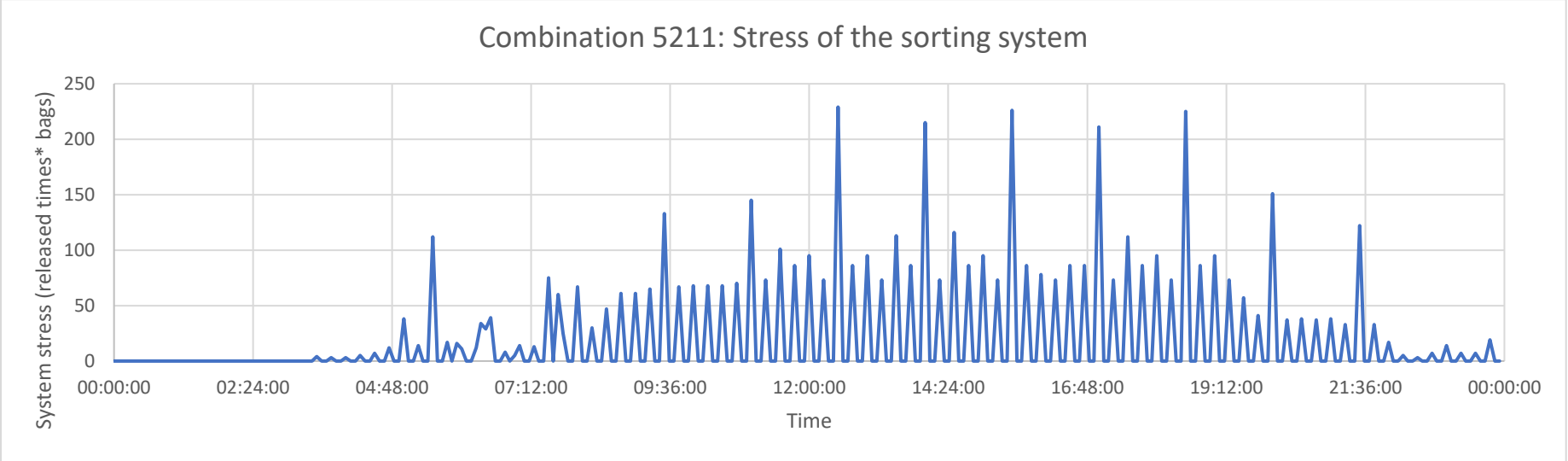


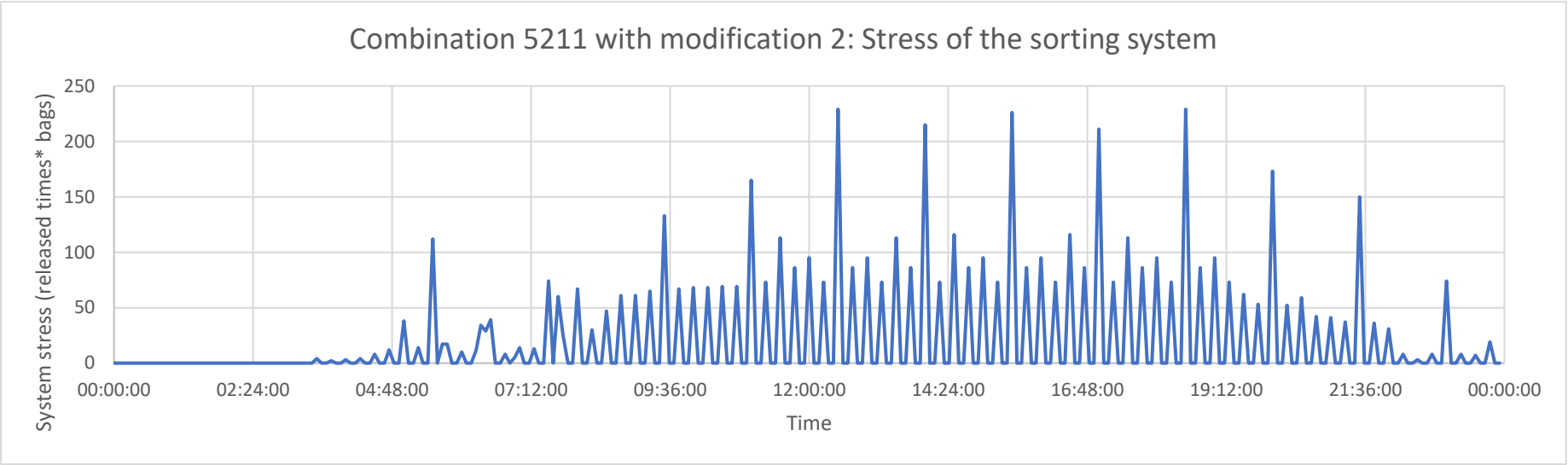


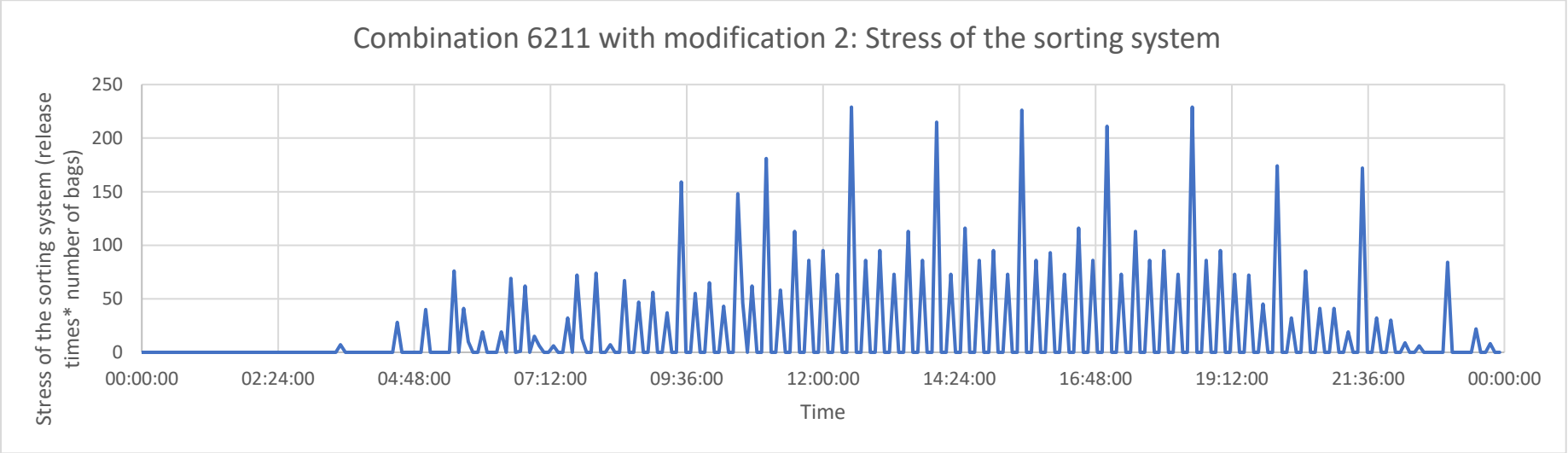

Combination 3211: Stress of the sorting system
Stress of the sorting system (release times* number of bags)
250
200
150
100
50
0
00:00:00
02:24:00
04:48:00
07:12:00
09:36:00
12:00:00
14:24:00
16:48:00
19:12:00
21:36:00
00:00:00
Time

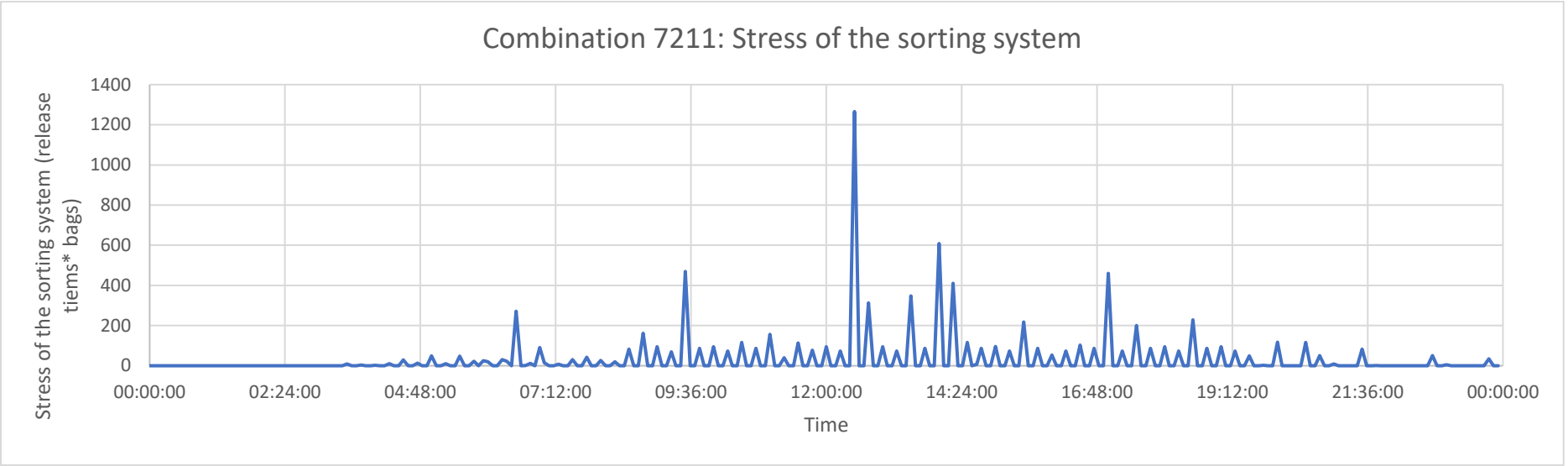

Combination 3211 with modification 2: Stress of the sorting system
Stress of the sorting system (release times* number of bags)
250
200
150
100
50
0
00:00:00
02:24:00
04:48:00
07:12:00
09:36:00
12:00:00
14:24:00
16:48:00
19:12:00
21:36:00
00:00:00
Time

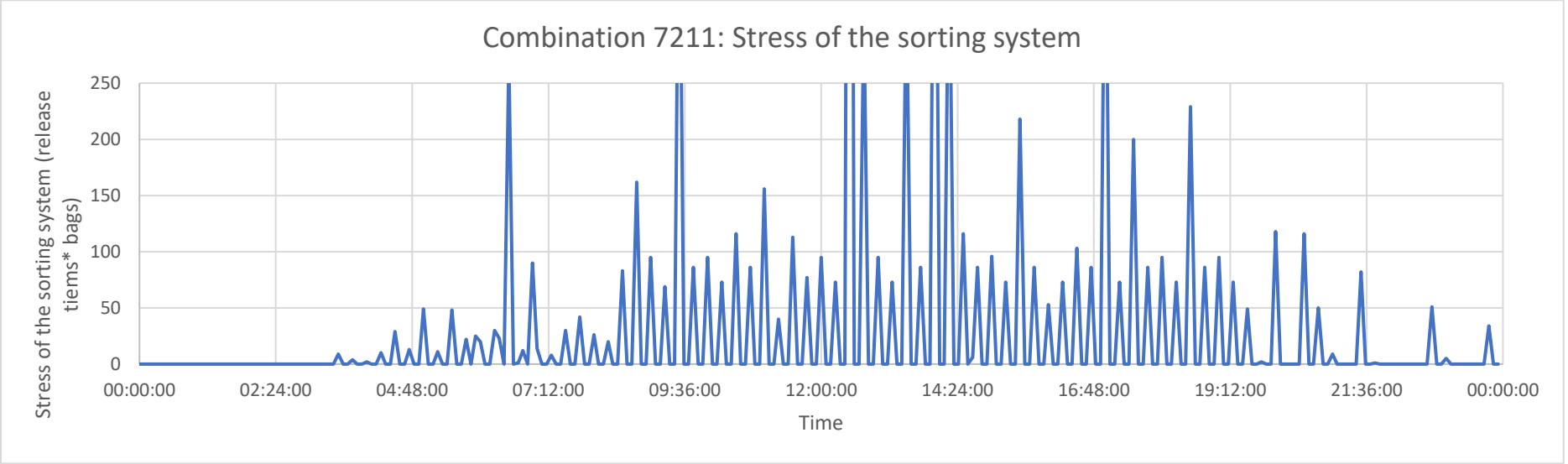

Combination 4211 with modification 2: Stress of the sorting system
Stress of the sorting system (release times* number of bags)
250
200
150
100
50
0
00:00:00
02:24:00
04:48:00
07:12:00
09:36:00
12:00:00
14:24:00
16:48:00
19:12:00
21:36:00
00:00:00
Time

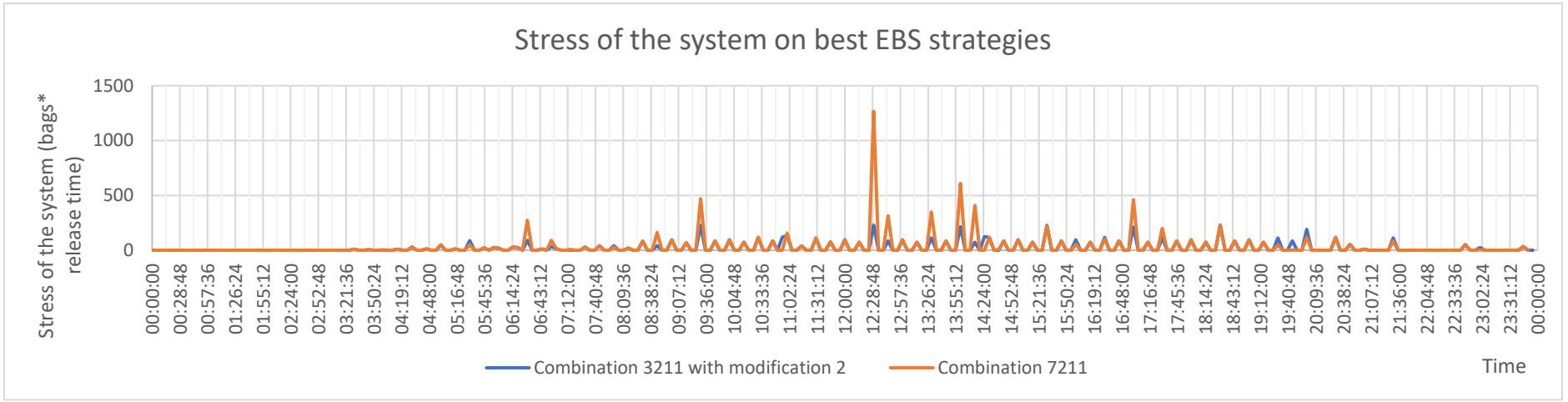

Combination 5211: Stress of the sorting system
System stress (released times* bags)
250
200
150
100
50
0
00:00:00
02:24:00
04:48:00
07:12:00
09:36:00
12:00:00
14:24:00
16:48:00
19:12:00
21:36:00
00:00:00
Time

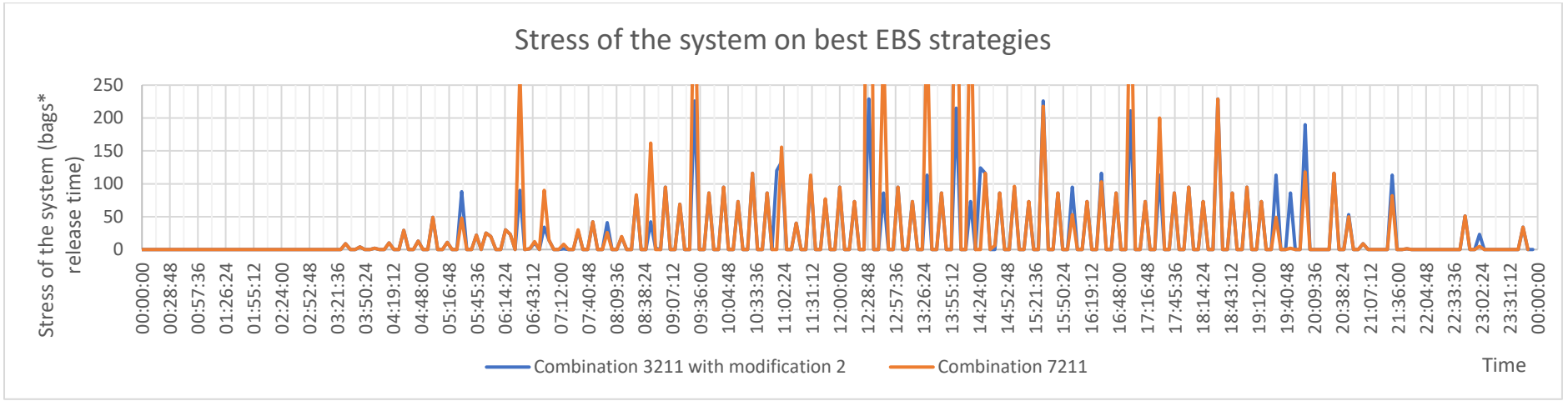

Combination 5211 with modification 2: Stress of the sorting system
System stress (released times* bags)
250
200
150
100
50
0
00:00:00
02:24:00
04:48:00
07:12:00
09:36:00
12:00:00
14:24:00
16:48:00
19:12:00
21:36:00
00:00:00
Time


Combination 6211 with modification 2: Stress of the sorting system
Stress of the sorting system (release times* number of bags)
250
200
150
100
50
0
00:00:00
02:24:00
04:48:00
07:12:00
09:36:00
12:00:00
14:24:00
16:48:00
19:12:00
21:36:00
00:00:00
Time


Combination 7211: Stress of the sorting system
Stress of the sorting system (release tiems* bags)
1400
1200
1000
800
600
400
200
0
00:00:00
02:24:00
04:48:00
07:12:00
09:36:00
12:00:00
14:24:00
16:48:00
19:12:00
21:36:00
00:00:00
Time


Combination 7211: Stress of the sorting system
Stress of the sorting system (release tiems* bags)
250
200
150
100
50
0
00:00:00
02:24:00
04:48:00
07:12:00
09:36:00
12:00:00
14:24:00
16:48:00
19:12:00
21:36:00
00:00:00
Time


Stress of the system on best EBS strategies
Stress of the system (bags* release time)
1500
1000
500
0
00:00:00
00:28:48
00:57:36
01:26:24
01:55:12
02:24:00
02:52:48
03:21:36
03:50:24
04:19:12
04:48:00
05:16:48
05:45:36
06:14:24
06:43:12
07:12:00
07:40:48
08:09:36
08:38:24
09:07:12
09:36:00
10:04:48
10:33:36
11:02:24
11:31:12
12:00:00
12:28:48
12:57:36
13:26:24
13:55:12
14:24:00
14:52:48
15:21:36
15:50:24
16:19:12
16:48:00
17:16:48
17:45:36
18:14:24
18:43:12
19:12:00
19:40:48
20:09:36
20:38:24
21:07:12
21:36:00
22:04:48
22:33:36
23:02:24
23:31:12
00:00:00
Combination 3211 with modification 2
Combination 7211
Time


Stress of the system on best EBS strategies
Stress of the system (bags* release time)
250
200
150
100
50
0
00:00:00
00:28:48
00:57:36
01:26:24
01:55:12
02:24:00
02:52:48
03:21:36
03:50:24
04:19:12
04:48:00
05:16:48
05:45:36
06:14:24
06:43:12
07:12:00
07:40:48
08:09:36
08:38:24
09:07:12
09:36:00
10:04:48
10:33:36
11:02:24
11:31:12
12:00:00
12:28:48
12:57:36
13:26:24
13:55:12
14:24:00
14:52:48
15:21:36
15:50:24
16:19:12
16:48:00
17:16:48
17:45:36
18:14:24
18:43:12
19:12:00
19:40:48
20:09:36
20:38:24
21:07:12
21:36:00
22:04:48
22:33:36
23:02:24
23:31:12
00:00:00
Combination 3211 with modification 2
Combination 7211
Time

9.8.4 Results of the best strategies against different arrival rates

9.8.4.1 Earliest possible belt strategy

Combination	Normal day	Arrival rate decreased by 10%	Arrival rate decreased by 20%	Arrival rate increased by 10%	Arrival rate increased by 20%
3211	205	205	0	549	1056
3212	404	0	0	1047	1593
3213	0	0	0	0	0

9.8.4.2 Current CPH Airport Strategy

Combination	Normal day	Arrival rate decreased by 10%	Arrival rate decreased by 20%	Arrival rate increased by 10%	Arrival rate increased by 20%
7211	435	37	0	688	1119
7212	505	0	0	1047	1593
7213	0	0	0	0	0

9.8.5 Early Baggage Storage used as a buffer. Comparison of the best strategies

9.8.5.1 Number of manually handled bags on the best strategies

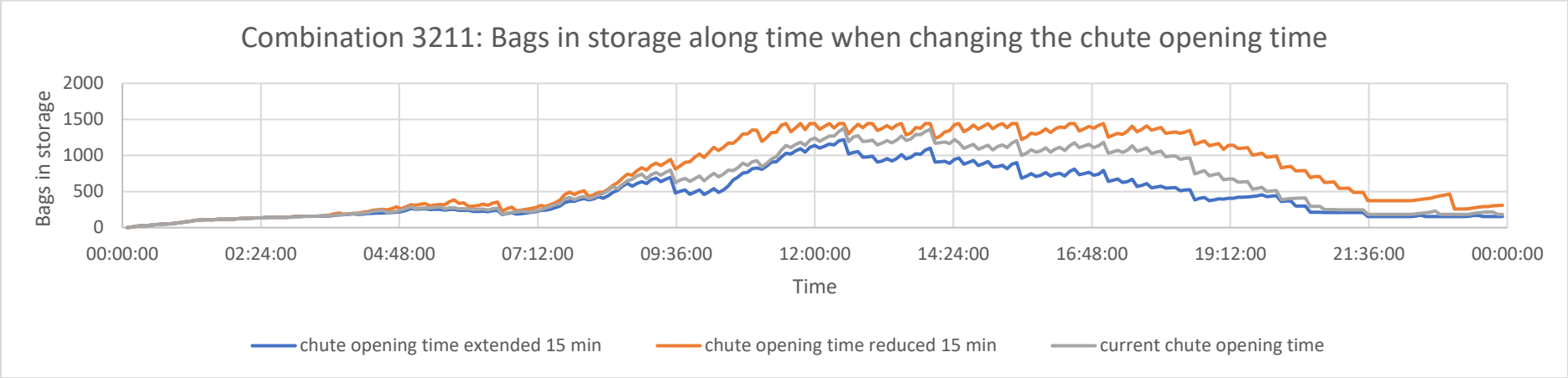
9.8.5.1.1 Earliest possible belt strategy

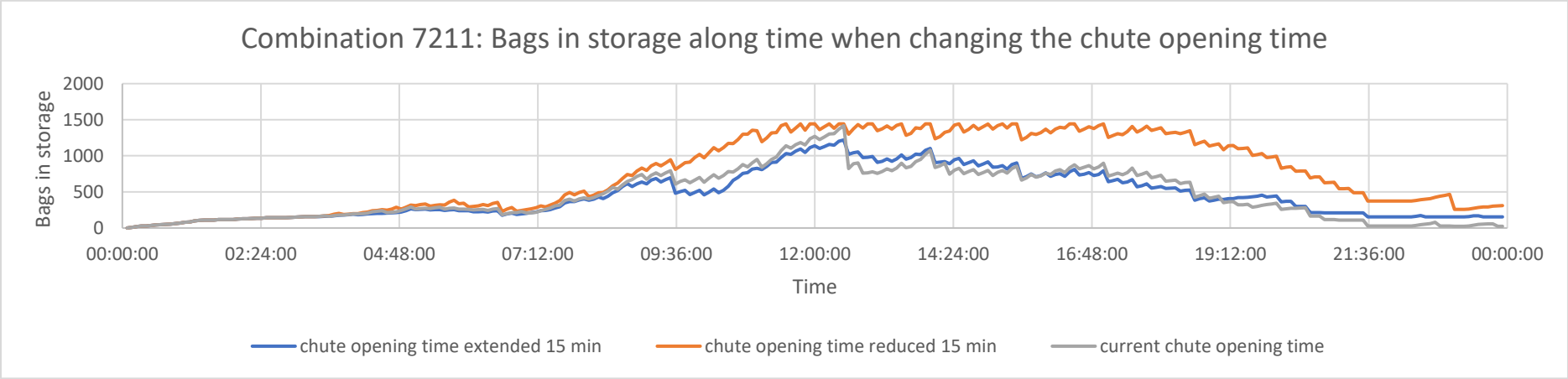
Combination	Normal day	Chute opening time extended 15 min	Chute opening time extended 30 min	Chute opening time reduced 15 min	Chute opening time reduced 30 min
3211	205	0	0	1464	3208
3212	404	0	0	2106	3879
3213	0	0	0	0	0

9.8.5.1.2 Current CPH Airport strategy

Combination	Normal day	Chute opening time extended 15 min	Chute opening time extended 30 min	Chute opening time reduced 15 min	Chute opening time reduced 30 min
7211	435	0	0	1440	3208
7212	505	0	0	1705	3879
7213	0	0	0	0	0

9.8.5.2 EBS storage along time on the best EBS strategies





9.8.5.3 Stress of the sorting system on the best EBS strategies