

**GridPick: A High Density Puzzle Based Order Picking System
with Decentralized Control**

by

Onur Uludağ

A dissertation submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Auburn, Alabama

May 3, 2014

Keywords: puzzle based storage systems, grid based intralogistics, decentralized control, semi automated order picking, message passing algorithms, flexible and modular logistics solutions, multi agent modeling with petri nets, verification of structural properties with petri nets, deadlock free analysis.

Copyright 2014 by Onur Uludağ

Approved by

Kevin R. Gue, Associate Professor of Industrial and Systems Engineering,
Auburn University

Jeffrey S. Smith, Professor of Industrial and Systems Engineering, Auburn University

Kai Furmans, Professor of Institute for Conveying Technology and Logistics,
Karlsruhe Institute of Technology

Abstract

We develop a novel semi-automated order picking system called GridPick that enables high levels of throughput, space usage, and flexibility. The system uses unit sized conveyor modules in a grid architecture to bring requested items to and away from the pick face dynamically and as needed to fulfill orders to the system. GridPick uses a message passing and negotiation algorithm with decentralized control rules, which is a relatively unexplored area in conveyor based material handling systems. In a second part of the dissertation, we modify the GridPick system to allow picking from two sides of the grid. This enhancement allows higher throughput and an improved use of space when compared with single sided, traditional systems. To investigate structural properties of the system, we present a Petri Nets model. We use the model to verify deadlock conditions for the single-sided version of GridPick. We show that the system is deadlock free for up to a 4×4 grid and conjecture that the general case is also true.

Acknowledgments

This research study is supported by the National Science Foundation (CMMI 0926346) under the supervision of principal investigator, Dr. Kevin R. Gue (Grant number: 200267-130501-2000). I cordially appreciate the support of the National Science Foundation and Dr. Kevin Gue for the initiation of this research. I also appreciate the feedback on the research from committee members, Dr. Kai Furmans and Dr. Jeffrey S. Smith.

I would like to thank my wife, Ilknur Uludag for her great support and love. She understood my difficult situations and provided encouragement. We have received another valuable gift, Kerem Alper Uludag, along the knotted way of PhD.

Also, without the support and love of my family, it would be very difficult for me to make it through school. Knowing their support and good wishes even from a long distance, my parents Leyla & Ibrahim Uludag and my sister Incinur Uludag helped me to build necessary courage and patience.

I also want to express my appreciation to my friends for the spiritual support and journey in Auburn. There are many of them to name, but some of them are Yusuf Celikbag, Said Cakir, Yunus Sisman, Emin Ciftci, Vedat Cetinkaya. Also, they were there to help me when I am in need of anything in Auburn. I am grateful for their support and energy to my friends in Turkey: Emrah Harmanbasi, Yasin Yazar, Omer Ergul, Fatih Gulay.

Finally, I want to dedicate the value and appreciation to the true owner of this dissertation and I would like to acknowledge the real source of information. “They replied: God, You are limitless and flawless in your power and sovereignty. We do not have knowledge other than you taught us. You are the alone art all-knowing and truly wise” (Quran, Bakara (2):32).

Table of Contents

Abstract	ii
Acknowledgments	iii
List of Figures	vii
List of Tables	xvi
1 Introduction	1
1.1 Motivation	2
1.2 Background on Order Picking Systems	3
1.3 Background on Grid Based Storage Systems	5
1.4 Research Objectives	6
1.5 Organization of the Dissertation	7
2 Literature Review	9
2.1 Order Picking Systems	9
2.1.1 Structural Decisions and Design Questions	10
2.1.2 Picking Methods	13
2.2 Decentralized Control and Multi Agent Systems	24
2.3 High Density Puzzle Based Storage Systems	27
2.3.1 Unit Sized Transportation Modules	29
3 The GridPick System	32
3.1 Explanation of the Control Algorithm	35
3.1.1 System Overview	35
3.1.2 Symbolism and Representation	37
3.1.3 Negotiation for the Assignment of the Replenishing Items	42
3.1.4 Assignment of the Replenishing Items for the Convoy Movement	47

3.1.5	Negotiation Process for Movement Decisions	52
3.1.6	North-South Negotiation for Vertical Movement	53
3.2	Considerations on the Application of GridPick	58
3.2.1	Pick to Light Systems	59
3.2.2	Pick by Voice Systems	60
3.3	Model Representation	61
3.3.1	Message Passing	63
3.3.2	Determining Buffer Lengths for Cyclic Events	65
3.4	Statistical Analysis for Steady-State Parameters	68
3.4.1	Determining the Warmup Period	68
3.4.2	Determining the Number of Replications	70
3.4.3	Paired t-Tests for the Configuration Comparisons	71
3.5	Performance Analysis	72
3.5.1	Aspect Ratio Analysis	84
3.5.2	Variable k Analysis	92
3.5.3	Analysis with Multiple Copy Configurations	95
3.5.4	Comparison with a Flow Rack Configuration	102
3.5.5	Conclusions	104
4	A Modified Version of the GridPick System: Picking From Two Sides	106
4.1	System Description	108
4.2	Solution Approach	109
4.2.1	Discrete Events and Balance Rules for the Operations	110
4.3	Determination of Buffer Lengths for the Series of Events	126
4.4	Statistical Analysis for Steady-State Parameters	127
4.4.1	Determining the Warmup Period	127
4.4.2	Determining the Number of Replications	128
4.5	Performance Analysis	129

4.5.1	Performance of the Two Sided Model for Several Parameters	129
4.5.2	Aspect Ratio Analysis	139
4.5.3	Distribution of Empty Cells	145
4.5.4	Performance Comparison of the Two Workers	153
4.5.5	Performance Comparison with the One Sided Model	156
4.6	Conclusions	162
5	A Petri Nets Model for Grid Based Storage Systems	167
5.1	Discrete Event Systems Modeling	168
5.1.1	Petri Nets Overview	168
5.2	Fundamental Concepts	171
5.3	Colored Petri Nets	173
5.4	Modeling of GridStore System	175
5.5	Petri Nets Modeling of GridPick	183
5.5.1	Contributions of the Petri Nets Model and Deadlock Conditions	191
6	Conclusions and Contributions	194
A	Appendix	206
A.1	Statistical Data and Confidence Intervals for the Analysis	206
A.2	Paired t Tests	206

List of Figures

1.1	The GridStore System	6
2.1	Classification of Order Picking Systems According to the Automation Level [De Koster, 2004]	13
2.2	Classification of Order Picking Systems with the Multiple Criteria [Dallari et al., 2009]	14
2.3	Bin shelving and Gravity Flow Rack [NAVSUP, 1985]	15
2.4	Automated Storage and Retrieval System with Three Cranes [NAVSUP, 1985]	18
2.5	A Carousel with Rotating Shelves [NAVSUP, 1985]	19
2.6	Kiva Robot (on the left, from http://www.kivasystems.com) and Autostore (on the right, from http://www.mwpvl.com/html/swisslog_autostore_review.html)	20
2.7	A-Frame System, a Worker Replenishes Manually [Caputo and Pelagagge, 2006].	21
2.8	Flow in a Distribution Center with a Forward Reserve Inventory System [Bartholdi and Hackman, 2011]	22
2.9	Abstract Architecture of the Agent [Wooldridge, 2002]	25
2.10	Architecture of the State Maintaining Agent [Wooldridge, 2002]	25
2.11	Different Types of Neighborhood in the Cellular Automaton [Krühn et al., 2010]	26

2.12	Rush Hour Board Game	28
2.13	FlexConveyor	30
2.14	Swiveling Wheel Modules [Overmeyer et al., 2010]	30
3.1	Physical Devices for the Proposed Model: the Unit Sized Conveyor and Storage Container	32
3.2	The Grid of Modules with Four Neighbors	33
3.3	Outline of GridPick	33
3.4	GridPick: an Order Picking System for Carton Picking From Pallets or Piece Picking From Cartons	34
3.5	Example steps for the GridPick System	36
3.6	Representation of the States and the Messages for the Negotiation Algorithm	39
3.7	Example steps for GridPick	56
3.8	A Representation of the Pick to Light Systems	59
3.9	Pick by Voice Systems	60
3.10	GridPick with Several Levels	61
3.11	Series of Synchronous Events in Each Time Step (Source: Anylogic help files).	62
3.12	Each event can take a different process time depending on the length of negotiation (Source: Anylogic help files).	63
3.13	Cyclical Events in Each Time Step	64

3.14 Representation of events executing cyclically.	66
3.15 Initial Transient for Throughput	69
3.16 Initial Transient for Waiting Time	69
3.17 Initial Transient for Walking Time	70
3.18 Initial Transient for Flow Time	70
3.19 Throughput Plot for Different Expected Order Size Levels	73
3.20 Throughput changing with the order size and empty cells per row.	74
3.21 Walking Time per Pick for Different Expected Order Size Levels	76
3.22 Walking time for various expected order sizes and empty cells per row.	77
3.23 Average Retrieval Time for Different Expected Order Size Levels	79
3.24 Flow time for various expected order sizes and empty cells per row.	80
3.25 Waiting Time per Pick for Different Expected Order Size Levels	82
3.26 Waiting time for various expected order sizes and empty cells per row.	83
3.27 Throughput Plot	85
3.28 Throughput for different aspect ratios	86
3.29 Average Retrieval Time Plot	89
3.30 Flow time for different aspect ratios	89
3.31 Waiting Time for Aspect Ratio Configurations	90

3.32	Waiting time for different aspect ratios	90
3.33	Walking Time for Aspect Ratio Configurations	91
3.34	Walking time for different aspect ratios	91
3.35	Throughput Plot	92
3.36	Throughput for variable k configurations	93
3.37	Waiting Time for Variable k Configurations	95
3.38	Waiting time for Variable k Configurations	96
3.39	Walking time for Variable k Configurations	96
3.40	Average Retrieval Time Plot	97
3.41	Flow time for variable k configurations	97
3.42	Throughput Plot for Multi Copies of SKUs	98
3.43	Throughput for multi copies	98
3.44	Average Retrieval Time Plot	100
3.45	Flow time for multi copies	101
3.46	Waiting Time for Multi Copy Configurations	101
3.47	Waiting time for multi copy configurations	102
3.48	Walking time for multi copy configurations	102
3.49	Visual Comparison of GridPick and the flow rack	103

3.50	Productivity Increase in GridPick Compared to a Flow Rack	103
3.51	Comparison of Walking Times for the Flow Rack and the GridPick	104
4.1	Abstract Representation of the Two Sided Picking Model	106
4.2	Visualization of the Two Sided Picking Model	107
4.3	Representation of the States and Messages for the Negotiation Algorithm	109
4.4	Initial Transient for Throughput	127
4.5	Initial Transient for Waiting Time	127
4.6	Initial Transient for Walking Time	128
4.7	Initial Transient for Flow Time	128
4.8	Throughput Plot for Different Expected Order Size Levels	130
4.9	Throughput Plot for Different Expected Order Size Levels	130
4.10	Walking Time per Pick for Different Expected Order Size Levels	133
4.11	Walking Time per Pick for Different Expected Order Size Levels	133
4.12	Average Retrieval Time for Different Expected Order Size Levels	135
4.13	Average Retrieval Time for Different Expected Order Size Levels	135
4.14	Waiting Time per Pick for Different Expected Order Size Levels	138
4.15	Waiting Time per Pick for Different Expected Order Size Levels	138
4.16	Throughput Plot	140

4.17	Throughput Plot for Different Aspect Ratios	141
4.18	Average Retrieval Time Plot	143
4.19	Average Retrieval Time for Different Aspect Ratios	144
4.20	Waiting Time for Aspect Ratio Configurations	144
4.21	Waiting Time per Pick for Different Aspect Ratios	145
4.22	Walking Time for Aspect Ratio Configurations	146
4.23	Walking Time per Pick for Different Aspect Ratios	146
4.24	Different Ways to Allocate Empty Cells in the Two Sided Picking Model	147
4.25	Waiting Time for Variable k Configurations of the Two Sided Model	148
4.26	Waiting Time per Pick for Variable k Configurations	148
4.27	Throughput Plot	149
4.28	Throughput Plot for Variable k Configurations	149
4.29	% Difference on Throughput	151
4.30	Average Retrieval Time Plot	152
4.31	Average Retrieval Time for Variable k Configurations	152
4.32	Walking Time per Pick for Variable k Configurations	153
4.33	Throughput Plot	154
4.34	Throughput Plot for Two Workers	154

4.35	Average Retrieval Time Plot	157
4.36	Average Retrieval Time for Two Workers	157
4.37	Waiting Time for Aspect Ratio Configurations	158
4.38	Waiting Time per Pick for Two Workers	158
4.39	Walking Time for Two Workers	159
4.40	Walking Time per Pick for Two Workers	159
4.41	Throughput Plot	160
4.42	Throughput Plot for Two Models	160
4.43	Average Retrieval Time Plot	162
4.44	Average Retrieval Time for Two Models	163
4.45	Waiting Time for One Sided vs. Two Sided	163
4.46	Waiting Time per Pick for Two Models	164
4.47	Walking Time for One Sided vs. Two sided	164
4.48	Walking Time per Pick for Two Models	165
4.49	Percent Difference on Throughput for Two Sided vs. One Sided GridPick	165
4.50	Percent Difference on Throughput for $2 \times$ Onesided vs. Twosided GridPick . .	166
5.1	Marking before transition firing [Murata, 1989]	169
5.2	Marking after transition firing [Murata, 1989]	169

5.3	Graphical Representation of an Example Petri Net [Zurawski and Zhou, 1994]	173
5.4	Vertical Convoy Movement in Column 0	178
5.5	Vertical Movement in Column 1	179
5.6	Vertical Convoy Movement in Column 2	180
5.7	Horizontal Movement in Row 1	181
5.8	Horizontal Movement in Row 2	181
5.9	Transition from Replenishing to Occupied	182
5.10	Transition for the Replenishment From the Top Row	182
5.11	Transition for the Release of the Request	183
5.12	Output of LoLA Software Package for a 3×3 GridStore	184
5.13	Example of Balance Rule - 1	186
5.14	Example of Balance Rule - 2	186
5.15	Example of the Convoy Movement Encouragement	187
5.16	Example of the Vertical Movement for the Requested Items	187
5.17	Example of the Vertical Movement for the Replenishing Items	188
5.18	Example of the Horizontal Movement for the Requested Items	188
5.19	Example of the Horizontal Movement for the Replenishing Items	189
5.20	Transition Example from Requested to Being Picked Item	189

5.21 Transition Example from Being Picked to Occupied item 189

5.22 Example of the transition from Replenishing to Occupied Item 190

5.23 Example of the transition from Occupied to Replenishing Item 190

5.24 Example of the transition from Occupied to Requested Item 190

5.25 Example of the transition for the New Release of the Requested Items 191

List of Tables

3.1	Possible states of the conveyor module	38
3.2	Additional Negotiation Conditions	39
3.3	Rule Examples	41
3.4	First Ten Steps of Balance Rule 1 with the In advance Movement	43
3.5	Last Three Steps of Balance Rule 1 with the In advance Movement	44
3.6	Communication Rules for Balance Rule 1	45
3.7	Steps of Balance Rule 2 with the Latter Movement	46
3.8	Communication Rules for Balance Rule 2	47
3.9	Steps of the First Event of the Convoy Movement	48
3.10	Communication Rules for the First Event of the Convoy Movement	48
3.11	Steps of the Second Event of the Convoy Movement	49
3.12	Communication Rules for the Second Event of the Tandem Movement	50
3.13	Steps of the Third Event of the Convoy Movement	51
3.14	Communication Rules for the Third Event of the Convoy Movement	52
3.15	Negotiation conditions for the movement decisions	53
3.16	Example 1 for North-South Negotiation	54
3.17	Communication Rules for North-South Negotiation	55
3.18	Example for East-West Negotiation	57
3.19	Communication Rules for East-West Negotiation	58
3.20	CI's for the Negotiations (Units are in milliseconds)	67

3.21	CI for the Four Performance Measures	71
3.22	p values for throughput with various expected order sizes and empty cells per row	75
3.23	t values for throughput with various expected order sizes and empty cells per row	75
3.24	CI for throughput with various expected order sizes and empty cells per row .	76
3.25	p values of walking time with various expected order sizes and empty cells per row.	78
3.26	T values of walking time with various expected order sizes and empty cells per row.	78
3.27	CI of walking time with various expected order sizes and empty cells per row. .	79
3.28	p values for flow time with various expected order sizes and empty cells per row	80
3.29	t values for flow time with various expected order sizes and empty cells per row	81
3.30	Confidence Intervals for flow time with various expected order sizes and empty cells per row	81
3.31	p values for waiting time with various expected order sizes and empty cells per row	83
3.32	t values for waiting time with various expected order sizes and empty cells per row	84
3.33	CI for waiting time with various expected order sizes and empty cells per row .	84
3.34	Configurations with Different Aspect Ratio Values	85
3.35	p values for throughput with various aspect ratios	86
3.36	t values for throughput with various aspect ratios	87
3.37	CI for throughput with various aspect ratios	87
3.38	CI for throughput with various aspect ratios	87
3.39	CI for throughput with various aspect ratios	88
3.40	Distribution of Empty Modules for 3 Configurations	92
3.41	p values for throughput with variable k configurations	93
3.42	t values for throughput with variable k configurations	94
3.43	CI for throughput with variable k configurations	94
3.44	p values for throughput with null copy configurations	99

3.45	t values for throughput with mull copy configurations	99
3.46	CIs for throughput with mull copy configurations	100
4.1	Additional Negotiation Conditions	111
4.2	Rule Examples	112
4.3	Steps of Balance Rule-1 for Requested-1 in Two Sided GridPick	113
4.4	Communication Rules of Balance Rule-1 for Requested-1	114
4.5	Steps of Balance Rule-1 for Requested-2 in Two Sided GridPick	116
4.6	Communication Rules of Balance Rule-1 for Requested-2	117
4.7	Steps of Balance Rule-2 for Requested-1 in Two Sided GridPick	118
4.8	Communication Rules of Balance Rule-2 for Requested-1	119
4.9	Steps of Balance Rule-3 for Requested-1 in Two Sided GridPick	120
4.10	Communication Rules of Balance Rule-3 for Requested-1	121
4.11	Steps of Balance Rule-3 for Requested-2 in Two Sided GridPick	122
4.12	Communication Rules of Balance Rule-3 for Requested-2	123
4.13	Steps of Balance Rule-4 for Requested-2 in Two Sided GridPick	124
4.14	Communication Rules of Balance Rule-4 for Requested-2	125
4.15	Steps of Balance Rule-5 for Requested-2 in Two Sided GridPick	125
4.16	Communication Rules of Balance Rule-5 for Requested-2	126
4.17	Length of the Message Passing for Negotiation of Events	126
4.18	Confidence Intervals and Statistical Data for 30 Replications	129
4.19	p values of Throughput with Different Expected Order Size Levels	131
4.20	t values of Throughput with Different Expected Order Size Levels	131
4.21	CIs of Throughput with Different Expected Order Size Levels	132
4.22	p values of Walking Time with Different Expected Order Size Levels	132
4.23	t values of Walking Time with Different Expected Order Size Levels	134

4.24	CI's of Walking Time with Different Expected Order Size Levels	134
4.25	p values of Flow Time with Different Expected Order Size Levels	136
4.26	t values of Flow Time with Different Expected Order Size Levels	136
4.27	CI's of Flow Time with Different Expected Order Size Levels	137
4.28	p values of Waiting Time with Different Expected Order Size Levels	137
4.29	t values of Waiting Time with Different Expected Order Size Levels	139
4.30	CI's of Waiting Time with Different Expected Order Size Levels	139
4.31	Configurations with Different Aspect Ratio Values	140
4.32	p values for throughput with various aspect ratios	141
4.33	t values for throughput with various aspect ratios	142
4.34	CI's for throughput with various aspect ratios	142
4.35	CI's for throughput with various aspect ratios	142
4.36	CI's for throughput with various aspect ratios	143
4.37	Distribution of Empty Modules for 3 Configurations	147
4.38	p values for Throughput with Variable k Configurations	150
4.39	t values for Throughput with Variable k Configurations	150
4.40	CI's for Throughput with Variable k Configurations	151
4.41	p values for Throughput of Two Workers	155
4.42	t values for Throughput of Two Workers	155
4.43	CI's for Throughput of Two Workers	156
4.44	p values for Throughput of Two Models	161
4.45	t values for Throughput of Two Models	161
4.46	CI's for Throughput of Two Models	162
5.1	Some Interpretations of Places and Transitions [Murata, 1989]	172
5.2	Variables for the Arc Inscriptions	177

5.3	Results of the Petri Nets Models	192
A.1	Statistical Data for the Throughput of One Sided GridPick	207
A.2	Statistical Data for the Aspect Ratio Configurations of One Sided GridPick	208
A.3	Statistical Data for the Variable k Configurations of One Sided GridPick	208
A.4	Statistical Data for the Multi Copy Configurations of One Sided GridPick	281
A.5	Statistical Data for the Throughput of Two Sided GridPick	282
A.6	Statistical Data for the Aspect Ratio Configurations of Two Sided GridPick	283
A.7	Statistical Data for the Variable k Configurations of Two Sided GridPick	283
A.8	Statistical Data for the Two Workers Comparison of Two Sided GridPick	284

Chapter 1

Introduction

In a dynamic economy, supply chains with shorter lead times and lower inventory have a significant advantage because they respond to customer needs much more quickly. With the recent advancements such as the internet and globalization, companies should provide very short delivery times and a high variety of products. With the help of e-commerce, companies often experience high growth rates, which leads to their pursuit of more developed systems.

In conjunction with the internet, global competition has led to new trends in distribution operations and logistics. Efficient management of the material flow is the primary goal of logistics systems. A logistics system consists of transportation operations, material flow management, and physical distribution systems. While transportation operations and material flow management take care of the movement for the manufacturing plants, the distribution system determines the customer service dimension of the problem.

Distribution operations receive goods from suppliers and fulfill customer orders. These operations are also referred to as intralogistics, which is the science of moving physical goods within distribution facilities. Short product flow times and decreasing batch sizes require sophisticated operations in intralogistics. High growth rates and advanced operations require relocation of facilities or capacity expansion of current facilities, which can be a big issue. Companies strive to have flexible material handling systems for their intralogistics operations.

A key component for flexible and responsive supply chains is the distribution center. A distribution center's operations are receiving, labeling, picking, replenishing, packing, sorting, sequencing, controlling inventory, and shipping the customer orders. Order picking is one of the major functions in a distribution center. Order picking is typically the most costly

operation in the distribution center [De Koster et al., 2007]. It is also crucial determinant of customer service.

1.1 Motivation

In order picking operations, companies adopt automated material handling systems for a number of reasons: to decrease labor costs, to reach high performance goals, and to deliver customer orders on time. The task of an automated storage system is to provide short retrieval times of goods while having high utilization of the worker, and using as little space as possible. While achieving these goals, automated systems bring other drawbacks. Automated systems are typically expensive and are not as flexible as manual systems. With a manual system, we can increase the throughput capacity by simply increasing the number of workers. When there is a need for additional capacity in an automated system, an overhaul of the entire system is required in most cases. Furthermore, automated systems should handle large amounts of goods and process a number of operations at the same time.

For sophisticated and large volume operations, automated material handling systems embody several levels of mechanical, electrical, and data systems. Even a small change requires many adjustments, which are costly and time consuming. Companies usually postpone these small changes and tuneups, which leads to poor performance. For this reason, *adaptability* to different needs while sustaining high performance requirements is an important capability. Companies prefer automated material handling systems that have flexible and adaptive features.

Design of an automated order picking system can be quite complex considering design goals such as space, costs, and automation level. Traditional design goals are maximization of throughput, minimization of storage space, and minimization of response time. Recently other factors have emerged: companies also desire a flexible and adaptable system that can respond to changes. But, these features require other capabilities.

To have a flexible and adaptive order picking system, system components should be *modular*, and should be able to *integrate* with each other. Modularity can provide a *reconfigurable* system, which enables easy combination and replacement of system components. Furthermore, modular system components can enable a *scalable* system. When there is a need for additional capacity or a requirement for less capacity, modules can be added or removed from the system. In case of a breakdown, modules should be changed without adjustment of the whole system. Furmans et al. [2010] introduced design principles for plug and work material handling systems. We can obtain highly adaptive and flexible automated material handling systems with these features.

Consequently, one might wonder how to obtain a more flexible automated order picking system by handling complications such as *scalability*, *modularity*, *reconfigurability*, and *space constraints*. A semi-automated order picking system may handle known restrictions of manual and automated order picking operations. *Centralized control* is also a drawback because it is not able to enable flexibility and modularity properties. A centralized control system includes a central command unit for the decision making process. On the other hand, in decentralized control, each module has its controller unit, which enables independent operability of system components. Independent ability of communication and decision making among the system components provides a *decentralized control* system instead of a central controller, which enables adaptability and flexibility features.

This research is motivated by the lack of flexible automated order picking systems in the literature and in practice. We introduce a novel order picking design with the considered adaptability and flexibility features. We also analyze specific properties of the system through simulation and computational models.

1.2 Background on Order Picking Systems

Once an order is received for a product in the distribution center, it should be picked and prepared for shipping to the customer. Order picking includes traveling to the item,

searching for the item, reaching and extracting the item, and bringing it to the shipping point. There can be additional activities such as labeling, documenting, and sorting the items. The designer of the order picking system should consider several design parameters, strategies, and policies.

Most warehouses are divided into two major areas: reserve and forward areas. In the reserve area, goods are stored in large quantities and sizes such as pallets. Forward areas are arranged for order picking, and they include frequently requested stock keeping units (SKUs) for fast picking. The main objective behind an order picking system is to increase SKU density, which means there are many SKUs with small quantities. Therefore, the worker can reach a large number of different products with less travel. Generally, forward areas are replenished from the reserve area. There can be multiple levels of forward areas that provide different order picking forms and replenish each other [Jernigan, 2004]. Order picking can be in the form of piece picking, carton picking, or pallet picking. Piece picking is the retrieval of small numbers of the product, and workers generally consolidate the products to a tote. Carton picking is picking with cases and consolidation is to a truck or pallet. Pallet picking is sending the goods directly with large pallets. Replenishment to a forward area is mostly infrequent and may happen in the facility's break periods.

The automation level of the order picking can also differ. There can be a manual order picking system in which workers handle all the material movement. In a semi automated system, a robotic system can handle the travel of goods to the picker. Some order picking systems employ fully automated approaches that handle both the travel and picking processes.

Minimization of the space usage is also an objective of the order picking system design. Increasing the number of storage levels, having deeper storage locations, and narrowing the aisles are a couple of ways to increase the storage density [Gue, 2006]. However, they all have additional disadvantages such as requirement for specialized vehicles, increased safety issues, and decreased performance.

When there is a necessity to move other items to reach a requested item, we can speak of a “high density” order picking system [Gue, 2006]. Puzzle based storage systems are an alternative to the aisle based designs, which form the aisles as needed to retrieve the requested items [Gue and Kang, 2001]. Puzzle based storage systems provide a high density configuration with the movement of only one item at a time [Gue et al., 2013]. Grid based storage systems consider multiple items moving at the same time in a high density configuration. Simultaneous movement of storage units requires a design methodology of complex control algorithms.

1.3 Background on Grid Based Storage Systems

We introduce a high density order picking system called GridPick. It relies on a grid based design methodology introduced with GridStore [Gue and Furmans, 2011, Gue et al., 2013]. This is a new field of storage and retrieval system that can be applied in several different intralogistics operations. GridStore has defined the main state transition scheme for the handling of the movements. GridPick has a decentralized control algorithm with a message passing protocol in which unit sized conveyors communicate with their local neighborhood to decide on the cooperative movements.

GridPick is based on the decentralized control algorithm of GridStore (see Figure 1.1), in which unit sized conveyor modules form a rectangular grid [Gue and Furmans, 2011, Gue et al., 2013]. Each conveyor module has its own controller, sensors, and mechanical controls. They can communicate with each other and recognize the totes or cartons via RFID or other means of communication. Conveyor modules move the storage units without a centralized command. In this system, conveyor modules communicate only with their four neighbors (north, south, east and west neighbors). The task is to retrieve requested items from the takeaway conveyor at the bottom of the grid. Restocking of the items is from the replenishment conveyor at the top of the grid. Requests come from an external source to

the items via RFID or other means of communication. Conveyor modules realize that they carry a requested item, which needs to be moved down and reach the takeaway conveyor.

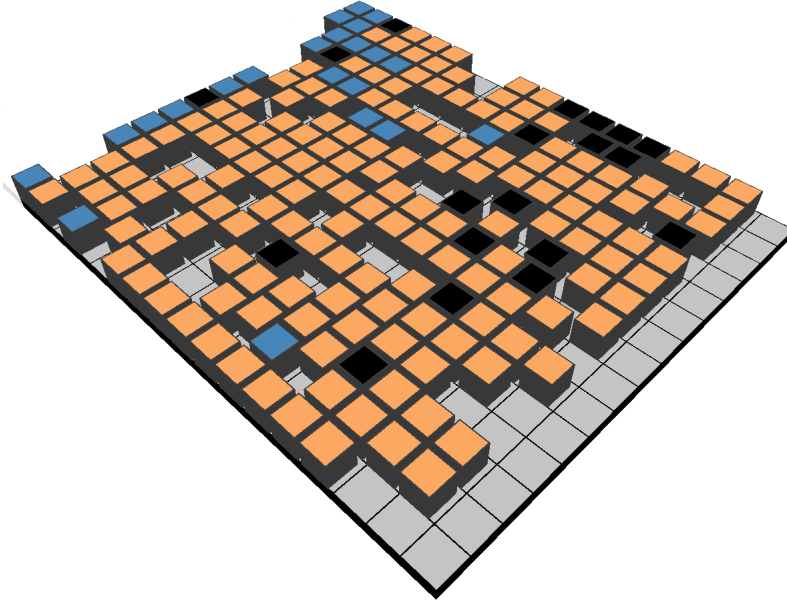


Figure 1.1: The GridStore System

In Figure 1.1, black items are requested and required to reach the bottom takeaway conveyor. Other items are either stored items moving east or west to allow the passage of requested items or replenishing items that are restocked from the top replenishment conveyor.

A decentralized control algorithm is adopted in GridStore, which enables modularity, reconfigurability, and scalability. In each time step, conveyor modules assess their own state, negotiate with their neighbors, and do the conveying. This state transition scheme iterates in each time unit, and concludes with the required decisions to move requested items to the takeaway conveyor.

1.4 Research Objectives

In an order picking system design problem, the main objective is to have a high SKU density, and consequently a high pick density. This will decrease the amount of time spent for traveling and increase the picks per unit time. Storage density is also preferred to

be high in order to minimize space related costs such as electricity, air conditioning, and land costs. It is also desired to maximize the worker utilization and eliminate activities unrelated to the picking process. From a customer satisfaction point of view, we want to have a responsive system that delivers orders on time. With recent trends, flexibility is an important capability that can enable a responsive system. Flexibility is in conjunction with adaptability to changing needs.

These objectives are usually incompatible and not easy to achieve at the same time. For instance, we can store items in a very high density configuration, but this may decrease the throughput of the system. We can try to have a very responsive system and keep workers ready for orders, but this can decrease worker utilization. Furthermore, a system can have very good performance in current conditions, but it may not be able to respond to changes. For instance, a failure to expand capacity can result in late deliveries or unmet demand.

The main objective of this research is to establish a flexible and efficient order picking system by taking advantage of decentralized control. We target a high density and high throughput order picking system by having picking operations with short travel and high pick density.

1.5 Organization of the Dissertation

In chapter 2, we provide a literature review for related topics. We discuss conventional order picking systems such as flow rack and bin shelving. We explain design principles and sections of a distribution center such as forward pick areas and bulk areas. We discuss performance measures in use for order picking systems and point out a few other goods-to-man automated systems. We describe decentralized control and agent based modeling concepts and mention applications in logistics and manufacturing systems.

Chapter 3 introduces the GridPick system and explains the message passing algorithm. It covers performance analysis and presents a comparison with a traditional order picking system. We discuss key insights such as keeping empty cells in each row. We also determine

how system parameters such as aspect ratio, empty cells per row, expected order size, and distribution of empty cells affect system performance.

Chapter 4 describes a major enhancement of the GridPick system to retrieve requested items from two sides of the grid. We discuss the additional balance rules that are introduced with a two sided model. We have also analyzed the performance of the two sided model compared to the initial GridPick model for a number of parameters such as expected order size, empty cells per row, and aspect ratio.

Chapter 5 presents the modeling and structural analysis of GridPick with Petri nets. This chapter includes a review of Petri nets theory, which is a discrete event system modeling tool that is able to model concurrent and parallel systems. The Petri nets model does not describe the message passing scheme explicitly, but it exactly mimics the movement patterns of the algorithm. Therefore, we are able to determine the properties of the system such as deadlocks and boundedness. Chapter 6 summarizes the research study and outlines the contributions.

Chapter 2

Literature Review

To be familiar with the concepts and to get the understanding of introduced models presented in this work, we discuss previous research and studies in related fields. We also give explanations of necessary terms, classifications and topics. Section 2.1 describes warehouse design problems and order picking operations. Section 2.2 presents decentralized control concepts. Section 2.3 introduces discrete event and concurrent system modeling tools.

2.1 Order Picking Systems

A logistics system integrates material flow, management and physical devices into an overall delivery system [Tompkins et al., 2003]. Therefore, an efficient logistics structure should effectively manage and perform the flow of goods and services from manufacturers to customers. For the companies, warehouses play a major role for the storage and distribution of goods and services. According to De Koster et al. [2007] if the facility mainly performs storage and buffering operations, *warehouse* is the more appropriate term. When the distribution of goods is the main concern, *distribution center* is the correct term. We use the terms interchangeably because the storage and distribution activities are both vital and integrated to each other.

There are several terms in the warehousing literature. A *stock keeping unit (SKU)* is a number or code that identifies each product. So, each item type in a warehouse has a unique number that cannot be mixed with another product type. When an order is received, it generally includes a number of products as a list, which is called an *order line* [Bartholdi and Hackman, 2011]. Each request in this list provides information such as required quantity, product type, and SKU number. The Warehouse Management System (WMS) reorganizes

the order line and converts it to a *pick line* according to the operations and the configurations of the warehouse. WMS is for the organization and control of warehouse operations, and it includes the needed information such as product availability, product location, and quantity. The pick line is the complete guide for the picker such as where to go, what to pick, in which quantity, etc. [Bartholdi and Hackman, 2011]. The surface of the storage areas in which SKUs are available is called a *pick face*. Ideally, warehouses want to have more SKUs available per unit of area in the pick face, which will decrease required travel and increase the *pick density*. So, the *pick density* is the number of picks made from per unit of area; we want it to be high for an efficient order picking operation [Bartholdi and Hackman, 2011].

2.1.1 Structural Decisions and Design Questions

Designing a warehouse is a difficult task that involves to determine a number of parameters. With the design and control of the warehouse, we should tackle all design problems and handle the operations by using as few resources as possible. There are several analytical and simulation based approaches for the warehouse design problems [Ashayeri and Gelders, 1985], [Rouwenhorst et al., 2000] and [Baker and Canessa, 2009]. Van den Berg [1999a] reviewed planning and control related decisions in the warehouse such as inventory management, location assignment, order batching and routing. Gray et al. [1992] introduced models and applications for warehouse design and planning problems such as technology selection, item location, zoning, picker routing, pick list generation, and order batching. Kostrzewski [2012] described a warehouse designing method in a procedure for the optimization of functional and spatial areas with a software.

Order picking is a major activity in distribution centers. The design and selection of appropriate order picking system is vital for operations. Brynzer et al. [1994] introduced a methodology called zero based analysis to compare order picking systems that divides resource consumption into three parts: necessary work (reaching, grabbing and transferring items), time losses (traveling, waiting, etc.), and administrative tasks (picking preparation,

quality check, etc.). By dividing activities that are common in all order picking systems, different order picking system designs can be compared. Yoon and Sharp [1996] introduced a structured procedure for order picking system design and control by considering relationships between functional areas (retrieval, picking, sorting, etc.). Dallari et al. [2009] proposed an order picking system classification and developed a design methodology based on Yoon and Sharp [1996]. The methodology uses data from companies such as picking rates, outflow, order size and response time. After that, they specify the operating strategies, storage capacities and equipment requirements with analytical expressions. They compared the final order picking system designs with a number of considerations and constraints.

Load Policies

There are several types of picking operations according to the number of tasks assigned. A single command cycle includes picking or storing one item, usually a pallet or a carton due to its size and weight. After the completion of task, the worker or the crane performs another single command cycle. Each time, the worker or the crane picks a unit and comes back to the deposit location. Dual command cycles have two tasks in a single trip. The worker or the crane stores an item to the location and makes a pick before returning to the pick and deposit point.

Unit load AS/RSs can be an example to both single command and dual command cycles. Cranes can pick and drop an item with a single command cycle or can store an item, pick another one in the same trip as a dual command cycle. If the worker or crane picks several items in a single trip, it is called multiple command order picking. Usually storage areas are large, and travel time comprises a major part of the processing time. For this reason, multiple picks in a trip allocate travel time among a number of items.

Storage Policies

After the decision of which order picking system to use, we should assign items to storage locations. There are several ways to assign SKUs to the slots in a storage area. One can adopt the *random storage policy* in which pickers place items in randomly selected locations. The random storage policy can result in a high space utilization. On the other hand, it can lead to increased travel time for order picking operations [De Koster et al., 2007]. Because it is easy to implement, randomized storage is preferred by many distribution centers.

Warehouses can also adopt a *dedicated storage policy*, in which items are assigned to one or multiple fixed locations. In this policy, it is likely to have a low utilization due to having an assigned location for out of stock items; but, it has the advantage of learning curve for order pickers with the fixed location of SKUs [De Koster et al., 2007]. In most cases, the dedicated storage assignment policy depends on constraints such as weight, size, fragility, temperature, and humidity conditions [Tompkins et al., 2003].

Some warehouses adopt a *volume based* or *full turnover storage policy* in which items are allocated according to demand rate. Petersen and Schmenner [1999] claimed that volume based storage reduces travel time significantly. The demand based assignment policy has a considerable drawback: demand rates of the products change frequently and it requires reassignment of SKUs [De Koster et al., 2007]. Also, the *cube per order index (COI)* based storage assignment policy can be adopted, which is based on an index value calculated as the ratio of the item's required space over number of required trips for the demand [Lee, 1992].

A combined policy is the *class based storage policy*. In this assignment policy, SKUs are divided into classes based on a measure of demand and pick volume. Items are allocated randomly within each class. The assumption in this policy is the 15-20% of the items correspond to the 80-85% of the activity in the order picking operations [De Koster et al., 2007]. We should assign this class of fast moving items to the best locations. We can also allocate the SKUs with respect to their similarities in the orders. Items that are frequently

ordered together will be placed close to each other, which is called *family based storage* [Frazelle and Sharp, 1989]. These storage assignment policies are used in both manual systems and AS/RSs [Roodbergen and Vis, 2009].

2.1.2 Picking Methods

Depending on the degree of automation, we can classify order picking systems into several groups. Dallari et al. [2009] proposed a classification for the order picking systems (see Figure 2.2) based on the classification introduced by Sharp [1992] and reviewed by De Koster [2004] (see Figure 2.1).

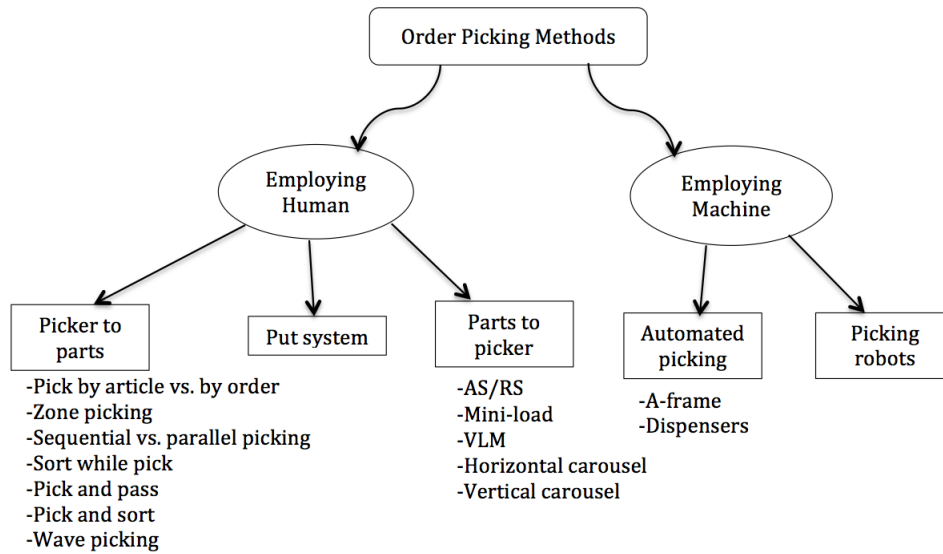


Figure 2.1: Classification of Order Picking Systems According to the Automation Level [De Koster, 2004]

De Koster [2004] classified order picking systems into two major groups as employing human and employing machine (see Figure 2.1). Without any automation, the picker should travel along aisles to fulfill orders (picker to parts). Order picking systems that employ human may still have an automation level that brings the items to the picker (parts to picker).

Dallari et al. [2009] pointed out the classification rationale and stated several criteria such as use of conveyors for the connection of picking zones (see Figure 2.2). So, there are

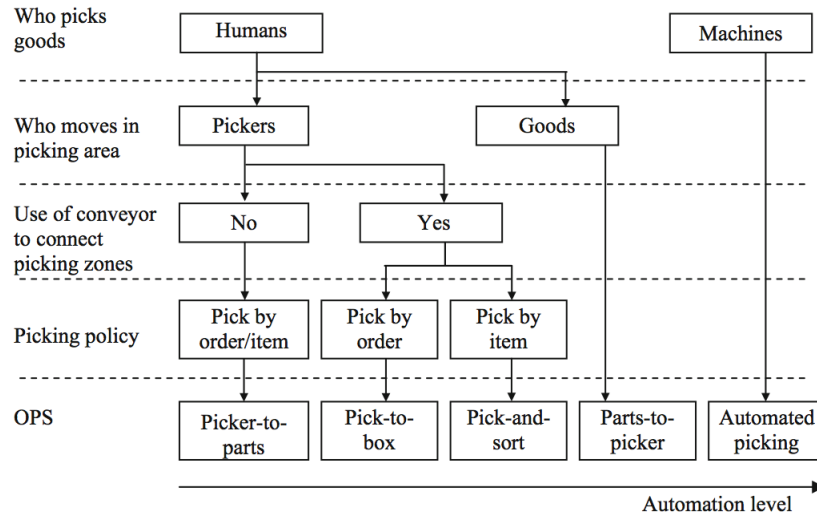


Figure 2.2: Classification of Order Picking Systems with the Multiple Criteria [Dallari et al., 2009]

many levels of automation starting from the manual picking and usage of basic tools to highly automated cranes and picker robots. Parts to picker systems provide a larger throughput capacity with its automation capabilities. Furthermore, parts to picker system solve the problem of labor constraints and costs. Therefore, the companies have an increasing interest on the parts to picker systems or goods to man systems. They put more effort and time for developing novel systems. We explain different types of order picking systems in the consequent subsections.

Picker to Parts Systems

In a picker to parts system, the order picker travels to the picking location by walking or driving through aisles. It is the most common method and can be low level picking from racks or bin shelves, or high level picking with a lift truck or crane [De Koster, 2004]. In high level picking, there can be many vertically oriented storage locations, and a crane enables picking from the storage places. Order picking includes several elements depending on the system in use. If we employ a picker to parts system, there are many worker-oriented

activities. The picker should search for the picking location and travel to, from, and between the picking locations. After that, the picker should reach and retrieve the item.

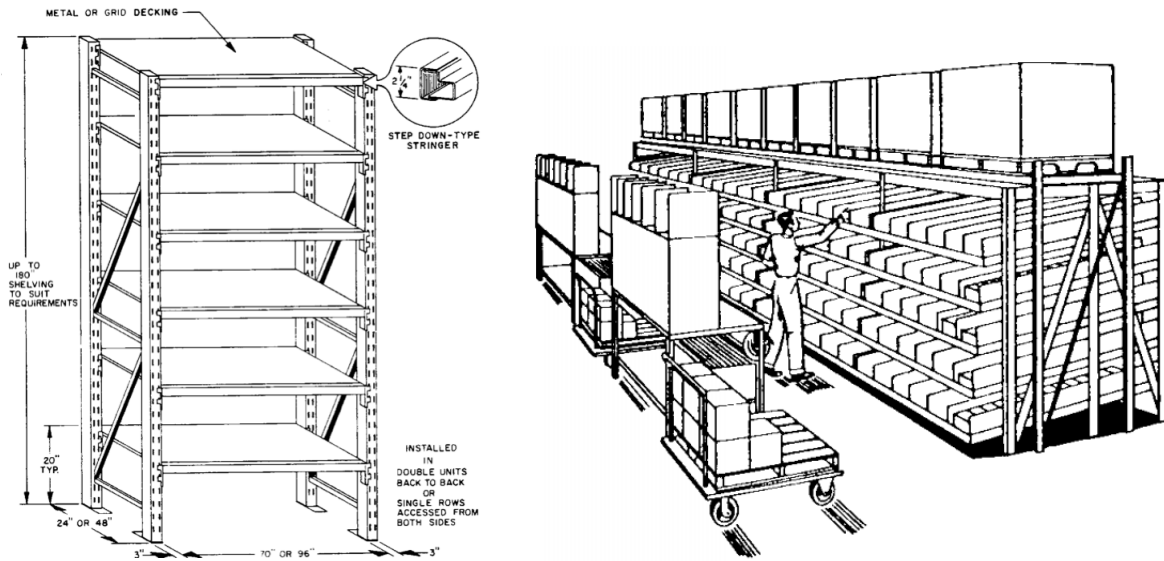


Figure 2.3: Bin shelving and Gravity Flow Rack [NAVSUP, 1985]

Bin shelving is the most common and least expensive storage equipment [Bartholdi and Hackman, 2011]. It requires minimal maintenance and is easy to adjust. It allows storage of many different kinds of items like cartons, paper containers, bins and small items. The shelves are typically shallow and storage of large quantities for each SKU results in fewer SKUs on the pick face. For the bin shelving, 50-100 picks/hour is the common picking rate [Bartholdi and Hackman, 2011]. Picking and replenishment is typically from the same side of the shelves.

A storage mode with higher SKU density is gravity flow rack. It is for broken case or piece picking of items with high turnover. It has multiple levels that are tilted, or works with rollers to slide items to the front after an item is picked from the pick face. So, only one copy of a SKU occupies the pick face and SKU density is high, resulting in high pick density and less travel. Replenishment is from the back and picking is from the front; so,

it maintains FIFO principle. Generally, replenishment is in large quantities and is triggered when one copy or a few copies of the SKU is left.

Assistive Technologies for the Pickers

There are different assisting techniques for picking the right item in the right quantity from its location. Picker may use either paper based pick lists or digital control systems having stored pick lists. There are also other paperless picking technologies, such as light directed or pick to light systems. In a pick to light system, lights on the racks direct the picker to the location and indicate the quantity. In a voice-directed system, the picker uses a headset to receive voice commands for the information such as location and required quantity of the SKU. Some companies use handheld radio frequency directed devices to communicate with the picker.

Picking Strategies

There are also different picking strategies for picker to parts order picking systems. The order can be batched to pick multiple orders in a single tour, and routing algorithms can optimize the travel time by rearranging the pick list and determining an appropriate route [Roodbergen and De Koster, 2001]. In the batch picking policy, since the picker accumulates multiple orders in a tour, the picker may sort the items immediately (sort while pick system), and place them in different totes in the picking cart. Another approach is to have a sortation workstation or a chute conveyor that can sort items out according to the orders.

In another picking policy, the picking area is divided into zones and assigned to the pickers. Pickers pass the order tote or the carton to the picker operating in the next zone (pick and pass system). In the zone picking policy, there is no need for sorting and picking is completed order by order. There might be a need of sortation for the delivery destination of the orders. Self organizing order picking systems such as bucket brigades are also example of pick and pass order picking with a self stabilizing zone picking policy [Bartholdi and

Eisenstein, 1996]. In a regular zone picking system, one should manage workload balancing among picking zones. Warehouses may prefer zone picking when the picking frequency is high with small sized items.

Distribution centers also sort items after the picking of multiple orders as in batch picking. In this case, orders are fed into a takeaway conveyor and sortation conveyors send each item to a destination bay (pick and sort system). These systems are often integrated with wave picking [De Koster et al., 2007], in which the WMS releases orders to balance the workload of the picking operations. So, it is expected to have a high number of orders or large batches in each wave. In some cases, warehouses prefer to have manual sortation activities. Especially in piece picking operations, the order picking and sortation operations are costly to automate, and hard to handle. For this reason, after the retrieval of the items, they are sent to workers for the allocation of customer orders (put system) [De Koster et al., 2007]. This system can give good performance for the orders with high quantities of small items.

In a picker to parts system, travel time is the largest part of the processing time. A number of studies developed routing policies to determine the sequence of picks and determined the routes that each picker will follow [Ratliff and Rosenthal, 1983], [Roodbergen and De Koster, 2001]. Ratliff and Rosenthal [1983] modeled the routing policy as a traveling salesman problem and adopted dynamic programming. Hall [1993] discussed several routing policies such as traversal, midpoint return, largest gap return for narrow-aisles, wide-aisle and zoned warehouses. Petersen and Aase [2004] used a simulation model to evaluate several picking, storage and routing policies, and determined which policies provide savings in order picker travel.

Parts to Picker Order Picking Systems

In parts to picker systems, an automated system brings items to the pickers. A common parts to picker system is the automated storage/retrieval system (AS/RS), which consists

of multi level storage locations with a crane carrying items out of the storage area through aisles (see Figure 2.4). If the order picking system is piece picking, the conveyors bring totes or cartons back to the storage cranes of the AS/RS for putting back. These systems are called end of aisle systems or mini load AS/RS for the bin storing systems. In case of carton picking, cartons leave the system and newly arrived items are stored in the AS/RS. Double deep storage can be employed in case of a low variety of SKUs with high turnover rate [Tompkins et al., 2003].

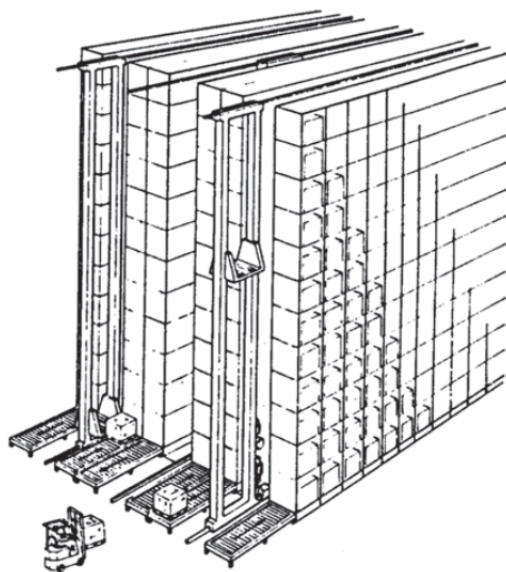


Figure 2.4: Automated Storage and Retrieval System with Three Cranes [NAVSUP, 1985]

The main advantage of an AS/RS is the picking cost reduction with the initial investment of automation; but it has the possibility of bottlenecks and low productivity in case of a small number of cranes [Dallari et al., 2009]. There are also savings in floor space and high order accuracy. AS/RSs have a large number of options, most common version has one crane in each aisle and works with the unit load command [Roodbergen and Vis, 2009]. Other studies have introduced flow rack variations of AS/RSs ([Sari et al., 2005] and [De Koster et al., 2008b]). Sari et al. [2005] developed travel time expressions for a flow rack AS/RS that has a retrieval machine on the pick face of the AS/RS and a storage machine at the back. There

is one pick up location, one drop off location and a restoring conveyor for the link between the picking and replenishment side. De Koster et al. [2008b] considered an AS/RS with gravity or powered flow racks. A crane moves the items to and from the storage area. They developed travel time expressions and addressed the sizing problem.

Hu et al. [2005] introduced a new AS/RS for heavy loads like shipping containers and provided travel time analysis for such systems. In this system, a vertical platform moves items among levels of the rack, and several horizontal platforms transfer the items to a specific storage location. For a higher utilization, two racks share the horizontal platforms. The main advantage of this system is the ability to move heavier loads at a higher speed by separating horizontal and vertical drives [Hu et al., 2005].

A relatively new class of AS/RSs is called autonomous vehicle storage and retrieval systems (AVS/RS) , in which autonomous vehicles perform storage and retrieval operations [Zizzi, 2000]. Malmberg [2002] discussed the design features of AVS/RS technology. The autonomous vehicles move on rail guides from and to the storage locations. Vertical movement is handled by mounted lifts to the vehicle. Vertical moves of the lifts are significantly slower than the horizontal movements, which affects the throughput performance directly. Heragu et al. [2009] and Kuo et al. [2008] studied performance analysis and modeling of AVS/RSs with simulation based and queueing-theoretic approaches.

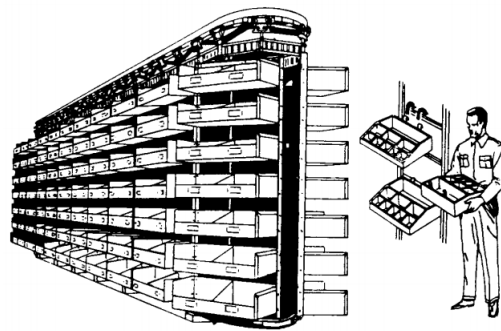


Figure 2.5: A Carousel with Rotating Shelves [NAVSUP, 1985]

A carousel is a set of shelves that are able to rotate and bring required items to the pick face of the worker rather than the picker traveling to the storage location. Since shelves with

requested orders come in the front of the picker, there is no need for aisles for traveling to the storage location. Therefore, one can place carousels in a high density configuration. To compensate for possible picker idleness during the rotating of the carousel, one should employ multiple carousels for high volume picking [Bartholdi and Hackman, 2011]. However, in most cases, each carousel includes a number of items for the order, which causes an unavoidable waiting of the worker.

Picking robots as a goods to man system

Companies have recently introduced new approaches to automated storage and retrieval systems. Conventional AS/RSs are often not flexible to changes in demand. They may require huge investments for additional capacity, and throughput is restricted to the few cranes operating in the system.

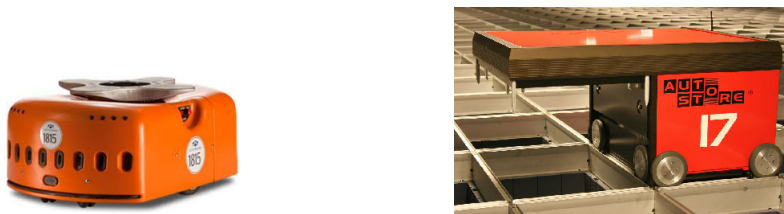


Figure 2.6: Kiva Robot (on the left, from <http://www.kivasystems.com>) and Autostore (on the right, from http://www.mwpvl.com/html/swisslog_autostore_review.html)

Newer systems use unit sized devices and grid based systems for more flexibility. Features such as modularity and scalability enable easy adjustments in case of a change in demand, moving, and sizing the systems for different needs. It also provides reduction in space and labor costs. For instance, there are two recent systems introduced: Kiva robots and Swisslog's Autostore. Kiva robots are able to go under shelves or other portable storage units and bring them to the pickers. Autostore units are standing on stacks of totes and they carry items to and from the pick up and drop off points. Both systems have the advantage of flexible order picking with the cost of automation. To justify an investment on these systems, high volume picking of fast moving items should be the case in the distribution center. These

systems are not for the picking process itself but to operate as a parts to picker order picking system. With these devices, in case of a change in needs, the storage and retrieval system can be scaled. Additional units can be easily added to the system for additional throughput or units can be used somewhere else if there is low demand in a particular system.

Automated Picking Systems

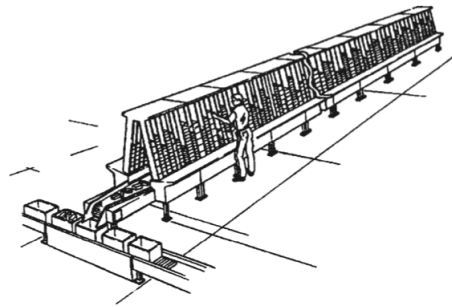


Figure 2.7: A-Frame System, a Worker Replenishes Manually [Caputo and Pelagagge, 2006].

There are also order picking systems employing automation for the picking process itself such as dispenser based systems. The main idea in these systems is automatically dispensing small items to a takeaway conveyor that are stacked in the channel like racks [Caputo and Pelagagge, 2006]. A-frame picking systems are common examples, which have stacked dispensing shelves side by side on two sides of the takeaway conveyor. In especially distribution centers for pharmaceutical products and small items of retail and e-commerce stores, A-frame systems are suitable.

The A-frame picking system can either fulfill the order one at a time and the takeaway conveyor can drop items to a tote for the specific order, or totes can travel within the takeaway conveyor and system can dispense items directly into the tote. A-frames can perform up to 750,000 picks/day with a high order accuracy [Caputo and Pelagagge, 2006]. Replenishment is generally manual with a worker filling the channel racks (please see Figure 2.7). A high volume and small items order picking can justify this kind of investment.

Order Picking Areas

Most distribution centers have two different sections: bulk and forward areas. Efficient picking areas are generally called forward pick or fast pick areas. In the bulk storage area, SKUs are stored in large, pallet quantities. Most SKUs are stored in the forward pick areas in small amounts to complete orders with a short travel time and high picking efficiency. The picking areas get replenished from the bulk storage or from other forward pick areas. Therefore, there is a benefit of picking with the cost of replenishment. Often warehouses use separate picking operations depending on the size of the orders. Case picking, piece picking, broken case picking, and pallet picking are common types of order picking operations. Items can be picked from both forward and bulk areas. Picking from the forward area may reduce the cost of picking depending on the size of the order and restocking costs. There can also be multiple forward areas, which are restocked from another forward area or from bulk, which is called a “multi tier inventory system”.

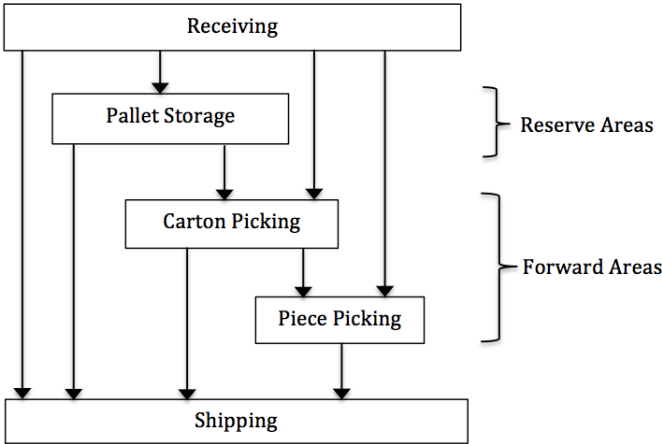


Figure 2.8: Flow in a Distribution Center with a Forward Reserve Inventory System [Bartholdi and Hackman, 2011]

Piece picking operations such as the bin shelving, flow rack, and A-frame automated systems can significantly decrease picking cost. At the same time, these systems will need restocks and we want it to be as low costly as possible. A common forward picking area type is the first level of the pallet rack; pickers can pick cartons of popular items and replenish

from the upper levels [Bartholdi and Hackman, 2011]. For less costly restocking, we may increase the capacity of the current forward area, but it is generally expensive to expand it and it can increase the picking costs with a longer travel time. We may adopt an intermediate forward area between bulk storage and forward pick area with less capital investment and without increased picking costs for the current forward area [Jernigan, 2004].

Hackman and Rosenblatt [1990] introduced a model to assign SKUs to forward areas and to determine the volume of each SKU in the forward area. They provided a knapsack based heuristic and compared it with a ranking based approach considering real data. Van den Berg et al. [1998] studied a forward reserve problem that considers which products in what quantities should be stored to the forward area in order to minimize labor time. They assumed that there are busy periods with picking periods and idle periods for the replenishment from the reserve area. They formulated the problem as a binary programming problem and introduced a heuristic that provides a performance guarantee. Also, they considered the LP relaxation of this model and showed the optimal solution with fractional values.

Heragu et al. [2005] developed a mixed integer programming model and a heuristic algorithm for the product allocation and size determination problems in the forward reserve areas and cross docking. They were able to obtain good results for large problems with the heuristic algorithm. Frazelle et al. [1994] considered product allocation while using the size of the forward area as a variable. Bartholdi and Hackman [2008] compared equal space and equal time allocations of the products with the optimal allocation of the fast pick area. In equal space allocation, the same space is assigned to each SKU, and in equal time allocation, SKUs spend the same time in the forward area. Bartholdi and Hackman [2008] showed that these two common allocation policies result in the same number of restocks, and that the optimal allocations model of Hackman and Rosenblatt [1990] is significantly better. Gu et al. [2010] developed a branch and bound algorithm for the forward reserve allocation problem that is able to solve practical problems quickly.

2.2 Decentralized Control and Multi Agent Systems

We can adopt several different types of control architecture for a complex system. Mayer [2009] indicates three types of control concepts. If an independent unit operates the complete decision process and includes the entire logic for the system, we call this centralized control. In the case of unavailability of the central control unit, the entire system does not know what to do and a change in the system requires an update to the central control. Another approach is a hierarchical control, in which responsibilities and decision making processes are divided into subunits. This allows local decisions at lower levels and takes some workload from the central controller. In hierarchical control, subcomponents still need decisions and control from the central unit for the interoperability of system components. In the decentralized control approach, each system component decides individually and communicates on the same level. There can be specific capabilities and different functionalities of the units. Specific capabilities do not affect the equal ability of communication.

A closely related modeling approach for decentralized control is multi-agent systems. Architecture of the agents and interaction protocols are two main concerns of the multi-agent systems. Wooldridge [2002] introduced an abstract architecture of agents based on the concept of perception and action (see Figure 2.9). Common visualization of the agents is to assign attributes that determine the action and behavior of the agent. Wooldridge [2002] stated that architecture of an agent can be reactive, deliberative or a hybrid of both. In a multi-agent system, we can refer to some surroundings or circumstances called the environment. So, interaction between agents are either direct or indirect through the environment. For the indirect reaction, an agent may change things in the shared environment, which results in some reaction from other agents. Reactive agents are able to perceive the environment and react within the defined logic. Deliberative agents follow a goal directed manner, which means they act with the defined objectives and try to reach a specific goal.

In Figure 2.9, the perception part captures changes in the environment. It is a function to sense the environment and to convert the state of the environment to an impression. The

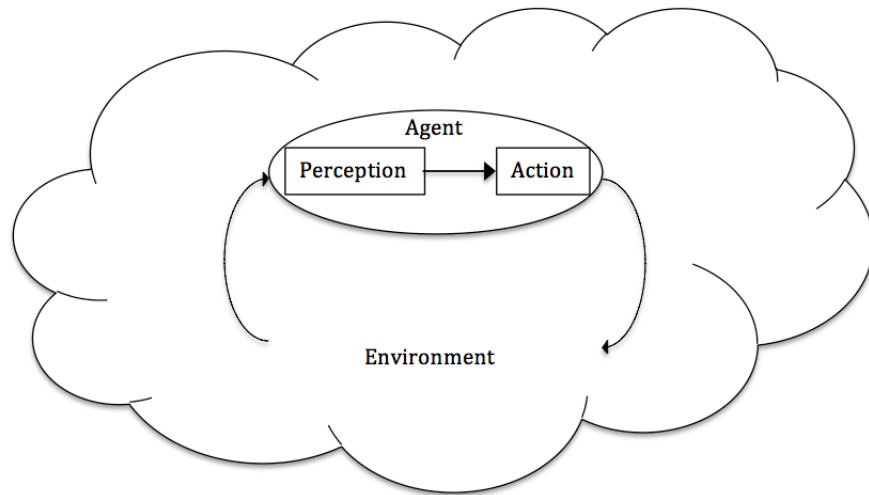


Figure 2.9: Abstract Architecture of the Agent [Wooldridge, 2002]

perception is an input for the action function, and action decides on a behavior according to perceptions.

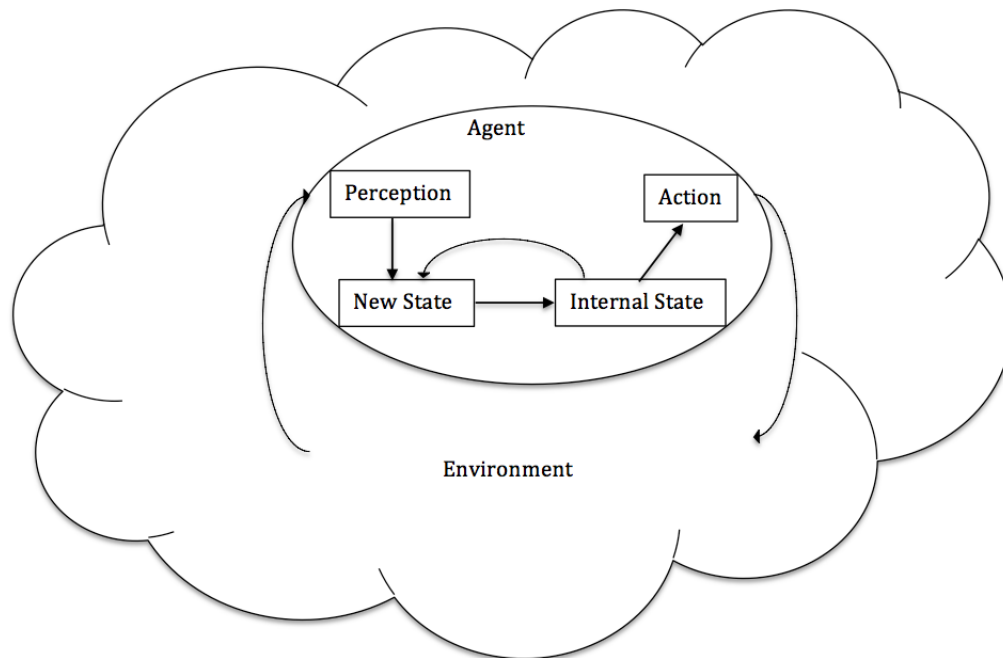


Figure 2.10: Architecture of the State Maintaining Agent [Wooldridge, 2002]

The agent can exchange information and interpret messages with a communication protocol [Weiss, 1999]. The protocol defines the set of rules for the understanding of messages

and the language among the agents. With this interaction mechanism, agents may cooperate for goal oriented behavior, compete for resources, or do negotiations for self defined rules. Multi agent systems have broad application in areas such as computational science, biology, and logistics [Davidsson et al., 2005].

An agent can have a state, which is a position already taken or a decision making mechanism. With perception from the environment, the agent can go to a new state and also act to change the environment. An internal state provides the computational power and a mechanism for the agent specialization (see Figure 2.10).

Another closely related modeling approach is the cellular automaton. It is a discrete modeling architecture including a cellular space in which each cell can take a finite number of states [Smith, 1971]. A certain set of cells stands as the neighborhood for each cell. We assign an initial state for each cell. According to the rules, each cell goes to a new state depending on the current state of the cell, and the state of the cells in the neighborhood. Generally, for each cell, state updating rules and the neighborhood types are the same, and there are no specialized cells.

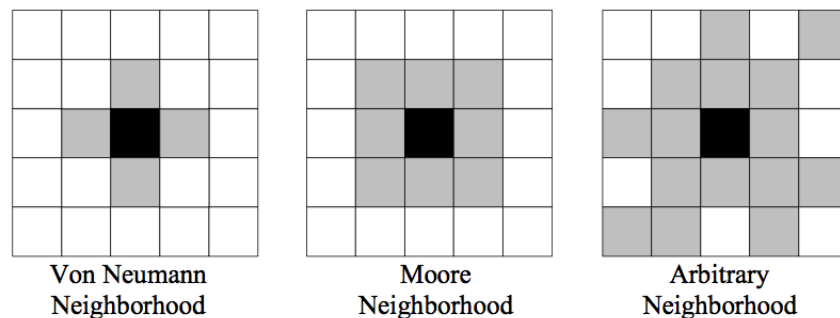


Figure 2.11: Different Types of Neighborhood in the Cellular Automaton [Krühn et al., 2010]

Common types of neighborhoods are the von Neumann neighborhood [von Nuemann, 1966] and the Moore Neighborhood [Moore, 1962] (see Figure 2.11). Conway’s “Game of Life” is a well known cellular automaton application in which cells are born or die depending on their Moore neighborhood on a two dimensional cellular space [Gardner, 1970]. A live cell stays alive with two or three live neighbors and a dead cell is born with exactly three

live neighbors. In all other cases, the cell dies due to overcrowding or loneliness. With these simple rules, interesting patterns emerge.

Efforts for high flexibility material handling systems are introduced under the term “Internet of Things” (IoT). This refers to a self configuring network of material handling systems with decentralized control [Ten Hompel et al., 2006]. Materials with RFID tags can be recognized and flow through the individual objects of a network of the material handling system. With decentralized control, “what you see is what you get” from a physical device. So, modularity and reconfigurability can be provided by independent subcomponents of a system. Montreuil [2011] introduced a similar term called “Physical Internet” and argued that current logistic systems are not sustainable. The Physical Internet is based on a metaphor between the digital internet and global material flow. Highly integrated and standardized logistics systems may provide globally efficient material handling systems.

2.3 High Density Puzzle Based Storage Systems

Due to high costs and limited availability of places for distribution centers, effective use of space is an objective for companies. One way to have higher density is to narrow the aisles. But, it brings additional complications like specialized vehicles and the possibility of pickers blocking each other. Another way might be to make the racks taller. However, it also requires specialized vehicles and it increases safety issues. There are also storage areas with deeper storage locations. Lane depth is the maximum number of items stored in a storage location. Generally, a lane is dedicated to a single SKU to prevent additional material handling. We can refer to a storage system as “high density” when it is usually necessary to move the interfering items according to reach the required item [Gue, 2006]. Assuming that the lane will be replenished when it is depleted, storage positions close to the aisle will be empty after picked until it is replenished again, which is called “honeycombing waste” [Bartholdi and Hackman, 2011]. Gue [2006] introduced an algorithm to fill multi

deep storage spaces with high density and showed that with a maximum lane depth of k , the density of storage space cannot be greater than $2k/(2k + 1)$.

Gue and Kim [2007] introduced a high density storage system that defines puzzle based movement inspired by the children’s game, 15-Puzzle. They assumed that each cell is self propelled, which means the cell can move independently with its own devices. Furthermore, “puzzle movement” is defined as the independent motion of open cells only one at a time. There is one input/output point and one unit moves at a time. They developed an optimal algorithm with one empty cell, performed the expected time analysis for multiple empty cells and compared puzzle based systems with aisle based storage systems. Taylor and Gue [2010] presented design parameters and the best placement for multiple empty cells case of puzzle based storage systems. Alfieri et al. [2010] considered puzzle based storage systems with a limited number of vehicles. They developed heuristics and conducted the retrieval time analysis. These studies have a central control algorithm.

A related problem in the literature is the “Warehouseman’s Problem” studied in Hopcroft et al. [1984]. They showed the PSPACE-hardness of this problem, stronger than being NP-hard. The problem is basically arranging and moving different sized rectangle items in a large rectangular space to obtain a required configuration. Sharma and Aloimonos [1992] proposed some constraints and concepts of temporary storage space to obtain a tractable problem and developed polynomial time algorithms that give coordinated arrangements.



Figure 2.12: Rush Hour Board Game

Another associated problem is the board game “Rush Hour.” In this game, there are 3×1 trucks and 2×1 cars and one specific car needs to exit the board by moving other vehicles either vertically or horizontally (see Figure 2.12). Therefore, a certain number of moves in cardinal directions will form an aisle for this specific car, allowing it to exit. Flake and Baum [2002] showed that “Rush Hour” is PSPACE-hard with a reversible logic approach. Hearn [2006] introduced a framework called nondeterministic constraint logic that can be used to evaluate the complexity of Rush Hour as well as other puzzle games.

The “Discrete Warehouse Problem” is also a related problem studied by Sarrafzadeh and Maddila [1995], in which unit sized objects such as robots, movable and non-movable obstacles occupy a grid-like space. The task is to construct a clear path for the robot with a motion planning procedure. They consider two types of movement for the obstacles: movement by a remote mechanism and movement by the robot. A more general case of motion planning problems with movable obstacles was studied by Wilfong [1988]. He showed that with one robot and movable obstacles, the problem is NP-hard. These problems consider one move at a time and central control. Today’s automated systems enable simultaneous movements by multiple modules and complexity of such integrated subsystems necessitate decentralized control concepts along with the benefits like high throughput and flexibility.

2.3.1 Unit Sized Transportation Modules

In this part, we mention several transportation modules working with decentralized control and developed by research institutions. These devices are unit sized conveyor modules that provide underlying technology for grid based storage systems.

Mayer [2009] presented a decentralized control system with identical modules, which includes the development of a unit sized conveyor module called the FlexConveyor (see Figure 2.13). His control algorithm for each module provides generation of a topological map, recognition of an incoming item, planning of the route to the destination, and deadlock

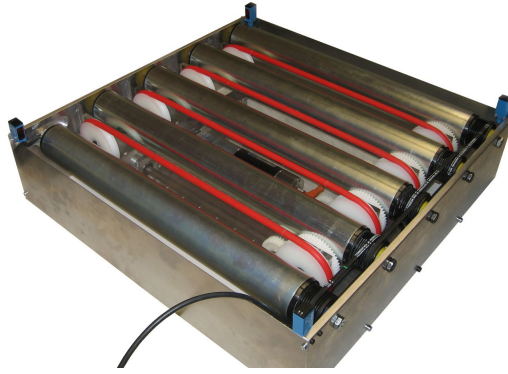


Figure 2.13: FlexConveyor

avoidance. The FlexConveyor is a modular, unit sized conveyor that can be plugged into other conveyor modules to form a conveyor network.

Each conveyor module has its own controller, which enables decentralized control, and its own motor and sensor control that enables it to convey in the four cardinal directions (north, south, east and west). It also has an RFID reader to recognize the tote or carton above the conveyor. So, it identifies the tote and determines the destination. The conveyor modules are able to communicate and send messages to each other for the aim of conveying in the required direction. Information of the totes and cartons can be also passed within the conveyor modules. The FlexConveyor has all the design principles mentioned above. We can reconfigure it easily by plugging it with other modules and we can scale the system with a smaller or larger number of modules. Furthermore, failures of the individual modules can be handled without the break down of the whole system while the system is still running, which is studied by Furmans et al. [2012].

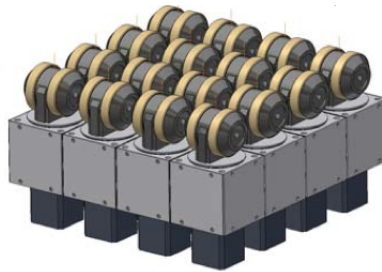


Figure 2.14: Swiveling Wheel Modules [Overmeyer et al., 2010]

Another small scaled modular system is the swiveling wheel [Overmeyer et al., 2010] (see Figure 2.14). An advantage of these modules is that they are able to roll in any x-y coordinate. The grid formed with the swivel wheel modules can be also expanded with a larger number of modules. Based on this technology, roller and belt conveyors can take on different functionalities such as sorting tasks, discharge and transfer with adjusted orientation. Krühn et al. [2010] introduced a decentralized, self organizing control for the swiveling wheel modules that fulfills the transfer and sorting tasks.

Chapter 3

The GridPick System

We introduce a novel order picking system called GridPick for carton picking from pallets or piece picking from cartons. GridPick is developed from the puzzle based architecture defined in GridStore [Gue et al., 2013]. Assumptions are based on unit sized conveyors able to convey loads in the four cardinal directions (see Figure 3.1). Unit conveyors have their own controllers, motors and sensor controls, and can be connected to form a grid for storage of items. They are able to send and receive messages digitally and items physically.

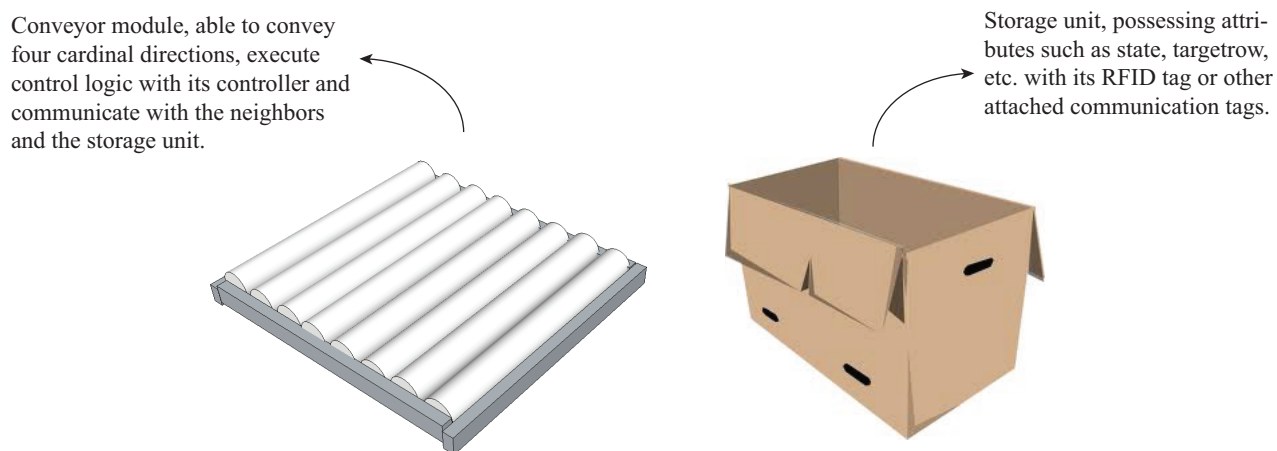


Figure 3.1: Physical Devices for the Proposed Model: the Unit Sized Conveyor and Storage Container

GridPick can be filled with storage containers at a high storage density. There are no fixed lanes or aisles; only empty locations distributed on the grid to allow the movement and retrieval of items. The problem is to provide a high pick rate by moving requested items to the pick face while avoiding any possible congestion.

Decentralized control addresses flexibility concerns by providing modularity, scalability and reconfigurability. Each conveyor module has identical control logic and executes the

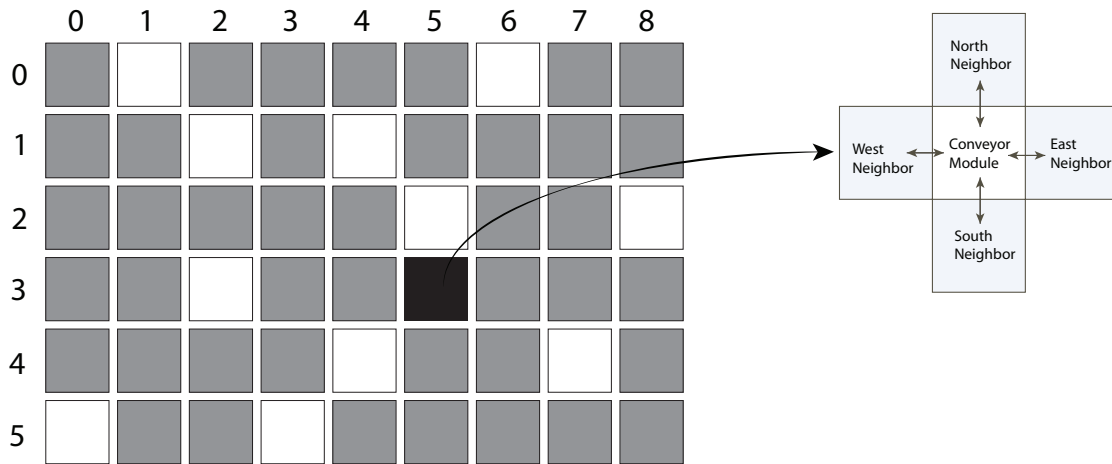


Figure 3.2: The Grid of Modules with Four Neighbors

same algorithm as a synchronous set of discrete events. The synchronous events guarantee that we have the modules in the same phase of the algorithm.

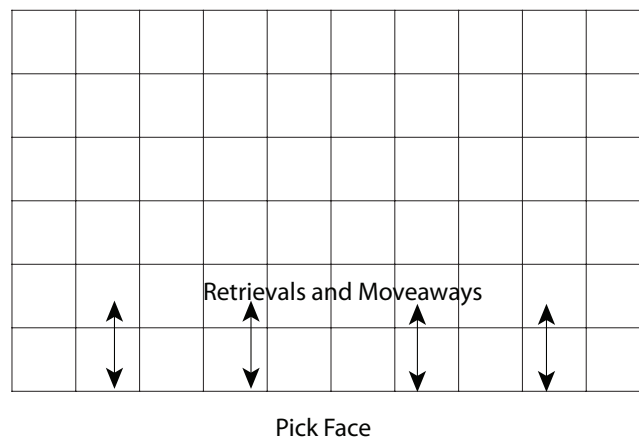


Figure 3.3: Outline of GridPick

Unlike GridStore, items in GridPick do not leave the grid. Requested items move down to the edge of the system. Other interfering items allow the passage of the requested items by moving east and west. There is also a backward vertical movement to balance the empty cells in each row. We sustain the system liveness or avoid deadlocks with these balancing vertical movements. A picker goes back and forth on the pick face to pick the cartons from the pallet or the pieces from a carton (see Figure 3.4). The worker accumulates the order to a storage container or a picking cart.

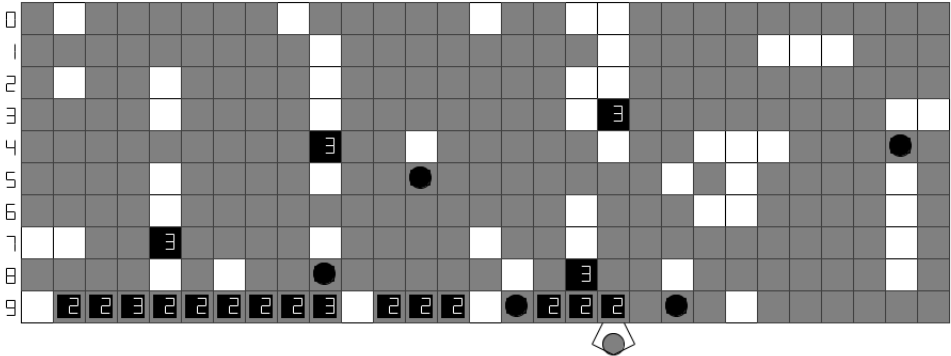


Figure 3.4: GridPick: an Order Picking System for Carton Picking From Pallets or Piece Picking From Cartons

Figure 3.4 shows an instance of GridPick. Gray items are the stored items that are not requested in the current order or in orders soon to be picked. Black items are requested and they travel to the pick face for processing by the worker. Gray items with black circles are balancing items moving in the opposite direction from the pick face. The numbers on top of the items show the order numbers of the requested items. Once all the items of an order arrive to the pick face, the next order is released. Order release or requests come from an external source to the system that communicates and activates the storage containers. So, for a series of small orders, the system can release a number of orders. For a series of large orders, the system can activate the next order while the worker is picking the current order. Therefore, we want to have the next order ready before the worker starts picking the current order. GridPick provides a dynamically changing pick face that provides a high pick density.

Therefore, the proportion of walking time in the total process time will decrease and picks per unit time will increase.

Since the worker picks a few items out of the storage container, the storage containers do not deplete often in GridPick. For this reason, we assume that the storage containers circulate in the grid for a long time and they need less frequent restocks. After a while, a storage container may deplete and this can provide more empty cells. We assume that items stay in the grid and density does not decrease over time, which is a more conservative assumption than having a less dense grid over time. Furthermore, while having multiple copies of a SKU, the problem of which copy to request appears in GridPick. We assume a copy is chosen randomly, which is a more strict policy without an enforcement to choose the specific items. It provides the ability of realistic performance analysis for a new developed system without the approaches for choosing a specific copy of SKU. Additionally, we adopt an order release policy in which the items of the next order arrive to the pick face while the worker is processing the current order.

3.1 Explanation of the Control Algorithm

The decentralized control algorithm is based on a series of negotiations conducted by passing messages between the conveyor modules. Each module's environment consist of four neighbors and the storage container. The conveyor modules can receive the information from the environment. They can respond to the messages, and take new states. They can also change the environment by sending messages to neighbors or by changing the attributes of the storage container.

3.1.1 System Overview

Figure 3.5 shows an example of the algorithm steps. The grid consists of unit sized modules. Unit sized modules include storage containers. Black items are requested items, and they include goods that are required to be shipped to the customers. The picker operates

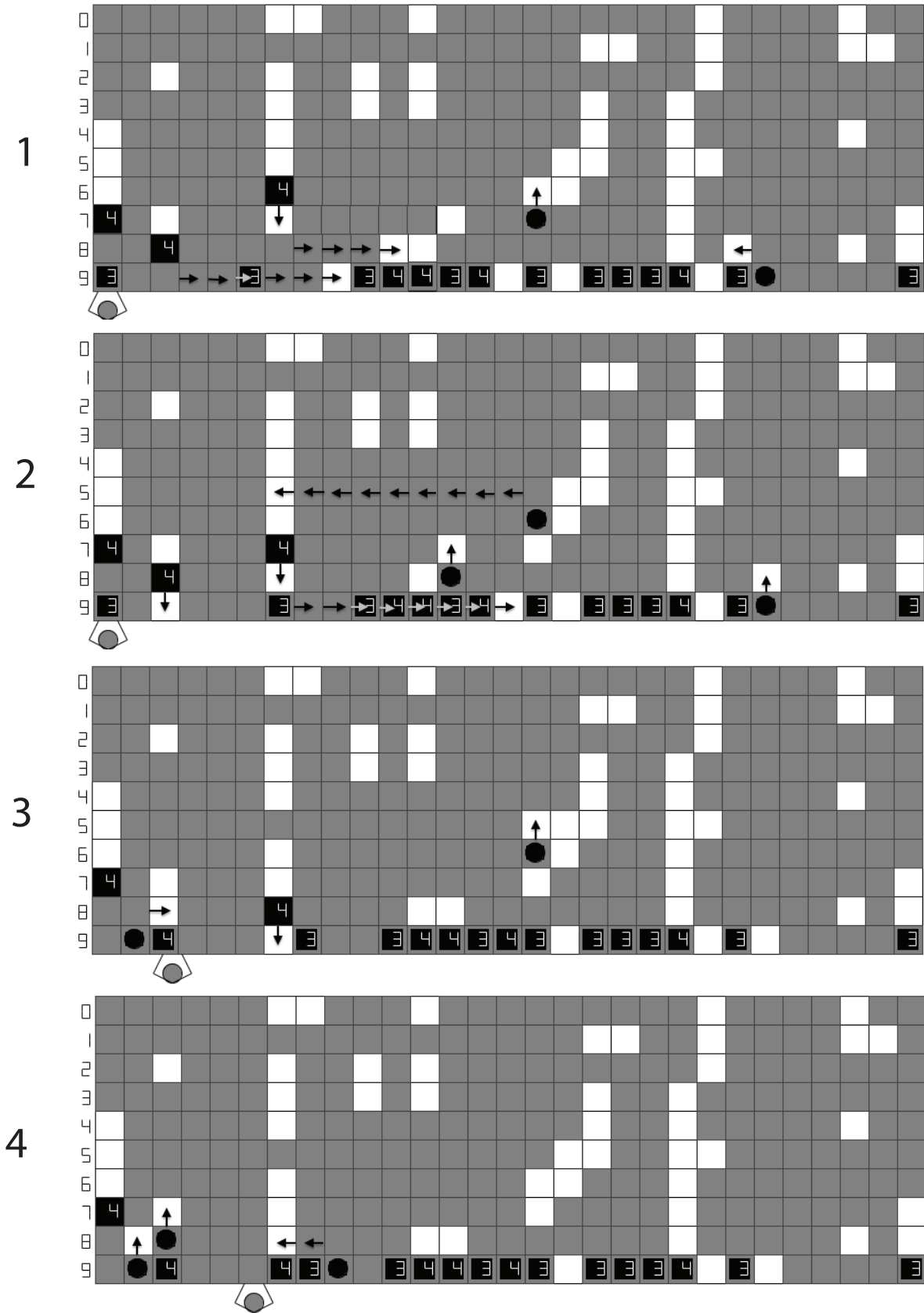


Figure 3.5: Example steps for the GridPick System

on one side of the grid, and makes some picks out of the storage containers. In the shown steps, the worker processes order number 3. Black requested items travel to the pick face to be picked by the worker. Gray items are not requested, but are occupied items in the grid. Occupied items move to the east or west to form empty cells and allow vertically moving storage units (see time steps 1 and 2). White cells show the empty locations. Some other items move away from the pick face to keep some empty cells in each row. So, the replenishing items keep the balance of the grid occupancy. Items with the black circles have the replenishing storage units (see time steps 3 and 4).

In each time step, each module executes the algorithm and negotiates with other modules to decide on the movements. In time step 1, row 6 has a requested item moving towards the pick face. Occupied items in row 8 move to the east to allow the passage of the requested item. A replenishing item in row 7 balances the movement of the requested item. In row 9, occupied items move to the east to allow the vertical movement of another requested item. In time step 2, row 5 has a group of occupied items moving to the west to open an empty cell for the replenishing item. Since order 3 has already arrived to the pick face, order 4 is released, and its items are traveling to the pick face.

3.1.2 Symbolism and Representation

The model consists of discrete synchronous events. We will go through these events and explain state changes, negotiation and decision steps. The model first runs setup functions to place the conveyor modules and stored items to form a grid. Setup functions mimic the physical establishment of the grid and initial configuration. The function for the placement of the items leaves a number of empty cells in each row, which is defined as a system parameter.

The conveyor modules take initial states, which are determined by the stored item. We have used conveyor “modules” and conveyor “units” interchangeably throughout the document. We say an item is *replenishing* when it is moving to balance the occupancy of a requested item. Requested items of the order move to the pick face for processing. Stored

items are stationary unless they must move to allow passage of Requested or Replenishing items. Table 3.1 represents the states taken according to the attributes of the storage container. The conveyor modules take negotiation positions and evaluate decisions based on these states. These states can change either by a movement of the item or as a result of the negotiations.





Symbol	Conveyor module state	Definition
	Empty	The conveyor module did not perceive an item. So, it is not occupied by any item.
	Requested	The conveyor module carries a requested item.
	Replenishing	The conveyor module carries a replenishing item.
	Occupied	The conveyor module is occupied by a stored item that is not intended to move without a trigger.

Table 3.1: Possible states of the conveyor module

Furthermore, we define each conveyor module’s neighborhood to mimic the physical environment. The conveyor module’s West port is connected to the West neighbor, North port to North neighbor, and so on. The conveyor modules get the information of their location (specifically, row and column number). Consequently, the modules can consider this information during their decision. A module that is on the edge of the grid will know it does not have a neighbor on one side, and it will not forward a message that way.

For the purpose of better understanding state changes, other symbols provide additional information used in the negotiation steps. In Figure 3.6, symbols represent the state of the conveyor module (black circle with gray square in the background, black square, etc.). Numbers on the top right of the symbol show either target row, departure row or current row depending on the state. So, a number on the Replenishing item always shows the target row. The target row represents the row to which a Replenishing item should go. A number on the Requested item always indicates the departure row. The departure row represents the row to which a Requested item belongs or has departed from. A number on the Occupied

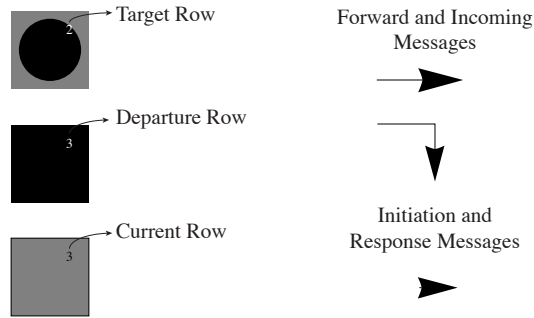


Figure 3.6: Representation of the States and the Messages for the Negotiation Algorithm

item always represents the present row of the stored item. An arrow with a bar shows a forward message or an arbitrary incoming message. An arrowhead without a bar shows the initiation message for the negotiation or the response message.





Symbol	Negotiation Condition	Definition
	Candidate to be Replenishing	The module has a stored item that is South of a replenishing module. It can turn to a replenishing module that enables a convoy.
	Willing to be Replenishing	An Occupied module is one step away from turning into a Replenishing module and will be a Replenishing module upon reception of a message.
	Candidate to be Occupied	A Replenishing module received a message from a Candidate to be Replenishing and has a chance to transfer its Replenishing state.
	Willing to be Occupied	The module has a replenishing item that received the first request from a module that is Willing to be Replenishing.

Table 3.2: Additional Negotiation Conditions

Convoy movement is a desired movement, in which Requested or Replenishing items can follow each other as a block and use only one empty cell for a movement of multiple items.

Otherwise, the items will use multiple empty cells for the movement of multiple Requested or Replenishing items.

There are additional negotiation conditions that are taken by the conveyor module during the negotiation for the assignment of Replenishing items and the encouragement of the convoy movement (see Table 3.2). These negotiation conditions will be understood more clearly in the following examples and rules for the related events.

A message is a package of information transmitted between modules electronically. An event can be described as the initiation of the negotiation for movement decisions. In the algorithm, there are sequential events for different purposes such as vertical movement of the requested item. Messages include several information. An important information included in the message is process number. A process number in the message points to a particular event. The next important piece of information is the included negotiation condition.

In the modeling architecture, message passing also mimics other communication protocols such as information flow between storage container and the conveyor module. It also reflects the possible interactions between the worker and the conveyor module. For instance, during picking from a location, the module should not convey to move the item in order to enable the picking. For this reason, the worker sends a message to the conveyor module that represents a button to lock the module during the pick. So, the module will not participate in the negotiation. In practice, the conveyor module can include two modes: active in negotiation and passive in negotiation. When the worker arrives for picking, she can bring the conveyor module to a passive mode to prevent possible negotiation and movements during the pick by pushing a button integrated to the conveyor module. After the worker is done with the picking, he/she can push again to bring the conveyor module to the active mode. Information transfer from the storage container is done in a similar manner. We mention other information included within the message in the respective events.

Tables throughout the next section denote the communication rules for the respective negotiation events. For a better understanding, the reader should follow a few guidelines

during the review of communication rules. These rules are only for the cases that require state change(s) and/or trigger of message(s). Combinations of states and messages that are not feasible or do not require any action are not shown in the tables. Also, only rules with the messages coming from the east are shown. The rules are axis symmetric and the same rules apply for the messages coming from the west. We classify the rules according to the type of action.

A module is responsible for the initial message to start the respective event. There is a target group that will respond to these initial messages, which are shown in the response message row. Some other modules are not directly related to the negotiation and they forward the messages back and forth. Toward the end of the negotiation, final state updates happen, which are shown in the final state change row.

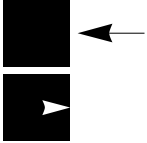
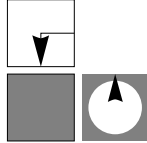
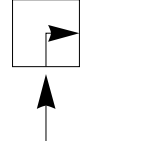
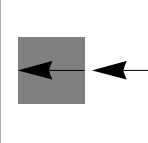
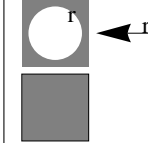
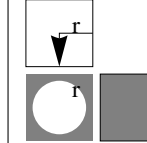
					
Response Message Type 1	Response Message Type 2	Forward Message Type 1	Forward Message Type 2	Final State Change Type 1	Final State Change Type 2

Table 3.3: Rule Examples

In the *response* messages, if the module receives a message from the east or west, we show the new state and the response to the south of the current state of the module (see Figure 3.3). If the module receives a message from the North or South, we show the new state and the response to the East of the current state of the module.

In the *forward* messages, the module passes the message in the direction shown with the arrow (see Figure 3.3). In the *final state change* rules, if the module receives a message from the east or west, we show the final state of the module to the south of the current state. If the module receives a message from the north or south, we show the final state of the module to the east of the current state. The messages in these events also include the

column number of the requested unit that has initiated the message and the column number of the occupied unit that has responded to the message (see Figure 3.3). Reader should follow these guidelines to review the communication rules in the latter sections.

3.1.3 Negotiation for the Assignment of the Replenishing Items

Before any negotiation for movement, there are events to determine which items will move north as Replenishing items and to encourage convoy movement. Since there are requested items moving downward to the pick face, congestion may occur if not managed properly. Our approach is to have some other items move upward to distribute the density evenly over the grid. There are two events that provide balance rules and turn some regular stored items into replenishing items. After the initial state update in each iteration, these two events initiate. For the balance rules, we use a departure row, target row and current row exchange.

Table 3.4 shows an example of Balance Rule 1. In the initiating message, the requested module includes its departure row information. Whenever, an empty module receives the message, it forwards the message to the south neighbor. If the south neighbor is an occupied module and has not already received a message for Balance Rule 1, it responds to the message and goes to a new condition called *candidate to be replenishing*. The response message includes the current row of the occupied module. The requested module will take the current row information upon reception of the message. So, the current row of the occupied module will be the new departure row of the requested module. The requested module will respond to this message including its old departure row. The response to the incoming message by the requested module is on a first come, first serve manner. In the example, the message of the module on the right comes first to the requested module. Therefore, the requested module takes its current row. Then, the occupied module will receive the old departure row information. It will turn into a replenishing module and will take the old departure row as a target row. The requested module will also respond to the message coming from the west

Instance of the Negotiation	Description
	<p>A requested unit is at its departure row and it initiates a message with a “process number” to make an occupied item move upward.</p>
	<p>Stored items forward the messages to their neighbors.</p>
	<p>On the left of the initiated message, another occupied unit forwards the message to the west. On the right of the initiated message, an empty unit is found and it forwards the message to the south based on the rule index number.</p>
	<p>On the left, an empty unit is found and it forwards the message to the south. On the right, an occupied unit receives and responds to the message.</p>
	<p>On the left, an occupied unit receives and responds to the message. On the right, an empty unit forwards the message to the east based on the column information of the message.</p>
	<p>On the left, an empty unit forwards the message to the east. On the right, an occupied unit forwards the message to the west.</p>
	<p>The requested unit received the respond message, it takes the row number information and sends its departure row with a new process number.</p>
	<p>On the left, the occupied unit forwards the message to the requested unit. But, the requested unit will respond to this message with new departure row information.</p>
	<p>The empty unit forwards the message to the south. The requested item responds to the message.</p>
	<p>The occupied unit received the message and turned to a replenishing unit with a target row “2”. The requested unit’s departure row is now “3”.</p>

Table 3.4: First Ten Steps of Balance Rule 1 with the In advance Movement

Instance of the Negotiation	Description
	The occupied unit forwards the response message to the west.
	The empty unit forwards the message to the south.
	The occupied unit receives the message that includes the same row number with its current row and it stays in the occupied state.

Table 3.5: Last Three Steps of Balance Rule 1 with the In advance Movement

side (see Table 3.5). It will send the updated departure row information to the occupied module and this will match with the current row of the occupied module. So, this unit will stay as stored.

Table 3.6 shows the communication rules for Balance Rule 1. See the guidelines at the end of the Symbolism and Representation section to review the rules properly. In the *final state change* rules, if the *willing to be replenishing* module receives a row information smaller than its current row, it becomes a replenishing state because it understands that it should arrive to the target row. If the received row information is the same as its row information it stays occupied.

Table 3.7 presents an example for Balance Rule 2. In this case, the requested unit could not exchange its departure row in Balance Rule 1. In Balance Rule 1, the initiated message should arrive to an empty unit and an occupied unit should be present to the south. There can be cases without this combination. Balance Rule 2 has a greater chance to find an occupied unit that can turn into a replenishing unit since it looks for an occupied unit in the same row. When the requested unit receives the response message, it takes the current

	Requested	Stored	Replenishing	Empty
Initiation Message(s)				
Response Message(s)				
Forward Message(s)				
Final State Change(s)				

Table 3.6: Communication Rules for Balance Rule 1

row of the occupied unit as the departure row and the requested unit sends its old departure row to the occupied unit. The occupied unit takes this information as a target row and transitions into a replenishing unit. The requested unit responds to the message in a first come, first serve manner. The message from the left comes first and the requested unit takes its row information. The other message from the east gets the new row information and the module stays in the occupied state.

Table 3.8 lists the communication rules for Balance Rule 2. The requested module initializes the message passing by sending a message to the east and west. An occupied unit will respond to it and other units will just forward the message with the information included. In the *final state change* rules, if the *willing to be replenishing* receives a row information smaller than its current row, it becomes a replenishing state because it understands that



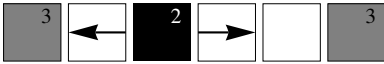







Instance of the Negotiation	Description
1 	Initially, Balance Rule 1 did not work for the requested unit and it has already moved while having a departure row “2”. Its current row is “3”.
2 	The requested unit initiates message passing to turn a regular occupied into replenishing unit. So, it will keep the occupancy of the row in balance.
3 	Empty cells forward the messages to the west and east.
4 	On the left, an occupied unit responds to the message with the column number information. On the right, another empty cell forwards the message to the east.
5 	On the left, empty cell forwards back the message to the east. On the right, an occupied unit is found and it responds to the message with the column number information.
6 	The requested unit receives the message, it takes the column number “3” and sends its departure row information with a message.
7 	On the left, an empty cell forwards the message to the west. The requested unit receives the forward message, and since it has already taken another unit’s column number information, it responds with the new departure row information.
8 	The occupied unit receives the departure row information and it becomes a replenishing unit with the target row of “2”.
9 	The empty cell forwards the response from the requested unit to the east.
10 	The empty cell forwards the response message. Incoming message’s row information matches with the unit’s current row. So, the module stays in the occupied state. The last two steps are combined for brevity.

Table 3.7: Steps of Balance Rule 2 with the Letter Movement

	Requested	Stored	Replenishing	Empty
Initial Message(s)				
Response Message(s)				
Forward Message(s)				
Final State Change(s)				

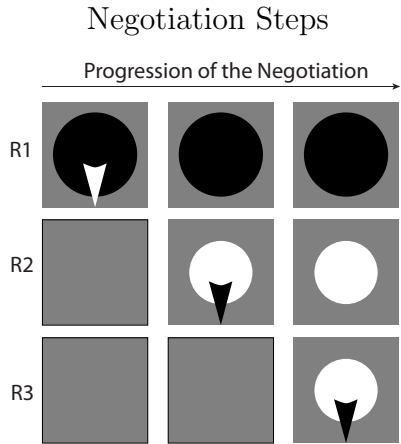
Table 3.8: Communication Rules for Balance Rule 2

it should move to the target row. If the received row information is the same as its row information it stays occupied.

3.1.4 Assignment of the Replenishing Items for the Convoy Movement

After the assignment of replenishing items, the system has some balancing items and the requested items do not have to go back to their initial row to balance the empty cells. However, the replenishing items mostly require one empty cell and there is no convoy movement. To achieve such a movement, some occupied items that are south of a replenishing item can take the responsibility of replenishing items and can turn into replenishing items themselves. For this purpose, we have defined three consecutive events that are using negotiation positions as a result of the previous event.

The first event for the convoy movement determines the *candidate to be replenishing* units. Table 3.9 shows an example of the first convoy movement event. The occupied units can turn into replenishing units and they will not require extra space to move upward.



Description

Index numbers on the left show the row numbers of the units. In the first column, a replenishing unit sends a message to the south neighbor. The south neighbor is an occupied module and it turns to *candidate to be replenishing* upon receipt of the message. In the second time step, it sends the same message to the south neighbor. If there is an occupied module, it also turns to *candidate to be replenishing*. The message passing goes on until a unit is found other than an occupied module.

Table 3.9: Steps of the First Event of the Convoy Movement

	Replenishing	Occupied
Initial Message(s)		
Forward Message(s)		

Table 3.10: Communication Rules for the First Event of the Convoy Movement

First event of the convoy movement require only two rules (see Table 3.10). A replenishing unit will initiate the message and if an occupied unit is present to the south, it will update its condition and forward the message.

Table 3.11 shows an example for the second convoy movement event. After the first event, the replenishing units initiate the second event for the convoy movement. They will send a message to the east and west. If a *candidate to be replenishing* is present in the row, it will respond to it. The replenishing module turns into a *candidate to be occupied* upon reception of the response message. Message passing rules for the second convoy movement event are shown in Table 3.12.







Instance of the Negotiation	Description
1 	The replenishing units initiate a message to the east and west to discover options to be a replenishing unit instead of itself.
2 	The occupied units forward the message of the replenishing unit in the middle. On the right, a <i>candidate to be replenishing</i> receives the message, responds to it and goes to a negotiation position, <i>willing to be replenishing</i> .
3 	An empty cell forwards the message of the replenishing unit in the middle. The replenishing unit on the right receives the response message and turns into a <i>candidate to be occupied</i> .
4 	On the left, the module responds to the message and turns to a <i>willing to be replenishing</i> condition.
5 	The empty cell forwards the message to the west.
6 	The occupied unit forwards the message to the replenishing unit and it takes the negotiating position of <i>candidate to be occupied</i> . We have omitted one step here for brevity.

Table 3.11: Steps of the Second Event of the Convoy Movement

We provide an example for the third event of the convoy movement (see Table 3.13). An important point is the third convoy movement event's cyclic behavior. After an occupied module turns into a replenishing module it will send a message to the south. If there is a *candidate to be replenishing* to the south, it will send another message to reinitiate the third event of the convoy movement. This is necessary to enable convoy movement of multiple replenishing items.

Table 3.14 shows the communication rules for the third event of the convoy movement. The replenishing unit will initiate the event by sending a message to the south. After getting








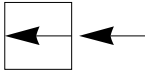
	Replenishing	Occupied	Requested	Empty
Initial Message(s)				
Response Message(s)	 	 		
Forward Message(s)				

Table 3.12: Communication Rules for the Second Event of the Tandem Movement

this trigger message, a *willing to be replenishing* unit will send a message to the east and west. The negotiation will happen between a *candidate to be occupied* and *willing to be replenishing* unit. The occupied, empty and requested units will forward the incoming messages. The message includes the target row of the replenishing unit and the current row of the occupied unit. In the final state change, if the replenishing unit sends its target row to an occupied unit, it will take the responsibility and turn into a replenishing unit. After the occupied unit becomes a replenishing unit, it will send its current row to the sender of the message. The receiver of this message will turn into an occupied unit. The replenishing unit will respond to later messages with its target row. The receiver of this message will stay in the replenishing state.

One might wonder about the possibility of performing the replenishing units assignment for the convoy movement with a single event or a shorter negotiation. In the first event, *candidate to be replenishing* can directly initiate a message to look for replenishing units to take over their responsibility. However, this does not guarantee a convoy, because other replenishing units can exchange with the occupied units, which results in disruption of the convoy. In the second event, we observe a similar complication. We do not know which replenishing units transfer their replenishing state. So, the second event determines

Instance of the Negotiation	Description
	<p>A replenishing module initiates a message to the south neighbor. There is a <i>willing to be replenishing</i> unit in the south.</p>
	<p>The unit with the <i>willing to be replenishing</i> condition sends message to both east and west neighbors. Messages will look for <i>candidate to be occupied</i> units.</p>
	<p>The occupied units forward the received messages.</p>
	<p>On the left, a <i>candidate to be occupied</i> unit receives the message, turns to the <i>willing to be occupied</i> condition and responded to the message.</p>
	<p>On the left, the occupied unit responds back the message. On the right, a <i>candidate to be occupied</i> unit receives the message, turns to a <i>willing to be occupied</i>, and responds to the message.</p>
	<p>The <i>willing to be replenishing</i> unit turns into a replenishing unit, responds to the message, and also initiates another negotiation by sending a message to the south neighbor.</p>
	<p>The occupied units forward the messages to the west.</p>
	<p>The unit with the <i>willing to be occupied</i> condition turns to an occupied unit upon reception of the message. The replenishing unit responds to the message coming from the east including its new state.</p>
	<p>The occupied unit forwards the message.</p>
	<p>The unit with the <i>willing to be occupied</i> condition stays as a replenishing unit. We have combined two steps here for brevity.</p>

Table 3.13: Steps of the Third Event of the Convoy Movement

	Replenishing	Occupied	Requested	Empty
Initial Message(s)				
Response Message(s)				
Forward Message(s)				
Forward Message(s)				
Final State Change(s)				

Table 3.14: Communication Rules for the Third Event of the Convoy Movement

the replenishing units that have a chance to transfer its replenishing responsibility. The replenishing modules without a chance to be an occupied module will initiate the third event and they will lead the convoy.

3.1.5 Negotiation Process for Movement Decisions

After the events for the assignment of the replenishing units, and the convoy movement, negotiation events execute for the movement decisions of the modules. At first, north-south negotiation events happen for the vertical movement decisions of the modules. So, the vertical movements have a priority over the horizontal since their decision process executes first. The logic of these events are based on the gridstore system [Gue et al., 2013]. Furthermore, the replenishing units have separate events from the requested units. The replenishing units have a priority over the requested units by taking the movement decision before the















	Request East or West		Requested
	Request East		Replenishing
	Request West		Commit North
	Willing East		Commit South
	Willing West		Commit East
	Available Empty		Commit West
	Available Occupied		No movement

Table 3.15: Negotiation conditions for the movement decisions

requested units. By having the priority for the replenishing units, we enable movement of the balancing items before the Requested items, which tends to avoid deadlock. Table 3.15 represents the negotiation conditions used in the events for the movement decisions.

3.1.6 North-South Negotiation for Vertical Movement

In the north-south negotiation, the replenishing and the requested units initiate the negotiation. The requested units send a message to their south neighbors and the replenishing units send messages to north neighbors to check the availability of the modules.

Table 3.16 shows two examples of the north-south negotiation for the upward movement of replenishing units. The left side of the Table 3.16 results in a *commit north* decision. So, in the next iteration, three units will convey upward and two replenishing units will move to the north. Downward vertical movement of the requested units take place in a similar manner.

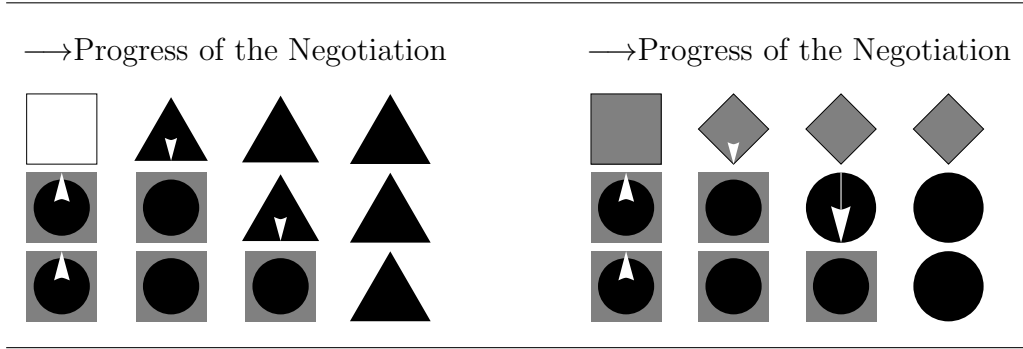


Table 3.16: Example 1 for North-South Negotiation

The right side of the Table 3.16 shows an example of the north-south negotiation for the replenishing units. In this case, the north neighbor is occupied and it needs go east or west to allow passage of the replenishing units. For this reason, the occupied unit goes to a new negotiation condition called *request east or west*.

Table 3.17 shows the message passing rules for the north-south negotiation.

East-West Negotiation for Horizontal Movement

When there are obstacles preventing the vertical movement of requested and replenishing units, they should move east or west. In the north-south negotiation, the requested and replenishing units look for an empty cell to send their storage container. If there is an occupied module, it receives the message that says it should become available.

Table 3.18 shows an example east-west negotiation. An occupied unit that knows it should move east or west turns into a new condition called *request east or west*. This unit sends a message to both east and west neighbors. If there is an occupied unit it forwards the message until an empty cell receives the message. If a module, other than the occupied or empty state, receives the message, its response informs that there are no empty cells. If an occupied unit on the edge receives the message it also informs that there are no empty cells. If an empty cell receives the *request*, it goes to a new condition called *willing* (east in this instance) and it responds to the message. When the occupied unit with *request east or*

	Requested	Occupied	Replenishing	Empty
Initiation Message(s)				
Response Message(s)		 		
Forward Message(s)	 		 	

Table 3.17: Communication Rules for North-South Negotiation

west condition, which initiated the message receives the response, it goes to a new condition called *commit* (east in this case) and sends a final message that turns other units into *commit* for the movement. Rules for the east-west negotiation are listed in the Table 3.19.

After documenting all the rules and negotiation steps, we can revisit the steps of the algorithm. If we take a close look in Figure 3.7, in step 1 row number 2, we can see that a replenishing item is assigned with the balance rule-1. In step 1 row number 9, a replenishing item is assigned with the balance rule-2. In step 4 rows 4 and 5, replenishing items has formed a convoy. The replenishing item in row 4 is assigned by the requested item in row 3.

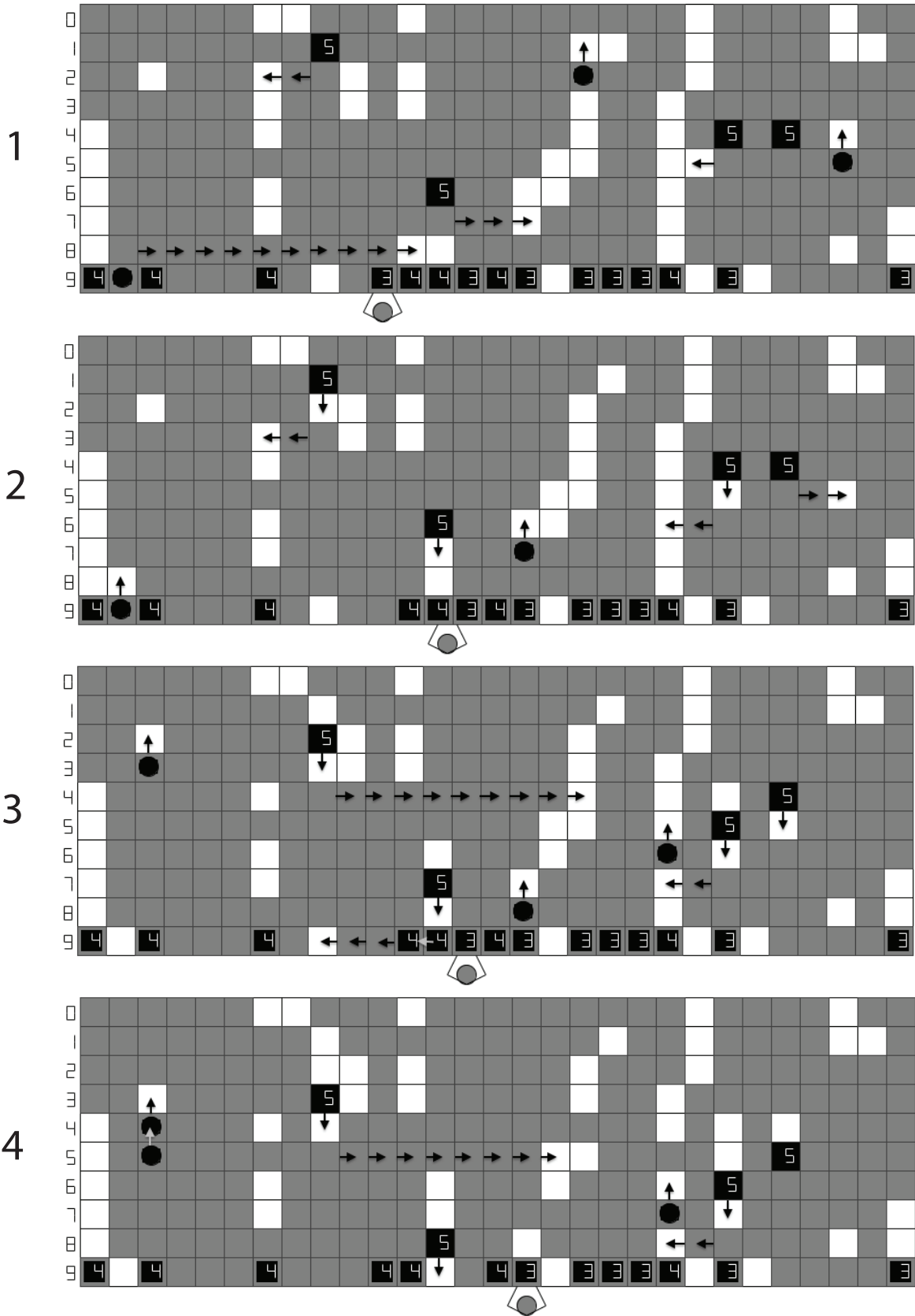


Figure 3.7: Example steps for GridPick

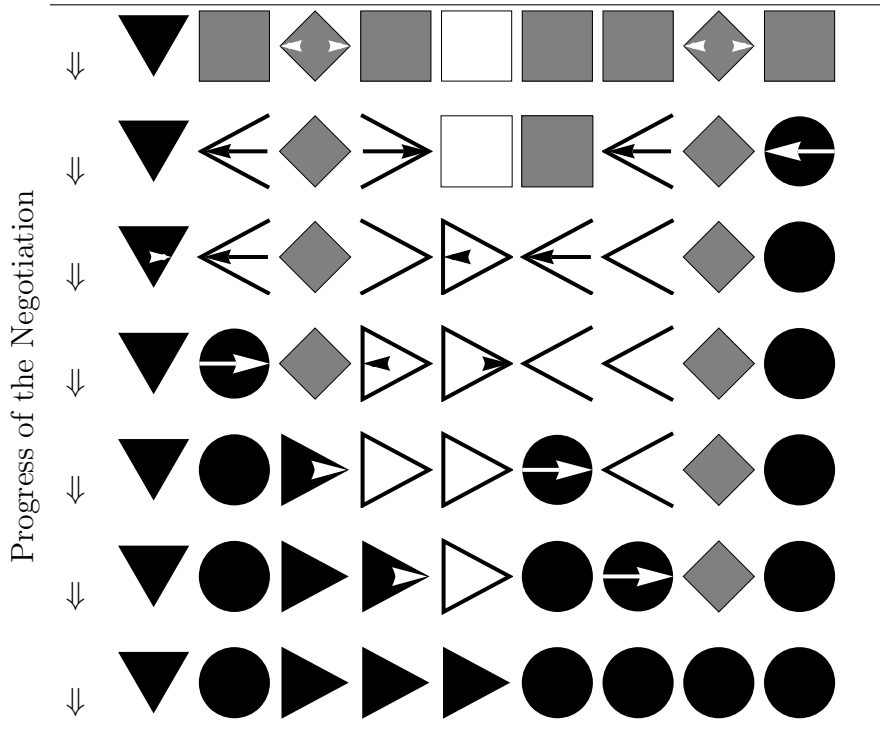


Table 3.18: Example for East-West Negotiation

The replenishing item in row 5 was at another location at the same row, but it is transferred to form a convoy.

Figure 3.7 shows steps of the GridPick algorithm in a 10×29 grid. At the beginning, the worker picks the 3rd order and items of the 4th order are moving toward the pick face. At the 8th step, all items of the 4th order have arrived and 5th order is released. For a better understanding, see the video (<http://www.youtube.com/watch?v=Yiv12N10h-I&feature=relmfu>).

	Request E or W	Occupied	Empty	Request	Willing
Initiation Message(s)					
Response Message(s)					
Forward Message(s)					

Table 3.19: Communication Rules for East-West Negotiation

3.2 Considerations on the Application of GridPick

There are aspects worth mentioning for the application of GridPick as an operating order picking system. GridPick should be able to integrate with the picker and rest of the distribution center. An entry on the request list is called an order line, which consists of the item and the quantity requested. Because carton picking and each picking are generally separate order picking operations, an order with both cartons and eaches should be handled separately. The Warehouse Management System checks and divides the order into pick-lines (for cartons and eaches) if necessary. These data management activities are also required for the operations of GridPick.

Pick-lines are detailed descriptions of the order: location of the order, product identity (SKU number), quantity, and units of measure. A pick-line may include more than one pick, which means the picker should reach into the location several times to complete the pick. Pick-lines are organized into pick-lists to enable multiple picks of the worker. Pick-lines can be reorganized to have the picks sequenced for efficient and reduced travel. A pick-list can be a sheet of paper, RF, printed shipping labels as well as pick by light or voice transmission. As with a flow rack configuration, all these components of instructions are required for a worker operating on the GridPick system. We will discuss two order picking interfaces which are appropriate for the GridPick system.

3.2.1 Pick to Light Systems

Pick to light order picking systems have light indicators and digital screens to direct the worker to the correct pick. These devices are generally mounted to flow rack or other storage locations to indicate the requests. When the light indicator turns on, it draws the attention of the worker. The picker knows that there is a pick required to be processed in the corresponding location. The required quantity is also displayed on the digital screen. So, the worker picks the shown quantity and presses the lighted button to indicate that the pick has been processed (see Figure 3.8).



Figure 3.8: A Representation of the Pick to Light Systems

For GridPick, in addition to these functions, there is a need for another press of the button at the start of the picking process. Therefore, the unit modules will know that there is a “being picked” item and they should not move until the process is done. Also, there are

occasional horizontal moves to allow other requested item to reach the pick face. The unit modules should be able to deactivate and activate the light indicator and the digital screen considering the status of the storage container. An arrow indicator showing the direction of the current movement would also be helpful to direct the picker. So, the worker would know the item is moving to another location and that he should wait for this movement. In this case, a red light indicator for both the origin and destination would be turned off and movement arrows will show up. After the movement is done, the red indicator light will display in the current requested location.

3.2.2 Pick by Voice Systems

Order pickers can be directed via voice messages that tell the worker about the order details (locations, quantity, SKU number). In the pick by voice system, the picker generally has a headphone to receive the requests and a microphone to respond to the messages. A main benefit of having a pick by voice system is that the picker does not have to carry a paper list or computer screen. It also saves some time because it does not require looking at sheets or screens (see Figure 3.9).



Figure 3.9: Pick by Voice Systems

For GridPick with a pick by voice system, the picker must know the column location of the requested items. Therefore, there should be column numbers indicated on the pick face so that the voice instructions can direct the picker to a column by its number. The instructions should also include other information (quantity, SKU number). The system should also

report any sudden changes in the column number due to an occasional horizontal movement on the pick face.

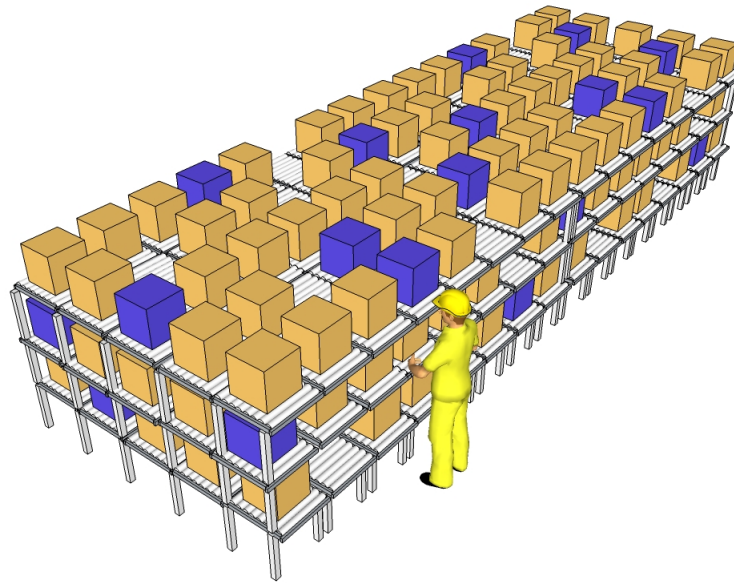


Figure 3.10: GridPick with Several Levels

Similar to a flow rack configuration, GridPick for totes could be built with several levels (see Figure 3.10). This would increase the SKU density and consequently the pick density. For a 3-level configuration, there is the possibility of having 3 requested items instead of 1 by having them stored vertically.

Since we have a restriction of order size up to the number of columns, the warehouse management system should divide orders into parts if necessary. Also, two consecutive orders cannot be larger than $2c - 1$, where c is the number of columns. Therefore, sequencing of the orders may be necessary. Restocking of the GridPick system could be done during off peak times or could be integrated into picking operations directly.

3.3 Model Representation

For the modeling of GridPick, we have used, fundamentally, an agent based modeling approach. In agent based models, each agent executes a step function, which is mainly a set of individual based rules. In our model, we create an object and all unit modules are

instances of this object. Therefore, they execute the same algorithm. In each time step, agents execute the same logic (see Figure 3.11). So, agents do not go to the next time step, and all of them are updated in a synchronous manner in each iteration.

Since computers execute instructions serially, agent instructions are executed sequentially. To prevent any bias, a random sequential order is determined and agents are executed in this order. In our case, random execution of parallel events is handled automatically by the AnyLogic software. All agents update their states before an agent executes the next event. This approach introduces a synchronous behavior. This is also called a pseudo distributed system, and it reflects the behavior of a parallel system. Use of parallel machines in agent based models is for the purpose of increased performance and a better computational time. They are not intended to reflect the dynamics of the parallel system.

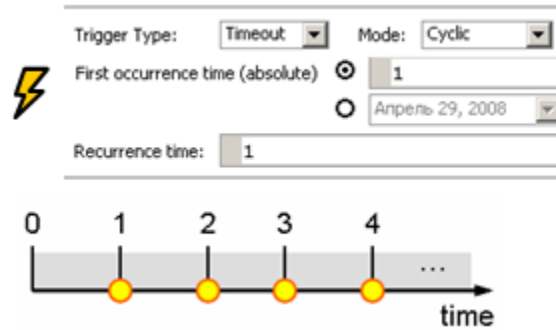


Figure 3.11: Series of Synchronous Events in Each Time Step (Source: Anylogic help files).

In each iteration of GridPick, there are cyclic events executed for different purposes of the negotiation. Each event corresponds to a specific negotiation. Each agent is scheduled to execute at the same time, so they do not go into the next event before all agents are done with the event. If an event includes negotiation, process time of the event depends on the length of the negotiations, which is not known beforehand. There are also events not including any negotiation and the process time is not negotiation dependent.

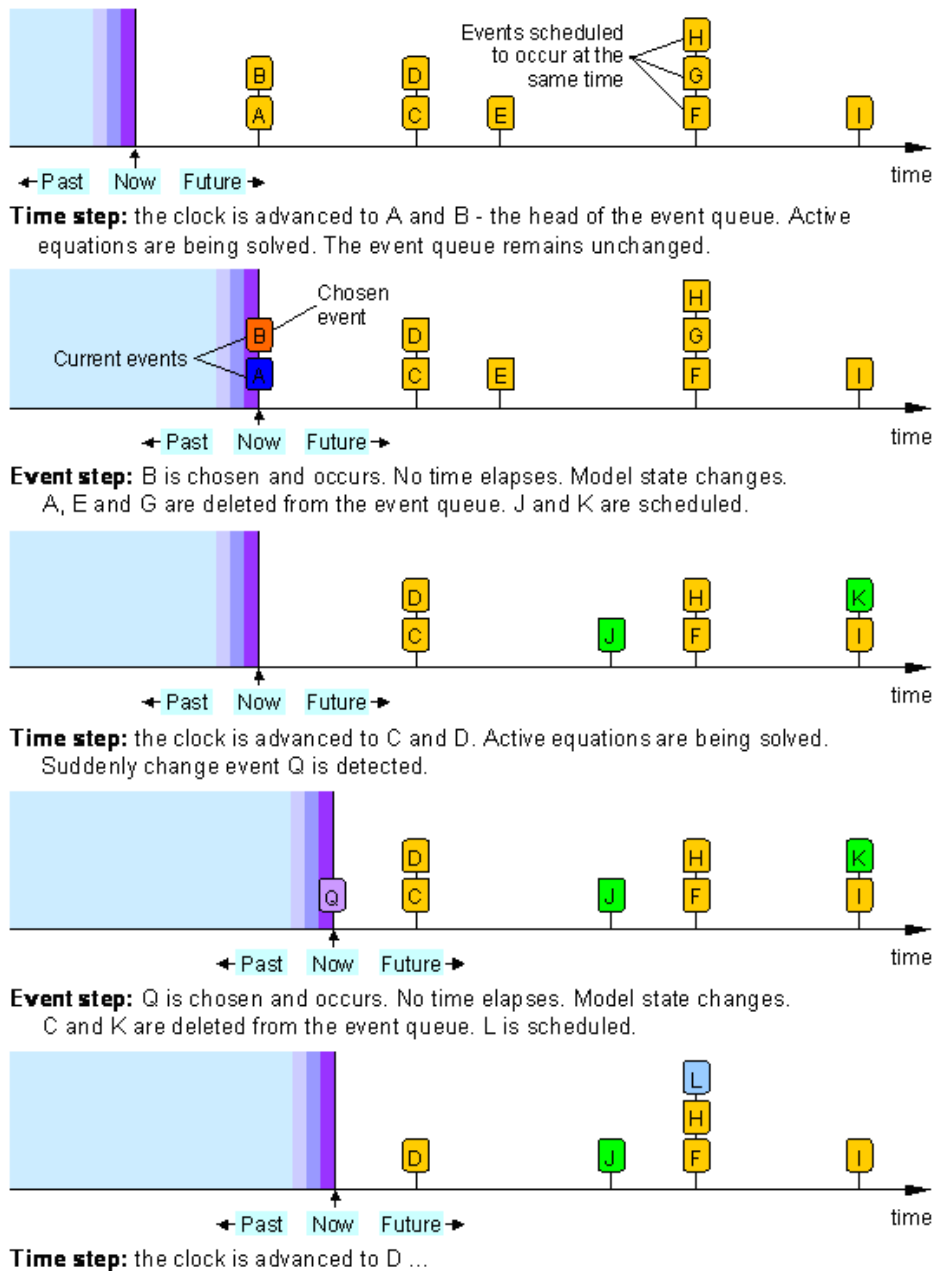


Figure 3.12: Each event can take a different process time depending on the length of negotiation (Source: Anylogic help files).

3.3.1 Message Passing

Message transfer time depends on the size of the message. In our model, a message includes a java class object. Therefore, it includes related set() and get() functions and comments, which increases the size. Actually, we only pass a few values to another unit

module. According to size calculations, message sizes can be up to 3,000 bytes. The transfer time of a message will take only up to 100 microseconds, which is 0.1 milliseconds [Dongarra and Dunigan, 1997].

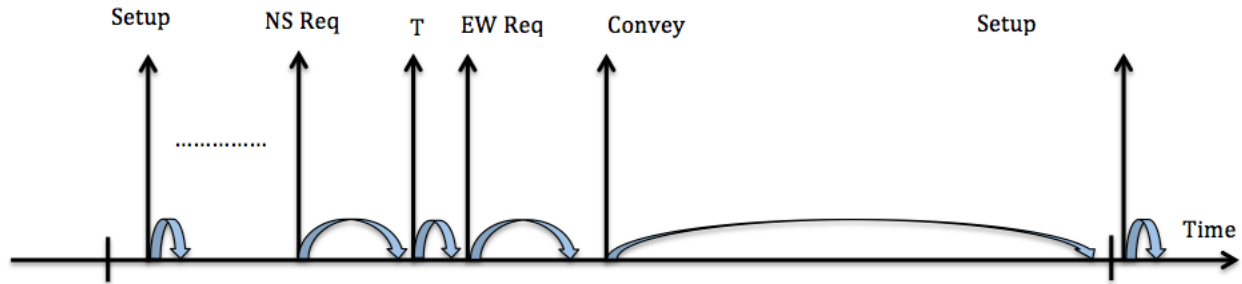


Figure 3.13: Cyclical Events in Each Time Step

At the startup of the model, there are several initial setup functions to form the grid and adjust the locations of the storage containers. After that there are synchronous cyclic events that is executed by each unit module. Synchronization is achieved by executing all subevents for each event before a module executes the next event. The model runs in continuous time with a series of events in each cycle. We specify a frequency for the iteration that is sufficient to run all negotiation events and for the movement of items. For simplicity, we have used 1 second as an iteration time or step length.

At first, there are two events called *setupcondition* and *sendcondition* for the detection of current state of the unit module. In the *setupcondition* event, unit modules set their state to empty. In the *sendcondition* event, the state of the stored items is received by the unit module, and they update their state (requested, replenishing, etc.) as well as other information such as target row and negotiation condition.

After that there are two balance rule events called *receive* and *climbreplenish*, which are executed for the assignment of balancing item. Unit modules are required to finish the negotiation before the next event begins. To enable tandem movement encouragement, there are three additional events.

To prioritize the movements of replenishing items, we execute the North-South negotiation and East-West negotiation events for replenishing items separately and before the negotiation events of requested items. There are also transition events between negotiation events, which are for transferring the current condition of the module into the format of the next negotiation event. To prevent confusion, we use separate variables and numbers for the negotiation conditions in each event. After all negotiation events and state assignments, all modules execute the *convey* phase, and they move in the appropriate direction depending on the result of the negotiations.

There are a few sources of randomness in the model. In the initial setup, orientation of the empty cells in each row is determined randomly. A major random component is the order release policy. All items are equally likely to be requested, which may change the movement during the execution. Therefore, when there is a request, any occupied item can be requested. Also, expected order size is generated from a Poisson distribution. Furthermore, during the initiation of the events, sending messages are serially executed. If the module is required to send two messages to East and West, it chooses one side randomly to send first.

In a physical system, controllers have timers to keep the clock time and determine the timings of the event executions. Occasionally, timers may drift, which will prevent accurate execution of the logic. This difficulty could be addressed by updating the clock times of controllers periodically. For instance, during each order release, a broadcast or message to the modules including the current time can help keeping the time updated and accurate. Depending on the severity of the time drift, the update could be more or less frequent.

3.3.2 Determining Buffer Lengths for Cyclic Events

The model includes a series of events, which execute sequentially and synchronously. When an event starts, there is a process time for the negotiation. One crucial consideration is to avoid overlapping negotiation processes. Therefore, the time between initiation of the events should be large enough to allow sufficient time for unit modules to finish the

execution and negotiation before the next event initiates (see Figure 3.14). Some events do not include negotiation. They are mostly for updating the states and transferring the negotiation conditions to the next event. These events, which typically take less than one millisecond, not include variability due to the negotiation.

We have recorded the finish time of the events that have negotiation during their process. So, we can assign timings of the events considering the length of the negotiation. We have obtained 50 replications for the negotiation length analysis.

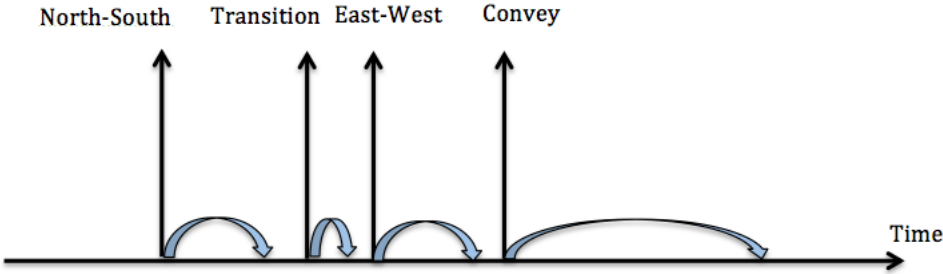


Figure 3.14: Representation of events executing cyclically.

Bonferroni Inequality

Since we have multiple events that should be finished within a specified interval, we would like to obtain an overall confidence for the respective events.

When there is a $100(1 - \alpha_s)$ percent confidence interval represented by I_s for the measure of μ_s , the probability that all k confidence intervals simultaneously contain their true measures satisfies

$$P(\mu \in I_s \text{ for all } s = 1, 2, 3, \dots, k) \geq 1 - \sum_{s=1}^k \alpha_s$$

This is known as the Bonferroni inequality. So, if we have multiple performance measures, the sum of α values provides the overall significance level. This means the confidence level will decrease with an increasing number of measures. One solution to this problem is setting the α values at a high level to obtain a sufficient confidence level. We have 99.5%

confidence intervals for each event, which results in at least 98% overall confidence level for the series of events.

Since we have a 98% confidence interval, we can say that we are 98% confident that the true value of the parameters lies in the determined confidence interval. If an event’s negotiation still does not finish on time, it will appear in the transfer event, because we have transfer events that translate the current negotiation condition into the new event’s related variables. We can easily detect if the negotiation does not finish by printing an error message. Furthermore, in the transfer event, wrong negotiation conditions can be translated as no movement, which will prevent a failure of the system.

Variable	N	Mean	StDev	SE Mean	99.5% CI	UCL
NS-rep	50	1.2909	0.5331	0.0719	(1.0805, 1.5013)	1.5066
EW-rep	50	1.926	1.226	0.167	(1.437, 2.415)	2.427
NS-req	50	1.491	0.791	0.107	(1.179, 1.803)	1.812
EW-req	50	2.455	1.399	0.189	(1.903, 3.007)	3.022

Table 3.20: CIs for the Negotiations (Units are in milliseconds)

Initially, we have determined large inter-startup times for the events. These times are determined by considering the number of columns and number of rows. Because the North-South negotiation happens only between North and South neighbors with two waves of messages, and considering that a message transfer time takes 0.1 milliseconds [Dongarra and Dunigan, 1997], “ $2 \times \text{number of rows} \times \text{transfer time}$ ” provides a long inter-startup time. This means that a North-South negotiation can take at most this length. For a configuration with 10 rows, North-South negotiation takes at most 2 milliseconds ($2 \times 10 \times 0.1$). For an East-West negotiation, there are three waves of messages for the event. For a configuration with 30 columns, the E-W negotiation can take at most 9 milliseconds ($3 \times 30 \times 0.1$). From the confidence intervals, we see that this is very conservative assumption, especially for the

East-West negotiation (see Table 3.20). We can determine the inter-startup times by having upper control limits for each negotiation length. We obtain upper control limits by adding three standard errors to the mean.

3.4 Statistical Analysis for Steady-State Parameters

3.4.1 Determining the Warmup Period

Observations at the startup of the simulation may not represent the steady state behavior of the model. To deal with this problem, generally we should “warm up” the model by removing initial data. A common approach to determine warmup period is Welch’s method. In this approach, at first a number of replications is obtained for the simulation. Then the average of the replications for each observation is acquired. To have a smooth the oscillations, we get a moving average with a window size w (such that $w \leq m/2$), where m is the simulation length.

For the throughput, results are low initially, and waiting times are longer at the startup of the simulation. Since initial bias is very short at the beginning of the simulation, by inspecting the plots, we determine that a warmup period of “200” is sufficiently large. So, the Welch’s method helps us determining the warmup period by inspecting the moving averages of data. We have used a window size of 10.

At the startup, due to the waiting of the first order, throughput is small. The throughput increases and this initial transient stabilizes quickly (see Figure 3.15). For the next order, items are already arrived to the pick face, and the picker does not wait for them.

Waiting time is large at the startup because the picker waits for the item for certain. While the picker is processing the current order, next order is released and they are for picking while the worker is done with the current order (see Figure3.16). A little bit of oscillation in the data can be removed by having larger window size for the moving average. Because it does not show any pattern and it is around the stabilized trend, a warmup period of “200” is decided to be sufficient.

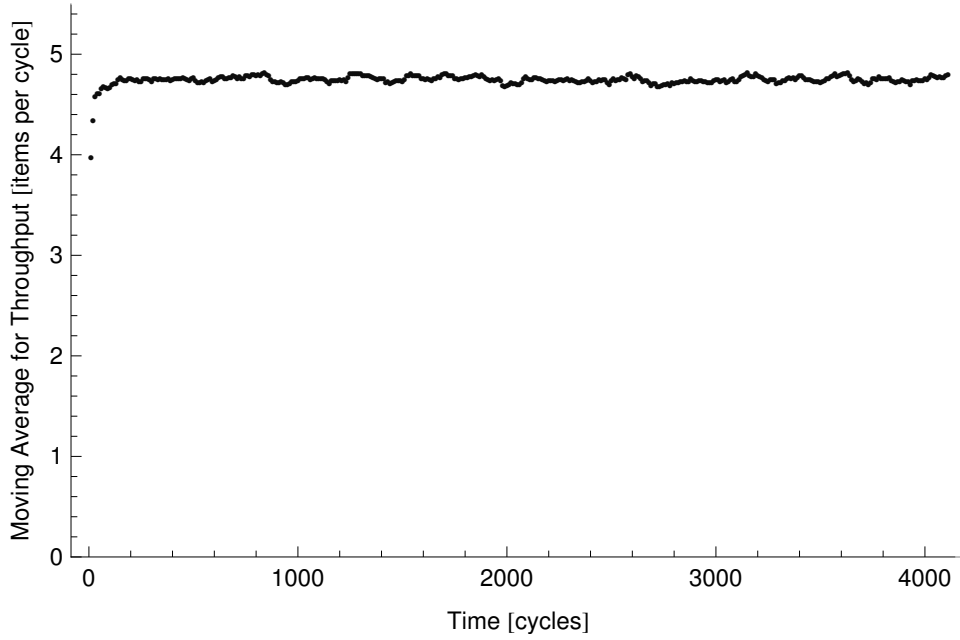


Figure 3.15: Initial Transient for Throughput

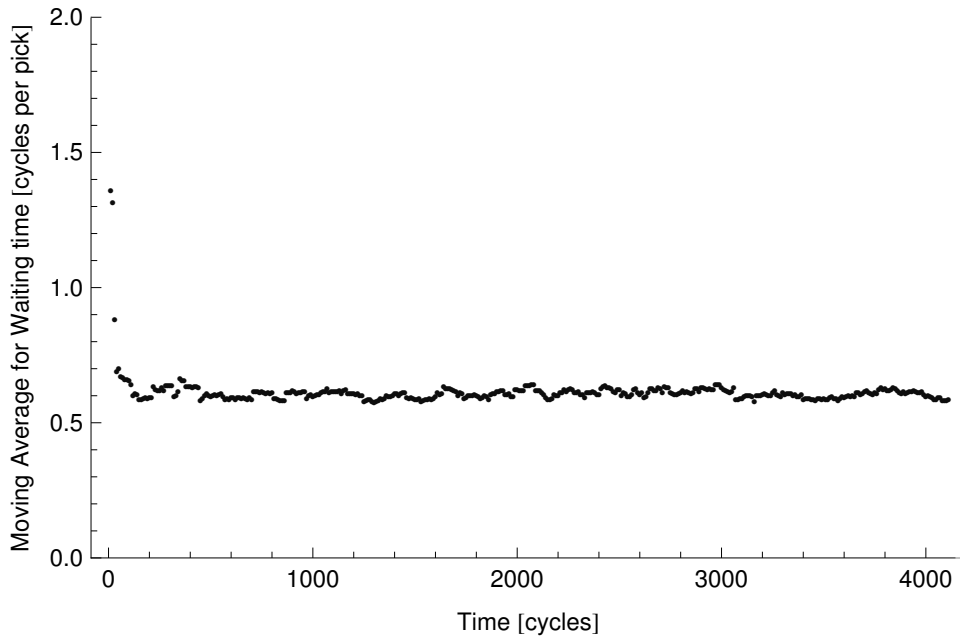


Figure 3.16: Initial Transient for Waiting Time

Walking time measure is not effected from the initial transient (see Figure 3.17). It is similar for all time instances.

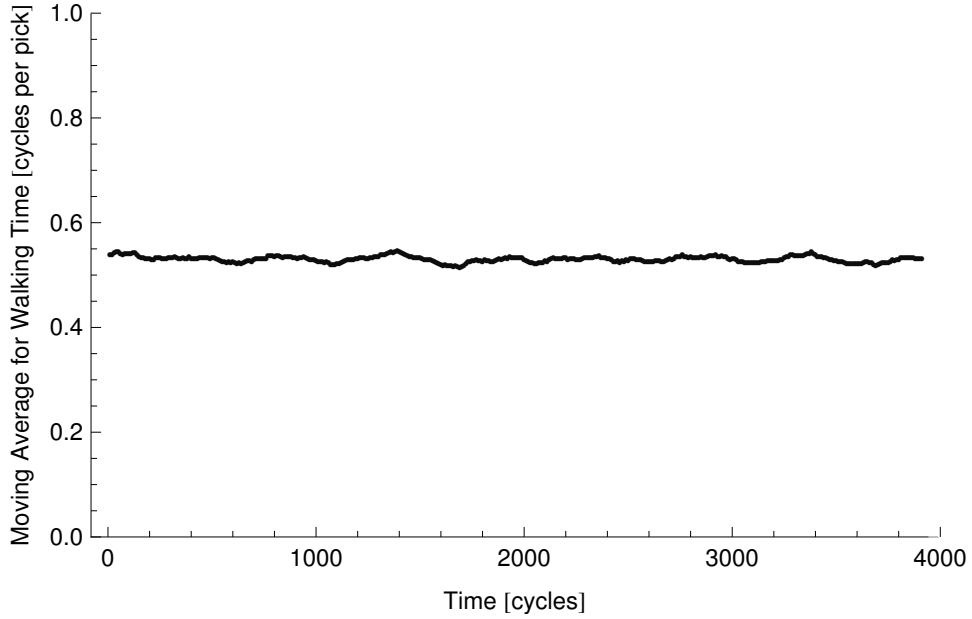


Figure 3.17: Initial Transient for Walking Time

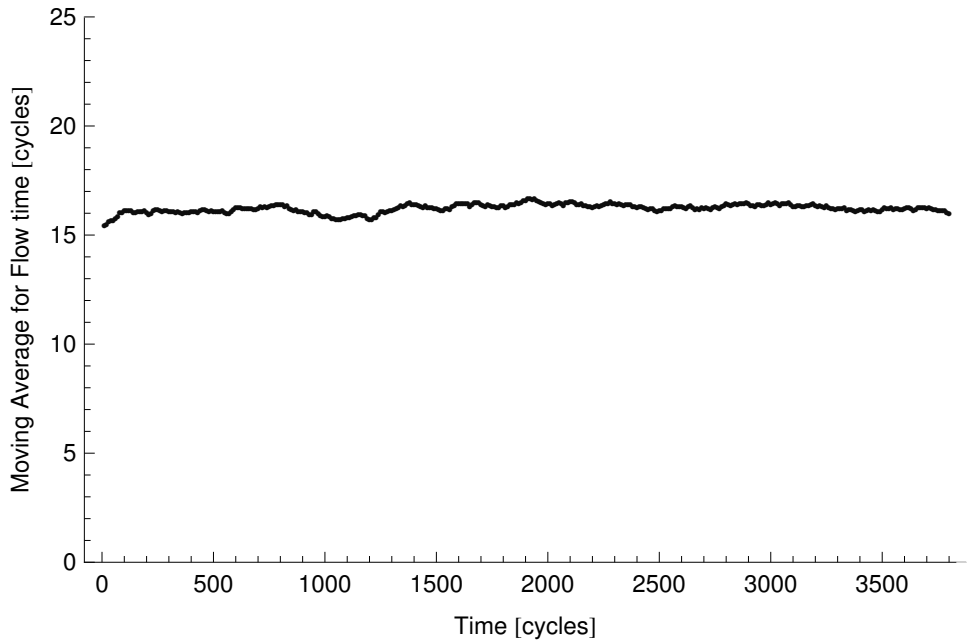


Figure 3.18: Initial Transient for Flow Time

3.4.2 Determining the Number of Replications

We have four performance measures for the simulation analysis (throughput, walking time, waiting time, and flow time). For this reason, we have considered bonferroni inequality

to acquire an overall confidence interval. Four 99 % confidence intervals provides at least 96 % overall confidence.

	Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	% Mean
Thr.	$k3 - 14$	30	0.473242	0.002792	0.00051	(0.471837, 0.474647)	0.001405	0.3%
Walking	$k3 - 14$	30	0.6707	0.01153	0.0021	(0.66490, 0.67651)	0.00581	0.87%
Waiting	$k3 - 14$	30	0.69261	0.0116	0.00212	(0.68677, 0.69844)	0.00583	0.84%
Flow time	$k3 - 14$	30	15.0662	0.2703	0.0494	(14.9302, 15.2023)	0.1361	0.9%

Table 3.21: CIs for the Four Performance Measures

Since halfwidths are not larger than 1 % of the mean for all confidence intervals, a value of “30” is sufficient for the number of replications (see Table 3.21). This result is obtained for a $k = 3$ configuration with an expected order size of 14. We have obtained confidence intervals for different k values and expected order sizes (small and large). They are included in the Appendix. These confidence intervals show similar results, halfwidth is approximately 1% of the mean.

3.4.3 Paired t-Tests for the Configuration Comparisons

In the next section, we have detailed performance analysis of the one sided GridPick. To statistically compare the configurations, we have used the paired t -test. We do not know the population’s variance and mean. Also, all other conditions are same for the simulations other than the comparison criteria.

Since the number of pairs is 30, which is the number of replications, degree of freedom is 29 for our configurations. Corresponding t table value for $\alpha = 0.05$ is 2.045. Therefore, we compare the t values with 2.045. Larger values indicates that configurations are significantly

different at the 5 % significance level. Values smaller than 2.045 show that there is no significant difference.

We can also compare p values with the significance level α to understand the result of the hypothesis testing. P values larger than $\alpha = 0.05$ implies that there is no significant difference. P values smaller than 0.05 shows a significant difference between configurations. We can also understand the case from the confidence intervals. If the confidence interval includes “0”, then there is no significant difference. Otherwise, there is a statistically significant difference.

3.5 Performance Analysis

Our main performance criteria for GridPick are picks per unit time, average retrieval time, walking time and waiting time of the worker. Furthermore, we evaluate the effect of the aspect ratio on system performance. In order to examine the performance, we compare GridPick to an equivalent flow rack picking system.

Performance of GridPick relies on several parameters and design policies. We explore the effects of the expected order size and empty cells per row on performance. Parameters and assumptions for the experiments are as follows: The grid includes 250 storage containers, and there are 2, 3, or 4 empty cells per row (k), which makes 20, 30, or 40 empty positions total. Each simulation run is with 4,000 iterations, 200 iterations for the warmup period, and 30 replications. The AnyLogic software package is used for the experiments. We analyze the system with a number of parameters for various configurations to capture the effect on the performance.

For this model, distance units are meters and time units are seconds. Picking times were derived from data of the Jo-Ann Stores Distribution Center in Opelika, Alabama and it is assumed to be 7.5 seconds. The conveyor speed is an important parameter, because it directly affects the possible waiting time of the worker. If the conveyor speed is high, it lowers the waiting time of the worker significantly. The common conveyor speed is given as

65 fpm (feet per minute), which we convert to 0.4 m/s for the simulation. Lastly, a worker has an assumed walking speed of 3.6 km/hour, which is approximately equal to 1.0 m/s. It is taken from a publication that has a direct warehouse application and analysis [Dekker et al., 2004].

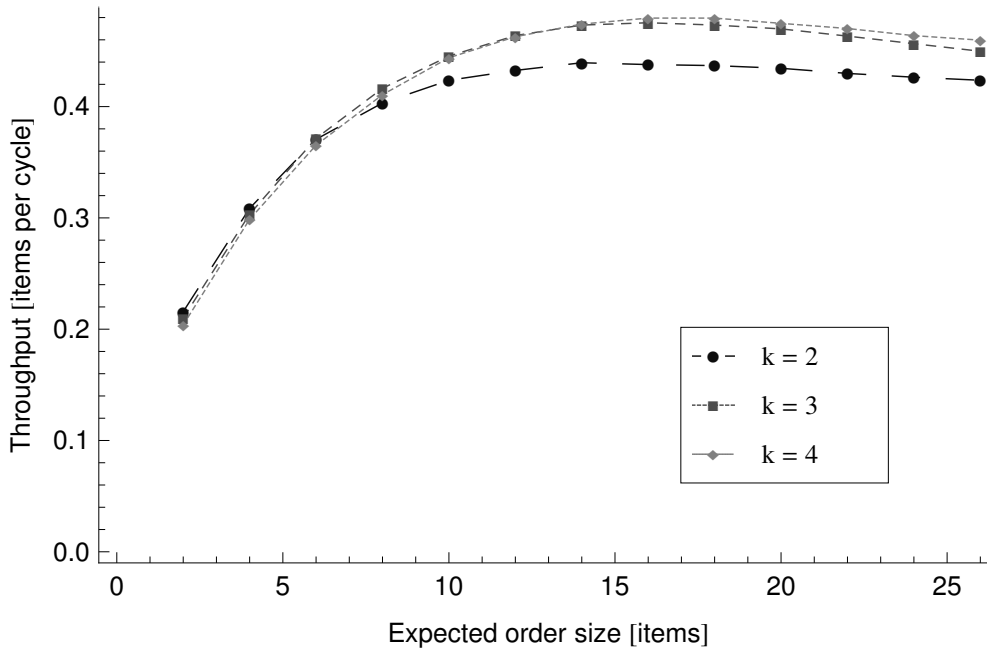


Figure 3.19: Throughput Plot for Different Expected Order Size Levels

Throughput of the example GridPick system increases with the increasing expected order size until the level of approximately 10 (see Figure 3.19). From expected order size of 10 to 28, throughput stabilizes and does not improve. This is due to stabilizing walking time per pick. Performance results of the expected order size will change for different configurations such as various aspect ratio values, which is the subject of the next section.

The throughput plot shows an expected result; configurations with more empty modules have better performance (see Figure 3.19). Increasing the number of empty cells per row increases throughput, but with diminishing returns.

Because all the experiments results in a distribution and does not represent a single value, we have prepared box whisker plots for the performance analysis. Distributions does

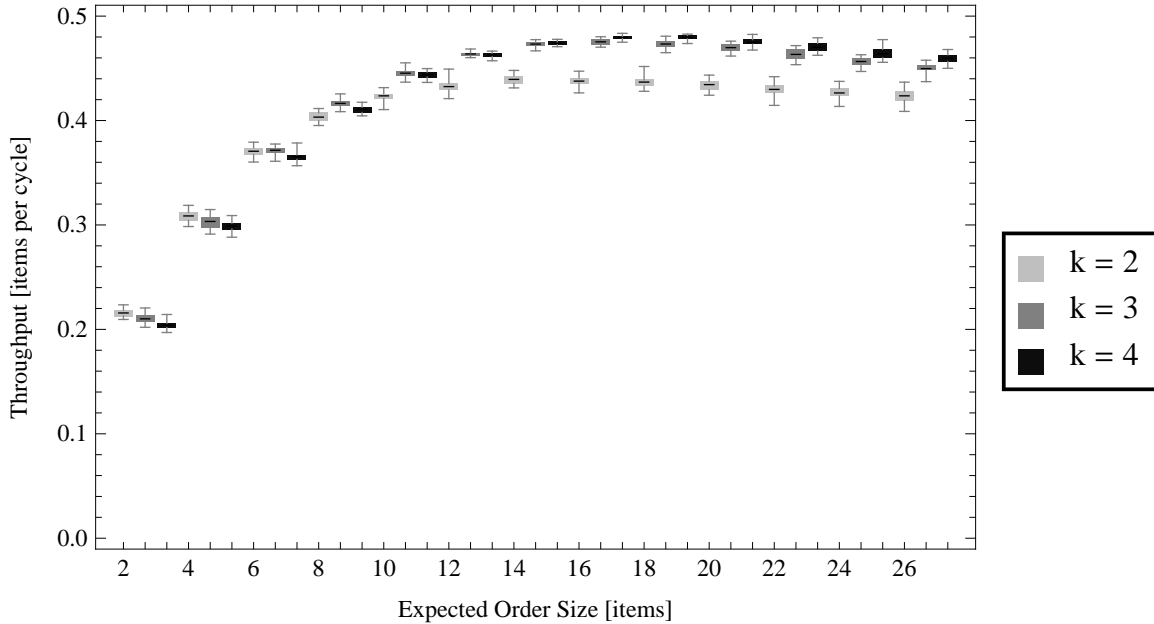


Figure 3.20: Throughput changing with the order size and empty cells per row.

not overlap and shown next to each other. For instance, in the throughput plot (see Figure 3.20) for the expected order size of 2 there are 3 box-whiskers, which shows the three k configurations. Therefore, we should compare first three, second three, third three configurations and so on. $k = 2$ configuration has longer whisker for larger expected order sizes, due to intense traffic and lower traffic capacity with small number of empty cells.

Table 3.22 shows significant and insignificant differences between configuration. A value smaller than 0.05, which is significance level, means there is a significant difference. Otherwise, two configurations are statistically same. Table ?? shows that only four configurations are statistically same and have values larger than 0.05.

We can interpret the results more easily by considering the t table (see Table ??). Values larger than 2.045 or smaller than -2.045 mean there is a significant difference. If there is a negative value second configuration in the comparison has a statistically larger value. Otherwise, first configuration has a statistically larger value. We see that smaller k configurations are significantly better for small order sizes. There is a breakeven point in which configurations arrive to a insignificant level of expected order size. This is expected

order size	k=2 vs k=3	k=2 vs k=4	k=3 vs k=4
2	0	0	0
4	0	0	0.004
6	0.458	0	0
8	0	0	0
10	0	0	0.15
12	0	0	0.12
14	0	0	0.086
16	0	0	0
18	0	0	0
20	0	0	0
22	0	0	0
24	0	0	0
26	0	0	0

Table 3.22: p values for throughput with various expected order sizes and empty cells per row

order size	k=2 vs k=3	k=2 vs k=4	k=3 vs k=4
2	4.47	12.11	5.88
4	3.92	7.51	3.11
6	-0.75	4.39	7.24
8	-14.44	-7.99	5.58
10	-18.02	-17.7	1.48
12	-30.8	-27.97	1.6
14	-39.38	-41.9	-1.78
16	-35.5	-40.56	-7.67
18	-27.69	-39.63	-8.1
20	-31.1	-30.25	-5.82
22	-21.7	-29.71	-5.07
24	-24.04	-24.47	-5.09
26	-17.98	-25.44	-9.15

Table 3.23: t values for throughput with various expected order sizes and empty cells per row

order size of 6 for $k = 2$ vs. $k = 3$ comparison and the break even is at expected order sizes of 10, 12, and 14 for $k = 3$ vs. $k = 4$ comparison. All differences are significant for $k = 2$ vs. $k = 4$ configurations.

order size	k=2 vs k=3	k=2 vs k=4	k=3 vs k=4
2	(0.00307, 0.00823)	(0.009835, 0.013832)	(0.00403, 0.00833)
4	(0.00253, 0.00805)	(0.00719, 0.01257)	(0.00157, 0.00761)
6	(-0.00341, 0.00158)	(0.00281, 0.00771)	(0.004430, 0.007920)
8	(-0.015003, -0.011280)	(-0.009337, -0.005530)	(0.00362, 0.00780)
10	(-0.02417, -0.01924)	(-0.02258, -0.01790)	(-0.000562, 0.003496)
12	(-0.03339, -0.02923)	(-0.03250, -0.02807)	(-0.000282, 0.002332)
14	(-0.035599, -0.032084)	(-0.036446, -0.033054)	(-0.001955, 0.000138)
16	(-0.03975, -0.03542)	(-0.04387, -0.03966)	(-0.005300, -0.003067)
18	(-0.03912, -0.03373)	(-0.04489, -0.04048)	(-0.007839, -0.004677)
20	(-0.03774, -0.03308)	(-0.04311, -0.03765)	(-0.006723, -0.003227)
22	(-0.03668, -0.03036)	(-0.04313, -0.03757)	(-0.00959, -0.00408)
24	(-0.03268, -0.02755)	(-0.04048, -0.03424)	(-0.01015, -0.00433)
26	(-0.02904, -0.02311)	(-0.03896, -0.03316)	(-0.01221, -0.00775)

Table 3.24: CIs for throughput with various expected order sizes and empty cells per row

We represent the confidence intervals for the differences between throughput configurations in Table 3.24. If the confidence interval includes 0, this means there is no significant differences between configurations.

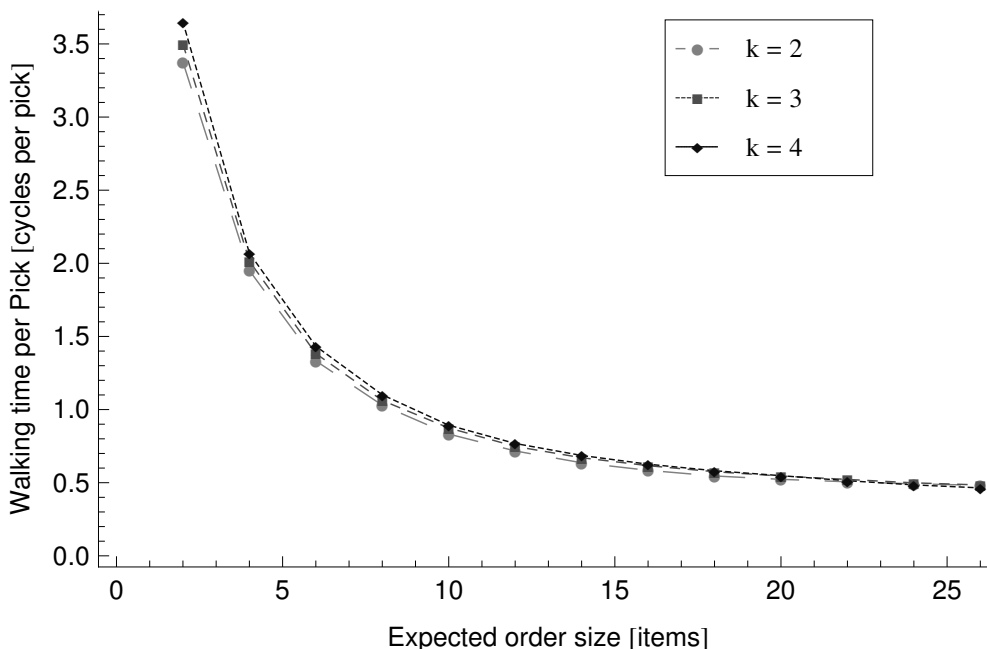


Figure 3.21: Walking Time per Pick for Different Expected Order Size Levels

Walking time per pick for all k configurations decreases with increasing expected order size, and stabilizes for larger expected order sizes (see Figure 3.21).

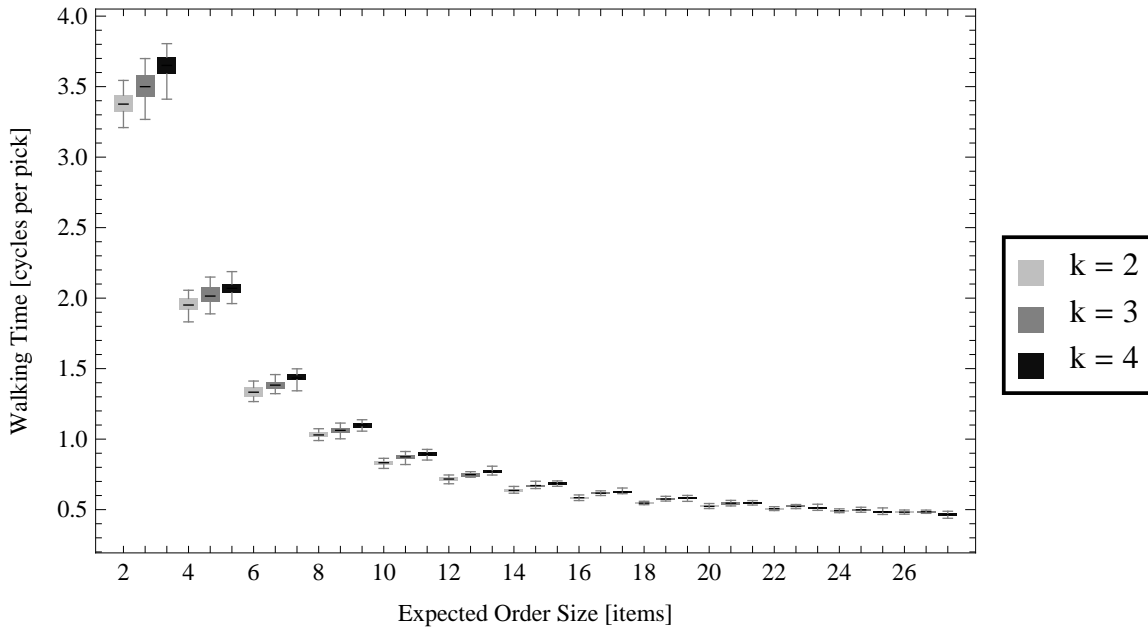


Figure 3.22: Walking time for various expected order sizes and empty cells per row.

For walking time per pick configuration, experiments have higher variation for small expected order sizes due to longer and variable walking times between picks. With larger expected order size values, walking time per pick decreases and becomes less variable.

For the $k = 2$ vs. $k = 3$ comparisons, only largest expected order size indicates an insignificant difference (see Table 3.25). For the $k = 3$ vs. $k = 4$ comparison, expected order size of 20 is insignificantly different.

T values shows that $k = 2$ configuration has a shorter walking time up to larger expected order sizes (see Table 3.26). Because there is another column of empty cells for larger k configurations. For very larger order sizes, $k = 4$ configuration became to have a shorter walking time because less empty cell configuration cannot handle the traffic well and has adjustment movements. These movements are due to having requested items waiting at the back and make other requested items move horizontally on the pick face. If the worker arrives after the movement decision, he adjusts the movement by walking to very next location.

	k=2 vs k=3	k=2 vs k=4	k=3 vs k=4
2	0	0	0
4	0	0	0.002
6	0	0	0
8	0	0	0
10	0	0	0.001
12	0	0	0
14	0	0	0
16	0	0	0.001
18	0	0	0.012
20	0	0	0.367
22	0	0.004	0.008
24	0.007	0.007	0
26	0.14	0	0

Table 3.25: p values of walking time with various expected order sizes and empty cells per row.

order size	k=2 vs k=3	k=2 vs k=4	k=3 vs k=4
2	-4.35	-12.2	-6.27
4	-4.53	-8.17	-3.49
6	-5.54	-10.3	-7.02
8	-6.04	-15.76	-6.13
10	-8.75	-11.56	-3.58
12	-8.61	-15.59	-5.4
14	-11.76	-20.03	-6.87
16	-17.04	-23.55	-3.85
18	-14.01	-15.8	-2.69
20	-9.57	-13.71	-0.92
22	-8.26	-3.11	2.83
24	-2.93	2.92	5.02
26	-1.52	6.58	7.58

Table 3.26: T values of walking time with various expected order sizes and empty cells per row.

Confidence intervals for the differences between walking time configurations are shown on Table 3.27. If the confidence interval includes 0, it means there is no significant difference.

Because there are more active items and higher traffic intensity with larger order sizes, average retrieval time increases with the expected order size (see Figure 3.24). This is due to the increased waiting time. Waiting time per pick increases with the expected order size,

order size	k=2 vs k=3	k=2 vs k=4	k=3 vs k=4
2	(-0.1819, -0.0656)	(-0.3202, -0.2283)	(-0.1996, -0.1014)
4	(-0.0922, -0.0349)	(-0.1483, -0.0889)	(-0.0874, -0.0228)
6	(-0.06859, -0.03162)	(-0.12046, -0.08056)	(-0.06508, -0.03573)
8	(-0.04336, -0.02143)	(-0.08054, -0.06203)	(-0.05186, -0.02592)
10	(-0.05159, -0.03205)	(-0.07124, -0.04982)	(-0.02941, -0.00801)
12	(-0.03853, -0.02374)	(-0.05796, -0.04452)	(-0.02772, -0.01249)
14	(-0.04039, -0.02842)	(-0.05506, -0.04486)	(-0.02018, -0.01092)
16	(-0.03572, -0.02806)	(-0.04557, -0.03829)	(-0.01537, -0.00471)
18	(-0.03335, -0.02486)	(-0.04055, -0.03125)	(-0.01196, -0.00163)
20	(-0.02753, -0.01783)	(-0.02885, -0.02136)	(-0.00784, 0.00299)
22	(-0.01990, -0.01200)	(-0.01193, -0.00246)	(0.00243, 0.01509)
24	(-0.01146, -0.00203)	(0.00212, 0.01202)	(0.00818, 0.01944)
26	(-0.00551, 0.00082)	(0.01210, 0.02302)	(0.01453, 0.02528)

Table 3.27: CIs of walking time with various expected order sizes and empty cells per row.

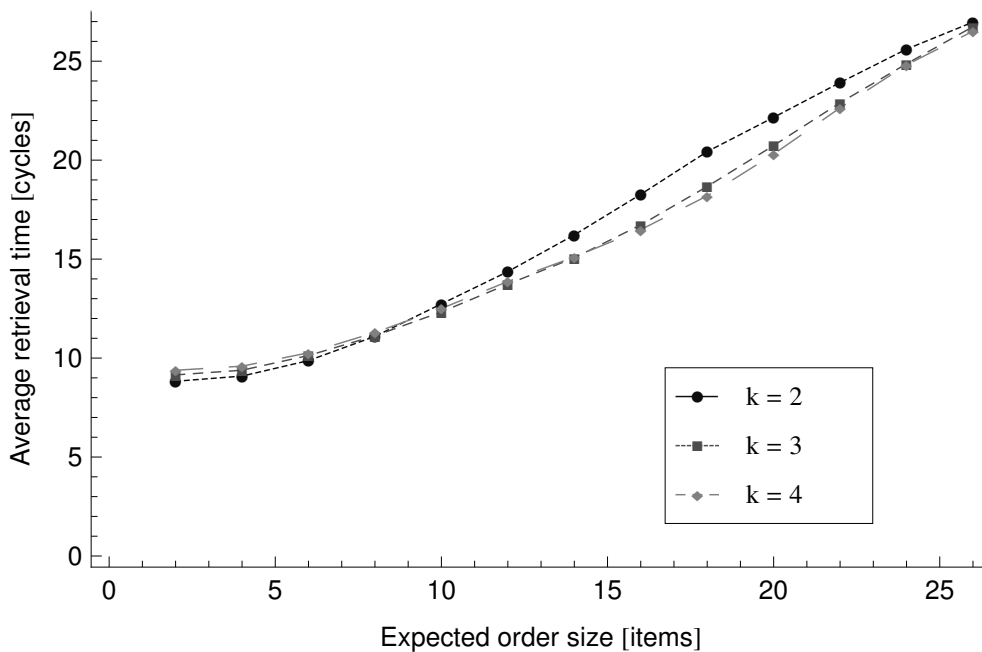


Figure 3.23: Average Retrieval Time for Different Expected Order Size Levels

which results in a longer retrieval time (see Figure 3.25). As expected, average retrieval time is shorter with more empty cells, especially for the larger expected order size values. Waiting time per pick is significantly larger for the $k = 2$ configuration. Since waiting time is combined with the travel time for the computation of the retrieval time, the difference on the average retrieval times becomes less significant.

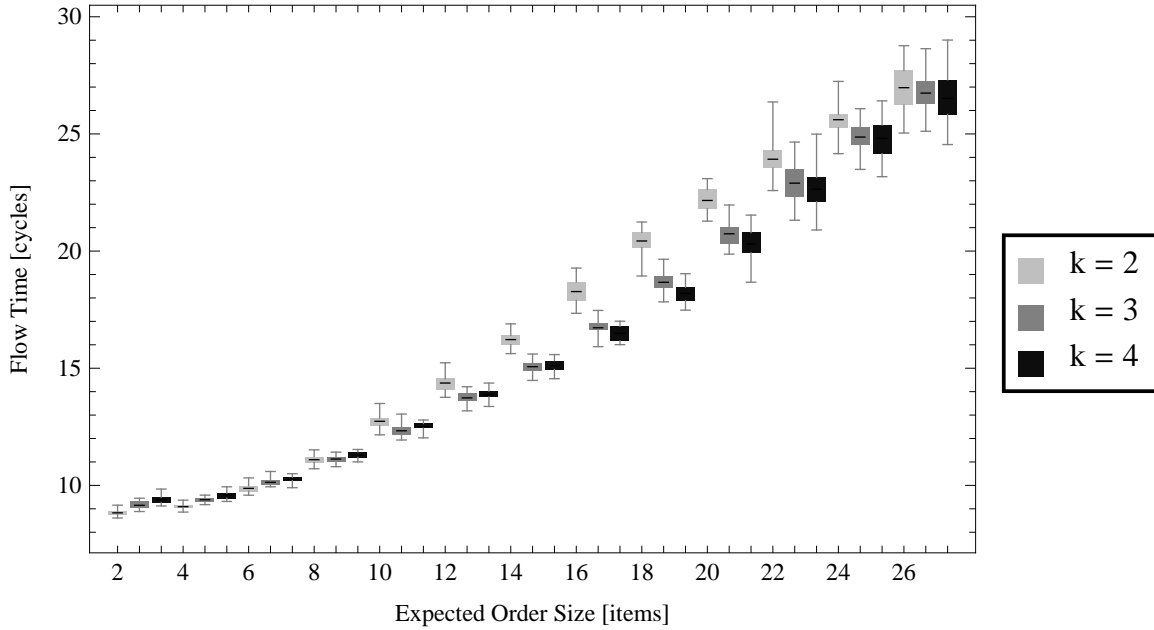


Figure 3.24: Flow time for various expected order sizes and empty cells per row.

Variation of the average retrieval times increase with the larger expected order sizes (see Figure 3.24). This is due to having more traffic with more requested items.

order size	k=2 vs k=3	k=2 vs k=4	k=3 vs k=4
2	0	0	0
4	0	0	0
6	0	0	0.001
8	0.568	0	0
10	0	0	0.006
12	0	0	0.066
14	0	0	0.524
16	0	0	0.009
18	0	0	0.001
20	0	0	0.009
22	0	0	0.279
24	0	0.001	0.77
26	0.332	0.061	0.322

Table 3.28: p values for flow time with various expected order sizes and empty cells per row

Similar to the throughput results, expected order size of 8 is insignificant for $k = 2$ vs. $k = 3$ configurations (see Table 3.28). $k = 3$ vs. $k = 4$ configurations show insignificant results for expected order size of 10 and 12. For very large expected order sizes, there are also insignificant results.

order size	k=2 vs k=3	k=2 vs k=4	k=3 vs k=4
2	-8.78	-15.24	-6.72
4	-9.33	-16.83	-5.85
6	-6.54	-9.19	-3.88
8	-0.58	-4.4	-3.95
10	7.01	4.11	-3
12	7.58	5.19	-1.91
14	12.95	15.73	-0.64
16	16.76	17.39	2.79
18	15.17	16.23	3.84
20	9.84	13.91	2.8
22	4.89	6.3	1.1
24	4.53	3.77	0.29
26	0.99	1.95	1.01

Table 3.29: t values for flow time with various expected order sizes and empty cells per row

order size	k=2 vs k=3	k=2 vs k=4	k=3 vs k=4
2	(-0.3959, -0.2464)	(-0.6303, -0.4812)	(-0.3059, -0.1632)
4	(-0.3614, -0.2315)	(-0.5619, -0.4401)	(-0.2762, -0.1330)
6	(-0.3381, -0.1770)	(-0.4817, -0.3063)	(-0.2083, -0.0645)
8	(-0.1226, 0.0686)	(-0.2836, -0.1037)	(-0.2530, -0.0803)
10	(0.2870, 0.5233)	(0.1162, 0.3463)	(-0.2926, -0.0552)
12	(0.4631, 0.8056)	(0.3023, 0.6956)	(-0.2806, 0.0097)
14	(0.9739, 1.3391)	(0.9696, 1.2593)	(-0.1753, 0.0913)
16	(1.3566, 1.7337)	(1.575, 1.995)	(0.0641, 0.4155)
18	(1.526, 2.001)	(1.954, 2.518)	(0.221, 0.724)
20	(1.127, 1.719)	(1.585, 2.131)	(0.118, 0.752)
22	(0.595, 1.448)	(0.864, 1.695)	(-0.220, 0.735)
24	(0.406, 1.075)	(0.364, 1.226)	(-0.319, 0.426)
26	(-0.249, 0.711)	(-0.022, 0.943)	(-0.236, 0.694)

Table 3.30: Confidence Intervals for flow time with various expected order sizes and empty cells per row

$k = 2$ configuration has a shorter average retrieval time for small expected order sizes (see Table 3.29). This gets to a breakeven and for large order sizes, $k = 3$ and $k = 4$ configurations becomes better because they handle the traffic flow better with more empty cells.

Confidence intervals for the differences between average retrieval times of k configurations are shown on Table 3.30.

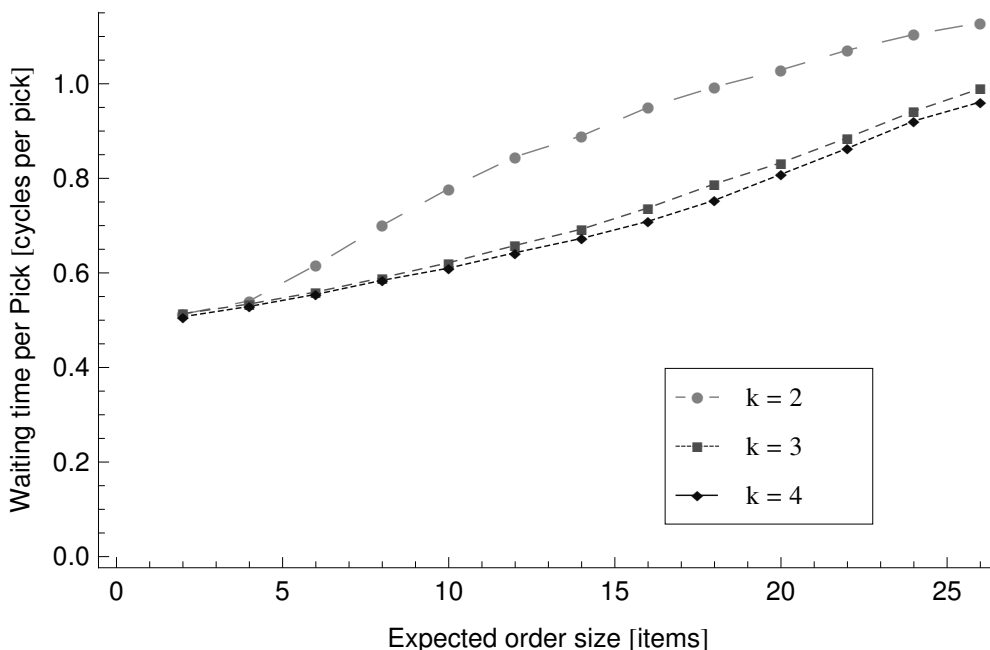


Figure 3.25: Waiting Time per Pick for Different Expected Order Size Levels

Waiting time per pick for $k = 2$ configuration has more variability because of lower traffic capacity with less empty cells (see Figure 3.26).

For very small expected order sizes, there is no significant difference between configurations (see Table 3.31). Also for larger expected order sizes difference between $k = 3$ vs. $k = 4$ becomes insignificant because both configurations have more difficulty handling the traffic.

T values shows that $k = 2$ has a significantly longer waiting time except the smallest order size of 2 compared to other two configurations (see Table ??). $k = 3$ configuration has a significantly longer waiting time due to less empty cells per row.

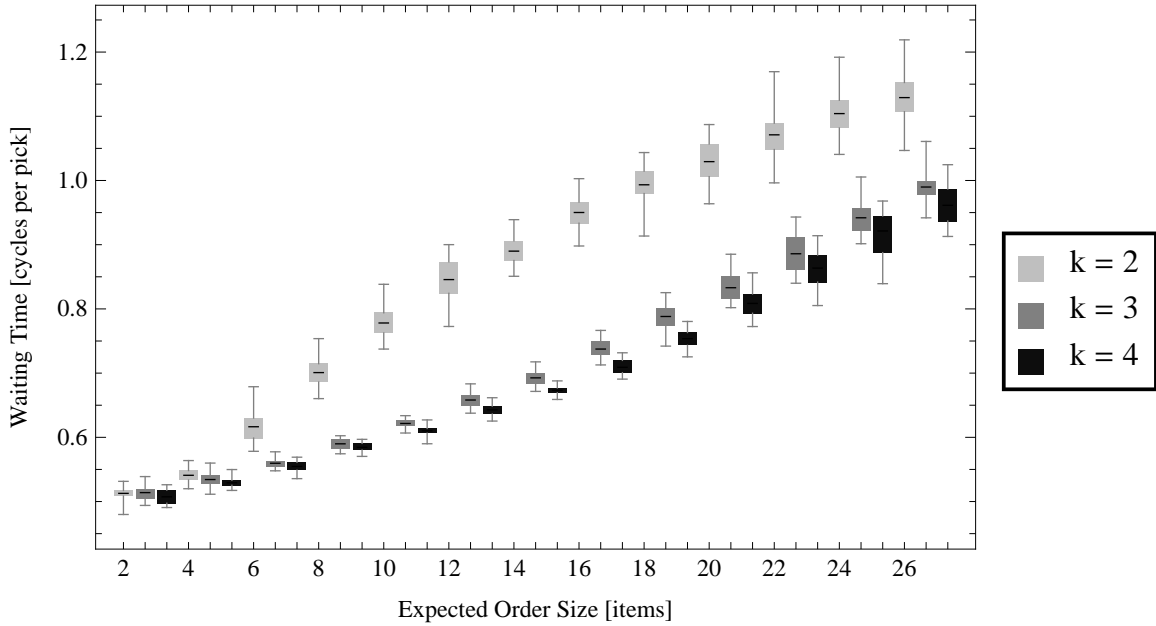


Figure 3.26: Waiting time for various expected order sizes and empty cells per row.

	k=2 vs k=3	k=2 vs k=4	k=3 vs k=4
2	0.704	0.057	0.068
4	0.035	0	0.032
6	0	0	0.023
8	0	0	0.009
10	0	0	0
12	0	0	0
14	0	0	0
16	0	0	0
18	0	0	0
20	0	0	0
22	0	0	0.013
24	0	0	0.026
26	0	0	0

Table 3.31: p values for waiting time with various expected order sizes and empty cells per row

Confidence intervals are shown for the waiting time per pick differences of k configurations.

	k=2 vs k=3	k=2 vs k=4	k=3 vs k=4
2	-0.38	1.98	1.9
4	2.21	5.01	2.25
6	13.33	14.48	2.41
8	23.54	28.49	2.82
10	32.33	34.38	5.49
12	32.49	35.43	5.32
14	44.41	49.55	8.08
16	37.67	49.59	10.42
18	28.12	38.25	7.87
20	29.18	29.19	4.53
22	20.69	25.36	2.63
24	19.93	19.28	2.35
26	17.22	18.7	4.03

Table 3.32: t values for waiting time with various expected order sizes and empty cells per row

	k=2 vs k=3	k=2 vs k=4	k=3 vs k=4
2	(-0.00748, 0.00511)	(-0.00017, 0.01083)	(-0.00051, 0.01353)
4	(0.00048, 0.01278)	(0.00693, 0.01649)	(0.00046, 0.00970)
6	(0.04812, 0.06557)	(0.05311, 0.07058)	(0.00075, 0.00925)
8	(0.10129, 0.12056)	(0.10809, 0.12480)	(0.00152, 0.00952)
10	(0.14660, 0.16640)	(0.15792, 0.17790)	(0.00716, 0.01567)
12	(0.17576, 0.19937)	(0.19117, 0.21459)	(0.00942, 0.02120)
14	(0.18827, 0.20645)	(0.20800, 0.22591)	(0.01463, 0.02454)
16	(0.20114, 0.22423)	(0.23114, 0.25103)	(0.02283, 0.03398)
18	(0.19019, 0.22002)	(0.22686, 0.25250)	(0.02559, 0.04356)
20	(0.18260, 0.21012)	(0.20533, 0.23628)	(0.01341, 0.03549)
22	(0.16691, 0.20352)	(0.19082, 0.22430)	(0.00499, 0.03969)
24	(0.14561, 0.17891)	(0.16341, 0.20220)	(0.00269, 0.03840)
26	(0.12272, 0.15580)	(0.14940, 0.18608)	(0.01403, 0.04293)

Table 3.33: CIs for waiting time with various expected order sizes and empty cells per row

3.5.1 Aspect Ratio Analysis

We investigate the effect of the aspect ratio (number of columns / number of rows) on performance. Aspect ratio configurations for the experiments are in Table 3.34. All configurations have approximately the same number of conveyors, storage containers, and empty cells.

Config. No	Aspect Ratio	Rows	Cols.	k	Conveyors	Items	Empty Cells
1	8 (6:50)	6	50	8	300	252	48
2	3 (10:30)	10	30	5	300	250	50
3	1 (16:19)	16	19	3	304	256	48
4	0.5 (25:12)	25	12	2	300	250	50
5	0.16 (42:7)	42	7	1	294	252	42

Table 3.34: Configurations with Different Aspect Ratio Values

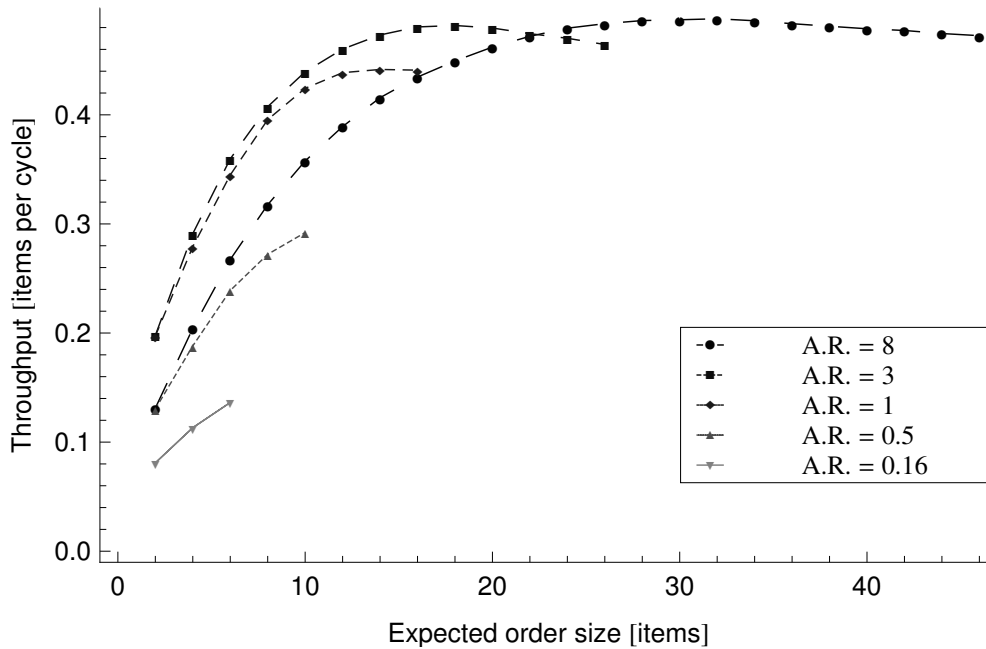


Figure 3.27: Throughput Plot

All aspect ratio configurations are significantly different from each other except aspect ratio of 0.5 vs. 8 and aspect ratio of 1 vs. 3 at the expected order size 2 (see Table 3.35). These points are breakeven points where longer walking time balances the effect of shorter waiting time and these configurations become statistically same.

T values shows that aspect ratio of 3 is better for all expected order sizes (see Figure 3.36). Aspect ratio of 8 is worse than the aspect ratios of 1 and 3 due to excess walking.

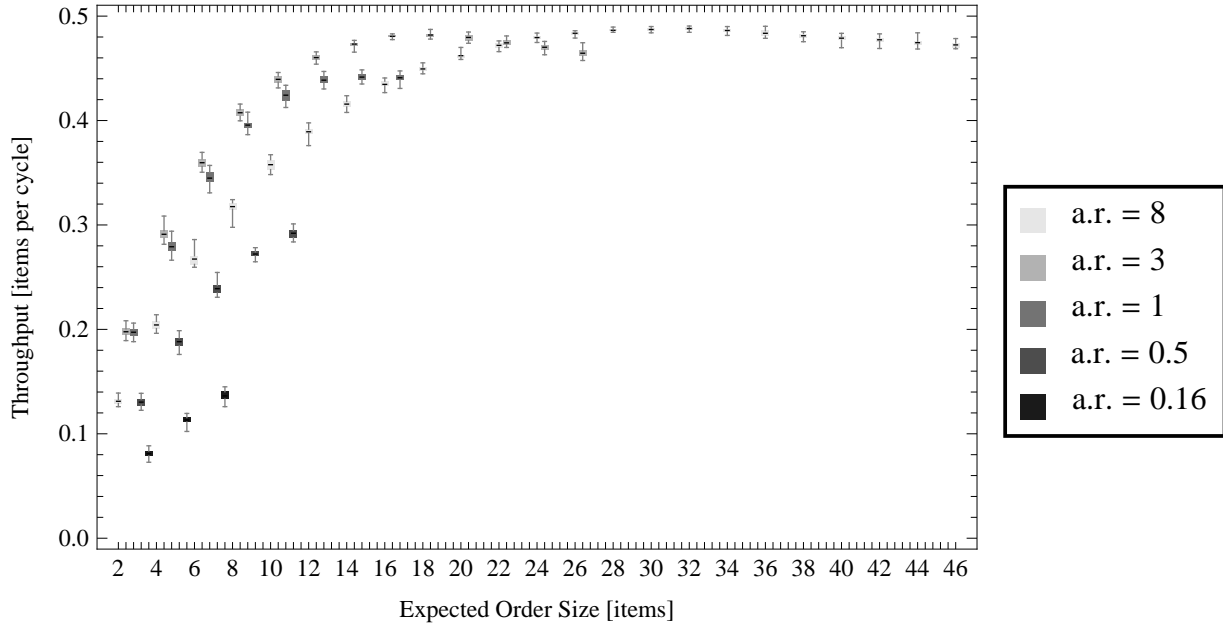


Figure 3.28: Throughput for different aspect ratios

	0.16vs0.5	0.16vs1	0.16vs3	0.16vs8	0.5vs1	0.5vs3	0.5vs8	1vs3	1vs8	3vs8
2	0	0	0	0	0	0	0.309	0.721	0	0
4	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
8	-	-	-	-	0	0	0	0	0	0
10	-	-	-	-	0	0	0	0	0	0
12	-	-	-	-	-	-	-	0	0	0
14	-	-	-	-	-	-	-	0	0	0
16	-	-	-	-	-	-	-	0	0	0
18	-	-	-	-	-	-	-	-	-	0
20	-	-	-	-	-	-	-	-	-	0
22	-	-	-	-	-	-	-	-	-	0.004
24	-	-	-	-	-	-	-	-	-	0
26	-	-	-	-	-	-	-	-	-	0

Table 3.35: p values for throughput with various aspect ratios

Aspect ratio of 8 is better than the aspect ratios of 0.16 and 0.5 because of shorter waiting time.

Confidence intervals for the aspect ratio configurations are shown on Tables 3.37, 3.38 and 3.39.

	0.16vs0.5	0.16vs1	0.16vs3	0.16vs8	0.5vs1	0.5vs3	0.5vs8	1vs3	1vs8	3vs8
2	-56.16	-135.13	-117.13	-56.04	-64.3	-56.26	-1.04	-0.36	71.08	60.86
4	-71.1	-115.74	-143.63	-86.88	-57.91	-79.12	-11.69	-8.12	59.23	80.96
6	-82.79	-134.49	-179.75	-94.34	-68.38	-96.99	-20.88	-9.26	49.69	80.04
8	-	-	-	-	-99.66	-147.87	-42.22	-9.56	64.52	70.86
10	-	-	-	-	-79.87	-120.56	-53.07	-11.88	45.48	68.87
12	-	-	-	-	-	-	-	-24.44	46.51	79.6
14	-	-	-	-	-	-	-	-46.6	24.96	69.1
16	-	-	-	-	-	-	-	-49.64	6.69	56.74
18	-	-	-	-	-	-	-	-	-	50.19
20	-	-	-	-	-	-	-	-	-	23.8
22	-	-	-	-	-	-	-	-	-	3.1
24	-	-	-	-	-	-	-	-	-	-14.65
26	-	-	-	-	-	-	-	-	-	-23.5

Table 3.36: t values for throughput with various aspect ratios

	0.16vs0.5	0.16vs1	0.16vs3	0.16vs8
2	(-0.0511, -0.0476)	(-0.118, -0.115)	(-0.1188, -0.1147)	(-0.052, -0.048)
4	(-0.077, -0.073)	(-0.1688, -0.163)	(-0.180, -0.175)	(-0.093, -0.089)
6	(-0.105, -0.100)	(-0.2117, -0.2054)	(-0.226, -0.2208)	(-0.134, -0.128)

Table 3.37: CIs for throughput with various aspect ratios

	0.5vs1	0.5vs3	0.5vs8
2	(-0.06906, -0.06480)	(-0.06984, -0.06494)	(-0.002403, 0.000787)
4	(-0.09415, -0.08773)	(-0.10554, -0.10022)	(-0.01885, -0.01323)
6	(-0.10904, -0.10271)	(-0.12319, -0.11810)	(-0.03115, -0.02560)
8	(-0.12600, -0.12093)	(-0.137197, -0.133453)	(-0.04763, -0.04322)
10	(-0.13558, -0.12881)	(-0.14986, -0.14486)	(-0.06816, -0.06310)

Table 3.38: CIs for throughput with various aspect ratios

Up to expected order size of approximately 15, throughput increases with the increasing aspect ratio until the aspect ratio of 3. For the lower expected order sizes, an aspect ratio of 8 has a lower throughput than the aspect ratio configurations with 3 and 1. For larger expected order sizes, throughput stabilizes. Since we restrict the expected order size with the number of columns, for different aspect ratio levels curves end at different expected order size levels.

	1vs3	1vs8	3vs8
2	(-0.00306, 0.00214)	(0.064222, 0.068028)	(0.06435, 0.06882)
4	(-0.01495, -0.00893)	(0.07231, 0.07749)	(0.08465, 0.08904)
6	(-0.01803, -0.01151)	(0.07431, 0.08069)	(0.08991, 0.09462)
8	(-0.01439, -0.00932)	0.07557, 0.08052)	(0.08731, 0.09249)
10	(-0.01778, -0.01256)	(0.06357, 0.06955)	(0.07930, 0.08415)
12	(-0.023425, -0.019808)	(0.04724, 0.05159)	(0.069208, 0.072858)
14	(-0.033004, -0.030229)	(0.02375, 0.02800)	(0.055790, 0.059193)
16	(-0.041249, -0.037984)	(0.004390, 0.008260)	(0.044286, 0.047598)
18	-	-	(0.031184, 0.033833)
20	-	-	(0.015966, 0.018967)
22	-	-	(0.000814, 0.003986)
24	-	-	(-0.010551, -0.007966)
26	-	-	(-0.020763, -0.017437)

Table 3.39: CIs for throughput with various aspect ratios

Waiting time per pick and wasted walking time plots help us to interpret the throughput and average retrieval time results (see Figures 3.31 and 3.33). For the worker, the ideal case is walking to the very next location and processing a pick. Wasted walking represents walking longer than the very next location between picks. Because the pick face is narrow for small aspect ratio values, waiting times are significantly larger, which increases the average retrieval time and decreases the throughput (see Figure 3.31).

For small expected order sizes, walking time increases with the increasing aspect ratio. After the aspect ratio of 3, walking times becomes larger per pick, which decreases the throughput for the aspect ratio of 8. For the lower expected order sizes, aspect ratio of 8 shows an interesting phenomenon: several orders are able to arrive to the pick face since order sizes are small and the worker is walking excessively, which leads to waiting time of the items (not the worker waiting), which increases the average retrieval time for the requested items. Average retrieval time keeps increasing with the increasing expected order size and stabilizes when change in the walking time per pick is small.

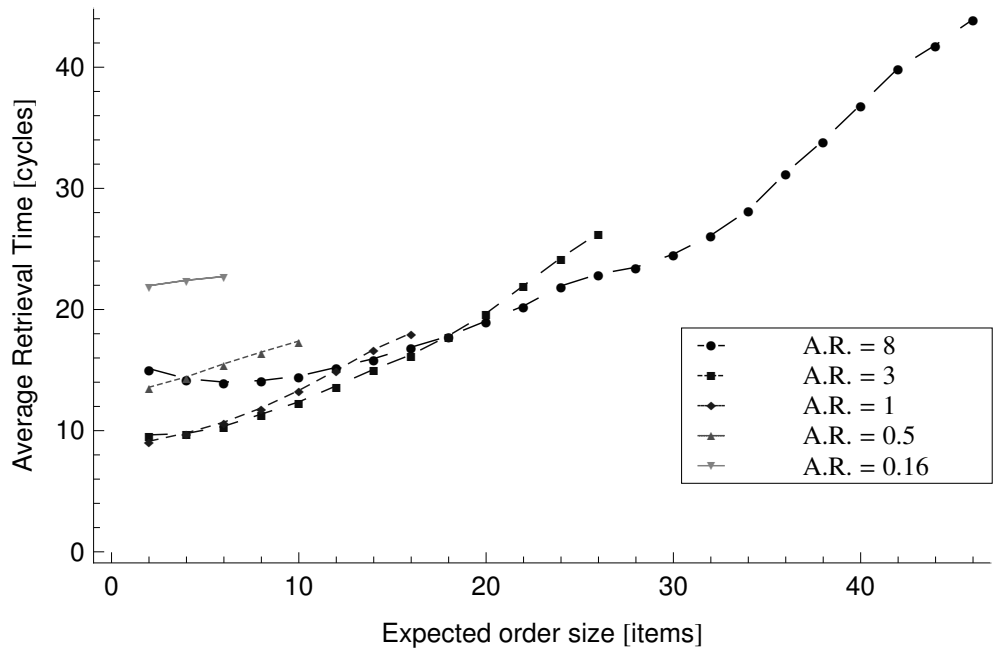


Figure 3.29: Average Retrieval Time Plot

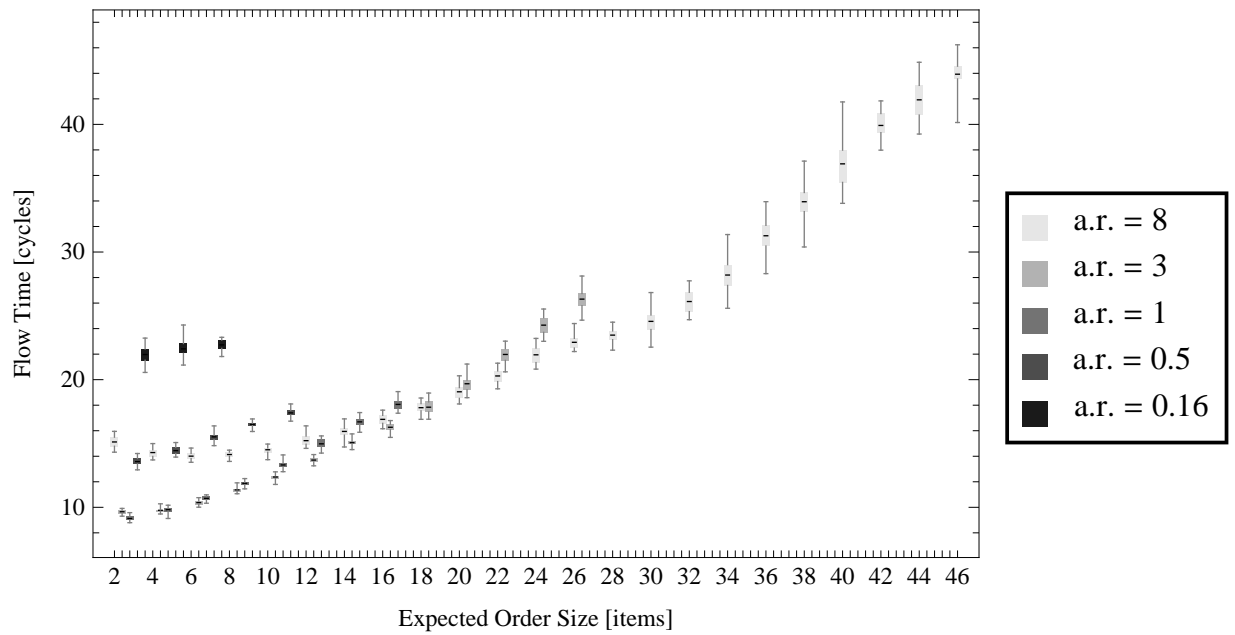


Figure 3.30: Flow time for different aspect ratios

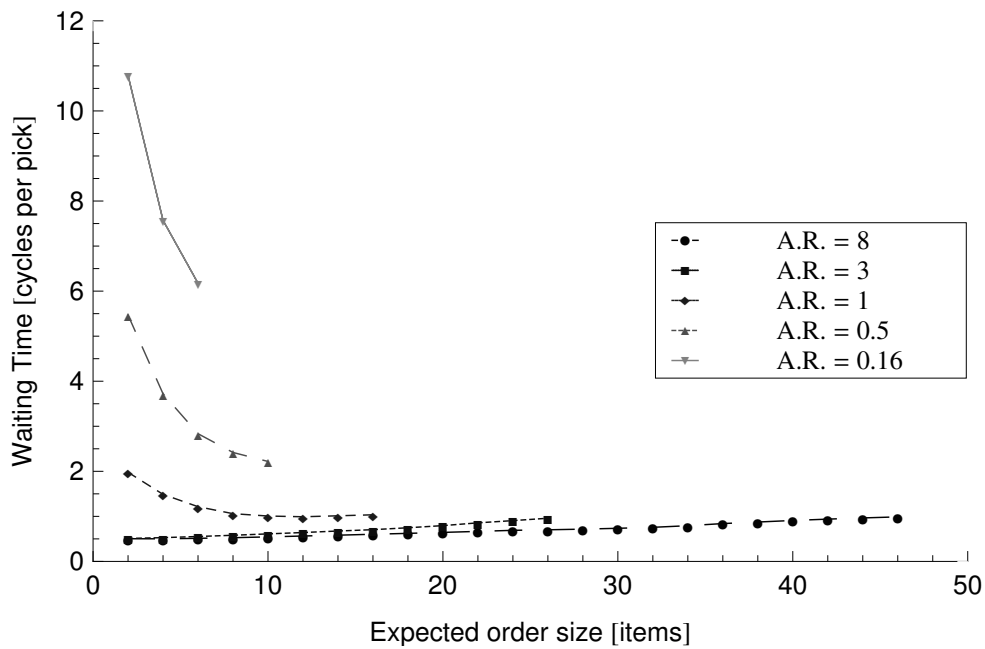


Figure 3.31: Waiting Time for Aspect Ratio Configurations

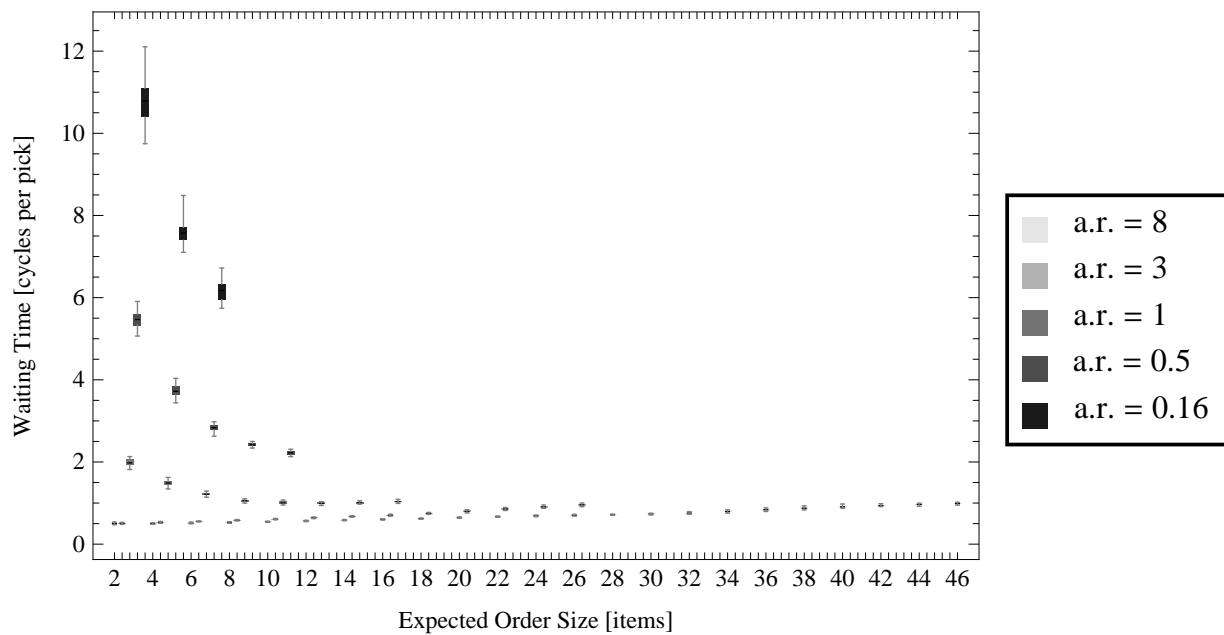


Figure 3.32: Waiting time for different aspect ratios

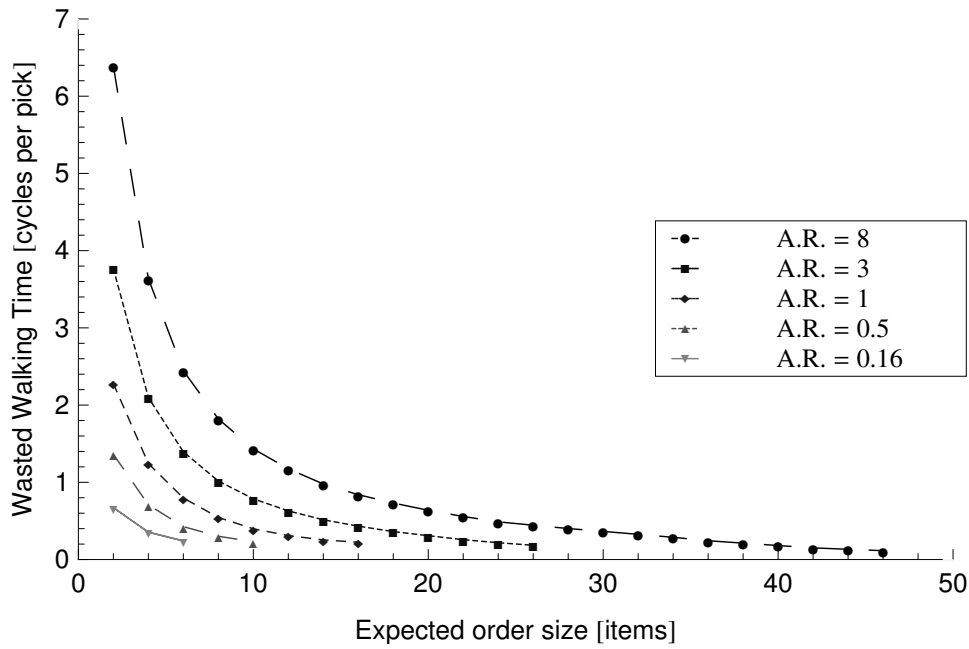


Figure 3.33: Walking Time for Aspect Ratio Configurations

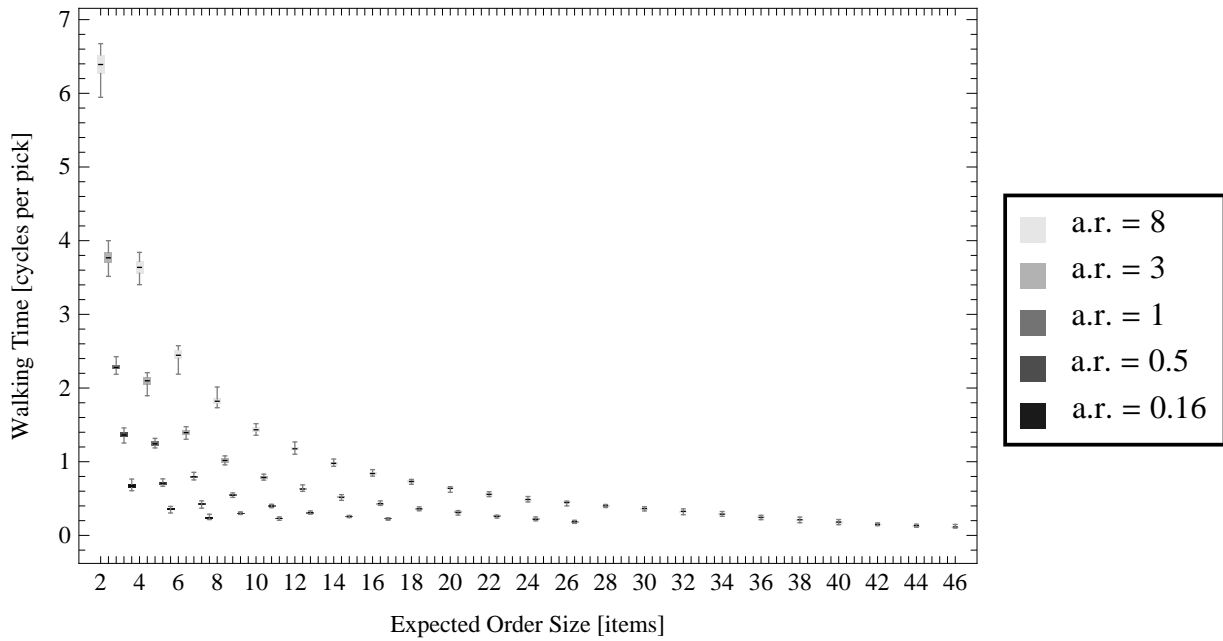


Figure 3.34: Walking time for different aspect ratios

3.5.2 Variable k Analysis

Empty cells do not have to be distributed uniformly throughout the grid. As long as there is at least one empty cell per row, we can distribute the empty cells arbitrarily. We investigate the effect of the distribution of empty cells with three configurations (see Table 3.40).

Row	Uniform	Decreasing k	Increasing k
1	4	6	2
2	4	6	2
3	4	6	2
4	4	4	4
5	4	4	4
6	4	4	4
7	4	4	4
8	4	2	6
9	4	2	6
10	4	2	6

Table 3.40: Distribution of Empty Modules for 3 Configurations

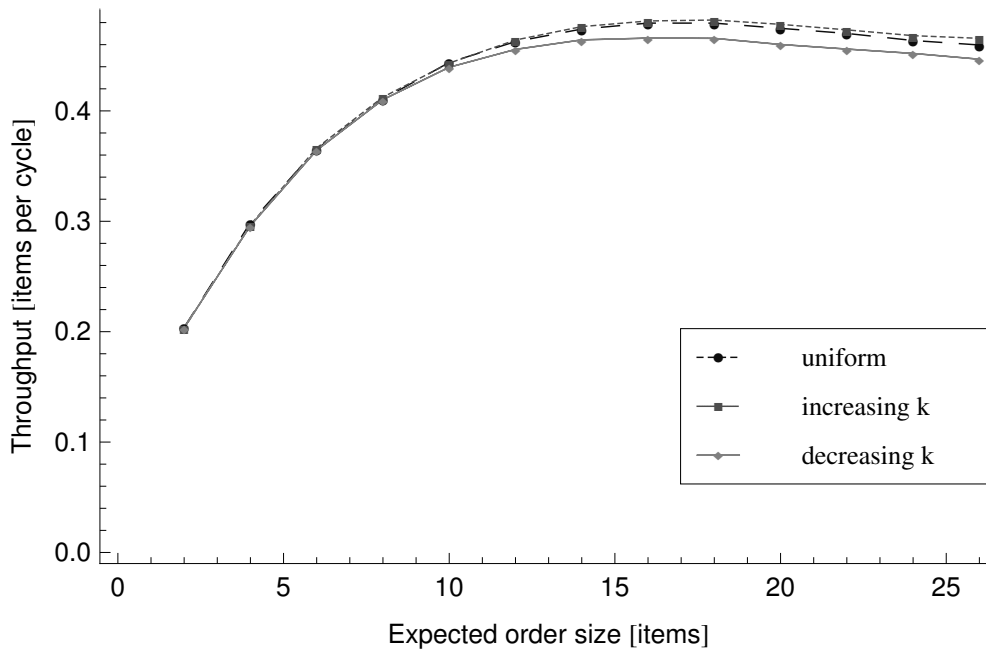


Figure 3.35: Throughput Plot

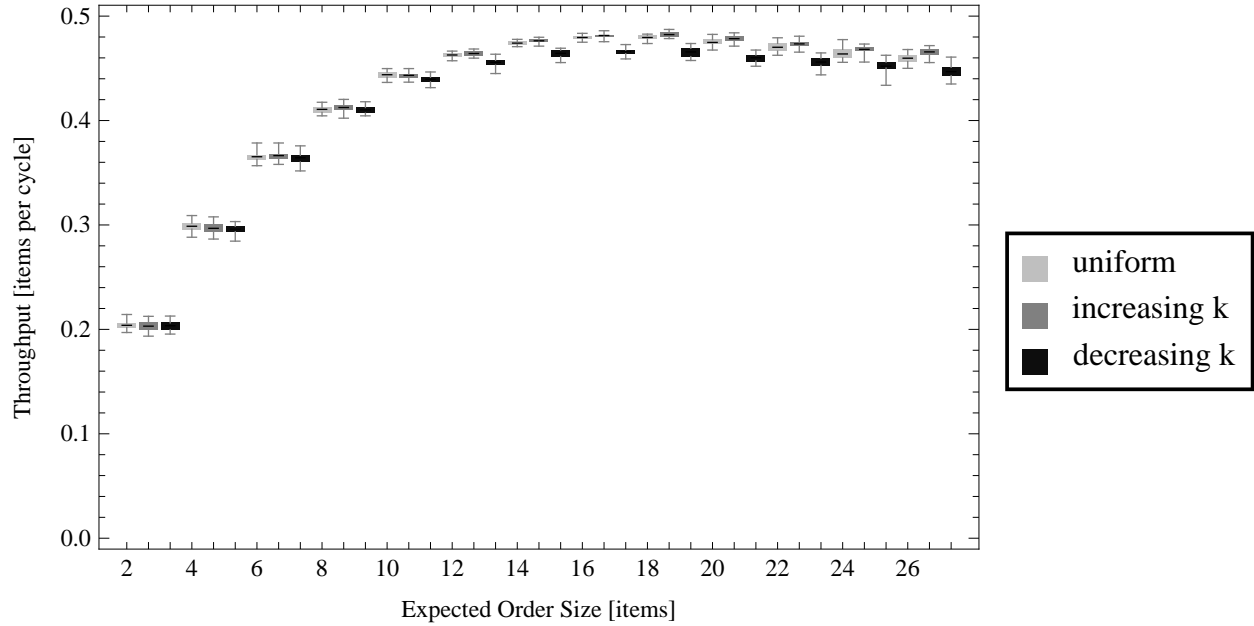


Figure 3.36: Throughput for variable k configurations

Throughput plots shows close results so we want to take a look at the statistical test results (see Figures 3.35 and 3.36).

	uniform vs. inc-k	uniform vs. dec-k	inc-k vs. dec-k
2	0.347	0.618	0.629
4	0.201	0.009	0.24
6	0.473	0.501	0.229
8	0.022	0.993	0.029
10	0.567	0	0
12	0.011	0	0
14	0	0	0
16	0.003	0	0
18	0	0	0
20	0	0	0
22	0.004	0	0
24	0	0	0
26	0	0	0

Table 3.41: p values for throughput with variable k configurations

For small expected order sizes up to 8, there is no significant difference between configuration because all of them handles the traffic well. For larger expected order sizes there is a significant difference.

	uniform vs. inc-k	uniform vs. dec-k	inc-k vs. dec-k
2	0.96	0.5	-0.49
4	1.31	2.79	1.2
6	-0.73	0.68	1.23
8	-2.41	0.01	2.3
10	0.58	5.3	4.71
12	-2.71	9.64	12.69
14	-4.68	14.05	16.66
16	-3.29	21.47	22.84
18	-4.73	13.48	15.99
20	-4.21	12.54	16.66
22	-3.15	12.53	18.09
24	-4.03	8	14.95
26	-4.74	7.88	15.8

Table 3.42: t values for throughput with variable k configurations

T values show that increasing- k configuration is significantly better compared to both uniform and decreasing- k configurations after the expected order size of 8 (see Table 3.42). The reason is probability of having requested items in a row increase when we get closer to the pick face. Therefore, we need more empty cells close to pick face. Uniform configuration is significantly better than decreasing- k configuration because it has more empty cells closer to the pick face.

	uniform vs. inc-k	uniform vs. dec-k	inc-k vs. dec-k
2	(-0.00127, 0.00351)	(-0.00176, 0.00291)	(-0.00281, 0.00173)
4	(-0.00094, 0.00427)	(0.00086, 0.00554)	(-0.00108, 0.00415)
6	(-0.00293, 0.00139)	(-0.00187, 0.00374)	(-0.00113, 0.00453)
8	(-0.003741, -0.000309)	(-0.001903, 0.001919)	(0.000229, 0.003838)
10	(-0.000972, 0.001739)	(0.002665, 0.006019)	(0.002239, 0.005678)
12	(-0.002663, -0.000370)	(0.005101, 0.007849)	(0.006704, 0.009279)
14	(-0.003341, -0.001309)	(0.008908, 0.011942)	(0.011185, 0.014315)
16	(-0.002717, -0.000633)	(0.012757, 0.015443)	(0.014362, 0.017188)
18	(-0.004334, -0.001716)	(0.01192, 0.01618)	(0.01489, 0.01926)
20	(-0.005485, -0.001898)	(0.01234, 0.01716)	(0.01618, 0.02071)
22	(-0.00585, -0.00125)	(0.01157, 0.01608)	(0.015411, 0.019339)
24	(-0.00671, -0.00219)	(0.00884, 0.01492)	(0.01410, 0.01857)
26	(-0.00804, -0.00319)	(0.00959, 0.01632)	(0.01617, 0.02098)

Table 3.43: CIs for throughput with variable k configurations

Confidence intervals for the throughput differences between variable k configurations are shown on Table 3.43. For small expected order sizes, the distribution of empty cells does not have a significant effect on the throughput. For larger expected order sizes, the decreasing k configuration performs worse than the other two configurations.

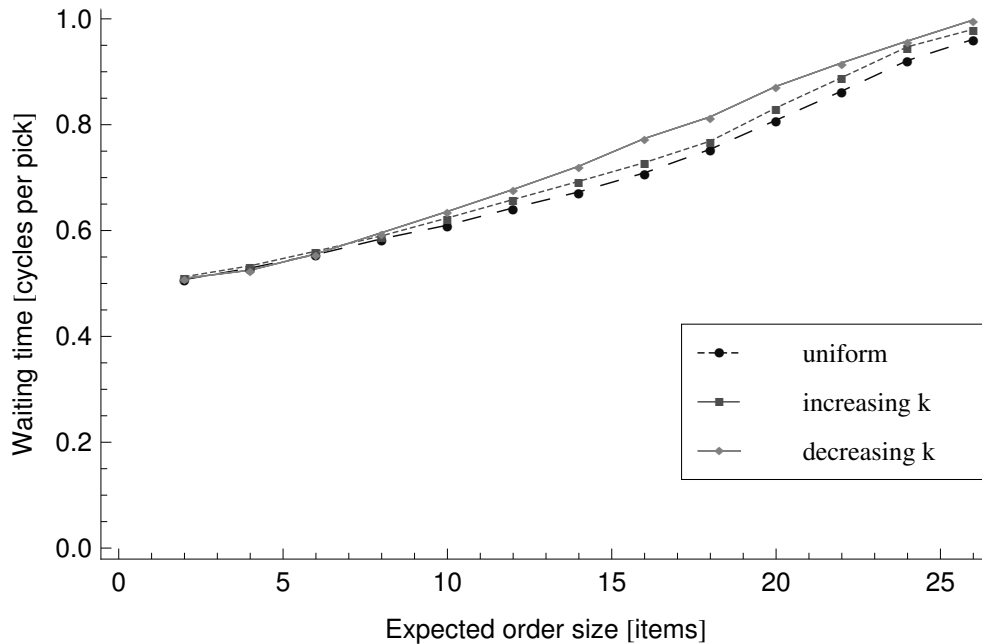


Figure 3.37: Waiting Time for Variable k Configurations

We can explain this difference with the average retrieval time and waiting time of the worker. Because all items try to reach the pick face, traffic density increases when items get closer to the pick face row. Because there are fewer empty positions toward the pick face in the decreasing k configuration, waiting time per pick increases and throughput decreases (see Figure 3.37).

3.5.3 Analysis with Multiple Copy Configurations

Towards the larger expected order sizes, one copy vs. two copy configuration becomes statistically same (see Table 3.44). For the very large expected order size, all configurations are statistically same. This is because of high traffic density for all configurations and deepness does not matter.

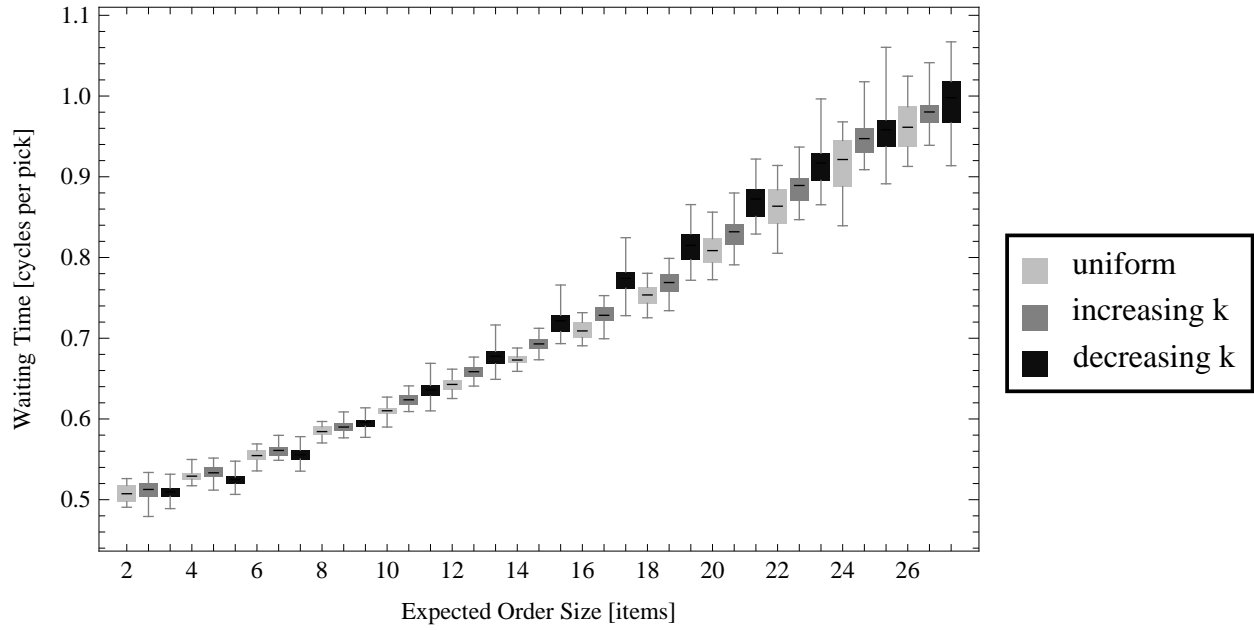


Figure 3.38: Waiting time for Variable k Configurations

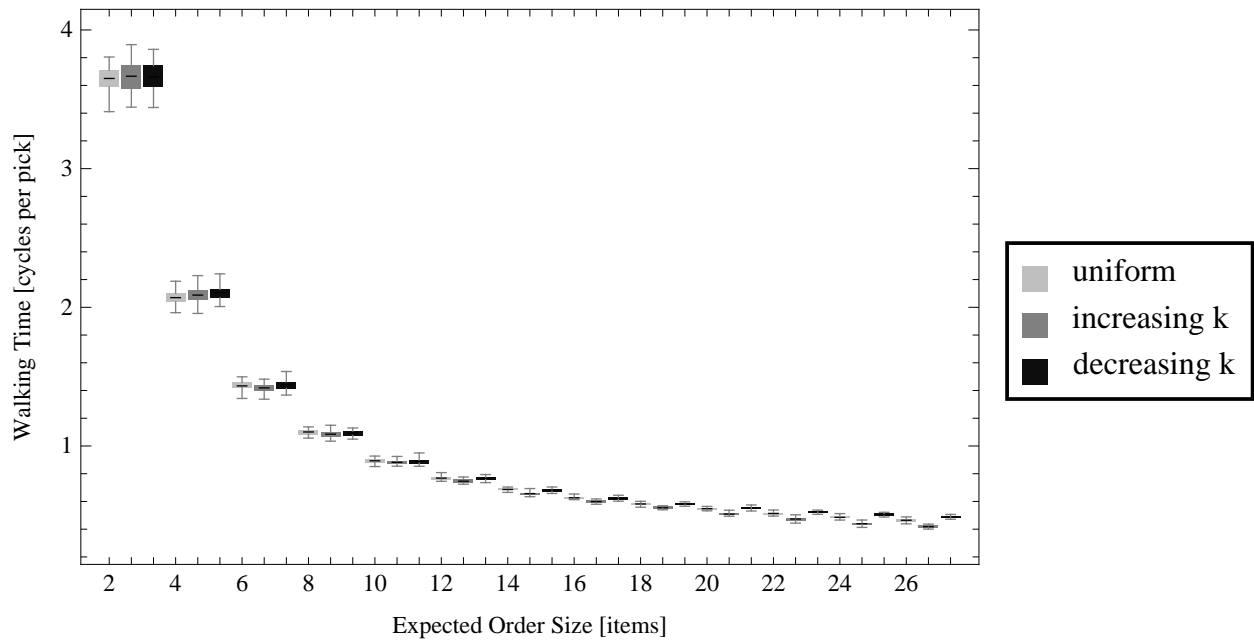


Figure 3.39: Walking time for Variable k Configurations

T values shows that one copy configuration is significantly better because of having shallow grid and shorter retrieval time (see Table 3.45). Two copy configuration is significantly

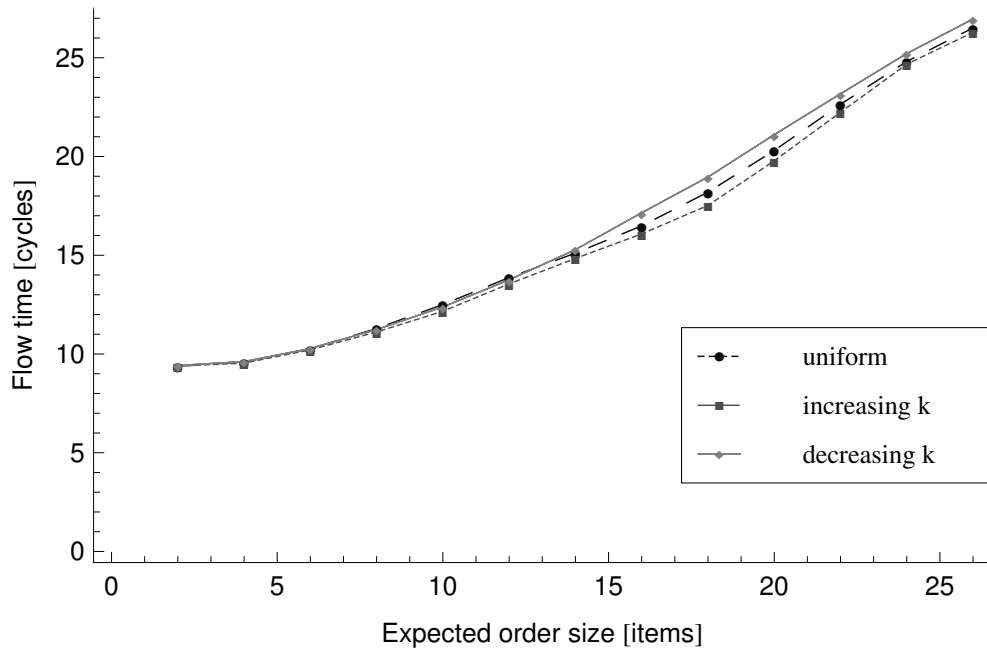


Figure 3.40: Average Retrieval Time Plot

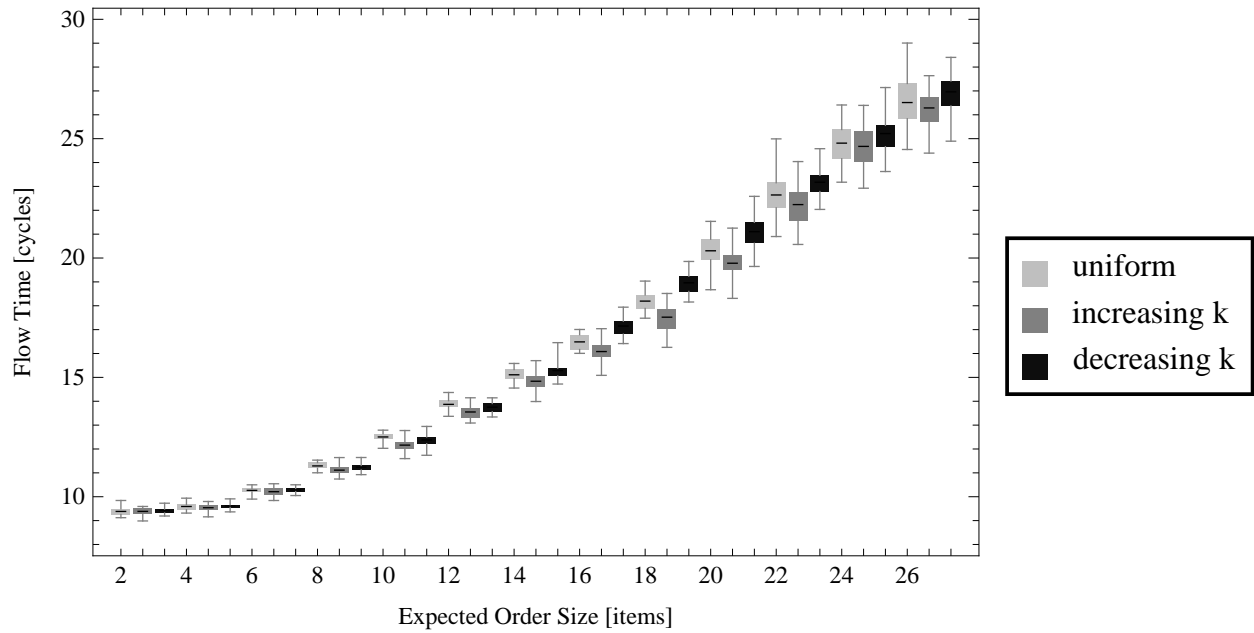


Figure 3.41: Flow time for variable k configurations

better than the three configuration due to same reasons shorter waiting time and average retrieval time.

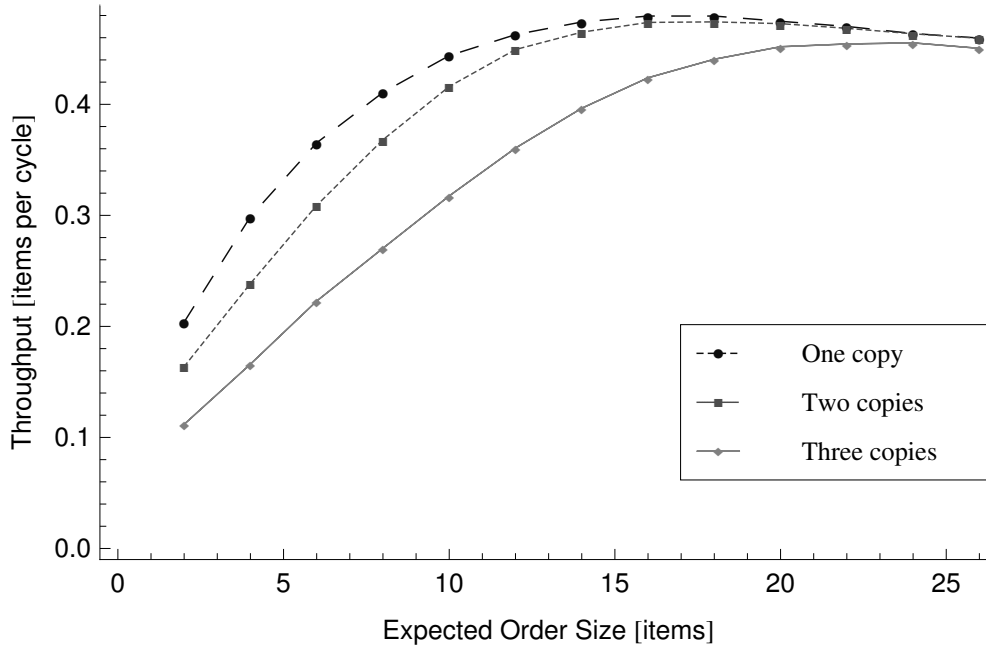


Figure 3.42: Throughput Plot for Multi Copies of SKUs

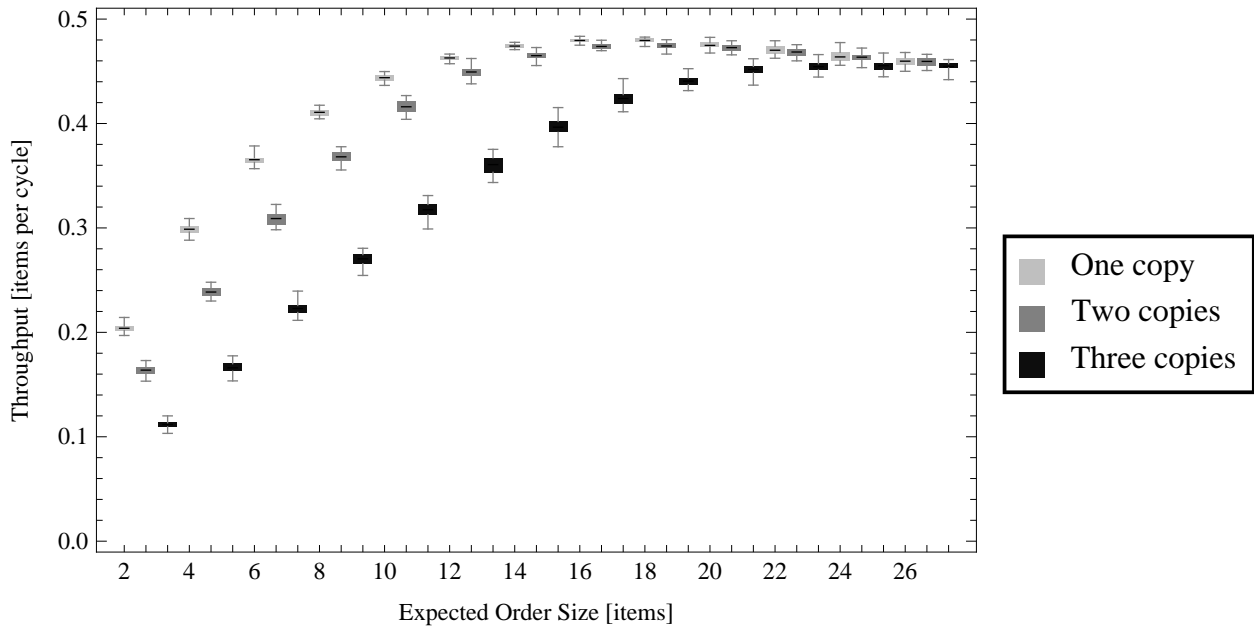


Figure 3.43: Throughput for multi copies

We also investigate the effect of having multiple copies of SKUs versus having a single copy from each SKU. Therefore, for one copy configuration, there are unique SKUs in the grid. Because the number of columns is the same for all configurations, the grid becomes

	one vs. two copy	one vs. three copy	two vs. three copy
2	0	0	0
4	0	0	0
6	0	0	0
8	0	0	0
10	0	0	0
12	0	0	0
14	0	0	0
16	0	0	0
18	0	0	0
20	0.057	0	0
22	0.204	0	0
24	0.811	0	0
26	0.93	0.109	0.1

Table 3.44: p values for throughput with mull copy configurations

	one vs. two copy	one vs. three copy	two vs. three copy
2	37.25	101.71	54.22
4	44.95	116.06	56.09
6	40.38	96.85	49.34
8	31.45	110.16	78.78
10	20.13	94	62.37
12	15.91	61.02	50.13
14	12.33	56.3	51.2
16	10.29	41.9	37.63
18	7.79	35.53	26.95
20	1.98	26.09	18.82
22	1.3	11.88	13.16
24	-0.24	6.93	8.51
26	-0.09	1.66	1.7

Table 3.45: t values for throughput with mull copy configurations

deeper with additional copies. Since we assume that storage containers do not deplete very often, density does not change over time.

Throughput rate is lower with more copies of each SKU (see Figures 3.42 and 3.44). When an arbitrary copy of the SKU is chosen, not necessarily closer item is chosen, which leads to preference with longer travel in some cases. Therefore, the worker waits longer for the requested items with the increased travel time of the storage containers. The significant

	one vs. two copy	one vs. three copy	two vs. three copy
2	(0.03764, 0.04201)	(0.090232, 0.093935)	(0.050287, 0.054229)
4	(0.05749, 0.06297)	(0.13093, 0.13562)	(0.07038, 0.07570)
6	(0.05309, 0.05876)	(0.13892, 0.14491)	(0.08243, 0.08956)
8	(0.04013, 0.04571)	(0.13701, 0.14219)	(0.09417, 0.09919)
10	(0.02426, 0.02975)	(0.12513, 0.13070)	(0.09760, 0.10422)
12	(0.012331, 0.015969)	(0.09845, 0.10528)	(0.08414, 0.09130)
14	(0.007966, 0.011134)	(0.07511, 0.08077)	(0.06566, 0.07112)
16	(0.004661, 0.006973)	(0.05216, 0.05752)	(0.04636, 0.05169)
18	(0.003891, 0.006659)	(0.03650, 0.04096)	(0.03092, 0.03600)
20	(-0.000061, 0.003695)	(0.021366, 0.025000)	(0.01905, 0.02369)
22	(-0.00088, 0.00393)	(0.01319, 0.01868)	(0.01217, 0.01665)
24	(-0.00261, 0.00206)	(0.00610, 0.01120)	(0.00678, 0.01107)
26	(-0.00242, 0.00222)	(-0.00230, 0.02187)	(-0.00200, 0.02177)

Table 3.46: CIs for throughput with mull copy configurations

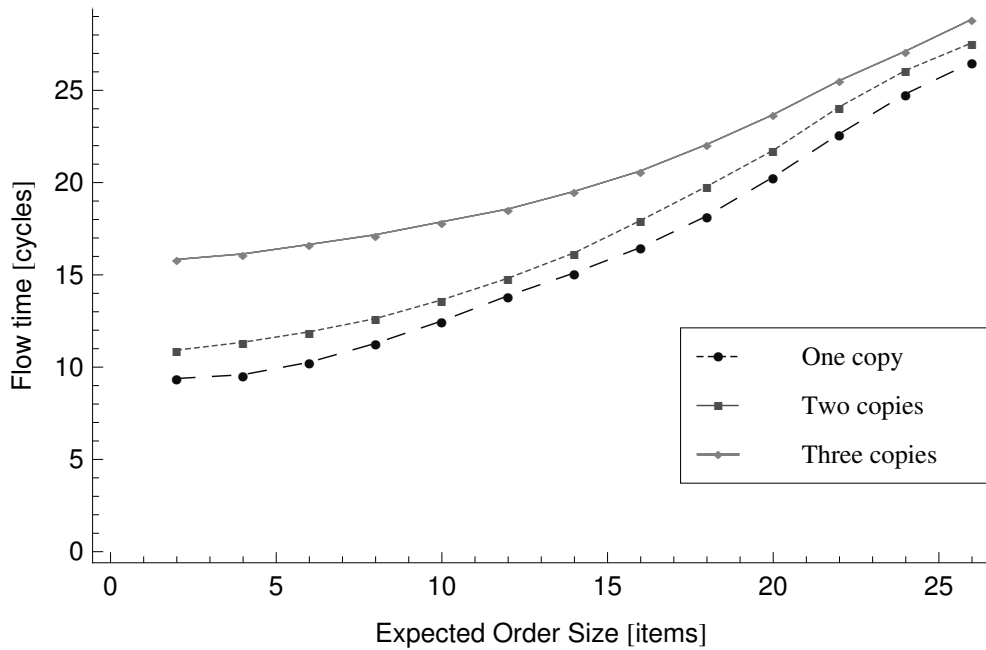


Figure 3.44: Average Retrieval Time Plot

difference between configurations decrease with the larger expected order sizes. In these cases, the worker is busy with the larger orders and the system can tolerate longer retrieval times. With the larger expected order sizes, traffic density increases in the grid, which causes the increase in the flow time in all configurations.

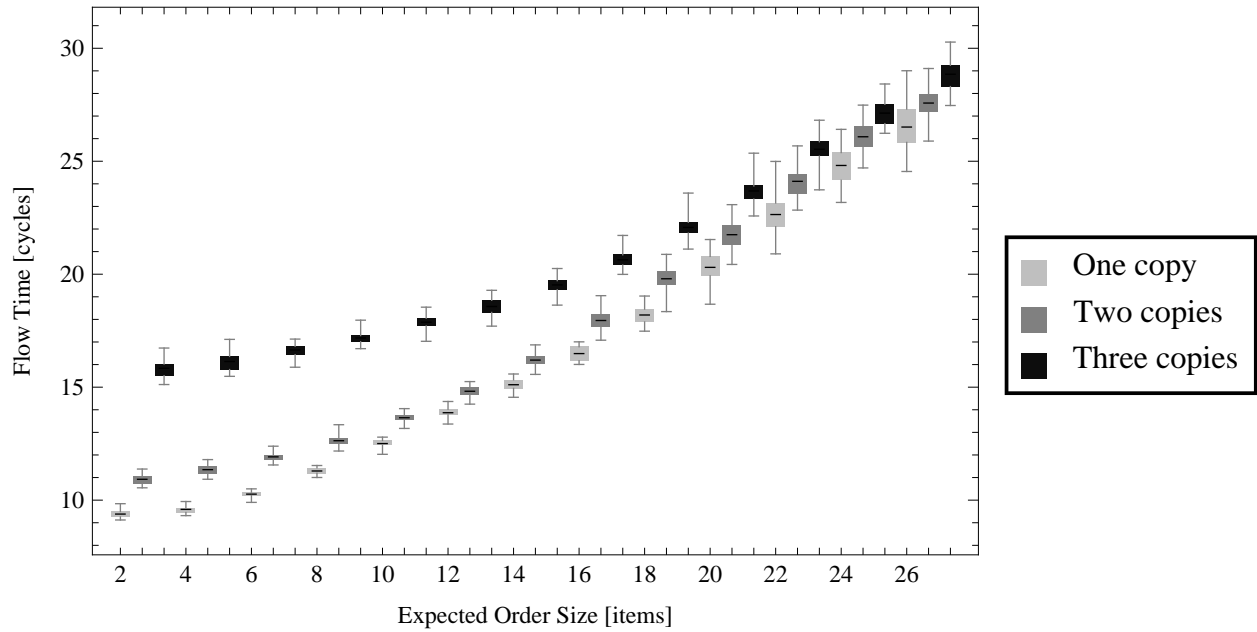


Figure 3.45: Flow time for multi copies

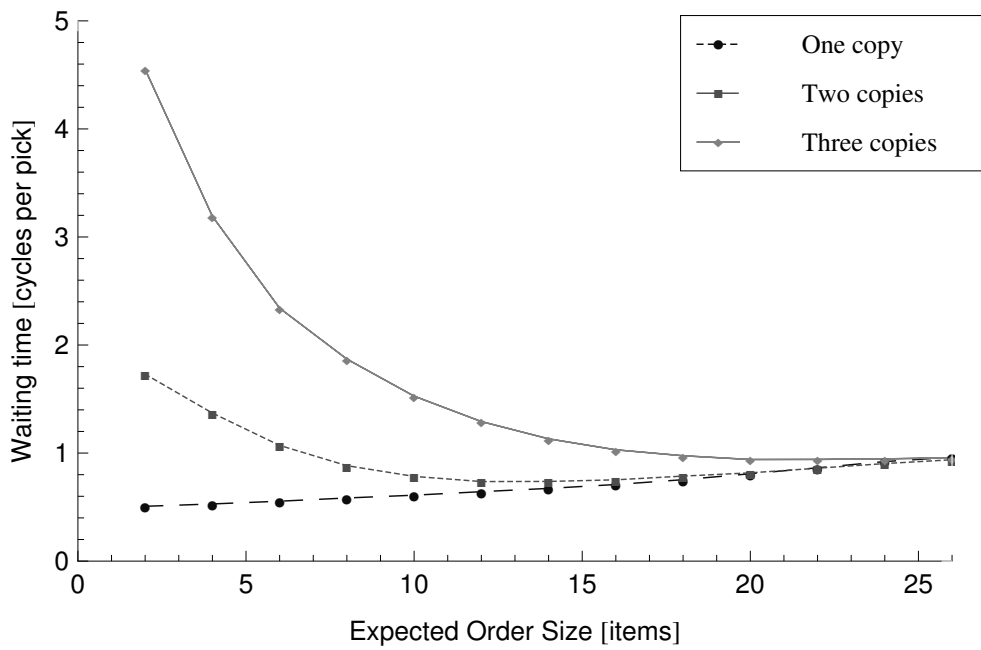


Figure 3.46: Waiting Time for Multi Copy Configurations

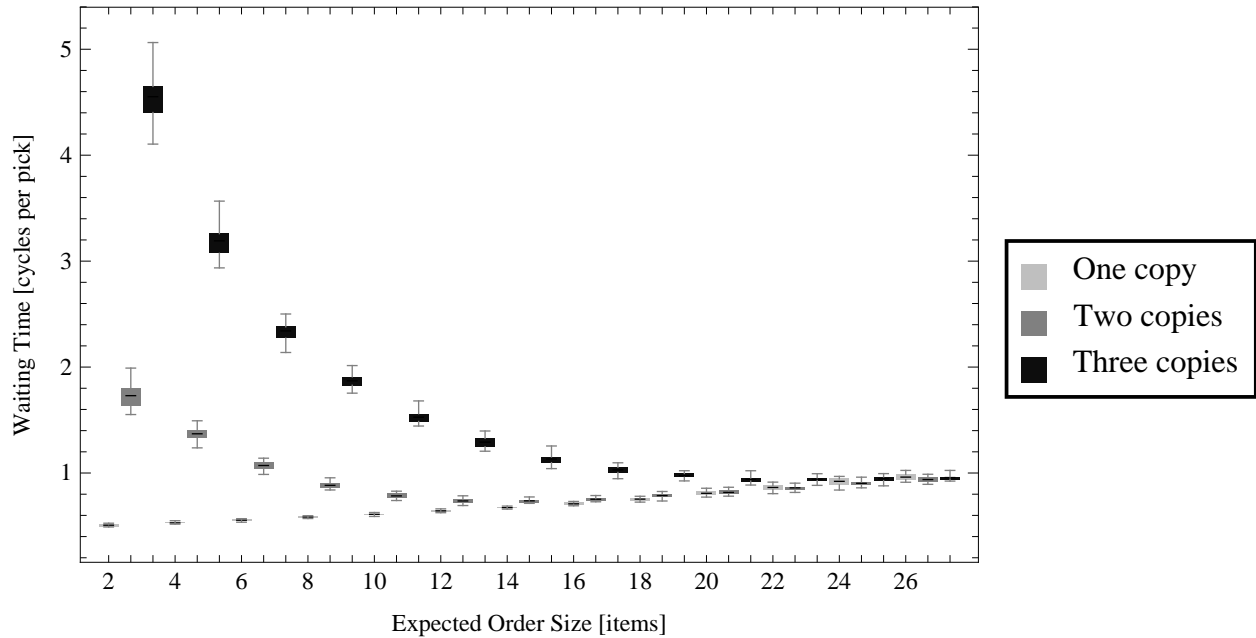


Figure 3.47: Waiting time for multi copy configurations

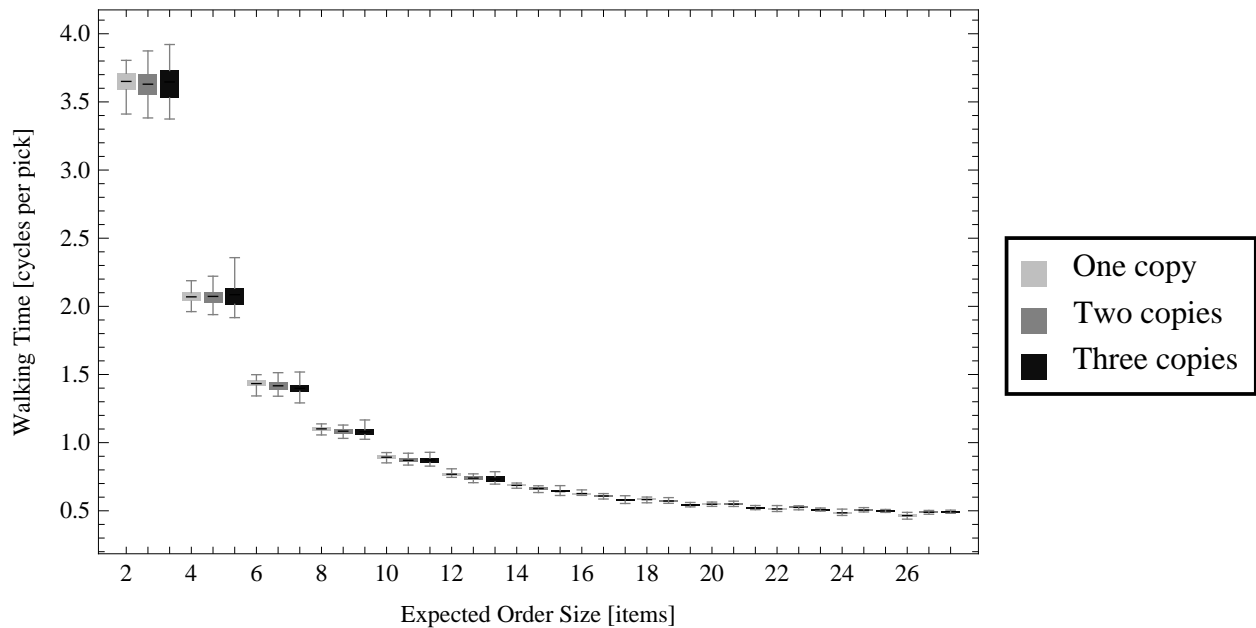


Figure 3.48: Walking time for multi copy configurations

3.5.4 Comparison with a Flow Rack Configuration

Flow rack is widely used in industry for high volume order picking operations. Flow rack guarantees a SKU density of one SKU per location. In flow rack, slot contain at most one SKU. GridPick allows effectively more than one SKU per slot.

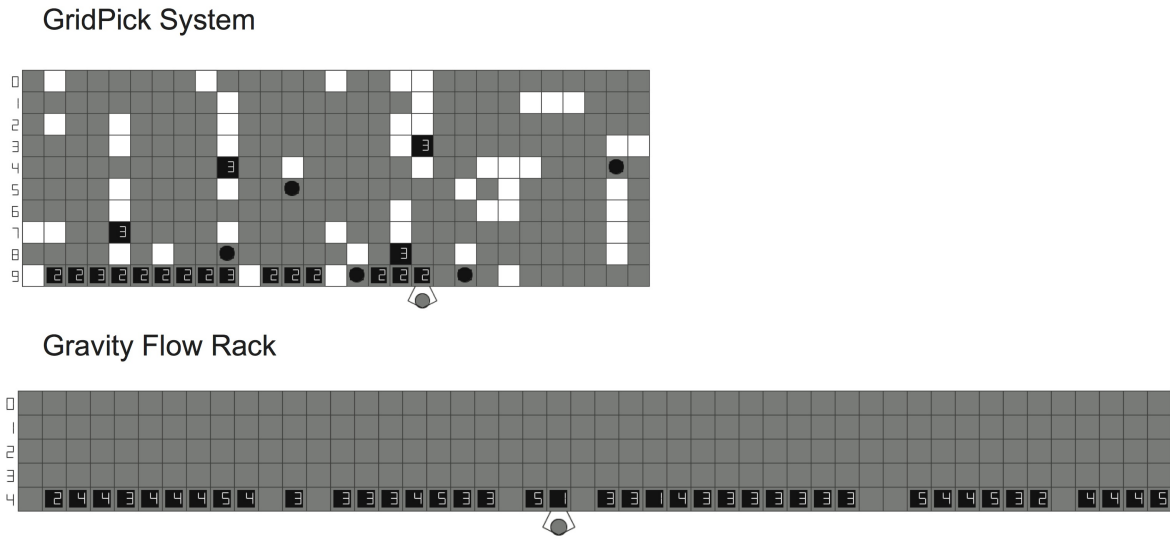


Figure 3.49: Visual Comparison of GridPick and the flow rack

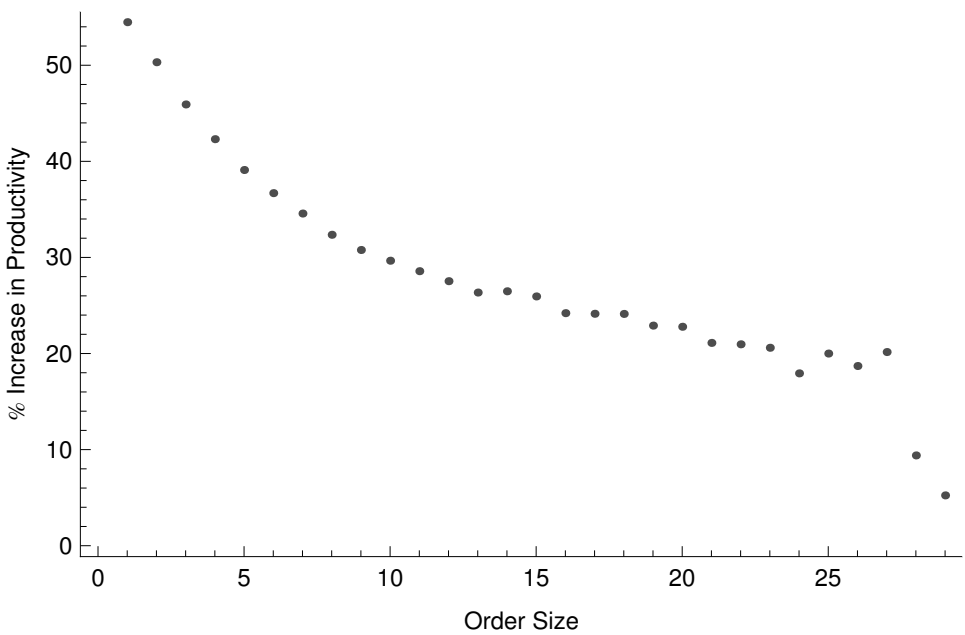


Figure 3.50: Productivity Increase in GridPick Compared to a Flow Rack

We compare the arrangements and performance of GridPick and an equivalent flow rack with the same number of SKUs. In both systems, there are 48 SKUs, 5 copies per SKU, and 240 storage containers. GridPick has 10 rows, 29 columns, and 5 empty cells per row. The flow rack configuration has 48 columns, and 5 rows. Having 48 columns is due to the

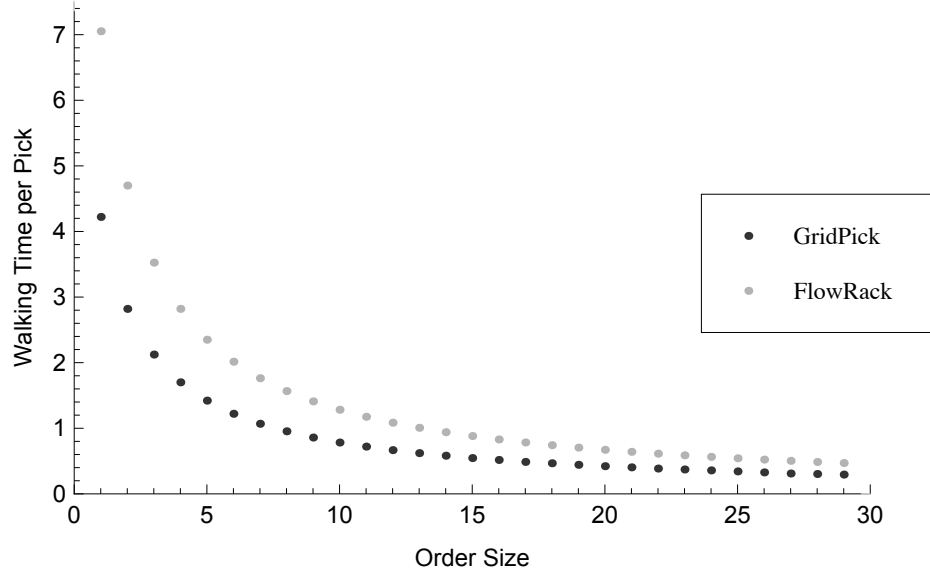


Figure 3.51: Comparison of Walking Times for the Flow Rack and the GridPick

presence of 48 SKUs. We assign a slot for each SKU. Figure 3.49 shows the orientation of both GridPick and the equivalent flow rack.

Figure 3.50 shows the percent difference in the throughput rate between GridPick and the flow rack configurations. There is a 20-50 % productivity improvement with GridPick. This is due to the decreased walking time per pick for all expected order sizes (see Figure 3.51). Because walking time in between picks decreases with the larger expected order sizes, improvement in the performance decreases with increasing expected order size. When the expected order size reaches the capacity of the pick face (number of columns), there is a sudden performance drop due to the traffic density in the pick face.

3.5.5 Conclusions

In brief, we introduce a new order picking system design that has a dynamically changing pick face. The requested items travel to the pick face when there is a demand for them. The system employs decentralized control as a design methodology, which enables flexibility and adaptability features. We develop the cell based rules that apply to each module. Decentralized control rules relieve the burden for the solution of large scale systems. GridPick provides

high SKU density, which results with a higher pick density. The system provides productivity improvement by reducing the travel time between items of an order. Simulation analysis shows the performance for different parameters and configurations. Expected order size, aspect ratio, and number of SKU copies are crucial factors on the performance. Performance improvement and flexibility are acquired with the cost of automation and algorithm development. Since the required hardware equipments are already available, the system can be implemented with current material handling technologies.

Chapter 4

A Modified Version of the GridPick System: Picking From Two Sides

We introduce a major enhancement to the one sided GridPick system that modifies the message passing algorithm to enable picking from two sides of the grid. In one sided GridPick, the system is constrained to the one side, which limits performance. Two sided picking offers another level of flexibility with the same sized grid. If there is a need for additional throughput capacity, an additional picker can work on the other side of the same grid to increase the picks per unit time. Therefore, we can scale the throughput by simply changing the labor use without extra costs of automation devices and inventory kept in the facility.

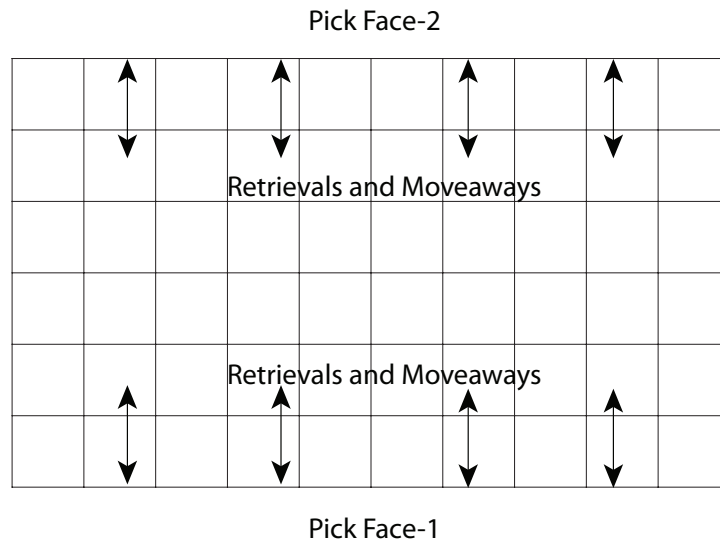


Figure 4.1: Abstract Representation of the Two Sided Picking Model

This problem is more difficult than one sided GridPick due to the higher traffic intensity, which may lead to conflicts and deadlocks if not managed properly (see Figure 4.1). There is a higher traffic intensity because the requested items are trying to move in two opposite directions.

Figure 4.1 shows the outline of the two sided GridPick model. There are requested items traveling to the two pick faces. Pick face-1 represents the bottom pick face and pick face-2 indicates the pick face on top. Requested items arrive at the edge of the system and workers on both sides are processing the orders by means of piece picking from cartons or carton picking from pallets. Whenever the worker processes the item, the storage container is free to move in the grid. We call the worker on pick face-1, worker-1, and the worker on pick face-2, worker-2.

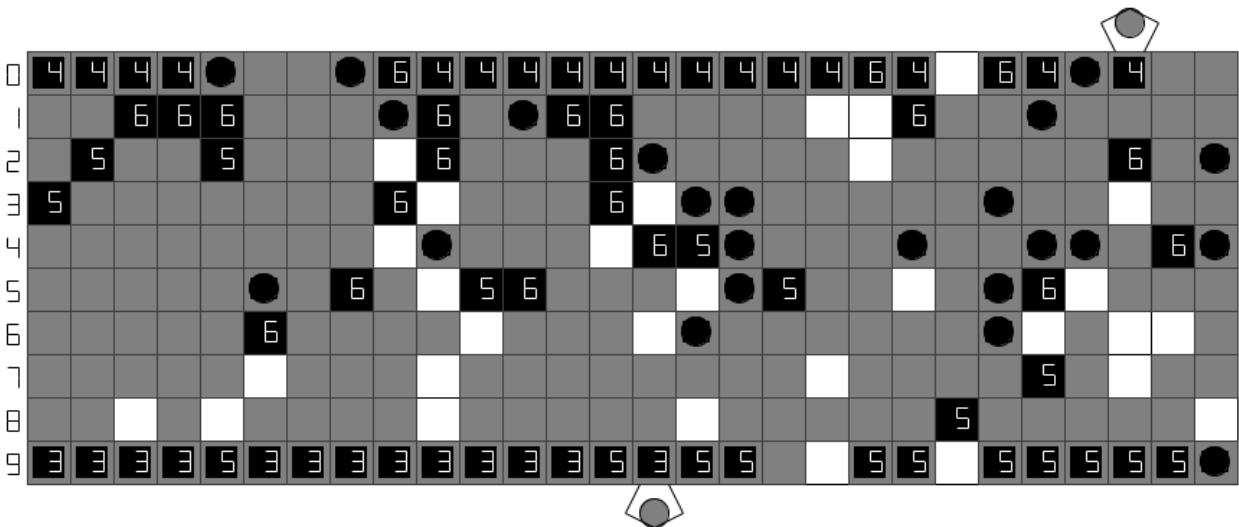


Figure 4.2: Visualization of the Two Sided Picking Model

Figure 4.2 represents a snapshot of the two sided picking model simulation. Worker-1 is picking order number 3, and items of order number 5 are still traveling to pick face-1. On the other side, worker-2 is picking order number 4, and items of order number 6 are still traveling to pick face-2.

By introducing this model, we expect to have a more convenient way of increasing throughput by enabling an alternative of one sided GridPick. With this model, in case of a higher demand, additional pickers can enable a larger throughput capacity. Additional hardware and equipment may not be available in these cases, especially when we consider the high cost of the investment. Therefore, with the same equipment, additional workers can provide required throughput capacity. Furthermore, this configuration does not require additional areas for a space restricted facility that needs extra activity for its operations. Without a two sided picking system, we would need a larger grid for additional performance. A basically equivalent system can be two one sided GridPick systems. This means additional inventory in the facility, which increases costs and decreases supply chain response time. We could add another worker to one sided GridPick on the same side of the grid. However, one side can only manage up to a certain traffic intensity. With the two sided GridPick enhancement, instead of having one two-deep picking location, we have two one-deep locations. Two sided picking enables a more shallow lane configuration than the one sided picking. It increases accessibility to the storage units by opening more space for order picking.

We will compare this model with the one sided GridPick model and will detect the performance improvement for the system. We do not address the optimization of which picking side is better for a particular order. Our study mainly achieves the design goals and fulfills the complex dynamics of decentralized control systems.

4.1 System Description

In the following negotiation figures, representation style is analogous to the rules representation of the one sided GridPick model. Row information is shown on the top right of the figure. It is the target row for the replenishing item. If it is a requested item, the number shows the departure or origin row. It indicates the current row for the stored item. Therefore, in the examples we can easily see the current row of a stored item. Arrow heads without

a bar show the initiation and response messages. Arrows with a bar show the forward and incoming messages (see Figure 4.3).

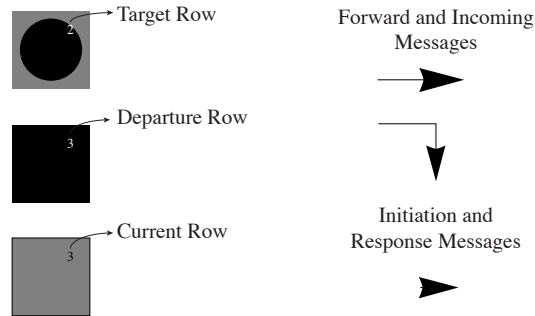


Figure 4.3: Representation of the States and Messages for the Negotiation Algorithm

4.2 Solution Approach

Similar to the initial model, we have adopted a decentralized control algorithm as a solution methodology. The conveyor module communicates with its four neighbors (north, south, east, and west). The decentralized control logic includes a series of events that initiate negotiations by passing messages between conveyor modules. Each conveyor module takes a state and negotiation condition depending on the state of the storage container and the phase of the negotiation. We will discuss the events that involve the balancing movements, which are the key events for the algorithm. These are the items moving to balance the requested items. There are also functions to setup initial configuration. We define events that update the states of the modules, and actuate the movement of the storage containers. Depending on the necessity of the information, messages between modules may include negotiation conditions of the modules, process number of the events, row information of the module, column number of the message sender and receiver, etc.

Another important point is: a conveyor module cannot send and receive multiple messages at the same time instance due to the discrete nature of the simulation. For instance,

during the initiation, the conveyor module cannot send messages to the east and west at the same time. One or the other is sent first. For this reason, we introduce randomness to the initiation by randomly choosing one of them to send first. Furthermore, the conveyor module cannot receive multiple messages at the same time instance, which prevents any conflict resulting from the incoming messages. Even if the incoming messages come from the same distance conveyor units, they will arrive the conveyor module in different time instances. Similar to the one sided GridPick model, vertical movements are prioritized over horizontal movements by having vertical movement negotiation events before the horizontal movement events. An important property for the system is: we have prioritized the movement towards the pick face-2 to avoid any conflict. We ensure this by executing the negotiation events of the items traveling towards pick face-2 before the items moving towards pick face-1.

4.2.1 Discrete Events and Balance Rules for the Operations

For handling the movement of requested items in two opposite directions, we have additional balance rules along with the rules already defined in one sided GridPick. While requested items move towards the pick faces, balancing items also move to balance the occupancy of the grid.

There are two types of requested items: *requested-1* and *requested-2* (see Table 4.1). Requested-1 needs processing on pick face-1 and requested-2 requires picking on pick face-2. There are also *replenishing* items that are moving to balance the movements of the requested items. *Replenishing-1* units move towards pick face-1, and *replenishing-2* units convey towards pick face-2, which will arrive at a target row and turn into a regular occupied storage unit.

There are also additional negotiating positions taken by the conveyor modules. If the requested item is about to change its row information, in the first step of the negotiation, it turns into *candidate to exchange*. If it is requested-1 originally it becomes *candidate to*









Symbol	Negotiation Condition	Definition
	Requested-1	The module has a requested storage container that will travel to pick face-1.
	Requested-2	The module has a requested item that will travel to pick face-2.
	Replenishing-1	The module has a replenishing item that is moving towards pick face-1.
	Replenishing-2	The module has a replenishing item that is moving towards pick face-2.
	Candidate Exchange-1	The module has a requested-1 item that may exchange its row information after the confirmation from the other negotiating side.
	Candidate Exchange-2	The module has a requested-2 item that may exchange its row information after the confirmation from the other negotiating side.
	Willing to be Replenishing-1	The module has an occupied item that will turn into replenishing-1 after the confirmation from the other negotiating side.
	Willing to be Replenishing-2	The module has an occupied item that will turn into replenishing-2 after the confirmation from the other negotiating side.

Table 4.1: Additional Negotiation Conditions

exchange-1. It turns into *candidate to exchange-2* if it is requested-2 originally. Also in the first step of the negotiation, if an occupied item receives a message to be replenishing, it turns into *willing to be replenishing-1* or *willing to be replenishing-2* depending on the direction it will move (see Table 4.1). There are 5 balance rules and 4 of them are defined for both requested-1 and requested-2 items, which makes a total of 9 balance rules.

This section includes tables for the examples of the balance rules and rules of the balancing movements. To understand the rule tables throughout the next section, the reader should follow a few guidelines for the negotiation events. These are the rules for the cases involving only state change(s) and/or trigger of message(s). State and message combinations that are not possible or do not necessitate an action are not shown in the tables. Also, for brevity, only rules coming from one side are shown. We have classified the rules with respect

to the type of action (forward, response, final state change, etc.). A module under certain states and conditions is responsible for the initiation of the respective event, which is checked by “if” statements. There is also a target group that will receive the messages and take action (response, forward, etc.) for the execution of the event. Other modules with unrelated states will forward the messages back and forth between the corresponding modules.

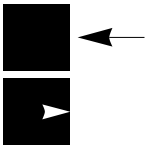
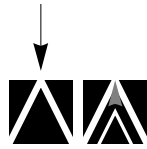
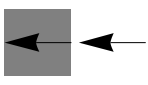
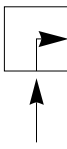
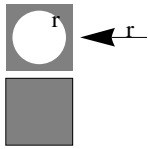
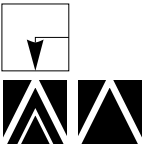
					
Response Message Type 1	Response Message Type 2	Forward Message Type 1	Forward Message Type 2	Final State Change Type 1	Final State Change Type 2

Table 4.2: Rule Examples

In the following rule tables, incoming messages and new states are shown. The reader should review the following guidelines for a better understanding of the communication rules. In the *response* message, if the module receives a message from the east or west neighbor, we show the new state of the module and the response message to the south of the module’s current state (see Table 4.2). If the module receives a message from the north or south, we show the new state and the response message to the east of the module’s current state. In the *forward* messages, the module passes the incoming message in the direction shown with the arrow. In the *final state change* rules, if the module receives a message from east or west, the new state of the module is shown to the south of the module’s current state. If the module receives a message from the north or south, the new state of the module is shown to the east of the module’s current state.

Balance rule-1 is between requested-1 and requested-2 items that executes the exchange of their row information. Therefore, the balance rule-1 regulates an existing balancing movement between requested items moving in opposite directions. Table 4.3 shows an example of

Instance of the Negotiation	Description
	<p>A requested-1 module initiates balance rule-1. Its current row is “3,” and it is at its departure row.</p>
	<p>On the left side of the initiated message, an empty module forwards the message to the south. On the right of the initiated message, an occupied module forwards the message to the east.</p>
	<p>On the left, a requested-2 module receives the message and responds to it by becoming <i>candidate to exchange-2</i>. It is at its departure row, which is “4.” On the right, an occupied module forwards the message to the east.</p>
	<p>On the left, an empty module forwards the response message to the east. On the right, another empty module forwards the message to the south.</p>
	<p>On the left, the requested-1 module responds to the message by sending its departure row information. It takes the row information from the first incoming message. Now, its departure row is “4.” On the right, requested-2 module responds to the message by becoming <i>candidate to exchange-2</i>.</p>
	<p>The empty modules forward the messages.</p>
	<p>On the left, the requested-2 module receives the message and takes the row information. Now, its departure row is “3,” which is the row it is about to move in. On the right, an occupied item forwards the message. Requested-1 module will respond to it with its exchanged row information, which will serve as a negative response to the requested-2 module. Remaining steps are omitted for brevity.</p>

Table 4.3: Steps of Balance Rule-1 for Requested-1 in Two Sided GridPick

balance rule-1 initiated by a requested-1 item. In this example, if the requested-1 item's departure row is smaller than or equal to its current row, it initiates balance rule-1 by sending a message to both east and west including the rule's process number, row information, and column number. Whenever an empty module receives the message, it forwards the message to the south. If there is a requested-2 item which is about to move to the north, it sends back its row information for a possible exchange. The requested-1 item takes the row information from the first incoming message and sends back its row information. Other incoming messages cannot execute the exchange. In this manner, requested items moving in opposite directions exchange their departure rows, which balances the system without an assignment of balancing items.

Initial Message(s)				
Response Message(s)				
Forward Message(s)				
Final State Change(s)				

Table 4.4: Communication Rules of Balance Rule-1 for Requested-1

Table 4.4 shows the communication rules for balance rule-1 for the requested-1 items. In the final state change rules, if the module receives row information other than its current row information, it updates its row information, and becomes the exchanging module in this

negotiation. Otherwise, it keeps its same row information, and remains as a requested-2 item.

A requested-2 item can also initiate balance rule-1 if its departure row is larger than or equal to its current row. Table 4.5 shows an example of balance rule-1 for the requested-2 item. The requested-2 item initiates the message by sending a message to both east and west neighbors. Whenever a message is received by an empty module, it forwards the message to the north. If there is a requested-1 module, it will respond to the message by sending its row information. The requested-2 module will record the row information from the first incoming message and will send its row information to the corresponding requested-1 module by specifying its column number. It will reply to other messages by simply sending the same incoming message without changing any information. Therefore, requested items moving in opposite directions will exchange their row information without assignment of any additional balancing items, which will prevent further traffic flow.

Table 4.6 indicates the rules of balance rule-1 for the requested-2 items. A requested-2 item initiates the row exchange before its movement. A requested-1 item in the row below replies to the message if its departure row is equal to or larger than its current row. In the final state change message, if the requested-1 item receives new row information, it exchanges its departure row number. Otherwise, it keeps the same row information.

A row exchange between two kinds of requested items can happen even if they are in the same row. Table 4.7 shows an example of balance rule-2. The requested-1 module initiates the negotiation by sending messages to both east and west if its departure row is smaller than its current row. A requested-2 module replies to the message if its departure row is larger than its current row. The requested-1 module evaluates the first incoming message, takes the row information, and sends its departure row. The requested-2 item receives the new departure row information. This rule does not require an empty spot in front of the requested item.


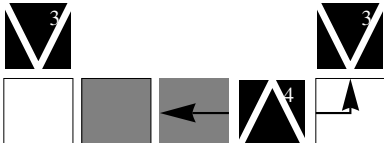





Instance of the Negotiation	Description
	<p>A requested-2 module initiates balance rule-1. Its current row is “4,” and it is at its departure row.</p>
	<p>On the left of the initiated message, an occupied module forwards the message to the west. On the right of the initiated message, an empty module forwards the message to the north.</p>
	<p>On the left, an occupied module forwards the message to the west. On the right, a requested-1 module responds to the message by turning into <i>candidate to exchange-1</i>.</p>
	<p>The empty modules forward the messages.</p>
	<p>The requested-2 module responds to the first incoming message with its departure row information. It takes the row information from the incoming message and makes its new departure row (3).</p>
	<p>The empty modules forward the messages.</p>
	<p>On the right, the requested-1 item takes the row information. Now, its departure row is “4,” which is the row it is about to move into. On the left, the respond message is on the way. It will be evaluated negatively. Remaining steps are not shown for brevity.</p>

Table 4.5: Steps of Balance Rule-1 for Requested-2 in Two Sided GridPick

Initial Message(s)				
Response Message(s)				
Forward Message(s)				
Final State Change(s)				

Table 4.6: Communication Rules of Balance Rule-1 for Requested-2

We list the communication rules for balance rule-2 in Table 4.8. If its departure row is smaller than its current row, a requested-1 item initiates the negotiation (see initial message). If a requested-2 item has a larger departure row than the current row, it responds to the message with its departure row information. The requested-1 item replies to the returning message with its departure row information (see response messages). Other items forward the incoming messages (see forward messages). In the final state change message, if the requested-2 item receives the new departure row information from the requested-1 item, it updates its row information. Otherwise, it keeps the same departure row number without a row change.

If the requested item cannot find another requested item to exchange its departure row, which is the convenient way without causing additional traffic flow. They initiate additional balance rules for the assignment of a balancing item, which replenishes to the departure row of the requested item for a more immediate balancing of occupancy. Balance rule-3 provides

Instance of the Negotiation	Description
1	The requested-1 module has already moved down, and its current row is larger than its departure row. The current row is “4,” and its departure row is “3.” It initiates balance rule-2 by sending messages to the both east and west.
2	On the left of the initiated message, an occupied module forwards the message. On the right of the initiated message, an empty module forwards the message.
3	On the left, a requested-2 module receives the message, and responds to the message by being <i>candidate to exchange-2</i> . On the right, an occupied item forwards the message to the east.
4	On the right, another requested-2 module responds to the message by being <i>candidate to exchange-2</i> . On the left, an occupied module forwards the response message.
5	The requested-1 module responds to the first incoming message with its row information. It takes the row information from the message.
6	Occupied and empty modules respond the messages.
7	On the left, the requested-2 module receives the message, and gets the row information. Now, its departure row is “3,” which is the row it is about to move in. On the right, the requested-1 module responds to the message with a negative reply.
8	The empty module forwards the response message.
9	The occupied item forwards the response message.
10	The requested-2 item receives the negative reply, and does not exchange its row information.

Table 4.7: Steps of Balance Rule-2 for Requested-1 in Two Sided GridPick


















Initial Message(s)				
Response Message(s)	 ← 	 ← 		
Forward Message(s)	 ← 	 ← 	 ← 	 ← 
Final State Change(s)	 ← 	 ← 		

Table 4.8: Communication Rules of Balance Rule-2 for Requested-1

a balancing item for the requested item to move into its departure row. It also encourages convoy movement by assigning an occupied item that is next to a requested item, which is going to move together vertically.

In balance rule-3, an occupied module forwards the message to the north, which is a possible replenishing item (see Table 4.9). If there is a requested-2 module in the north neighbor, it responds to the message. This message goes back to the initial requested-1 item. The requested-1 item evaluates the first incoming message, updates its departure row to current row number, and send its departure row information back. The occupied item receiving this message turns into a replenishing item with a target row, which is equal to the departure row of the requested-1 item. We list the communication rules of balance rule-3 for the requested-1 items in Table 4.10.

Table 4.11 shows an example of balance rule-3 for the requested-2 modules. If the departure row of the requested-2 item is larger than the current row, it initiates balance rule-3. Unrelated items forward the messages, and an occupied item forwards the message to the south neighbor. If there is a requested-1 module in the south, it replies back to this message. The requested-2 item evaluates the first incoming message, updates its departure

Instance of the Negotiation	Description
	<p>A requested-1 module initiates balance rule-3 by sending messages to both east and west.</p>
	<p>On the left of the initiated message, an occupied module forwards the message to the north. On the right of the initiated message, an empty module forwards the message to the east.</p>
	<p>On the left, a requested-2 module responds to the message, so, it is known that there is an requested-2 module, which will eventually move. On the right, an occupied module forwards the message to the north.</p>
	<p>On the left, an occupied module forwards the message to the east. On the right, requested-2 module replies to the message.</p>
	<p>On the left, a requested-1 module responds to the message with its row information. It takes the current row information from the message.</p>
	<p>Upon receiving the message, the occupied will turn into replenishing, and it will take the row information of the requested-1 module as a target row. The requested-1 module will not respond to the message coming from the other side.</p>

Table 4.9: Steps of Balance Rule-3 for Requested-1 in Two Sided GridPick



















Initial Message(s)				
Response Message(s)	 	 		
Forward Message(s)	 	   	 	 
Final State Change(s)	 			

Table 4.10: Communication Rules of Balance Rule-3 for Requested-1

row to the current row number, and sends a reply. The occupied item receives this reply message, turn into a replenishing by taking the departure row of the requested-2 item as the target row information. This means the replenishing item should move into the row indicated by the target row. We document the rules of balance rule-3 for the requested-2 items in Table 4.12.

We have also adopted balance rules similar to the ones defined in one sided GridPick. Balance rule-4 is analogous to balance rule-1 of one sided GridPick, but it is defined for both requested-1 and requested-2 modules. We have shown an example and rules of balance rule-4 for a requested-2 item (see Tables 4.13 and 4.14), not for a requested-1 item for brevity. The rules and example for the requested-1 items would be similar to the requested-2 items. In the example for balance rule-4, a requested-2 module initiates the negotiation if its departure row is equal to or larger than the current row number (see Table 4.13).

We show an example of two competing requested-2 items. An occupied item forwards the message to the north. If there is an occupied item, it evaluates the first incoming message,

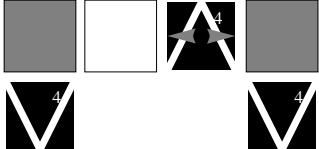
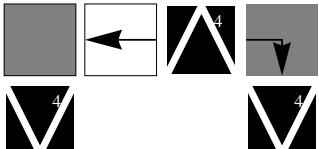
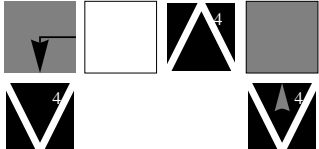
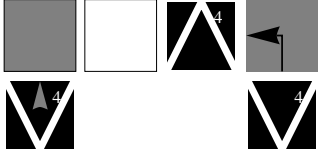
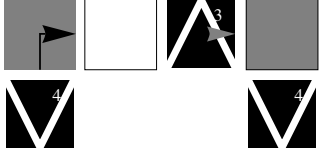
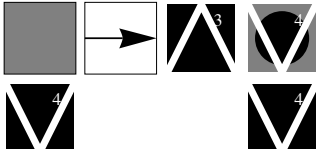
Instance of the Negotiation	Description
	<p>A requested-2 module initiates balance rule-3 by sending message to both east and west.</p>
	<p>On the left of the initiated message, an empty module forwards the message to the west. On the right of the initiated message, an occupied module forwards the message to the south.</p>
	<p>On the right, a requested-1 item responds to the message. So, it is known that there is a requested-1 module that will move. On the left, an occupied module forwards the message to the south.</p>
	<p>On the left, a requested-1 module responds to the message. So, it is known that there is a requested-1 module that will eventually move south. On the right, an occupied module forwards the message to the West.</p>
	<p>On the left, an occupied module forwards the message to the East. On the right, the requested-2 module responds to the message with its row information. Also, it takes the current row information.</p>
	<p>Upon reception of the message, the occupied module turns into replenishing. It takes the row information as its target row. The requested-1 module does not respond to the message coming from the other side.</p>

Table 4.11: Steps of Balance Rule-3 for Requested-2 in Two Sided GridPick


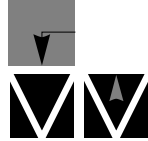
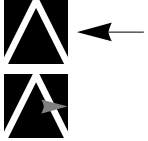


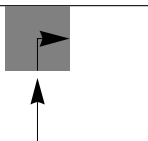

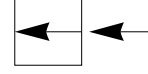
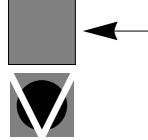
Initial Message(s)				
Response Message(s)				
Forward Message(s)		 		
Final State Change(s)				

Table 4.12: Communication Rules of Balance Rule-3 for Requested-2

replies to it, and becomes willing to be replenishing-1. With the response message, it takes the departure row of the requested-1 item as its target row, and turn into a replenishing item. We indicate the general rules of balance rule-4 for the requested-2 modules in Table 4.14.

Balance rule-5 for two sided GridPick is similar to balance rule-2 of one sided GridPick. We show the example and the rules of balance rule-5 only for the requested-2 items (see Tables 4.15 and 4.16). We show an example of two competing requested-2 items in the example (see Table 4.15). The occupied item evaluates the first incoming message. The requested-2 item updates its departure row to the current row number. In the last message wave, the occupied item turns into a replenishing item, and takes the departure row of the requested-2 item as its target row. We indicate the rules of balance rule-5 for the requested-2 modules in Table 4.16.

Instance of the Negotiation	Description
1	Two requested-2 modules initiate balance rule-4 by sending message to both east and west. In this example, we show the competition for an empty module.
2	Occupied modules forward the messages.
3	The empty module forwards the message to the north. The occupied item forwards the message to the west.
4	An occupied item responds to the message, and sends its row information by turning into <i>willing to be replenishing-1</i> . The occupied item does not respond to the other incoming message.
5	The empty module forwards the message to the west.
6	The occupied module forwards the message to the west.
7	The requested-2 module responds to the message with its row information. It takes the current row of the occupied module as its new row information.
8	The occupied module forwards the message to the east.
9	The empty module forwards the message to the north.
10	Upon reception of the message, the occupied module turns into replenishing module, by taking the row information as its target row, which is the row it is about to move.

Table 4.13: Steps of Balance Rule-4 for Requested-2 in Two Sided GridPick

Initial Message(s)				
Response Message(s)				
Forward Message(s)				
Final State Change(s)				

Table 4.14: Communication Rules of Balance Rule-4 for Requested-2

Instance of the Negotiation	Description
1	Two requested-2 modules initiates balance rule-5 by sending messages to both east and west.
2	Empty modules forward the messages.
3	The occupied module responds to the first incoming message from the east with its current row information, and it turns into <i>candidate to be replenishing-1</i> . It does not respond to the message coming from the west.
4	The empty module forwards the message to the east.
5	The requested-2 module responds to the message with its row information. It takes the row information of the occupied module.
6	The empty module forwards the message to the west.
7	The occupied module receives the message, and turns into replenishing by taking the row information of the requested-2 module.

Table 4.15: Steps of Balance Rule-5 for Requested-2 in Two Sided GridPick

Initial Message(s)			
Response Message(s)			
Forward Message(s)			
Final State Change(s)			

Table 4.16: Communication Rules of Balance Rule-5 for Requested-2

4.3 Determination of Buffer Lengths for the Series of Events

Again, we have captured the negotiation length of the message dependent events. We have obtained 99.5 % confidence intervals, which provides at least 98% overall confidence according to the Bonferroni Inequality.

Variable	N	Mean	StDev	SE Mean	99.5% CI	UCL
NS-rep	50	1.38	0.5303	0.075	(1.1595, 1.6005)	1.605
EW-rep	50	3.26	2.648	0.375	(2.159, 4.361)	4.385
NS-req	50	1.48	0.6773	0.0958	(1.1984, 1.7616)	1.7674
EW-req	50	3.208	2.699	0.371	(2.121, 4.294)	4.321

Table 4.17: Length of the Message Passing for Negotiation of Events

We have determined upper control limits for all negotiation events. Since we are interested in the upper control limits, UCL provides us a inter-startup time for the events. UCL is obtained by adding three standard errors to the mean.

4.4 Statistical Analysis for Steady-State Parameters

4.4.1 Determining the Warmup Period

For the throughput, results are low initially, and waiting times are longer at the startup of the simulation. Since initial bias is very short at the beginning of the simulation, a warmup period of “200” is sufficiently large.

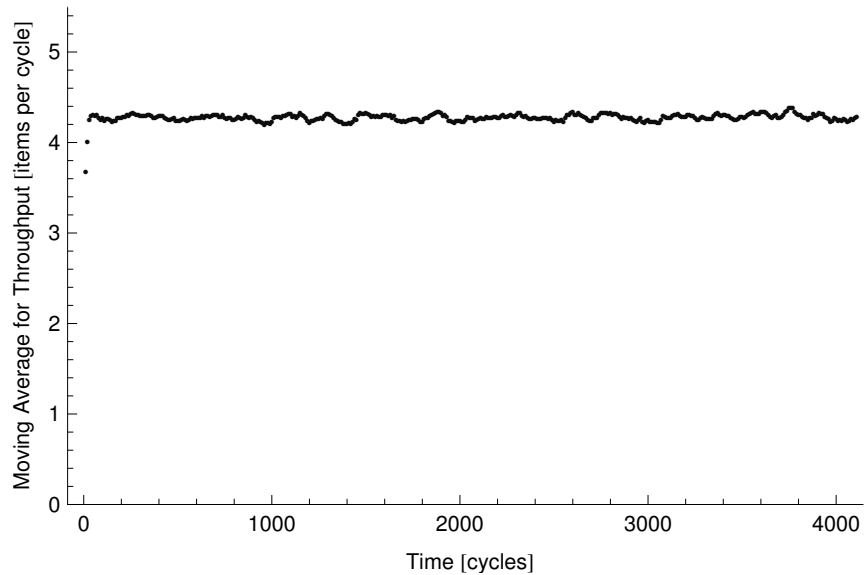


Figure 4.4: Initial Transient for Throughput

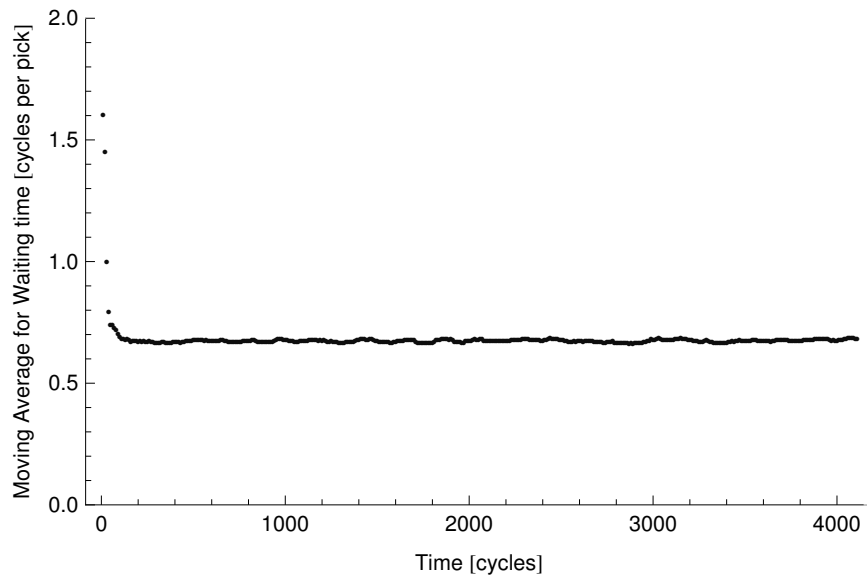


Figure 4.5: Initial Transient for Waiting Time

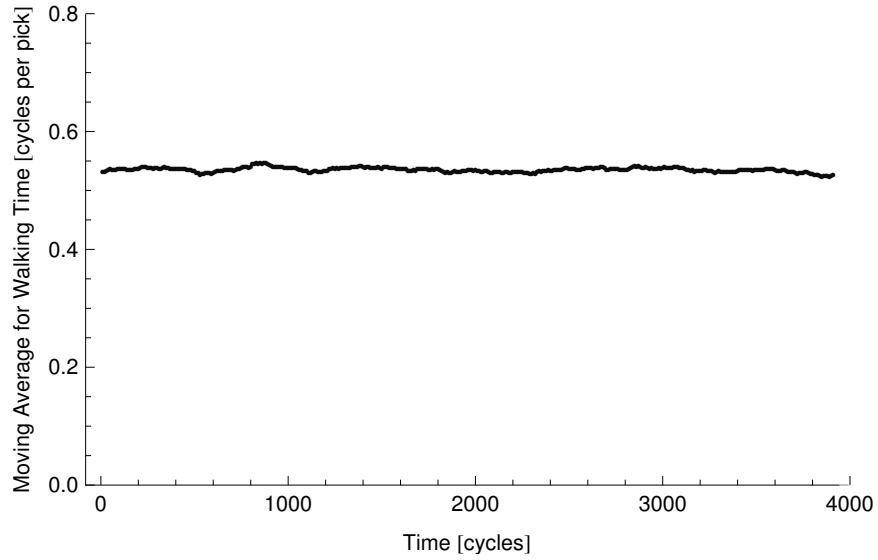


Figure 4.6: Initial Transient for Walking Time

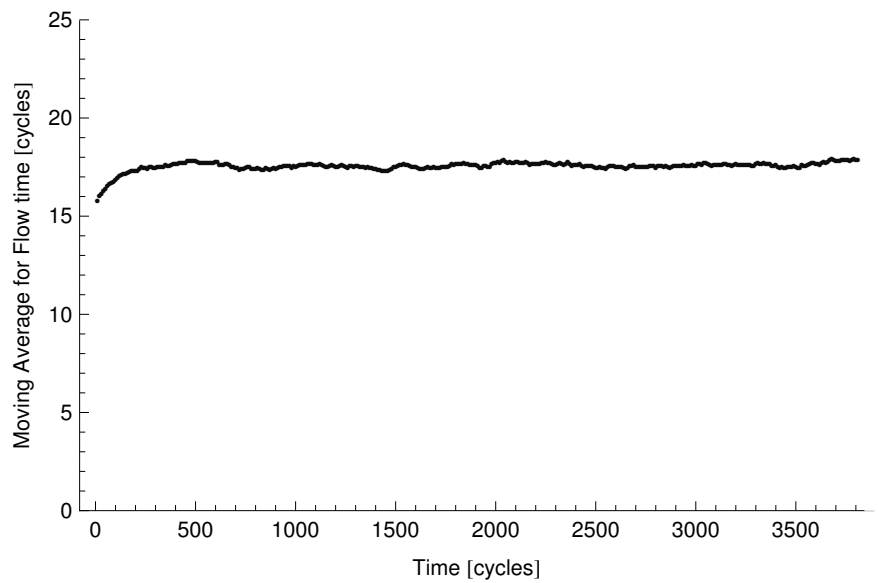


Figure 4.7: Initial Transient for Flow Time

4.4.2 Determining the Number of Replications

Since halfwidths are not larger than 1% of the mean for all confidence intervals, a value of 30 is sufficient for the number of replications. We have obtained at least 96% overall confidence by having 99% confidence intervals for each performance measure.

	Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	%Mean
thr.	k3-14	30	0.86712	0.00608	0.00111	(0.86406, 0.87019)	0.00307	0.35 %
walking	k3-14	30	0.67007	0.00926	0.00169	(0.66541, 0.67473)	0.00466	0.70 %
waiting	k3-14	30	0.88443	0.01484	0.00271	(0.87697, 0.89190)	0.00747	0.84 %
flow	k3-14	30	16.9712	0.3079	0.0562	(16.8162, 17.1261)	0.1549	0.91 %

Table 4.18: Confidence Intervals and Statistical Data for 30 Replications

4.5 Performance Analysis

We have performed a number of analyses to evaluate the performance of two sided GridPick. Similar to one sided GridPick, the main performance measures are pick per unit time (throughput), walking time per pick, waiting time per pick, and the average retrieval time per item (flow time). We have used an order profile of various expected order sizes with Poisson distribution. We evaluate the effect of expected order size and empty cells per row (k) as important system parameters.

We also consider the effect of aspect ratio on performance. We compare the performance of two workers to detect any possible bias. Furthermore, we perform variable k analysis to see the effect of the distribution of empty cells on the productivity. Finally, we detect the performance improvement compared to one sided GridPick. Considered assumptions (conveyor speed, walking speed, etc.) are analogous to one sided GridPick.

4.5.1 Performance of the Two Sided Model for Several Parameters

For the experiments with different expect order sizes and k values, we have used following parameters and assumptions:

There are 25 occupied columns, 10 rows and $k = 2, 3$ and 4 empty columns in the grid. Therefore, each configuration has 250 storage containers. Each simulation run has length of 4,000 iterations, 200 iterations for the warmup period, and 30 replications. We have used AnyLogic as the simulation software package.

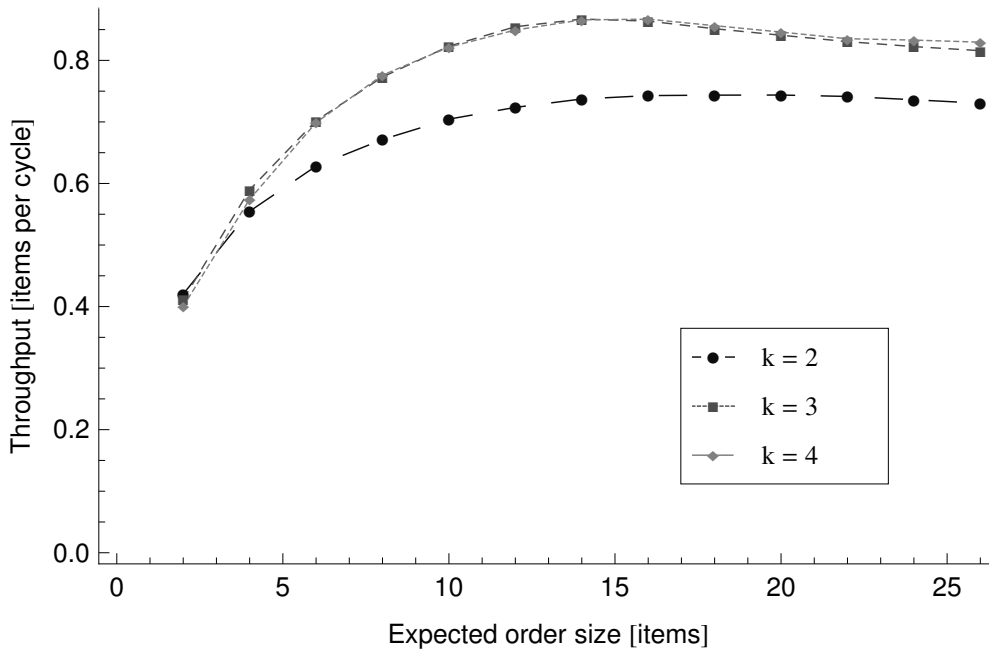


Figure 4.8: Throughput Plot for Different Expected Order Size Levels

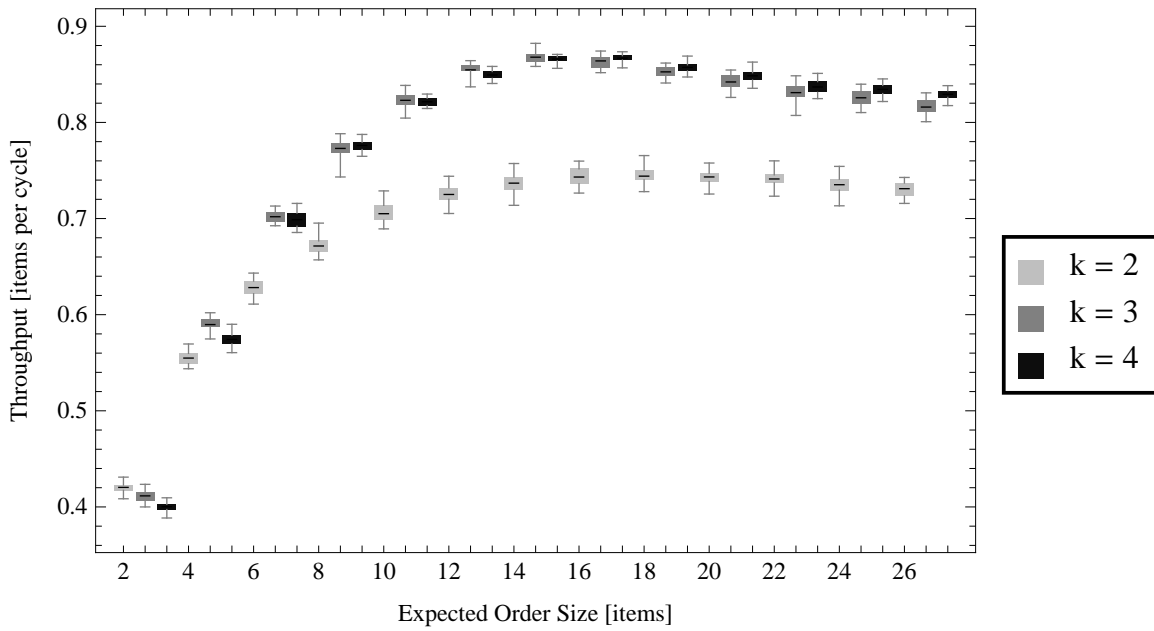


Figure 4.9: Throughput Plot for Different Expected Order Size Levels

	k=2 vs. k=3	k=2 vs. k=4	k=3 vs. k=4
2	0	0	0
4	0	0	0
6	0	0	0.151
8	0	0	0.185
10	0	0	0.324
12	0	0	0
14	0	0	0.264
16	0	0	0.027
18	0	0	0.002
20	0	0	0
22	0	0	0.002
24	0	0	0
26	0	0	0

Table 4.19: p values of Throughput with Different Expected Order Size Levels

	k=2 vs. k=3	k=2 vs. k=4	k=3 vs. k=4
2	5.62	14.62	8.39
4	-19.43	-11.54	8.52
6	-46.67	-37.33	1.48
8	-49.75	-48.59	-1.36
10	-64.36	-63.29	1
12	-75.38	-80.01	4.7
14	-65.76	-67.61	1.14
16	-62.37	-72.79	-2.32
18	-41.13	-49.79	-3.47
20	-54.37	-65.86	-4.18
22	-49.25	-52.02	-3.33
24	-49.62	-55.03	-6.38
26	-48.22	-57.52	-9.72

Table 4.20: t values of Throughput with Different Expected Order Size Levels

	k=2 vs. k=3	k=2 vs. k=4	k=3 vs. k=4
2	(0.00562, 0.01205)	(0.01735, 0.02300)	(0.00858, 0.01411)
4	(-0.03853, -0.03119)	(-0.02289, -0.01600)	(0.01172, 0.01912)
6	(-0.07693, -0.07047)	(-0.07459, -0.06684)	(-0.00115, 0.00712)
8	(-0.10567, -0.09733)	(-0.10898, -0.10017)	(-0.00771, 0.00156)
10	(-0.12187, -0.11436)	(-0.12038, -0.11284)	(-0.00157, 0.00459)
12	(-0.13452, -0.12741)	(-0.12887, -0.12245)	(0.00300, 0.00762)
14	(-0.13391, -0.12583)	(-0.13246, -0.12469)	(-0.00103, 0.00361)
16	(-0.12456, -0.11665)	(-0.12724, -0.12028)	(-0.00592, -0.00038)
18	(-0.11377, -0.10299)	(-0.11784, -0.10854)	(-0.00764, -0.00197)
20	(-0.10076, -0.09346)	(-0.10546, -0.09911)	(-0.00771, -0.00264)
22	(-0.09289, -0.08548)	(-0.09758, -0.09020)	(-0.00760, -0.00182)
24	(-0.08983, -0.08272)	(-0.10046, -0.09326)	(-0.01398, -0.00719)
26	(-0.08954, -0.08226)	(-0.10337, -0.09628)	(-0.01685, -0.01100)

Table 4.21: CIs of Throughput with Different Expected Order Size Levels

	k=2 vs. k=3	k=2 vs. k=4	k=3 vs. k=4
2	0	0	0
4	0	0	0
6	0	0	0
8	0	0	0
10	0	0	0
12	0	0	0
14	0	0	0
16	0	0	0
18	0	0	0
20	0	0	0
22	0	0	0
24	0	0	0
26	0	0	0

Table 4.22: p values of Walking Time with Different Expected Order Size Levels

Throughput of two sided GridPick increases with the expected order size up to approximately expected order size of 14. Then, it stabilizes, and there is a minor decrease (see Figure 4.8). The main reason behind this phenomenon is the traffic capacity of the system. Up to a certain number of active items (around the expected order size of 14), it handles additional traffic, and throughput keeps increasing. After the expected order size of 14, the system reaches its traffic volume, and it starts to perform a little worse due to greater

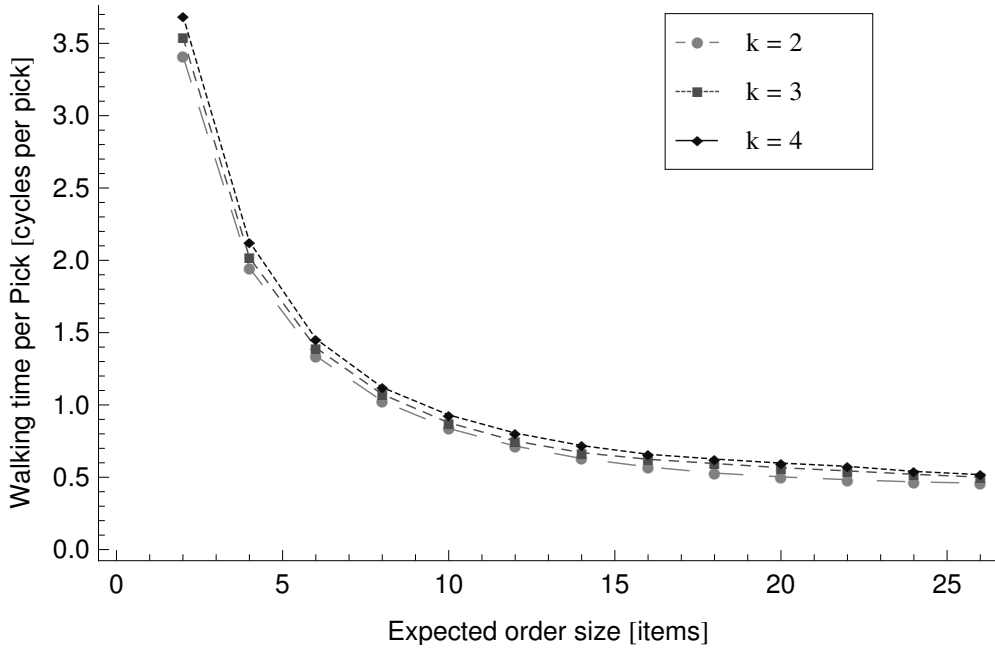


Figure 4.10: Walking Time per Pick for Different Expected Order Size Levels

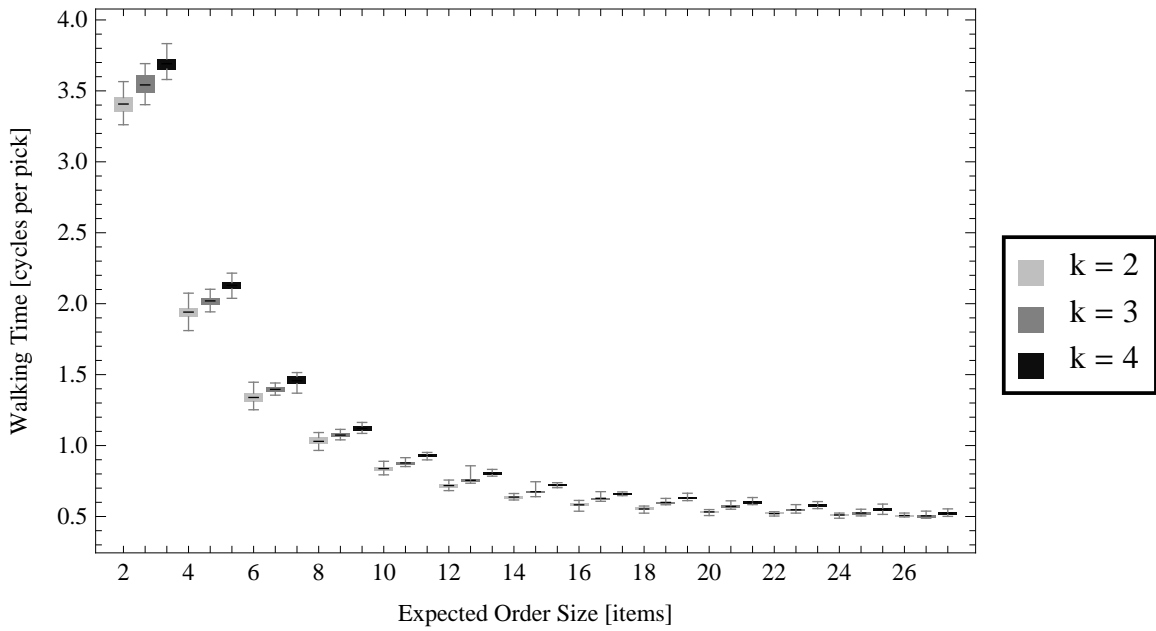


Figure 4.11: Walking Time per Pick for Different Expected Order Size Levels

congestion. More empty cells per row (k) has a diminishing benefit on the performance. From $k = 2$ to $k = 3$, throughput improvement is more significant; from $k = 3$ to $k = 4$, performance differs insignificantly (see Figure 4.8). This result suggests that the number of

	k=2 vs. k=3	k=2 vs. k=4	k=3 vs. k=4
2	-7.4	-17.74	-8.85
4	-7.73	-17.68	-10.38
6	-9.8	-16.06	-7.92
8	-8.15	-18.66	-8.08
10	-8.86	-25.26	-14.86
12	-13.05	-27.12	-18.61
14	-18.45	-37.19	-20.5
16	-27.31	-46.34	-17.01
18	-25.8	-56.68	-16.36
20	-28.45	-53.94	-12.64
22	-27.9	-40.99	-14.91
24	-27.49	-29.8	-7.35
26	-23.7	-25.65	-6.65

Table 4.23: t values of Walking Time with Different Expected Order Size Levels

	k=2 vs. k=3	k=2 vs. k=4	k=3 vs. k=4
2	(-0.1647, -0.0934)	(-0.3105, -0.2463)	(-0.1839, -0.1148)
4	(-0.09012, -0.05243)	(-0.2013, -0.1596)	(-0.1307, -0.0877)
6	(-0.06921, -0.04530)	(-0.13357, -0.10339)	(-0.07703, -0.04543)
8	(-0.05863, -0.03510)	(-0.10434, -0.08373)	(-0.05912, -0.03523)
10	(-0.04832, -0.03020)	(-0.10080, -0.08570)	(-0.06142, -0.04656)
12	(-0.04058, -0.02958)	(-0.09683, -0.08325)	(-0.06100, -0.04892)
14	(-0.04529, -0.03625)	(-0.09461, -0.08475)	(-0.05379, -0.04403)
16	(-0.05689, -0.04896)	(-0.09122, -0.08351)	(-0.03858, -0.03030)
18	(-0.06961, -0.05938)	(-0.09932, -0.09240)	(-0.03528, -0.02744)
20	(-0.06784, -0.05874)	(-0.09847, -0.09128)	(-0.03669, -0.02647)
22	(-0.06525, -0.05634)	(-0.09673, -0.08753)	(-0.03564, -0.02704)
24	(-0.05716, -0.04924)	(-0.07782, -0.06782)	(-0.02508, -0.01416)
26	(-0.04562, -0.03837)	(-0.06406, -0.05460)	(-0.02267, -0.01200)

Table 4.24: CIs of Walking Time with Different Expected Order Size Levels

empty cells is a restricting factor on the performance up to the $k = 3$ configuration. If we increase empty cells from $k = 3$ configuration to $k = 4$ configuration, empty cells do not add additional value to the system. Because there are already sufficient empty cells to the traffic density. Because there is less walking time for each pick and the worker walks less from pick to pick, walking time per pick gets lower with the larger expected order sizes (see

Figure 4.10). The performance difference with various k values is not related to the walking time, because there is not much change on the length of the pick face (see Figure 4.10).

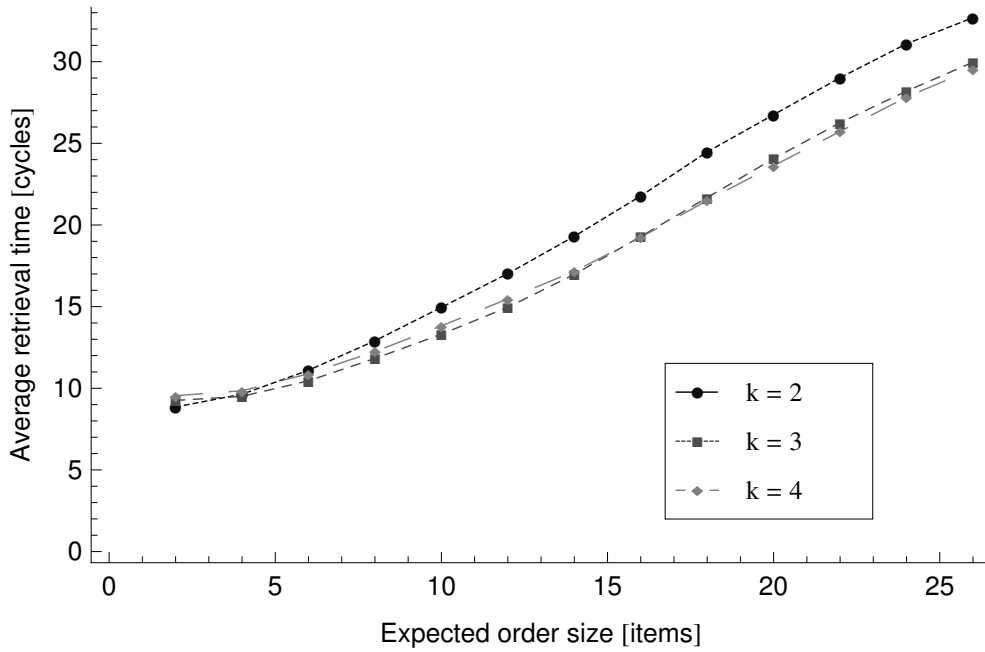


Figure 4.12: Average Retrieval Time for Different Expected Order Size Levels

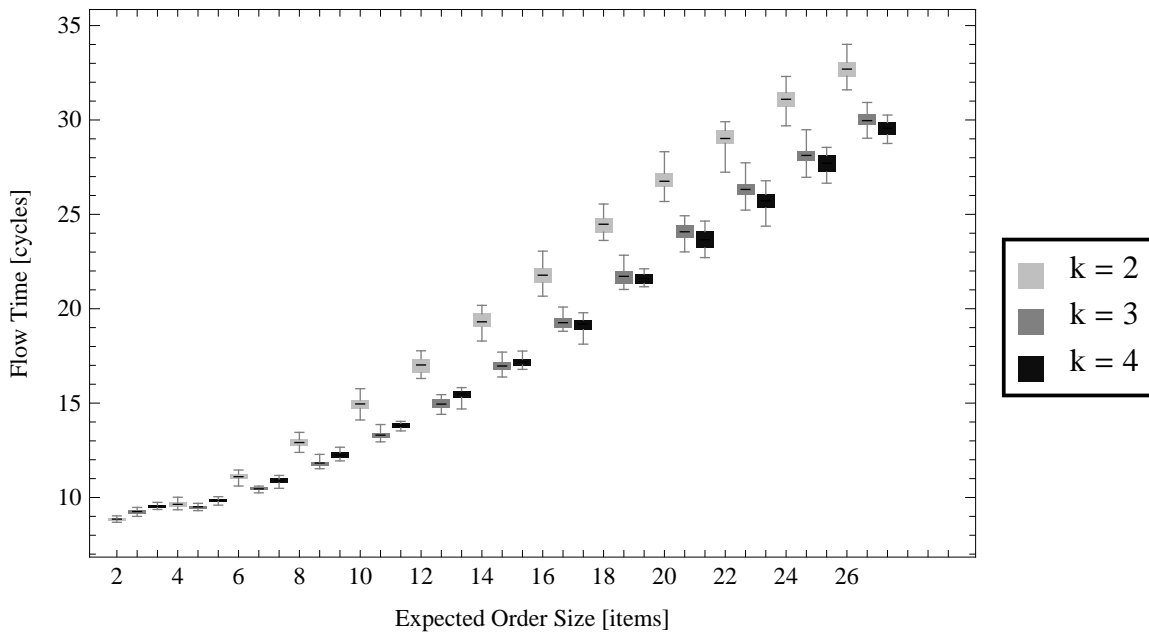


Figure 4.13: Average Retrieval Time for Different Expected Order Size Levels

	k=2 vs. k=3	k=2 vs. k=4	k=3 vs. k=4
2	0	0	0
4	0	0	0
6	0	0	0
8	0	0	0
10	0	0	0
12	0	0	0
14	0	0	0.008
16	0	0	0.639
18	0	0	0.077
20	0	0	0.001
22	0	0	0.001
24	0	0	0.011
26	0	0	0.001

Table 4.25: p values of Flow Time with Different Expected Order Size Levels

	k=2 vs. k=3	k=2 vs. k=4	k=3 vs. k=4
2	-14.63	-25.44	-11.67
4	4.76	-6.56	-17.72
6	15.01	4.47	-12.09
8	19.47	11.97	-10.92
10	20.08	15.45	-11.82
12	18.87	15.39	-7.85
14	23.39	21.33	-2.83
16	22.16	21.82	0.47
18	21.72	27.25	1.83
20	18.83	26.4	3.61
22	19.87	24.41	3.58
24	21.58	20.68	2.71
26	23.11	22.23	3.78

Table 4.26: t values of Flow Time with Different Expected Order Size Levels

Average retrieval time for the system is significantly longer for the $k = 2$ configuration, and other two configurations ($k = 3$ and $k = 4$) are close to each other (see Figure 4.12). The number of empty cells in the $k = 2$ configuration is not sufficient to handle the movements of the requested and balancing units. A small number of empty cells slows down the vertical movements significantly. A larger number of empty cells allows more traffic capacity, and shorter retrieval times. Empty cells have a diminishing benefit on the performance. The

	k=2 vs. k=3	k=2 vs. k=4	k=3 vs. k=4
2	(-0.4553, -0.3436)	(-0.7476, -0.6363)	(-0.3438, -0.2412)
4	(0.0742, 0.1860)	(-0.2780, -0.1459)	(-0.3815, -0.3025)
6	(0.5577, 0.7337)	(0.1154, 0.3101)	(-0.5062, -0.3597)
8	(0.9724, 1.2007)	(0.5494, 0.7758)	(-0.5034, -0.3445)
10	(1.4687, 1.8019)	(0.9779, 1.2763)	(-0.5962, -0.4202)
12	(1.837, 2.283)	(1.343, 1.755)	(-0.6436, -0.3775)
14	(2.1316, 2.5400)	(1.938, 2.349)	(-0.3308, -0.0533)
16	(2.278, 2.741)	(2.323, 2.804)	(-0.178, 0.286)
18	(2.503, 3.023)	(2.756, 3.203)	(-0.025, 0.457)
20	(2.379, 2.959)	(2.853, 3.333)	(0.184, 0.664)
22	(2.482, 3.052)	(3.027, 3.580)	(0.230, 0.843)
24	(2.622, 3.171)	(2.939, 3.584)	(0.090, 0.640)
26	(2.507, 2.994)	(2.863, 3.443)	(0.184, 0.619)

Table 4.27: CIs of Flow Time with Different Expected Order Size Levels

	k=2 vs. k=3	k=2 vs. k=4	k=3 vs. k=4
2	0	0	0
4	0	0	0
6	0	0	0
8	0	0	0
10	0	0	0
12	0	0	0
14	0	0	0
16	0	0	0
18	0	0	0
20	0	0	0
22	0	0	0
24	0	0	0
26	0	0	0

Table 4.28: p values of Waiting Time with Different Expected Order Size Levels

average retrieval time increases with expected order size for all configurations due to a larger number of requested items, and high traffic density. Longer average retrieval times in the $k = 2$ configuration cause a significantly longer waiting time per pick for the worker, which is the main reason for performance difference (see Figure 4.14). Furthermore, the $k = 4$ is not significantly different to the $k = 3$ configuration, which is seen from the waiting time per pick.

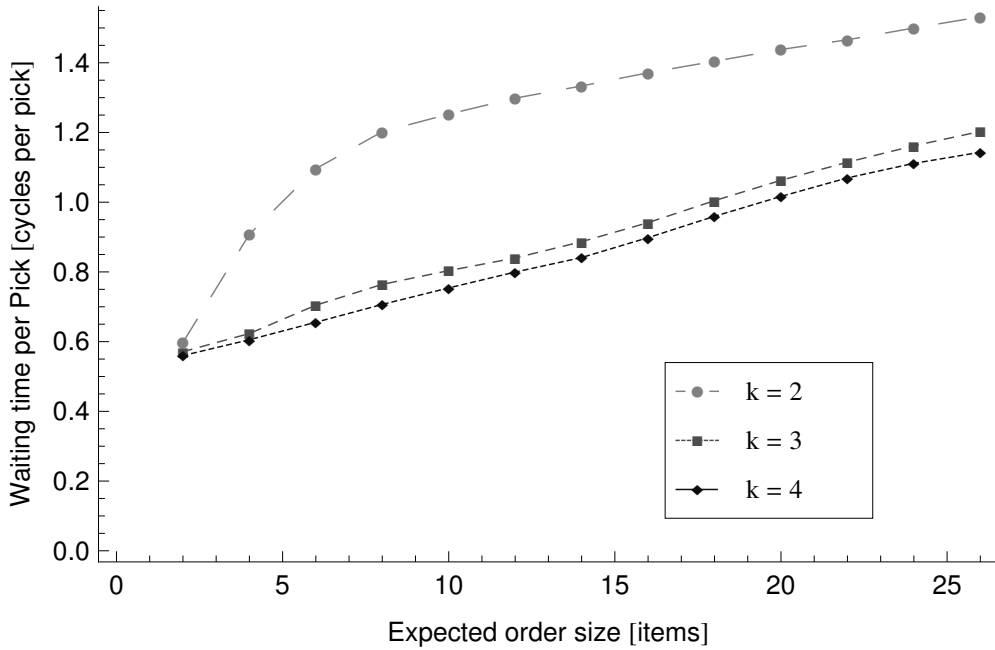


Figure 4.14: Waiting Time per Pick for Different Expected Order Size Levels

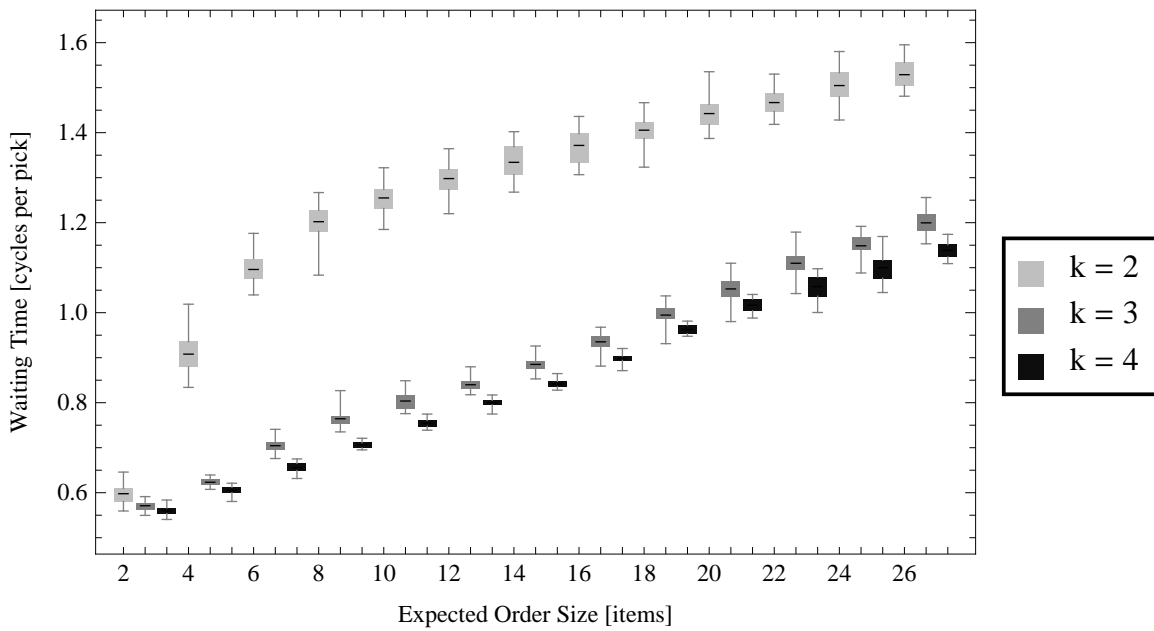


Figure 4.15: Waiting Time per Pick for Different Expected Order Size Levels

	k=2 vs. k=3	k=2 vs. k=4	k=3 vs. k=4
2	6.07	7.85	4.11
4	40.05	41.04	7.05
6	58.07	68.95	14.56
8	64.74	68.13	15.87
10	78.06	73.47	10.95
12	67.89	76.63	14.67
14	59.19	69.58	12.27
16	53.89	67.34	10.06
18	56.19	70.63	11.9
20	59.47	57.74	9.67
22	60.4	65.59	9.23
24	43.63	52.05	10.5
26	47.82	52.16	12.79

Table 4.29: t values of Waiting Time with Different Expected Order Size Levels

	k=2 vs. k=3	k=2 vs. k=4	k=3 vs. k=4
2	(0.01772, 0.03571)	(0.02795, 0.04764)	(0.00557, 0.01659)
4	(0.27011, 0.29918)	(0.28746, 0.31762)	(0.01270, 0.02309)
6	(0.37782, 0.40540)	(0.42741, 0.45354)	(0.04200, 0.05573)
8	(0.42405, 0.45172)	(0.48058, 0.51033)	(0.05015, 0.06499)
10	(0.43768, 0.46123)	(0.48519, 0.51297)	(0.04036, 0.05889)
12	(0.44590, 0.47360)	(0.48562, 0.51226)	(0.03373, 0.04465)
14	(0.43233, 0.46327)	(0.47656, 0.50542)	(0.03599, 0.05039)
16	(0.41523, 0.44799)	(0.45707, 0.48570)	(0.03169, 0.04787)
18	(0.38217, 0.41105)	(0.42521, 0.45057)	(0.03418, 0.04838)
20	(0.36003, 0.38567)	0.40301, 0.43261)	(0.03545, 0.05448)
22	(0.33161, 0.35485)	(0.37607, 0.40028)	(0.03498, 0.05491)
24	(0.31082, 0.34140)	(0.36172, 0.39131)	(0.04059, 0.06022)
26	(0.30929, 0.33693)	(0.36780, 0.39782)	(0.05016, 0.06924)

Table 4.30: CIs of Waiting Time with Different Expected Order Size Levels

4.5.2 Aspect Ratio Analysis

To understand the effects of aspect ratio, we used the same configurations as with one sided GridPick (see Table 4.31). All configurations have approximately same total number of conveyor modules, storage containers, and empty cells.

The data lines for the aspect ratio configurations end at different points due to having expected order sizes up to the number of columns (see Figure 4.16). The number of columns

Config. No	Aspect Ratio	Rows	Cols.	k	Conveyors	Items	Empty Cells
1	8 (6:50)	6	50	8	300	252	48
2	3 (10:30)	10	30	5	300	250	50
3	1 (16:19)	16	19	3	304	256	48
4	0.5 (25:12)	25	12	2	300	250	50
5	0.16 (42:7)	42	7	1	294	252	42

Table 4.31: Configurations with Different Aspect Ratio Values

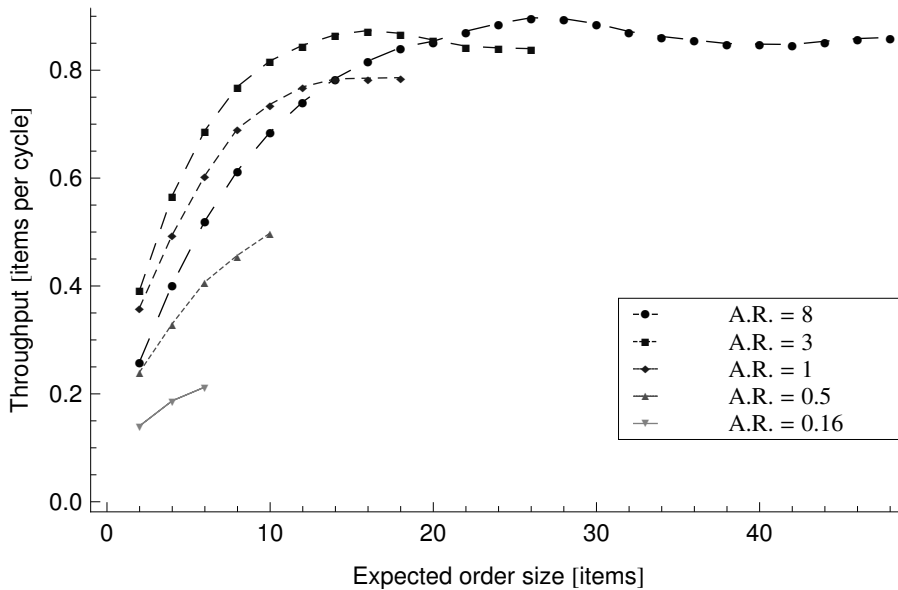


Figure 4.16: Throughput Plot

is distinct for each configuration. An aspect ratio of 3 is best for the expected order sizes up to approximately 20. Other aspect ratio configurations have a pattern of smaller aspect ratio with worse performance. Deeper grids have longer travel of requested items, an higher average retrieval time (see Figure 4.18). When the number of rows increases (smaller aspect ratio), the system gets deep, and it takes more time for the items to travel to the pick face. Therefore, the worker waits for the requested items, which result in worse performance. When aspect ratio increases the travel time improves. Up to aspect ratio of 3, the system performance improves. When aspect ratio increases again to 8, the system performance gets

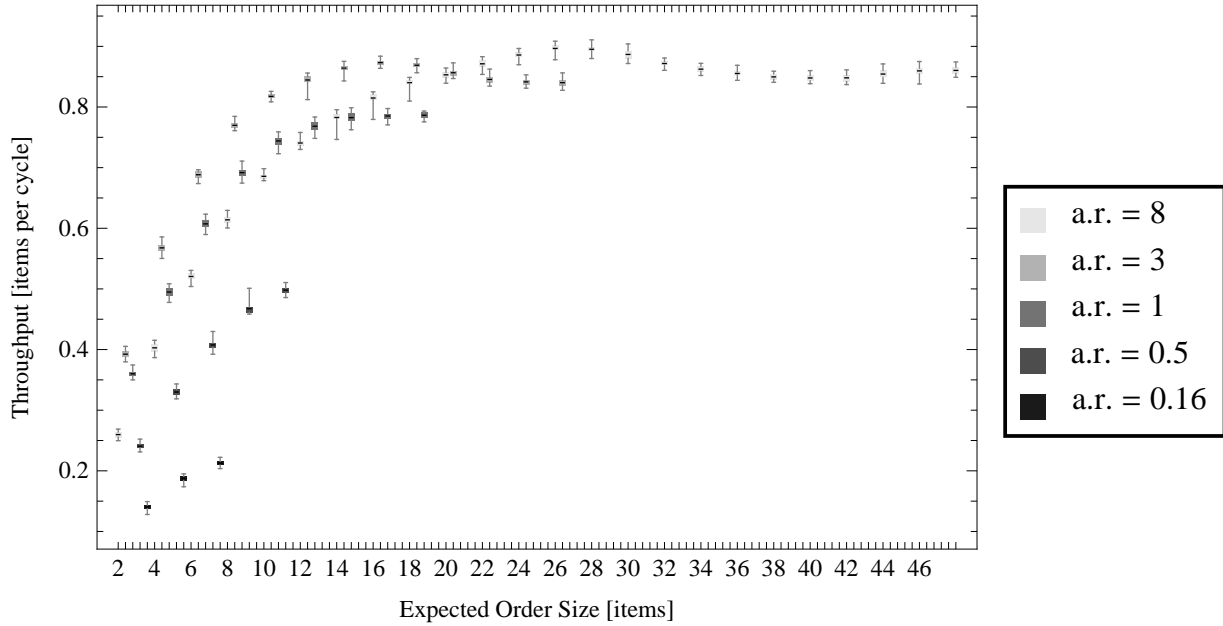


Figure 4.17: Throughput Plot for Different Aspect Ratios

	0.16vs0.5	0.16vs1	0.16vs3	0.16vs8	0.5vs1	0.5vs3	0.5vs8	1vs3	1vs8	3vs8
2	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
8	-	-	-	-	0	0	0	0	0	0
10	-	-	-	-	0	0	0	0	0	0
12	-	-	-	-	-	-	-	0	0	0
14	-	-	-	-	-	-	-	0	0.793	0
16	-	-	-	-	-	-	-	0	0	0
18	-	-	-	-	-	-	-	-	-	0
20	-	-	-	-	-	-	-	-	-	0.164
22	-	-	-	-	-	-	-	-	-	0
24	-	-	-	-	-	-	-	-	-	0
26	-	-	-	-	-	-	-	-	-	0

Table 4.32: p values for throughput with various aspect ratios

worse. The reason is: this configuration makes the worker walk excessively. In this case, requested items “wait” for the worker on the pick face, the SKU density is lower, and the picker walk a significantly longer distance.

We have analyzed the system’s waiting time per pick and wasted walking time per pick. While processing the picks, walking to the very next location is the ideal case. We can

	0.16vs0.5	0.16vs1	0.16vs3	0.16vs8	0.5vs1	0.5vs3	0.5vs8	1vs3	1vs8	3vs8
2	-83.21	-146.73	-174.09	-142.34	-75.83	-102.35	-13.19	-22.71	70.28	88.83
4	-83.71	-166.29	-213.76	-125.58	-104.3	-137.39	-38.02	-31.51	51.85	86.3
6	-78.94	-138.79	-325.73	-216.84	-57.52	-98.48	-43.54	-25.33	25.84	100.43
8	-	-	-	-	-39.75	-53.22	-27.18	-36.66	46.13	95.21
10	-	-	-	-	-30.28	-222.57	-147.54	-9.74	6.53	100.83
12	-	-	-	-	-	-	-	-48.88	18.68	74.51
14	-	-	-	-	-	-	-	-47.44	-0.26	63.87
16	-	-	-	-	-	-	-	-63.86	-24.03	47.33
18	-	-	-	-	-	-	-	-	-	24.45
20	-	-	-	-	-	-	-	-	-	1.43
22	-	-	-	-	-	-	-	-	-	-18.86
24	-	-	-	-	-	-	-	-	-	-32.33
26	-	-	-	-	-	-	-	-	-	-32.48

Table 4.33: t values for throughput with various aspect ratios

	0.16 vs 0.5	0.16 vs 1	0.16 vs. 3	0.16 vs. 8
2	(-0.10244, -0.09753)	(-0.22245, -0.21633)	(-0.25473, -0.24882)	(-0.120896, -0.117471)
4	(-0.14626, -0.13929)	(-0.31140, -0.30383)	(-0.38414, -0.37686)	(-0.21916, -0.21214)
6	(-0.20162, -0.19143)	(-0.39778, -0.38622)	(-0.47864, -0.47267)	(-0.31076, -0.30495)

Table 4.34: CIs for throughput with various aspect ratios

imagine a pick face with all requested items, and the worker processes each item in the pick face. Waiting time per pick decrease with the larger aspect ratio values (see Figure 4.20). A larger aspect ratio means shallow grid, shorter retrieval time, and a longer pick face, which results with longer walking for the worker and shorter travel of the requested item that minimizes the waiting time. Wasted walking time per pick decreases with the smaller aspect ratios (see Figure 4.22). The reason is: with a smaller aspect ratio, the SKU density increases and the pick face gets full with the requested items. Therefore, the picker walks to

	0.5 vs. 1	0.5 vs. 3	0.5 vs. 8
2	(-0.12263, -0.11619)	(-0.15482, -0.14876)	(-0.02218, -0.01622)
4	(-0.16807, -0.16161)	(-0.24126, -0.23419)	(-0.07679, -0.06896)
6	(-0.20243, -0.18852)	(-0.28493, -0.27334)	(-0.11656, -0.10610)
8	(-0.24620, -0.22210)	(-0.32415, -0.30015)	(-0.16805, -0.14452)
10	(-0.25497, -0.22270)	(-0.32234, -0.31646)	(-0.18990, -0.18470)

Table 4.35: CIs for throughput with various aspect ratios

	1 vs. 3	1 vs. 8	3 vs. 8
2	(-0.03530, -0.02947)	(0.09729, 0.10312)	(0.12954, 0.13564)
4	(-0.07761, -0.06815)	(0.08834, 0.09559)	(0.16094, 0.16876)
6	(-0.09041, -0.07690)	(0.07748, 0.09080)	(0.16438, 0.17122)
8	(-0.08235, -0.07365)	(0.07441, 0.08132)	(0.15252, 0.15921)
10	(-0.09749, -0.06365)	(0.03538, 0.06769)	(0.12942, 0.13478)
12	(-0.08049, -0.07403)	(0.02486, 0.03097)	(0.10229, 0.10806)
14	(-0.08487, -0.07785)	(-0.00393, 0.00303)	(0.07832, 0.08350)
16	(-0.09205, -0.08633)	(-0.03551, -0.02994)	(0.05403, 0.05891)
18	-	-	(0.02404, 0.02843)
20	-	-	(-0.00092, 0.00517)
22	-	-	(-0.03087, -0.02483)
24	-	-	(-0.04825, -0.04250)
26	-	-	(-0.06104, -0.05381)

Table 4.36: CIs for throughput with various aspect ratios

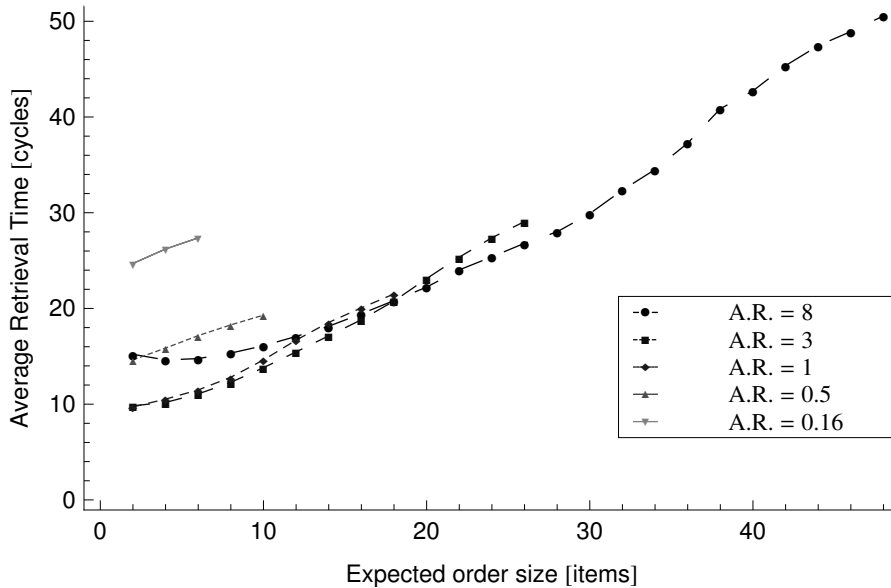


Figure 4.18: Average Retrieval Time Plot

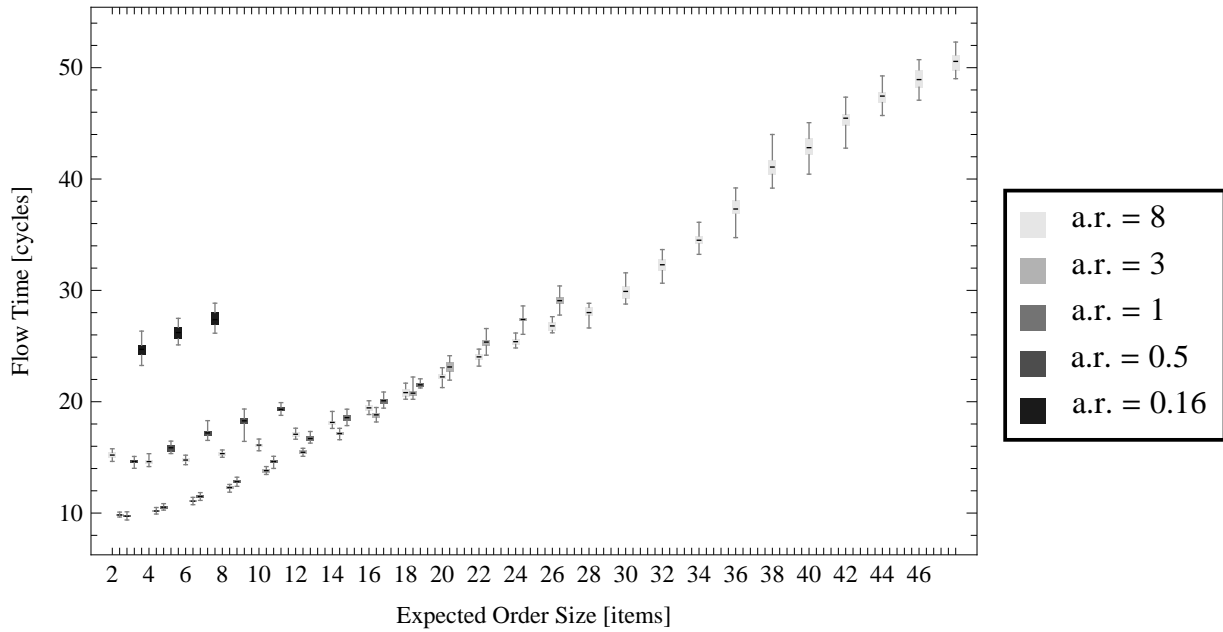


Figure 4.19: Average Retrieval Time for Different Aspect Ratios

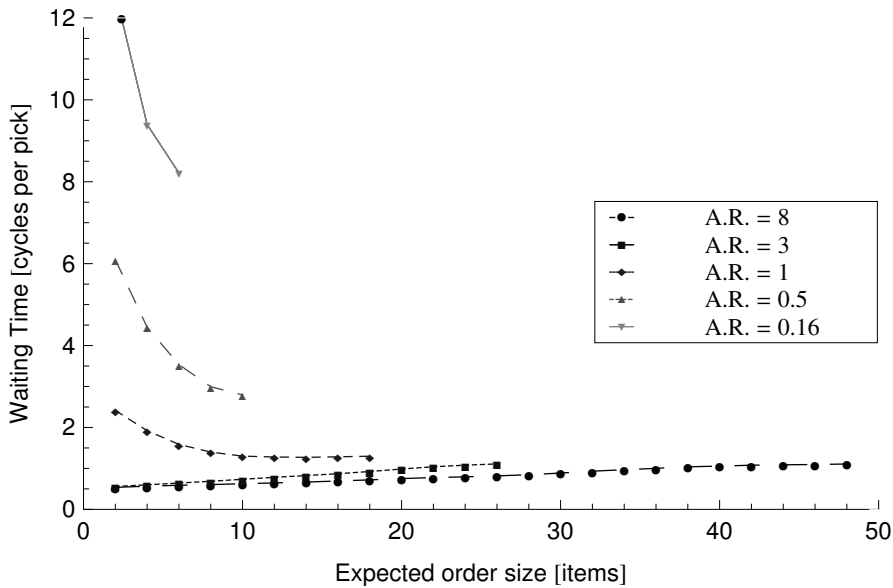


Figure 4.20: Waiting Time for Aspect Ratio Configurations

the very next location in the pick face in most cases. A smaller aspect ratio means a shorter pick face and shorter walking time for the worker.

We explain the middle level performance of the largest aspect ratio (8) configuration with the combination of waiting time per pick and walking per pick analysis. The largest

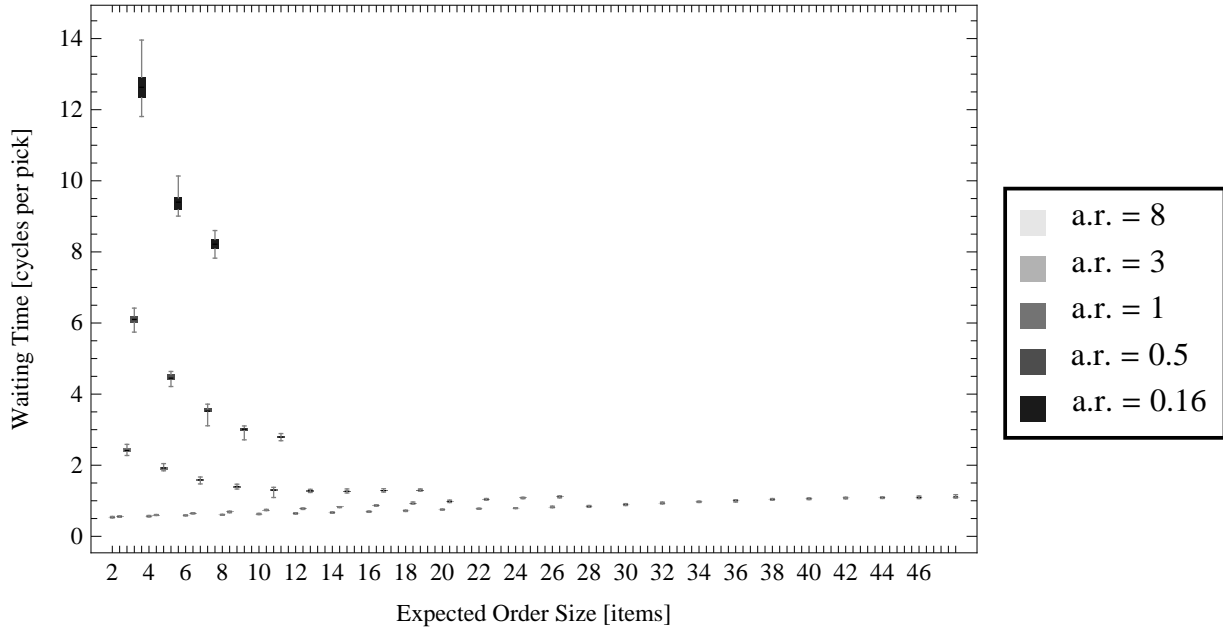


Figure 4.21: Waiting Time per Pick for Different Aspect Ratios

aspect ratio (8) configuration has the best performance for the waiting time per pick and the worst performance for the walking time per pick, which results in a moderate performance (see Figures 4.20 and 4.22). The smallest aspect ratio (0.16) configuration has the worst performance for both waiting time and wasted walking time per pick, which brings the worst productivity compared to other configurations. The largest aspect ratio (8) configuration has a decreasing throughput toward the larger expected order size values, due to reaching the traffic flow capacity of the system.

4.5.3 Distribution of Empty Cells

Considering that two sided GridPick will keep the empty cells in balance for each row, the distribution of empty cells can have a direct effect on the system performance. If uniform distribution of empty spots is not the best configuration, distribution of the empty cells in the two sided picking model could be different than the one in one sided GridPick.

Figure 4.24 shows possible different configurations for the distribution of empty cells. Having more empty cells on the edges can be a viable approach because the edges will have

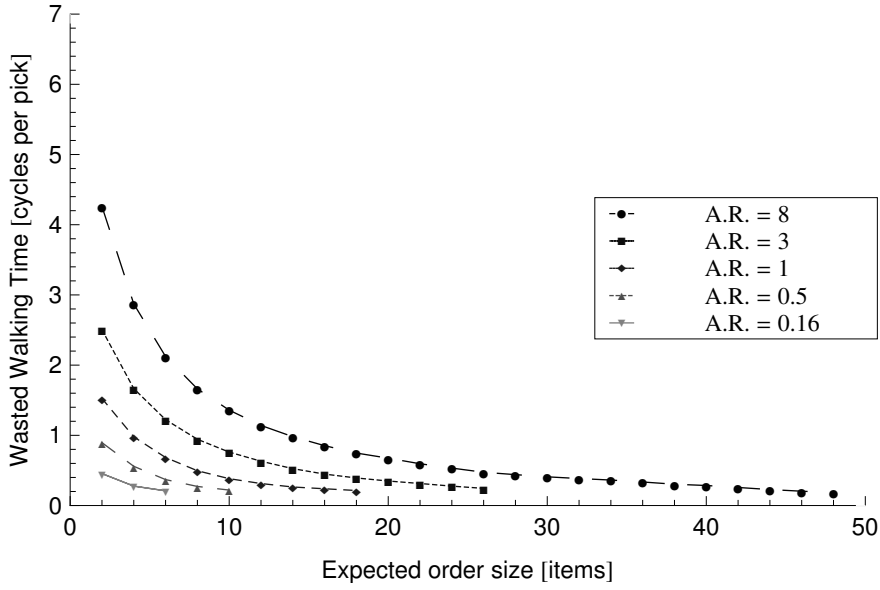


Figure 4.22: Walking Time for Aspect Ratio Configurations

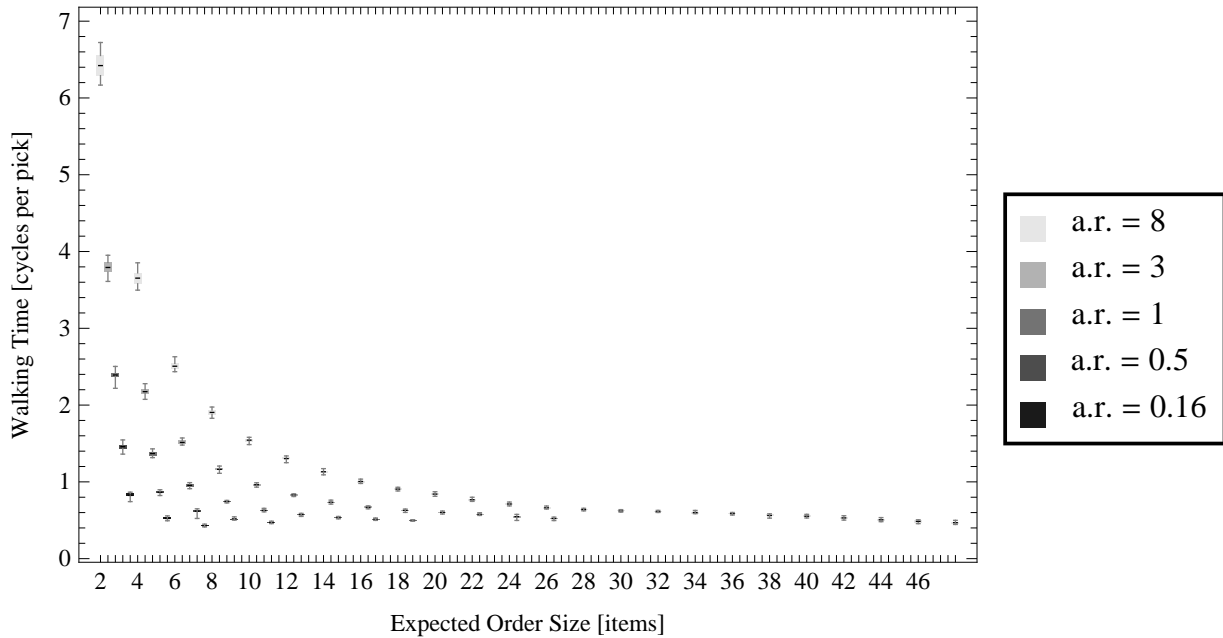
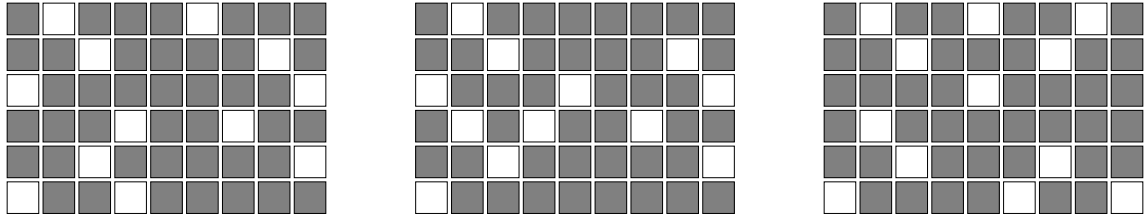


Figure 4.23: Walking Time per Pick for Different Aspect Ratios

the major activity and a dense traffic flow. Another plausible approach would be the opposite case – having more empty cells in the middle. In this case, more items will be close to the pick face, which might ease the travel of requested items.



(a) Uniform Distribution (b) Low Density in the Middle (c) Low Density on the Edges

Figure 4.24: Different Ways to Allocate Empty Cells in the Two Sided Picking Model

Row	Uniform Abundancy	Abundancy in the Middle	Abundancy on the Edges
1	4	2	6
2	4	2	6
3	4	4	4
4	4	6	2
5	4	6	2
6	4	6	2
7	4	6	2
8	4	4	4
9	4	2	6
10	4	2	6

Table 4.37: Distribution of Empty Modules for 3 Configurations

Table 4.37 shows the configurations used for the variable k analysis. All configurations have 25 occupied columns, 10 rows, and 40 empty cells. They differ only in the distribution of empty cells. The first configuration has uniform distribution of empty cells. The “Abundant Middle” configuration has more empty cells in the intermediate rows. “Abundant Edges” has more empty cells toward the pick faces.

We obtain the most distinct results for the waiting time per pick. “Abundant Edges” has a significantly longer waiting time per pick for relatively small expected order sizes. We interpret this result as follows: we can request items from all over the grid for both pick faces, therefore, some items have to travel all the way from one side another. Fewer empty

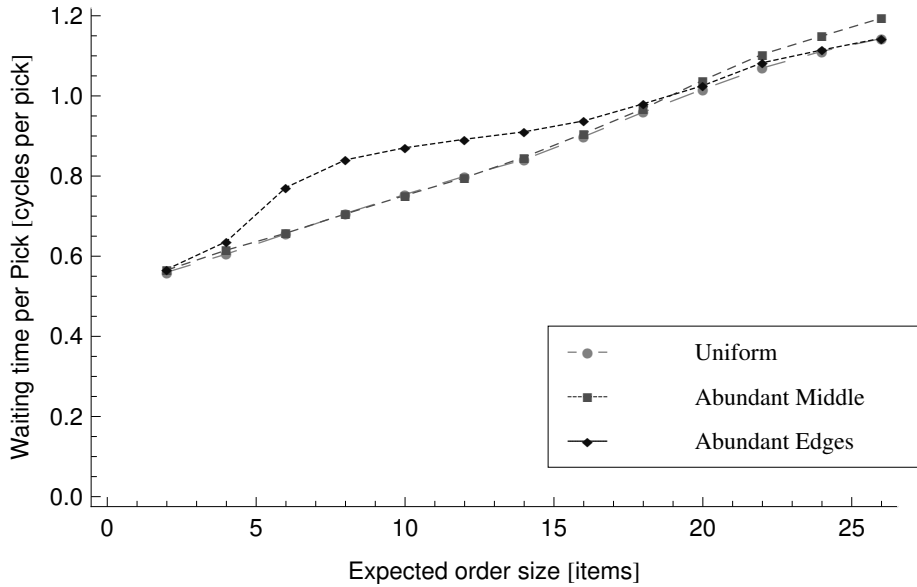


Figure 4.25: Waiting Time for Variable k Configurations of the Two Sided Model

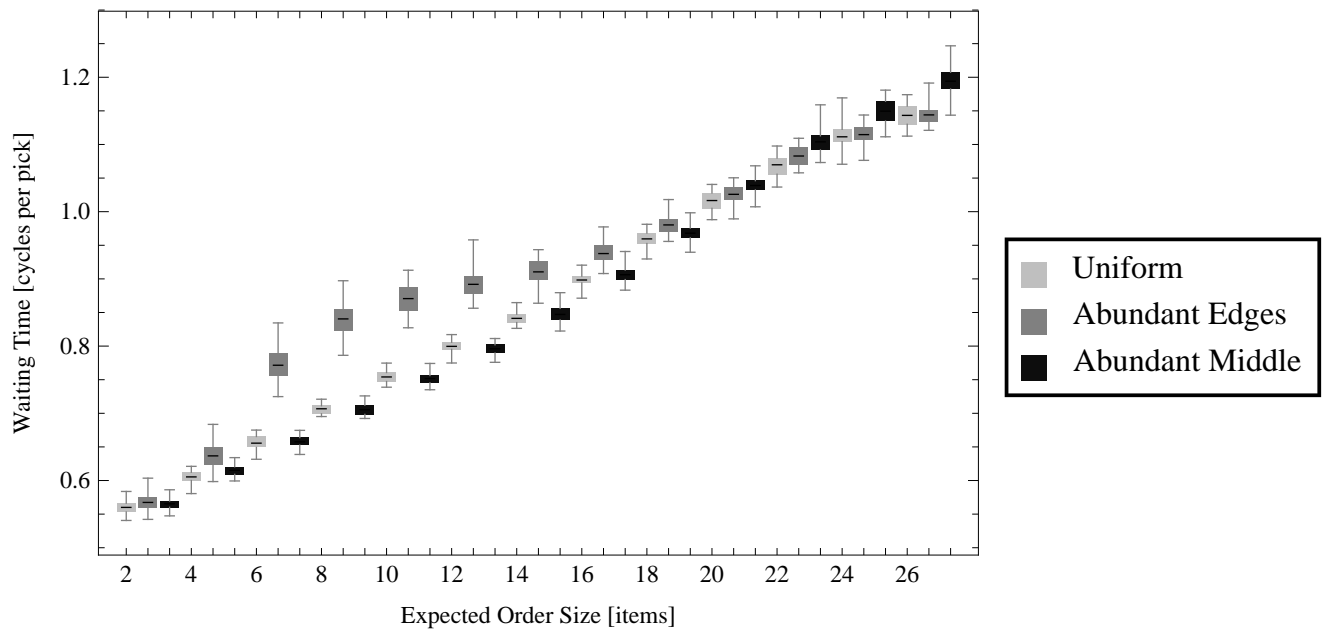


Figure 4.26: Waiting Time per Pick for Variable k Configurations

cells in the intermediate rows causes a bottleneck and slows down the travel of the requested items passing the intermediate rows.

Variable k configurations are statistically the same for a very small expected order size (2). We can see the percent difference of the throughput results from Figure 4.29. The

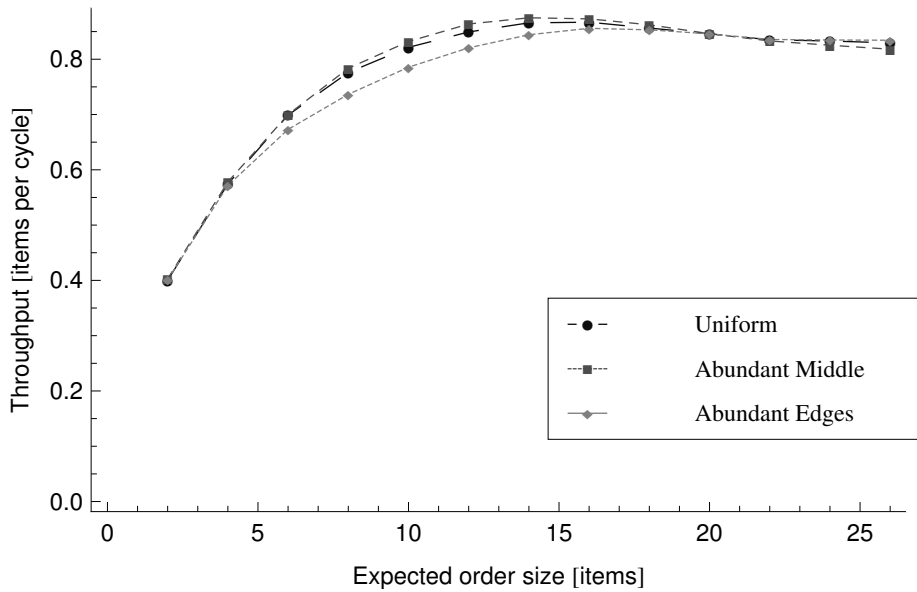


Figure 4.27: Throughput Plot

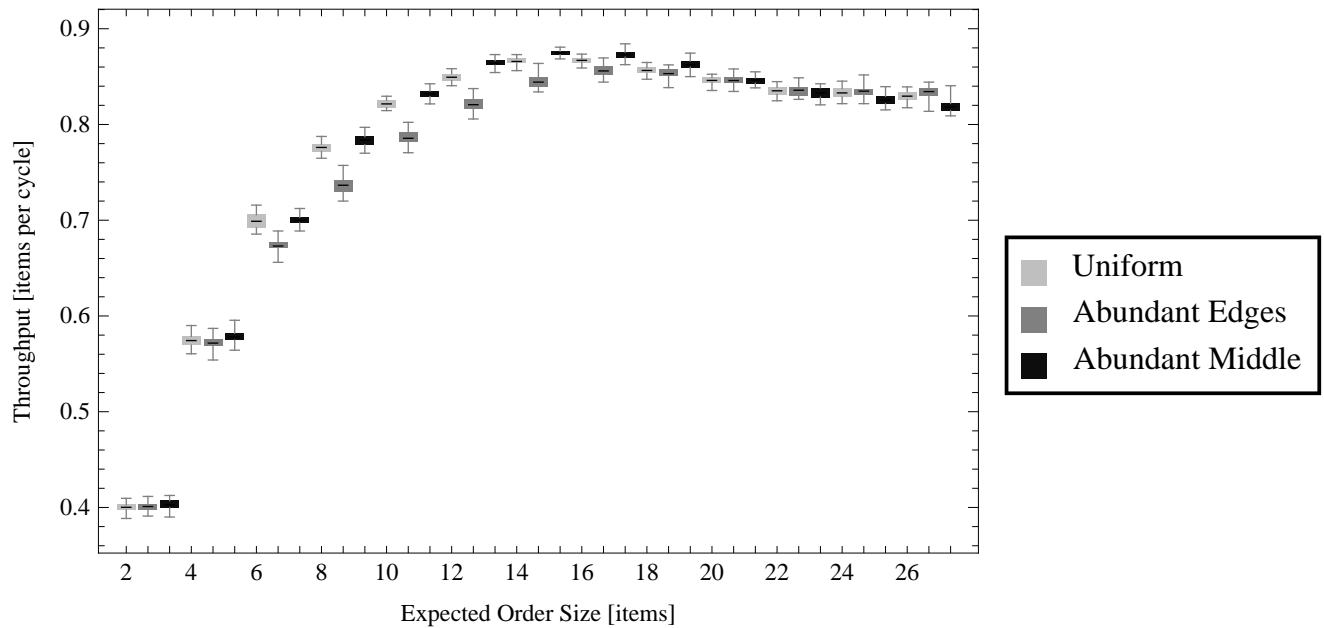


Figure 4.28: Throughput Plot for Variable k Configurations

Abundant Middle configuration is better than the Uniform configuration up to the expected order size of 20. Its pick point is around 1.7% at the expected order size of 12. Abundant Edges is worse than the Uniform configuration until the expected order size of 20. The largest difference is 5% at the expected order size of 8. After the expected order size of

	Uniform vs aEdges	Uniform vs aMiddle	aEdges vs aMiddle
2	0.598	0.055	0.203
4	0.116	0.018	0
6	0	0.632	0
8	0	0	0
10	0	0	0
12	0	0	0
14	0	0	0
16	0	0	0
18	0.014	0	0
20	0.737	0.875	0.654
22	0.524	0.191	0.099
24	0.342	0	0
26	0.002	0	0

Table 4.38: p values for Throughput with Variable k Configurations

	Uniform vs aEdges	Uniform vs aMiddle	aEdges vs aMiddle
2	-0.53	-2	-1.3
4	1.62	-2.51	-3.95
6	15.3	-0.48	-16.61
8	18.77	-5.54	-23.94
10	25.59	-7.43	-28.98
12	17.99	-13	-27.51
14	12.83	-11.36	-19.17
16	8.74	-5.85	-11.83
18	2.62	-4.6	-6.42
20	0.34	-0.16	-0.45
22	-0.64	1.34	1.7
24	-0.97	5.97	6.32
26	-3.37	6.34	9.2

Table 4.39: t values for Throughput with Variable k Configurations

20, the case is reversed. This is due to having so many requested items that should occupy the edges. When there is a high number of requested items, Abundant Edges configuration makes it easier to reach the pick face.

Throughput results are consistent with the waiting time per pick. Abundant Edges has a lower throughput for relatively low expected order sizes due to longer waiting times (see Figure 4.27). Variable k analysis results are interesting when we look at the average retrieval

	Uniform vs aEdges	Uniform vs aMiddle	aEdges vs aMiddle
2	(-0.00391, 0.00229)	(-0.00529, 0.00006)	(-0.00465, 0.00103)
4	(-0.00069, 0.00597)	(-0.00691, -0.00070)	(-0.00979, -0.00311)
6	(0.02237, 0.02928)	(-0.00492, 0.00304)	(-0.03006, -0.02347)
8	(0.03519, 0.04380)	(-0.01045, -0.00481)	(-0.05115, -0.04310)
10	(0.03305, 0.03879)	(-0.01343, -0.00763)	(-0.04973, -0.04317)
12	(0.02541, 0.03193)	(-0.01667, -0.01214)	(-0.04628, -0.03987)
14	(0.01814, 0.02501)	(-0.010767, -0.007483)	(-0.03398, -0.02742)
16	(0.00846, 0.01363)	(-0.00816, -0.00394)	(-0.02005, -0.01414)
18	(0.00079, 0.00640)	(-0.00851, -0.00327)	(-0.01250, -0.00646)
20	(-0.00210, 0.00293)	(-0.00232, 0.00198)	(-0.00322, 0.00205)
22	(-0.00337, 0.00176)	(-0.00113, 0.00539)	(-0.00059, 0.00648)
24	(-0.00506, 0.00181)	(0.00502, 0.01026)	(0.00627, 0.01226)
26	(-0.00778, -0.00190)	(0.00776, 0.01514)	(0.01267, 0.01991)

Table 4.40: CIs for Throughput with Variable k Configurations

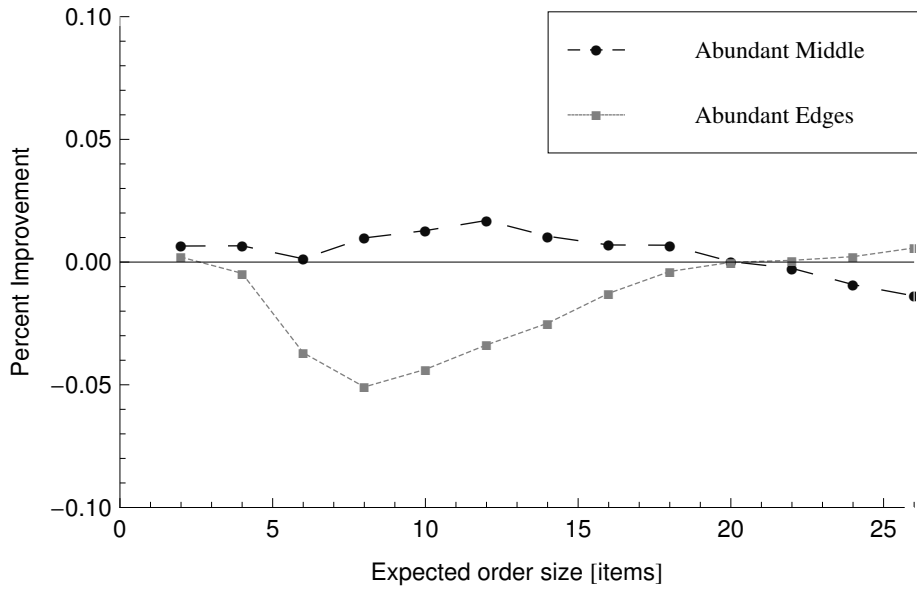


Figure 4.29: % Difference on Throughput

times for the three configurations (see Figure 4.30). They do not differ, unlike the waiting time per pick results.

Based on the observations, there are two kinds of delay in average retrieval time or flow time of the requested item. The first is the expected one – delay due to the high traffic flow, causing some requested items not to be able to move vertically in all iterations. The

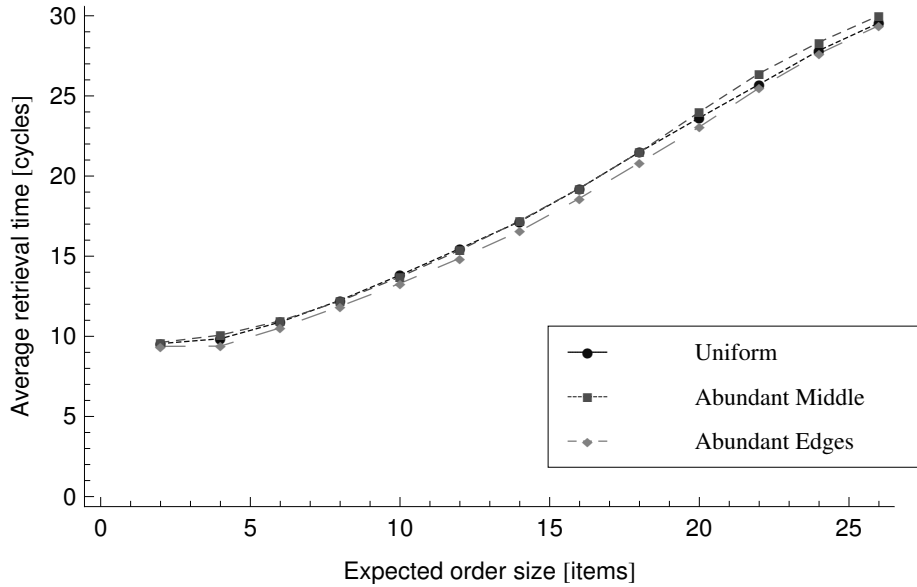


Figure 4.30: Average Retrieval Time Plot

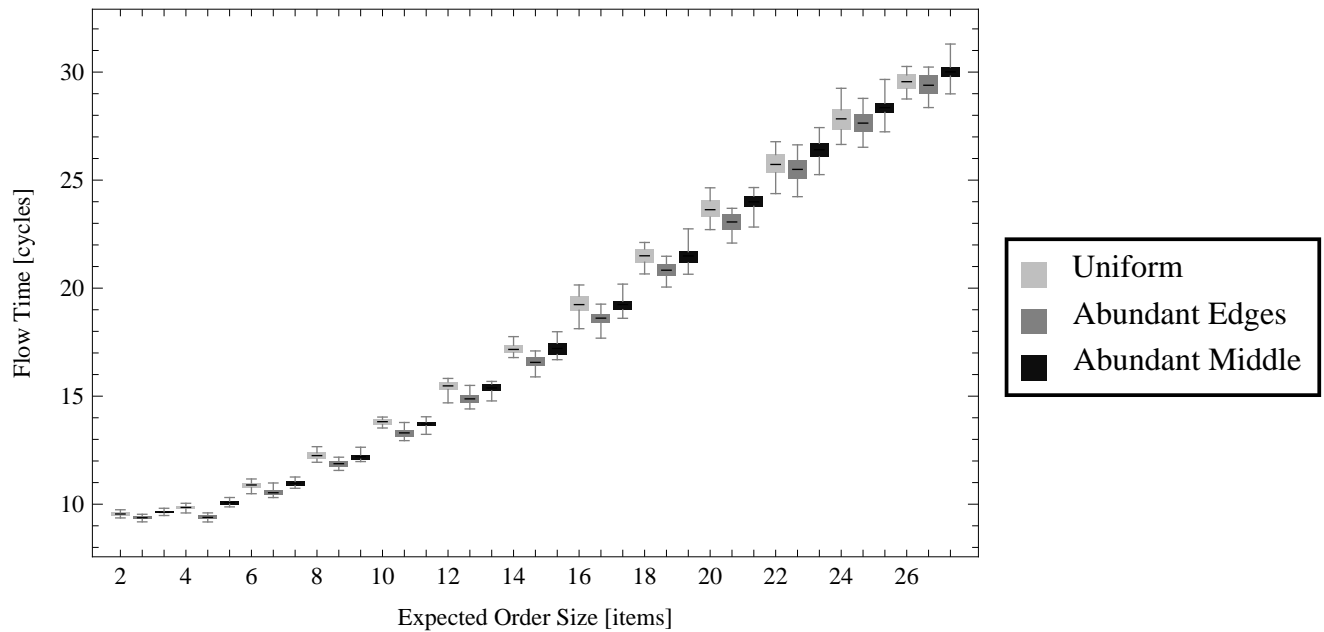


Figure 4.31: Average Retrieval Time for Variable k Configurations

second happens when the system is well adjusted, and items arrive to the pick face, and the worker is almost always busy. In this case, because the pick face is almost full with requested items, some items cannot reach the pick face. They wait at the back of the pick face in the second row. This phenomenon also causes a delay in the average retrieval time.

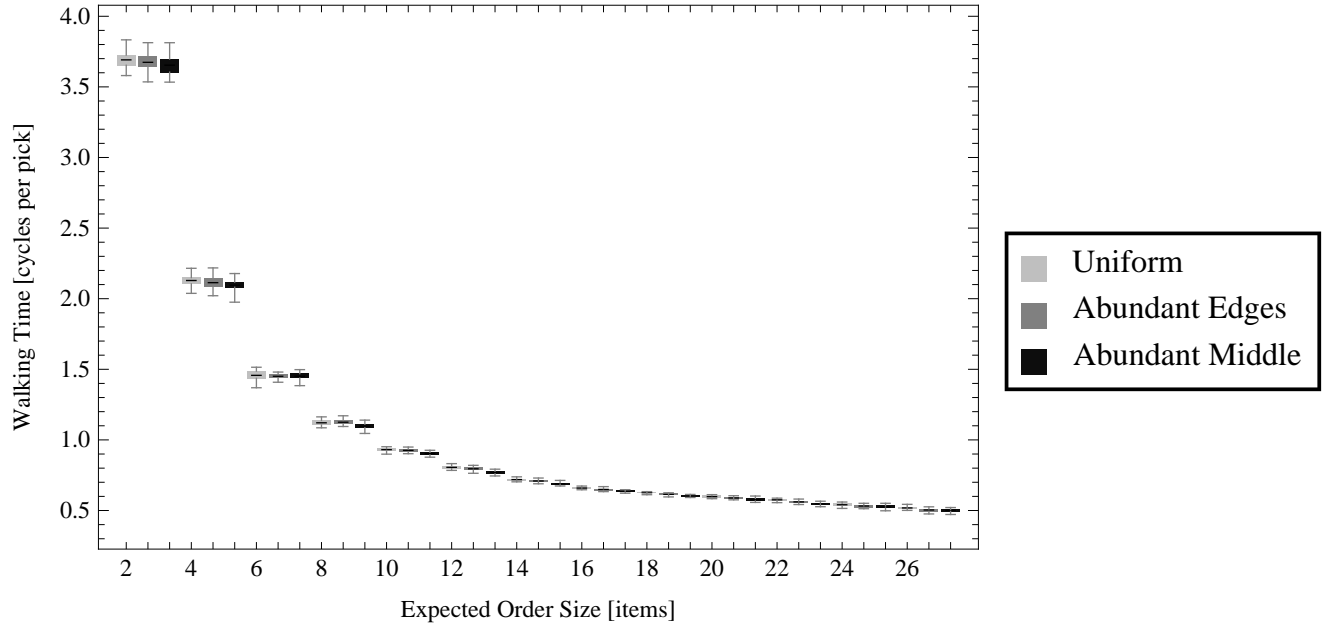


Figure 4.32: Walking Time per Pick for Variable k Configurations

Therefore, there are delays due to two cases, and only one of them results in worker waiting. As a result, we cannot see the effect of the average retrieval time results on the worker's waiting time (see Figure 4.30). Actually, up to the expected order size of 20, the Uniform and Abundant Middle configurations have a longer average retrieval time than the Abundant Edges configuration. Our intuition is that the second type of delay causes this result, which is the waiting time at the back of the pick face in a well adjusted system.

4.5.4 Performance Comparison of the Two Workers

We have also compared performance of the two workers to detect any possible bias in the two sided GridPick model. We used configuration of 25 occupied columns, 10 rows, and 4 empty cells per row (k).

Unexpectedly, Worker 1 has better performance compared to Worker 2 (see Figure 4.33). We expected a better performance for Worker 2, because we have a priority for the movement of items toward the pick face 2. We offer the following interpretation: occupancy of requested-1 items should be higher toward the pick face-1, and occupancy of requested-2 items should be

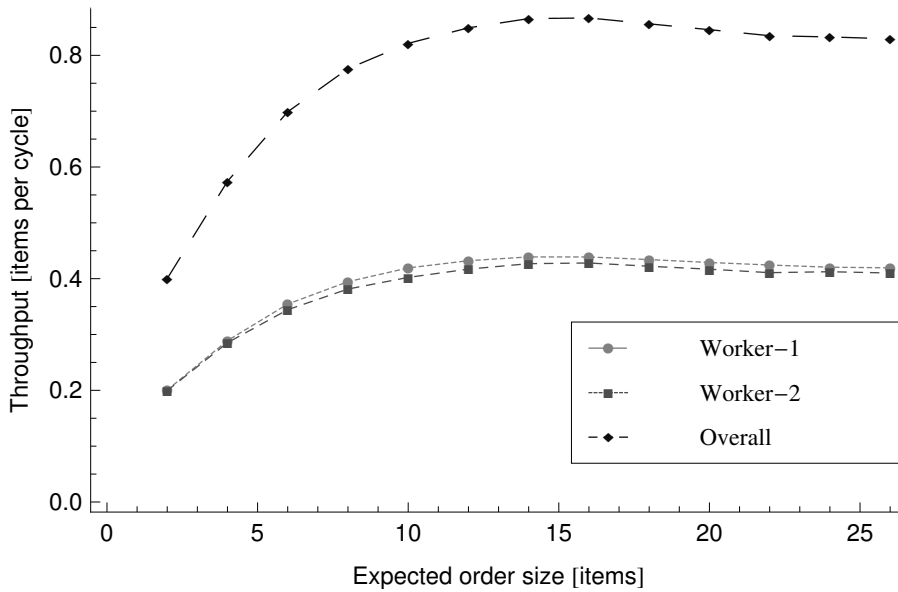


Figure 4.33: Throughput Plot

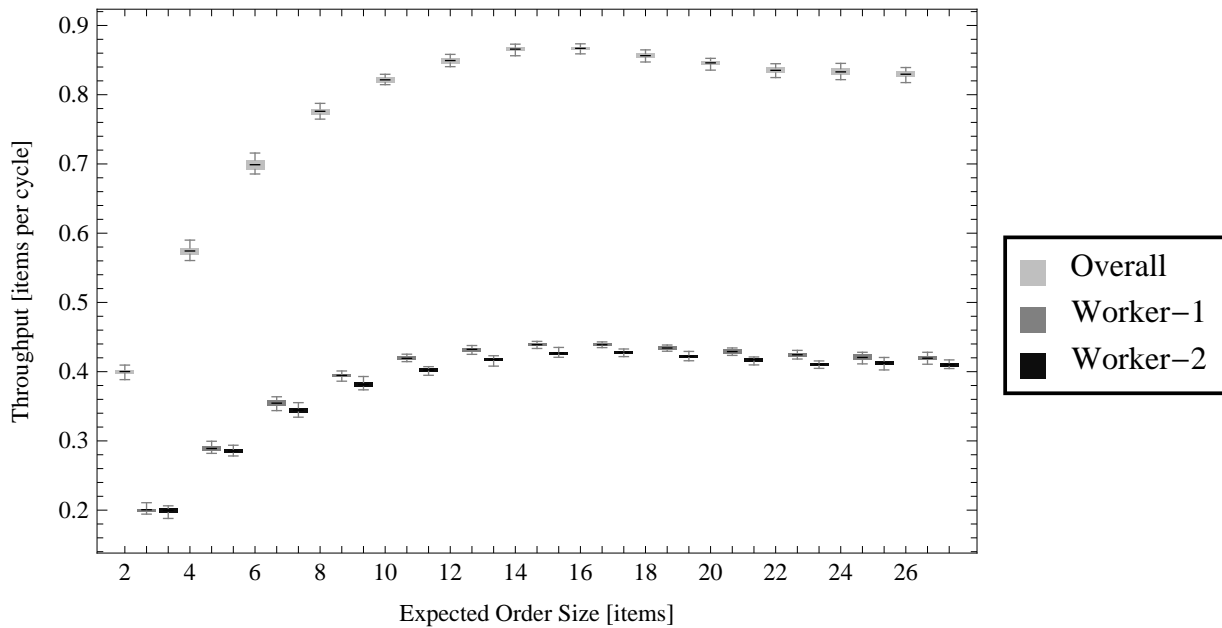


Figure 4.34: Throughput Plot for Two Workers

higher toward pick face 2. This means there will be more balancing items for the requested-1 items close to the pick face-1, and there will be more balancing items of the requested-2 items close to the pick face-2. Since there is a priority of movement for the items moving towards the pick face-2, replenishing items of the requested-1 items clear more quickly, providing space

	overall vs. w1	overall vs w-2	w-1 vs w-2
2	0	0	0.333
4	0	0	0.002
6	0	0	0
8	0	0	0
10	0	0	0
12	0	0	0
14	0	0	0
16	0	0	0
18	0	0	0
20	0	0	0
22	0	0	0
24	0	0	0
26	0	0	0

Table 4.41: p values for Throughput of Two Workers

	overall vs. w1	overall vs w-2	w-1 vs w-2
2	246.87	292.59	0.98
4	403.01	313.99	3.4
6	407.41	371.9	9.05
8	484.38	668.29	14.8
10	650.89	801.15	20.69
12	645.77	737.92	15.88
14	803.87	945.72	17.21
16	927.91	1029.57	16.23
18	644.47	877.77	14.78
20	672.24	717.85	12.7
22	807.69	643.38	20.5
24	519.99	469.64	6.29
26	673.97	559.46	10.14

Table 4.42: t values for Throughput of Two Workers

for the movement of the requested-1 items, which results in slightly better performance for Worker 1.

Average retrieval time shows consistent but different results from the throughput and waiting time analysis for reasons explained in the variable k analysis. Pick face 2 has a larger average retrieval time compared to pick face 1 (see Figure 4.35). Based on the observations of the simulation, because pick face 2 is almost full of requested items, later requested items

	overall vs. w-1	overall vs w-2	w-1 vs w-2
2	(0.197872, 0.201178)	(0.199231, 0.202036)	(-0.00119, 0.00341)
4	(0.283827, 0.286723)	(0.287134, 0.290899)	(0.00149, 0.00599)
6	(0.342671, 0.346129)	(0.352550, 0.356450)	(0.00782, 0.01238)
8	(0.379823, 0.383044)	(0.393384, 0.395799)	(0.011339, 0.014977)
10	(0.400969, 0.403497)	(0.418180, 0.420320)	(0.015334, 0.018699)
12	(0.415779, 0.418421)	(0.431035, 0.433431)	(0.013184, 0.017083)
14	(0.425797, 0.427969)	(0.438001, 0.439899)	(0.010633, 0.013501)
16	(0.427140, 0.429027)	(0.438011, 0.439755)	(0.009439, 0.012161)
18	(0.420885, 0.423565)	(0.433172, 0.435195)	(0.010304, 0.013613)
20	(0.415715, 0.418252)	(0.427794, 0.430239)	(0.010095, 0.013971)
22	(0.409710, 0.411790)	(0.423084, 0.425783)	(0.012318, 0.015049)
24	(0.410819, 0.414064)	(0.418694, 0.422356)	(0.00545, 0.01071)
26	(0.408905, 0.411395)	(0.417900, 0.420967)	(0.007411, 0.011155)

Table 4.43: CIs for Throughput of Two Workers

wait at the back of the pick face in the second row to arrive the pick face, and there is a delay of the requested items for pick face 2.

For the waiting time per pick and walking time per pick analysis, there is a difference between two workers (see Figures 4.37 and 4.39). The difference in the average retrieval time does not reflect the waiting time since there is no waiting time due to delay of the requested items.

4.5.5 Performance Comparison with the One Sided Model

In this section, we compare the performance of one sided and two sided GridPick models. Both models have the same configuration: 25 occupied columns, 10 rows, and 4 empty cells per row (k). In the one sided GridPick model, a single worker operates on one side of the grid, and in the two sided GridPick, there are two pick faces and two workers.

Throughput results show three configurations (see Figure 4.41). The first configuration indicates two sided GridPick, second one shows one sided GridPick, and third configuration represent two grids of one sided GridPick, which is equivalent to the operations of two sided GridPick. As the results point out, two sided GridPick performs significantly better than the one sided model. Two grids of one sided GridPick is better than the two sided model, which

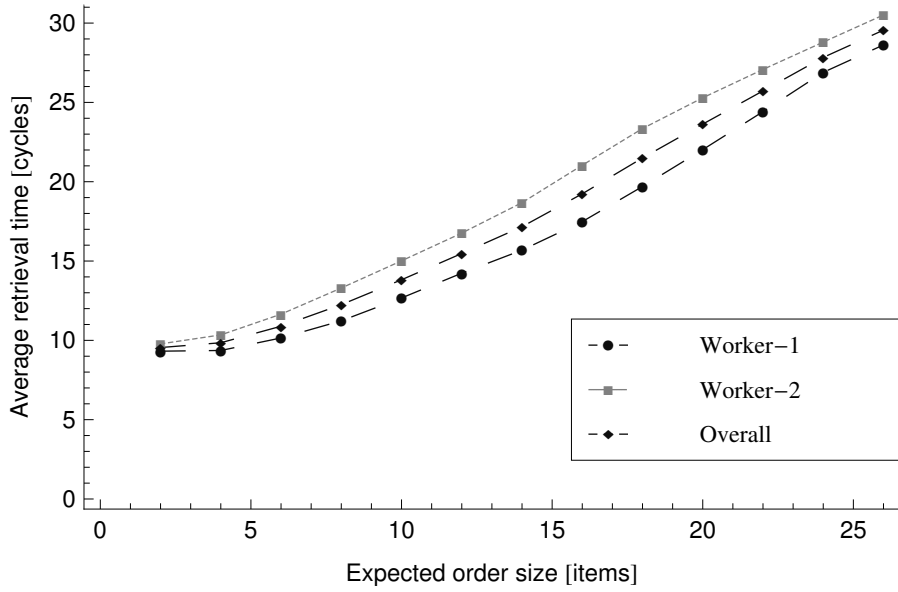


Figure 4.35: Average Retrieval Time Plot

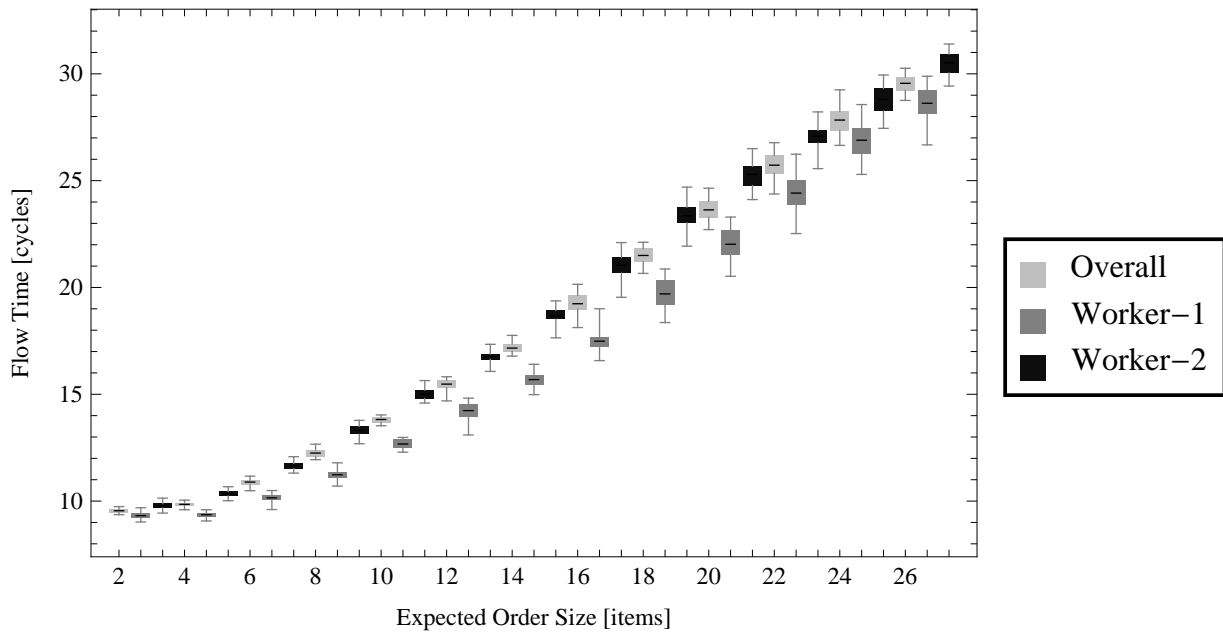


Figure 4.36: Average Retrieval Time for Two Workers

is expected due to having more traffic intensity in the two sided model. Average retrieval time of the two sided model is longer than the one sided GridPick (see Figure 4.43). Having more requested items leads to delay in the travel time of requested items.

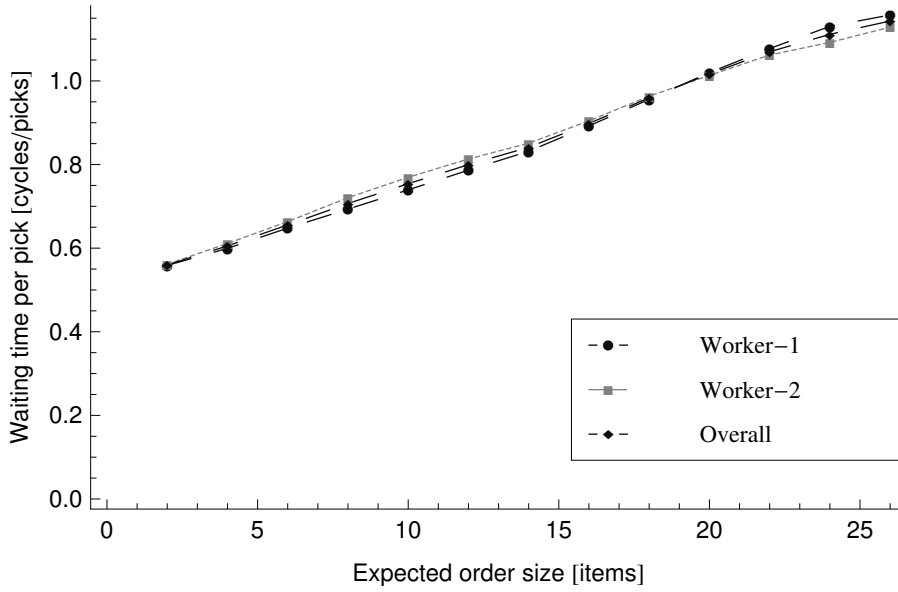


Figure 4.37: Waiting Time for Aspect Ratio Configurations

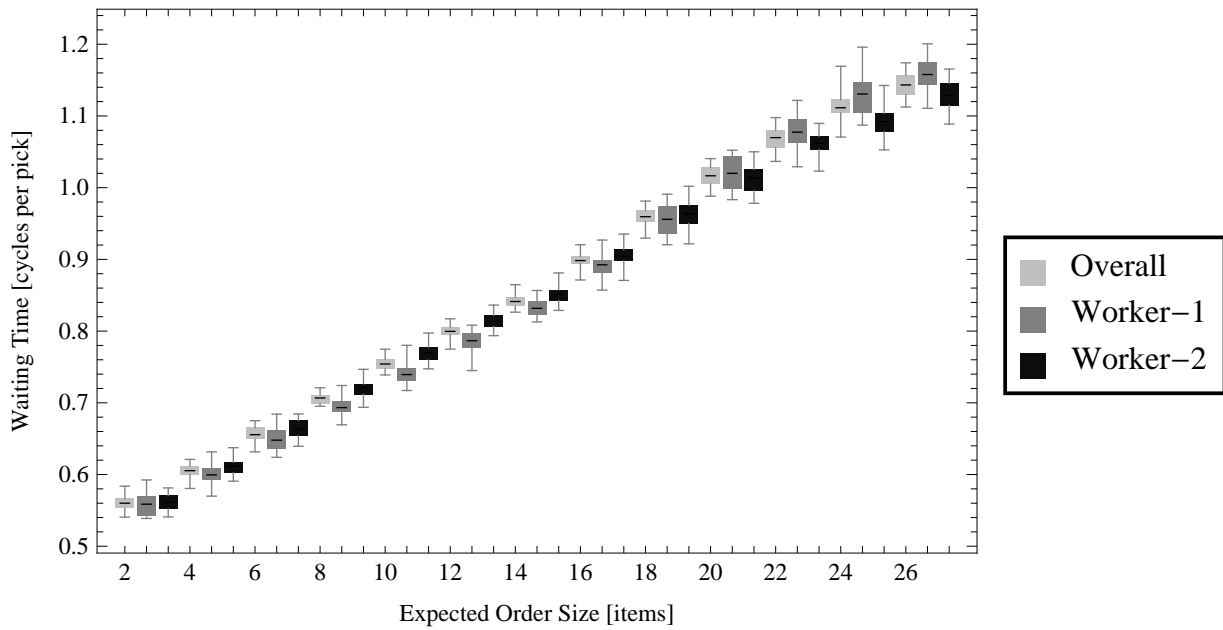


Figure 4.38: Waiting Time per Pick for Two Workers

Longer waiting times per pick in the two sided model are the result of longer average retrieval times (see Figure 4.45). Walking time per pick does not differ for one sided and two sided GridPick models (see Figure 4.47).

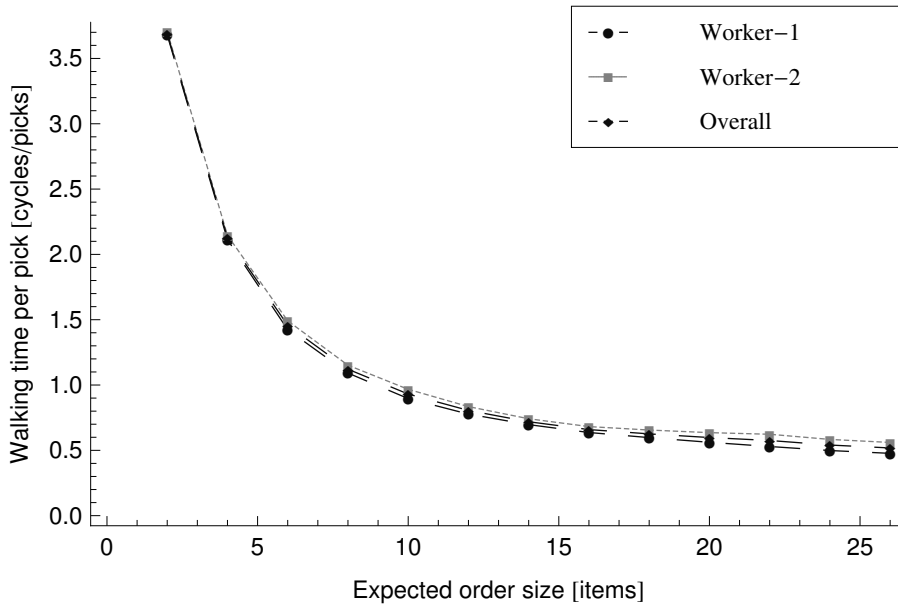


Figure 4.39: Walking Time for Two Workers

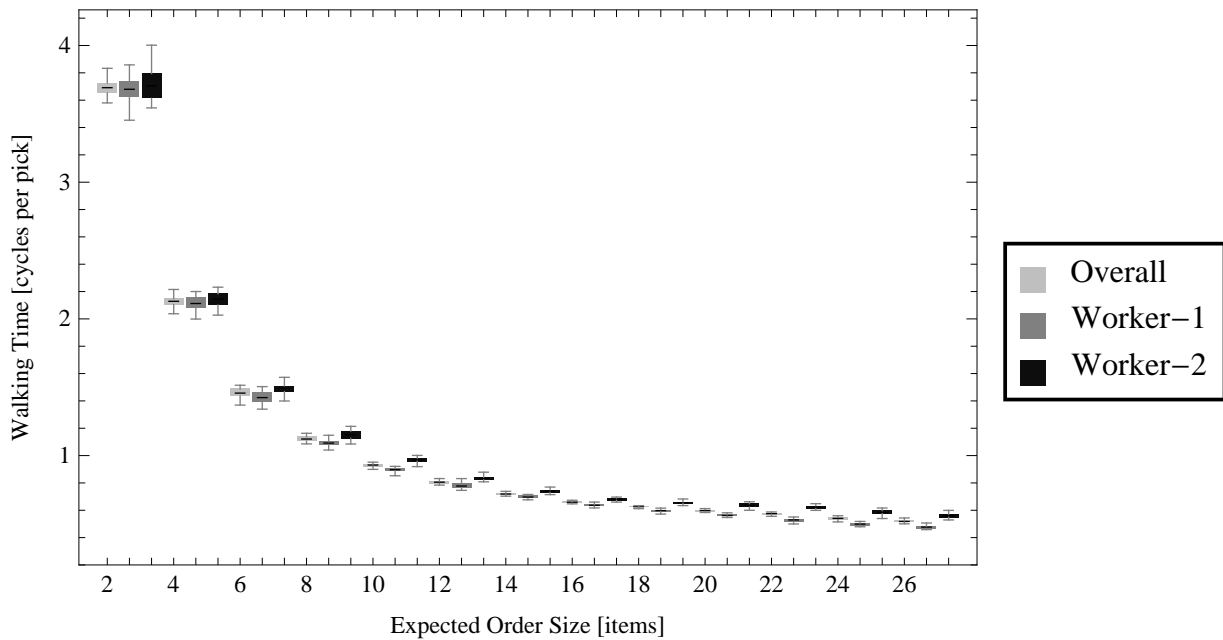


Figure 4.40: Walking Time per Pick for Two Workers

We have also looked at the percent difference between three configurations (see Figure 4.49 and 4.50). When the one sided GridPick model is the base case, two sided GridPick performs 80 - 95 % better than one sided GridPick with the cost of labor (one extra worker)

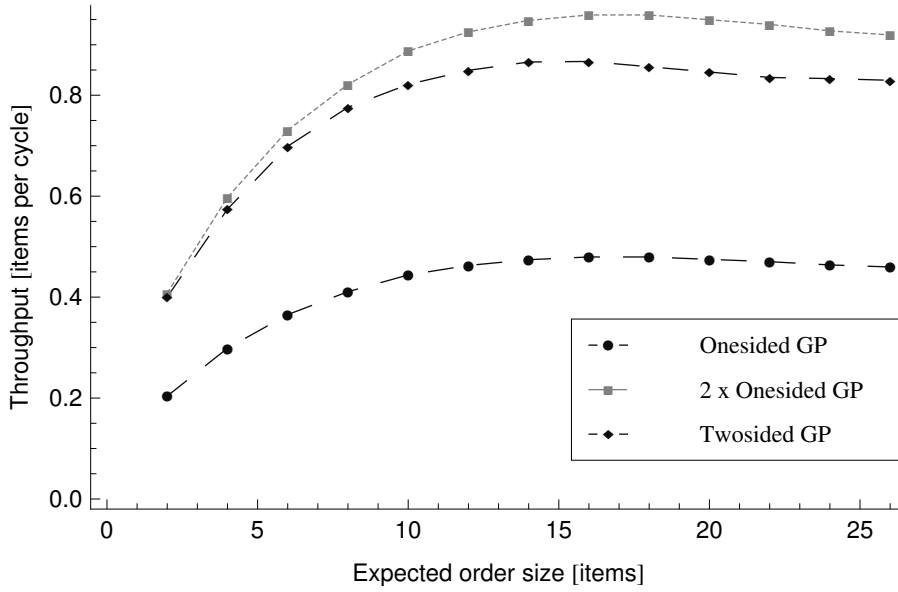


Figure 4.41: Throughput Plot

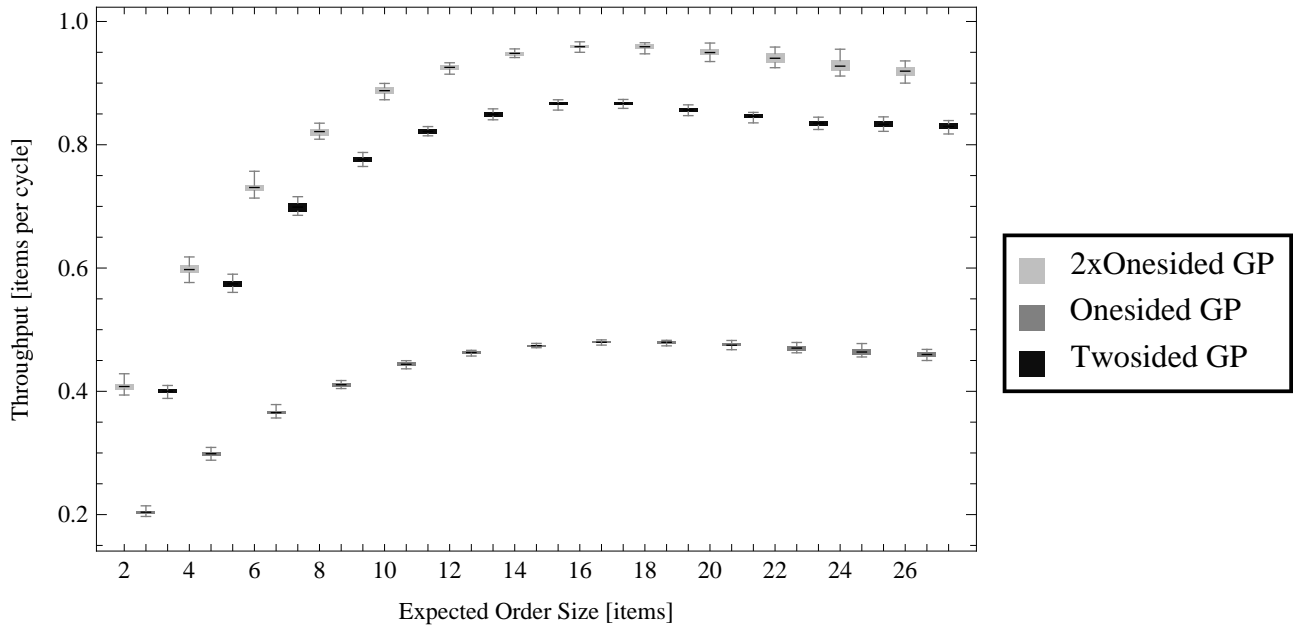


Figure 4.42: Throughput Plot for Two Models

(see Figure 4.49). Using two sides of the grid provides a great improvement on the throughput. If we employ two one sided GridPick instead of one, there would be a 100% improvement on the performance. Therefore, there is a difference of 5 - 20% in performance due to having two sided picking. The reason is increased traffic intensity in the grid. Vertical movements

	onesided vs. twosided	twosided vs. 2xonesided
2	0	0
4	0	0
6	0	0
8	0	0
10	0	0
12	0	0
14	0	0
16	0	0
18	0	0
20	0	0
22	0	0
24	0	0
26	0	0

Table 4.44: p values for Throughput of Two Models

T-value	onesided vs. twosided	twosided vs. 2xonesided
2	-145.3	-4.06
4	-161.15	-9.6
6	-183.22	-12.91
8	-277.24	-25.56
10	-399.42	-49.02
12	-442.83	-67.11
14	-483.75	-81.7
16	-566.57	-97.02
18	-415.43	-91.65
20	-382.01	-70.89
22	-253.77	-48.34
24	-253.11	-42.71
26	-318.17	-51.34

Table 4.45: t values for Throughput of Two Models

are constrained to the number of empty cells in the grid. More items moving in opposite directions causes a waiting for some items.

When two sided GridPick model is the base case, two grids of one sided GridPick model performs 1 - 12 % better than two sided GridPick, with the cost of another grid of unit modules (300 modules) (see Figure 4.50). This is a large capital investment for a 1- 10% improvement considering the availability of two sided picking.

	onesided vs. twosided	twosided vs. 2xonesided
2	(-0.19902, -0.19350)	(-0.01149, -0.00380)
4	(-0.27903, -0.27204)	(-0.02817, -0.01828)
6	(-0.33729, -0.32984)	(-0.03680, -0.02673)
8	(-0.36802, -0.36263)	(-0.04901, -0.04174)
10	(-0.379542, -0.375675)	(-0.06903, -0.06350)
12	(-0.388335, -0.384765)	(-0.07856, -0.07391)
14	(-0.393339, -0.390027)	(-0.08453, -0.08040)
16	(-0.388840, -0.386043)	(-0.094024, -0.090142)
18	(-0.378747, -0.375036)	(-0.10492, -0.10033)
20	(-0.373196, -0.369221)	(-0.10657, -0.10059)
22	(-0.36796, -0.36207)	(-0.10960, -0.10070)
24	(-0.37218, -0.36622)	(-0.09910, -0.09004)
26	(-0.37225, -0.36750)	(-0.09341, -0.08625)

Table 4.46: CIs for Throughput of Two Models

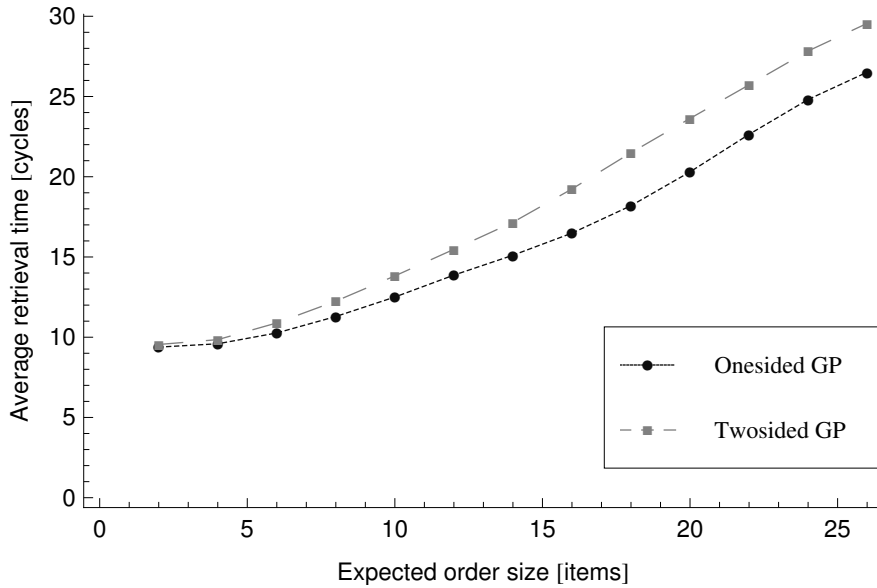


Figure 4.43: Average Retrieval Time Plot

4.6 Conclusions

In addition to the other flexibility features (modularity, reconfigurability, etc.), two sided GridPick provides another level of flexibility for the order picking system design for grid based storage systems. It provides the flexible scaling of throughput capacity in the

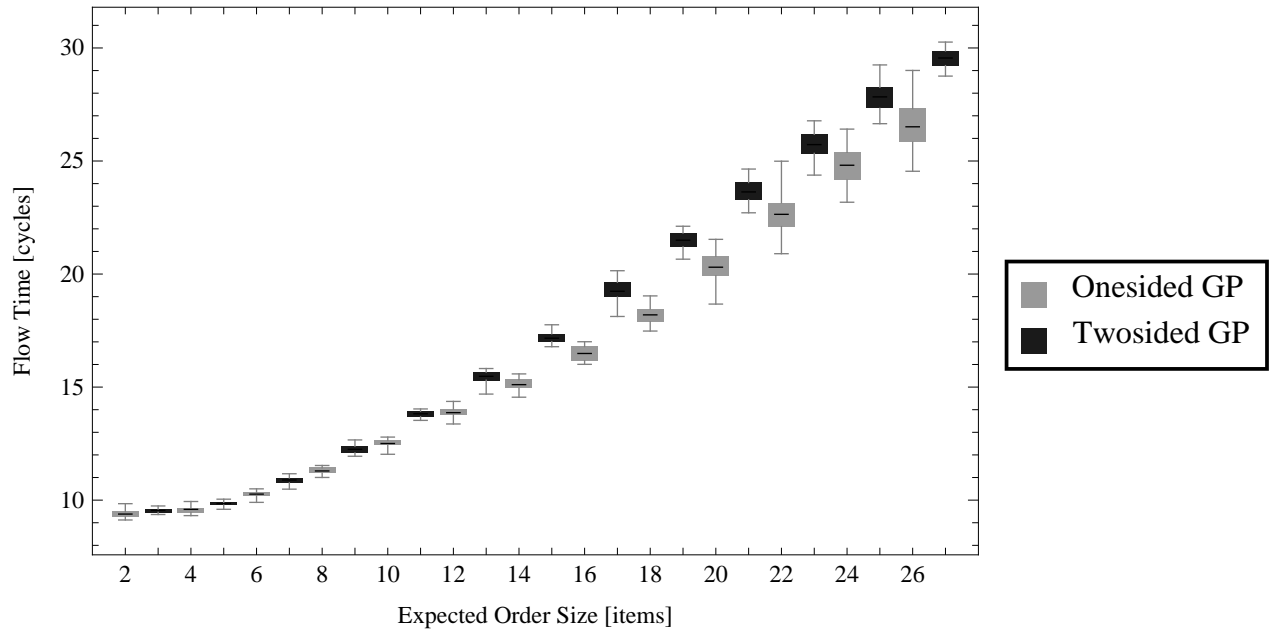


Figure 4.44: Average Retrieval Time for Two Models

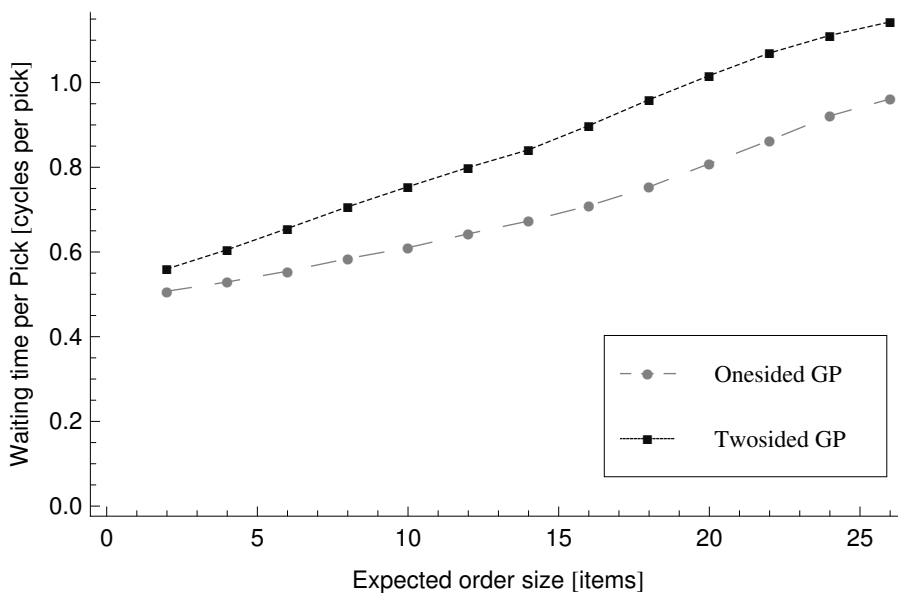


Figure 4.45: Waiting Time for One Sided vs. Two Sided

case of fluctuating demand levels. It also relieves the need for additional automation devices for the additional throughput.

We have discussed the effects of aspect ratio and distribution of empty cells on the performance. An application should consider the required expected order sizes and then

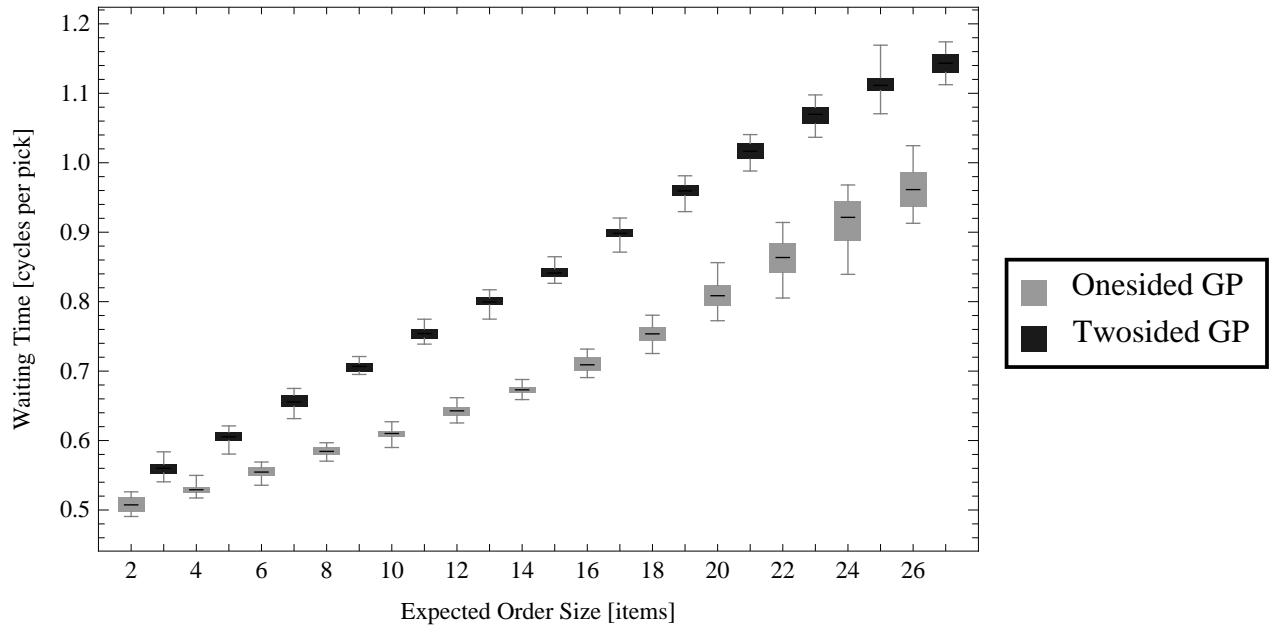


Figure 4.46: Waiting Time per Pick for Two Models

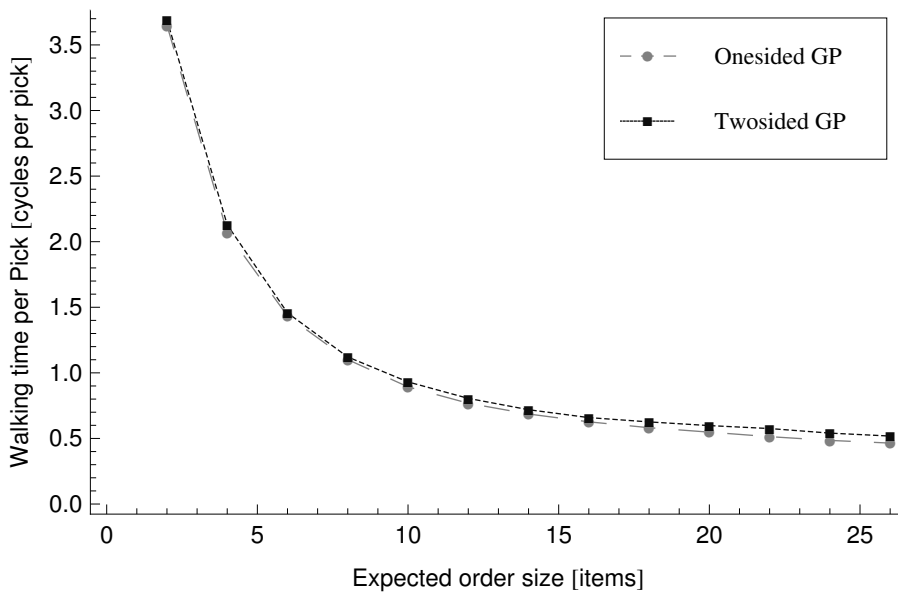


Figure 4.47: Walking Time for One Sided vs. Two sided

determine the appropriate configuration among the different values of parameters. An aspect ratio of 3 performs well with the respectively moderate expected order sizes (around 15 while number of columns is 30). Longer grids (aspect ratio of 8, for instance) can enable larger expected order sizes, but they result in worse performance. In the case of larger expected

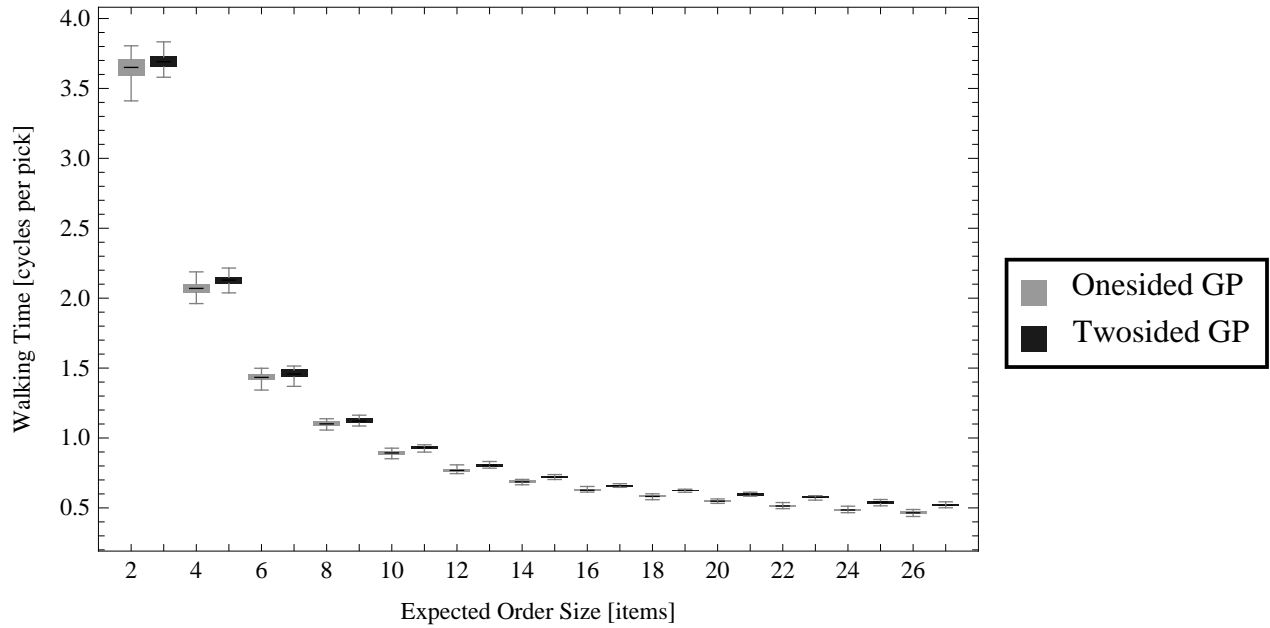


Figure 4.48: Walking Time per Pick for Two Models

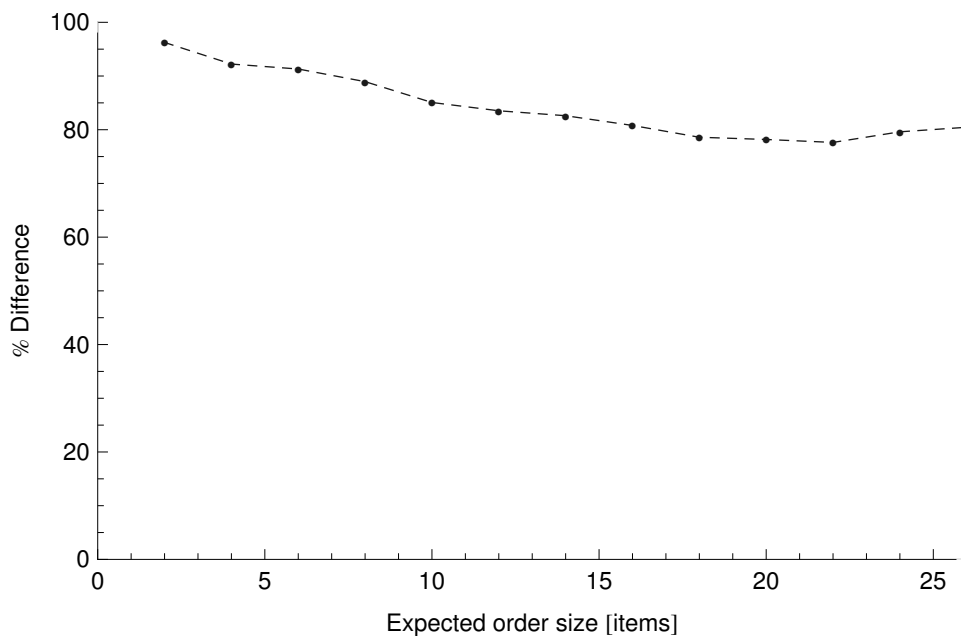


Figure 4.49: Percent Difference on Throughput for Two Sided vs. One Sided GridPick

order sizes, we may want to have a wider pick face. If we want to keep the number of columns large enough to have large order sizes, we should adjust the aspect ratio value accordingly. The performance will be better with a lower aspect ratio (aspect ratio of 3, for example), and additional conveyor modules. If we want to keep the number of modules lower, there

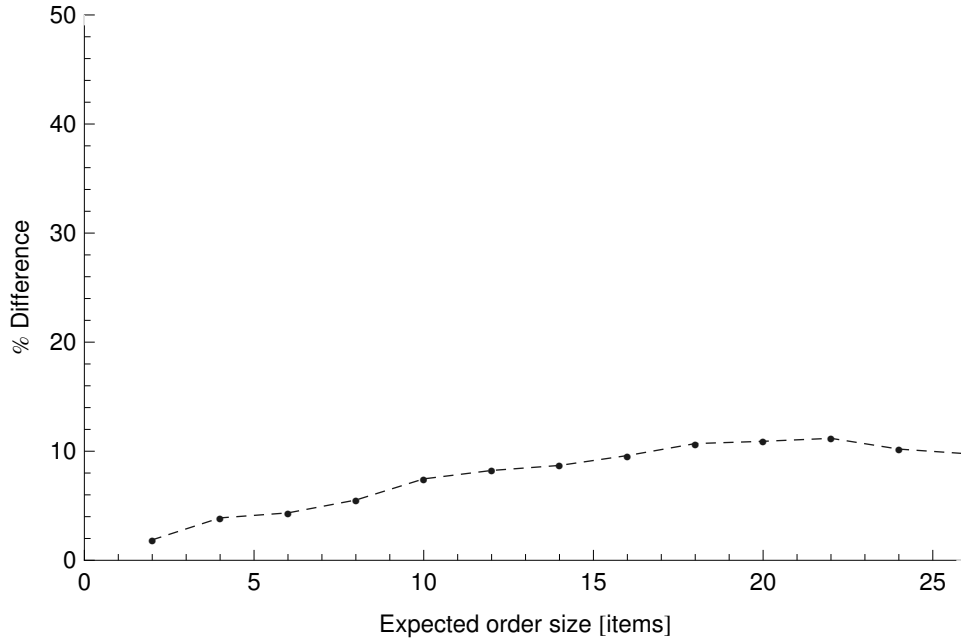


Figure 4.50: Percent Difference on Throughput for $2 \times$ Onesided vs. Twosided GridPick

would be a sacrifice in performance. Generally speaking, a configuration of 10 occupied cells to one empty cell ($k = 3$ in our experiments) seems to provide sufficient performance, which results in around 90% occupancy. Further increase in the number of empty cells provides small benefit.

Two sided GridPick is a significant extension with a 80-95% performance improvement with the cost of a single worker compared to the one sided GridPick. This extension provides the capability of additional throughput without any additional automation devices, which is a highly positive feature for automated system. Therefore, the automation system would be responsive to seasonal peaks and higher demand conditions. The two sided GridPick algorithm provides this availability without additional investment. In brief, two sided GridPick offers an effective alternative to the investments for a capacity expansion.

Chapter 5

A Petri Nets Model for Grid Based Storage Systems

Grid based storage systems are multi-agent systems in which conveyor modules interact and communicate to accomplish complex tasks. In multi-agent systems, individual units are expected to work together correctly in a large scale dynamic environment [Pujari and Mukhopadhyay, 2012]. A multi-agent system consists of autonomously working agents, which are evaluated as a discrete event dynamic system [Celaya et al., 2009]. These systems present concurrent and distributed computer system characteristics that have communication and interactivity based operations [Jensen, 1992]. The capabilities and complexity of multi-agent systems require formal analysis of system properties for these large scale dynamic systems [Celaya et al., 2009].

Simulation based models are able to show the presence of errors, validation of the system, and achievement of design objectives in a multi-agent system. In addition to the simulation environment, analytical methodologies are needed for the development of modeling and analysis approaches to verify formal system properties. Petri nets is an appropriate modeling tool because of its wide range of application areas such as communication protocols [Berthomieu and Diaz, 1991, Billington et al., 1988], distributed computer programs [Agerwala, 1979, Murata, 1989], flexible manufacturing design [Kamath and Vishwanatham, 1987], and material handling systems [Celaya et al., 2009, Basile et al., 2011, Dotoli and Fanti, 2002].

Petri nets modeling has a number of advantages: (1) Petri nets provide a graphical representation, which allows easy visualization of a large system. (2) Petri nets enable the capability of state space analysis for the detection of structural properties. (3) Corresponding systems can be modeled in various levels of abstraction or detail. (4) Performance analysis

of the system is available through timed Petri nets [Kamath and Vishwanatham, 1987]. (5) Petri nets can require less time and effort in the design, modeling and analysis of large software systems [Zurawski and Zhou, 1994].

5.1 Discrete Event Systems Modeling

A system is characterized as a discrete event system when its state space consists of discrete states and state changes happen in discrete time steps [Cassandras and Lafortune, 2008]. System components have several state changing actions or events at various time steps, which is also called an event driven structure. With the changing states of the sub-components, the whole system state also changes.

There are various discrete event modeling tools in the literature such as Automata theory, Markov chains, Petri nets, Supervisory control, and Queueing theory. We have chosen to use Petri nets because it allows analysis of structural properties. Petri nets theory is a strong discrete event modeling approach for a number of reasons: (1) Petri nets enable modeling of concurrent, distributed and parallel operating event driven systems. This makes it suitable for systems with decentralized control. (2) Petri nets provide analysis of structural properties such as boundedness, reachability, and liveness.

5.1.1 Petri Nets Overview

Petri nets theory is a graphical and mathematical modeling approach first introduced by Carl Adam Petri in 1962 [Murata, 1989]. It evolved over time with additional concepts and extensions. A basic Petri net includes a set of places (P), a set of transitions (T), a set of arcs (A) and an initial marking (M_0). Places are represented as nodes or circles in the net. Places are always connected to the transitions and transitions are connected to places with arcs. A marking represents the whole system state that is attributed to the tokens included in the places. Tokens travel through the net with the firing of the transitions. There can be significant variations of Petri nets. Arcs can have a weight function that represents the

required number of tokens flowing through the arc. Another property is having guards in the transitions. Guards are logical expressions that evaluate to true or false. If a guard is defined in a transition, it should also be true in addition to the required tokens in the places to enable the transition. Time delays and priorities can be also assigned to the transitions. Tokens can also have colors. So, the system can have specialized entities.

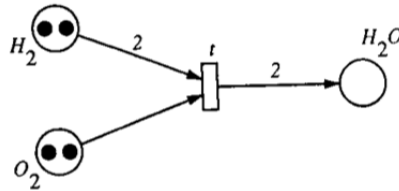


Figure 5.1: Marking before transition firing [Murata, 1989]

Figure 5.1 shows an example of an enabled transition. There are three places: H_2 , O_2 and H_2O . The transition requires two tokens from the H_2 and one token from the O_2 . Weight of the arcs is generally not shown, if the weight is one. An outgoing arc from the transition also has a weight of 2.

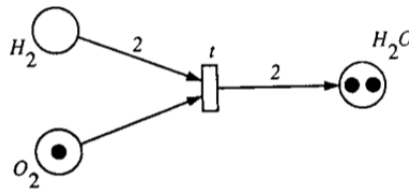


Figure 5.2: Marking after transition firing [Murata, 1989]

Figure 5.2 shows the markings after the firing of the transition. Now place H_2O has two tokens and the place O_2 is left with one token. Several structural properties can be applied to multi agent systems.

Reachability

The marking of all places corresponds to the state of the Petri net. The reachability graph has the marking of the Petri net as a node. So, a fully explored reachability graph

shows all possible system states of a net. If there is a sequence of transition firings from a marking M_i to a marking M_j , it is said that M_j is reachable from M_i .

Liveness

A Petri net is said to be live if it is always possible to fire a transition from a marking reachable from the initial marking. A closely related property is deadlock freeness. The liveness property ensures the absence of deadlocks. A marking of the net is dead if there is no enabled transition in this state. The Petri net is said to be live if there is no dead marking reachable from the initial marking. Deadlock is a set of places in which, for each transition, an output place is also an input place [Wegrzyn et al., 2004]. So, once all places are unmarked, they will never receive any token again.

Boundedness

The Petri net is said to be k -bounded, if the number of tokens in each place does not exceed a finite number, k , for any marking reachable from the initial marking $R(M_0)$. If k is 1, then the system is said to be a safe Petri net.

For more computational power, colored Petri nets and high level Petri nets are introduced. With colored Petri nets, tokens have distinct features. In high level Petri nets, tokens consist of data values. So, the tokens carry more information. We give more information about the high level Petri nets in the related chapter.

There are several Petri nets modeling tools in the literature. Colored Petri net (CPN) tools is one of the many developed software programs for Petri nets modeling. It is a popular program for the modeling of high level Petri nets, introduced by Jensen et al. [2007]. It has a graphical user interface (GUI) and support for state space analysis. Especially in computational sciences, CPN tools is used for Petri nets modeling and protocol verification. There is a number of applications that are using CPN tools as a modeling program. Xu et al. [2008] presented Petri nets modeling and formal verification of XML firewall with

CPN tools that is in use for web services of businesses. Sornkhom and Permpoontanalarp [2008] introduced a colored Petri nets model of Micali's fair contract signing protocol and analysis with CPN tools. These protocols play a crucial role in e-commerce in which a third trusted party ensures security of both sides. Wang et al. [2008] represented colored Petri nets modeling and verification of structural properties for a stream control transmission protocol with CPN tools. It is a protocol for reliable and secure transmission of applications over networks. There are several other tools such as LoLA [Schmidt, 2000], Alpina [Buchs et al., 2010], and Helena [Evangelista, 2005] providing condensed state space analysis.

Petri nets modeling approach is also used in logistics applications. Celaya et al. [2009] presented Petri nets modeling of a simple example of two agents carrying objects from one point to another cooperatively. They construct the reachability graph and show the liveness property of the system with invariants analysis. Dotoli and Fanti [2002] proposed a colored Petri net model for an AS/RS integrated with rail guided vehicles, which provides deadlock avoidance and control policies analysis. He et al. [2007] introduced Petri nets modeling of an AS/RS with three cranes, narrow aisles, and buffer stations. They used the model in a modular approach for the monitoring of control and scheduling policies. Hsieh et al. [1998] divided the AS/RS into four different modules: command typing module, profile module, compilation module and execution module. Then, they proposed a Petri net based structure for the operation modeling of AS/RS.

5.2 Fundamental Concepts

A Petri net is a graphical language for the modeling of concurrent systems, which are special bipartite directed graphs [Kamath and Vishwanatham, 1987]. Standard Petri nets consist of three kinds of components: a set of places, a set of transitions, and directed arcs. These arcs connect input places to the transitions, and they connect the transitions to the output places. Tokens flow through these places by firing transitions that represent a particular condition or state of the system.

Input Places	Transitions	Output Places
Conditions	Events	Conclusions
Input Signals	Signal Processors	Output Signals
Resources Needed	Tasks or Jobs	Resources Released
Input Data	Computation Steps	Output Data

Table 5.1: Some Interpretations of Places and Transitions [Murata, 1989]

There are several interpretations of input places, transitions, and output places (see Table 5.1). In a material handling system, places typically represent a resource (device) or a state of the system. Transitions typically represent the movement of the goods or an event.

The formal definition of a Petri net has a 5-tuple $N = (P, T, I, O, M_0)$ [Wang, 2007], where:

- (1) $P = \{p_1, p_2, p_3, \dots, p_m\}$ is a finite set of places,
- (2) $T = \{t_1, t_2, t_3, \dots, t_n\}$ is a finite set of transitions,
- (3) $I : P \times T \rightarrow N$ is the set of directed arcs from input places to transitions. N is a nonnegative integer.
- (4) $O : T \times P \rightarrow N$ is the set of directed arcs from transitions to output places. N is a nonnegative integer.
- (5) $M_0 : P \rightarrow N$ is the initial marking of the Petri net, which corresponds to the tokens present in places.

There are additional features and extensions to Petri nets. A weight function can be assigned to an arc in which the arc carries multiple tokens. Another variation is the inhibitor arcs that enable the transition in the case of absence of tokens in the input place. Inhibitor arcs are shown with a circle instead of an arrowhead. We can also assign priorities and guards to the transitions. Priorities determine the sequence of transition firings if multiple transitions are enabled at the same time. Guards are additional logical operators for enabling the transitions. In the presence of guards, tokens of the input places do not guarantee a sufficient condition for the transition firings. The guard should also evaluate to true for an enabled transition. Transitions may have either immediate firings or time delays.

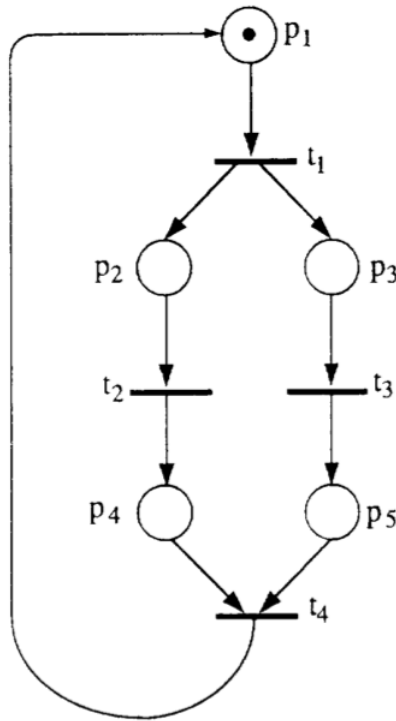


Figure 5.3: Graphical Representation of an Example Petri Net [Zurawski and Zhou, 1994]

Figure 5.3 shows a simple Petri net that has a token in the place 1. It consists of five places and four transitions. Place 1 is the input place for transition 1, and transition 1 is enabled in this configuration. Transitions 2 and 3 are synchronous.

5.3 Colored Petri Nets

Petri nets applications and modeled systems can be quite large and complex, depending on the level of detail required. Modeling of several identical and similar processes will necessitate a large net with a great number of places, arcs, and transitions. Therefore, in some cases representation by Petri nets will not be useful or even not possible. A more powerful and functional type of Petri nets is Colored Petri Nets (CPN) [Jensen, 1981, 1992, 1994, 1997]. A Colored Petri Net is the generalized form of Petri nets that has an associated color set for each token [Kamath and Vishwanatham, 1987]. Furthermore, a set of colors may be associated with places and transitions.

Tokens can carry complex data values with the help of color sets. Each place can include one or more tokens, which is called *marking* of the place. Additionally, each place will have a data domain or a place inscription that shows the type of token the place can carry. Color sets should be defined initially that represent the data formats of the tokens. Initial token assignment to the places refers to the *initial marking* (M_0) of the Petri net. With the transition firings, tokens will move within the places and a *current marking* (M_i) will refer to the current state of the Petri net. Directed arcs also have a data domain or an inscription that shows the type of data that particular arc can carry during the firing of a transition.

Occurrence of events refers to the firings of the transitions in a Petri net. A transition should be enabled to fire in a given marking. A transition is enabled when a *binding* of variables is present in the input places that matches with the arc inscriptions [Jensen et al., 2007]. During the firing of a transition, tokens are removed from the input places that match the arc inscriptions, and tokens are added to each output place according to the corresponding output arc inscription. Since the colored Petri net is a generalization of the Petri nets, colored Petri nets can unfold to be represented in a Petri net.

Most Petri nets research study groups have developed their own software tool that provides the modeling, analysis, and simulation environment for Petri nets applications [Murata, 1989]. There are a number of Petri nets tools developed by scholars, which include different capabilities and features. Some of them have a graphical user interface (GUI) for easy visualization. Some other tools focus on systems with large state spaces (condensed) that employ different techniques for the exploration of state spaces.

We have chosen to use “CPN Tools,” which provides a graphical user interface (GUI) and the capability of modeling in colored Petri nets [Jensen et al., 2007]. It also has a fairly good online documentation. Furthermore, CPN tools provides a simulation capability and error checker for the validity of developed models. CPN tools does provide state space analysis but, to the best of our knowledge, it does not use techniques to analyze condensed

state spaces. For state space analysis, we use “LoLA.” LoLA does not have a GUI but it has the ability to handle condensed state space analysis. LoLA requires the model in a textual format with its own modeling language. It runs in Linux or in a Cygwin environment on a Windows computer.

With the state space analysis of Petri nets, we can detect and analyze several properties of a system (boundedness, reachability, safeness, liveness, etc.). We are particularly interested in the absence (or presence) of deadlocks in our systems. Deadlock is a reached state that does not have a possible consecutive state and there is not any further available action [Iordache et al., 1999]. This definition matches the understanding of deadlock in Petri nets. We can examine the state space of the Petri net and determine a dead state, which does not have a subsequent state, in the presence of a deadlock.

5.4 Modeling of GridStore System

We start Petri nets modeling with GridStore system since it includes the negotiations for the movement decisions, which are also used in GridPick. The GridStore algorithm handles vertical movements of the requested items and horizontal movement of the occupied items. In GridStore, items leave the system and replenishing items reach the same row to keep the occupancy in balance. The GridPick model is a harder problem in which items do not leave the grid and occupancy should be adjusted in an alternative way. We keep the occupancy in balance with the items moving in opposite direction to “substitute” the requested items. Therefore, with GridStore we obtain the modeling of main movements. After observing that it is deadlock free, we can model a more complex model on top of it. By defining these events in GridStore Petri nets model, we acquire the modeling of fundamental events, so we can build on it with additional insights.

Different scenarios and approaches can be adopted for the modeling of a particular system. In our Petri nets modeling approach for grid based storage systems, nodes represent

the conveyor modules. Tokens represent the storage containers moving on the grid of conveyor modules. Each transition corresponds to an event for a particular movement or a state change of the storage container.

We have represented the variables and the color sets with the syntax defined in CPN tools. We have defined two color sets for the tokens. Each storage container has a state and target row information declared as:

colset state = int with 0..3;

colset targetrow = int with 0..2;

In the state color set, 0 represents the empty state, 1 shows the occupied state, 2 indicates the requested state, and 3 corresponds to the replenishing state. In the “targetrow” color set, integer values represent the target row of the storage container. For the domain or inscription of the places, we have defined a color set called “box” that includes the state and target row color sets, which means the places can only include the tokens (representing storage containers) in this format:

*colset box = product state * targetrow;*

“*” is the combination operator of the color sets, not a multiplication operator. It is the syntax of CPN tools. For instance, (3,1) represents a replenishing unit, which has target row of 1. Since we use target row value for only replenishing units, other tokens have a target row value of 0. For instance, (2,0) corresponds to a requested unit.

The replenishing and requested items move in a similar manner. The only difference is: the replenishing items turn into occupied when they have arrived to their target row. For this reason, we have defined a sub color set for the requested and replenishing units and another sub color set for the occupied and empty units. Now, we can refer to both requested and replenishing units by defining a variable from the color set *mov* and we can refer to occupied and empty units by defining a variable from the color set *nonmov*.

colset mov = subset state with [2,3];

colset nonmov = subset state with [0,1];

Another important point on the modeling approach is: we have tokens to represent the “empty” state. During a movement, empty tokens are also moving compatible with the particular movement. Therefore, we can follow the movement of empty spots.

In GridStore, since we have one synchronous event executing after another one, the events have a sequence and priority. For instance, North-South negotiation executes before East-West negotiation. Therefore, North-South negotiation has a priority over the East-West negotiation. CPN tools has a priority feature and we can assign priorities to the transitions in this respect (higher priority or a smaller value to the North-South negotiation). However, LoLA does not support priority assignment to the transition. For this reason, we have defined the transitions with a priority enforcement without any assignment of priority to the transitions.

We have defined places to form a 3×3 grid as $P00, P01, P02, P10, P11, P12, P20, P21, P22$. Additionally, we have a place to release request, $NReq$ and a place to replenish items, Rep . To limit the capacity of $NReq$ and Rep places, we have described anti-places, $AntiReq$ and $AntiRep$. Therefore, each time $NReq$ is an input place, $AntiReq$ will be the output place and vice versa. Also each time Rep is an input place, $AntiRep$ will be the output place and vice versa.

State Variables	Target Row Variables
$var\ s0 : mov;$	$var\ t0 : targetrow;$
$var\ s1 : mov;$	$var\ t1 : targetrow;$
$var\ s2 : mov;$	$var\ t2 : targetrow;$
$var\ s3 : mov;$	$var\ t3 : targetrow;$
$var\ snr : nonmov;$	$var\ t4 : targetrow;$

Table 5.2: Variables for the Arc Inscriptions

Several variables are defined to use in arc inscriptions and to assign color sets to the binding of the transitions (see Table 5.2). Variables $s0...s3$ refer to the presence of requested or replenishing items. Variables $t0...t4$ indicate the target row values. Variable snr represents an empty or occupied state.

We have enumerated all possible movements with the transitions in the model. So, there are a few places and many transitions. Graphical representation of this net is impractical due to the high number of arcs going in and out of the places. We have used a feature called fusion places to represent transitions separately. When we define transitions in different pages and fuse the places, they refer to the same places and the result of a transition firing is reflected in all nodes of the fusion set. Graphical representations are shown for better understanding (see figures starting from Figure 5.4).

Symbols in the circles represent the name of the node, and the numbers on the top right corner of the node show the current marking of the node with the token inscription. Arc inscriptions on the arcs indicate the binding of the transition with the defined variables. Definitions in the transition bar denote the transitions' names and we assign a time delay for representation purposes. We also define a boolean expression as a transition guard at the top right corner of the transition if necessary.

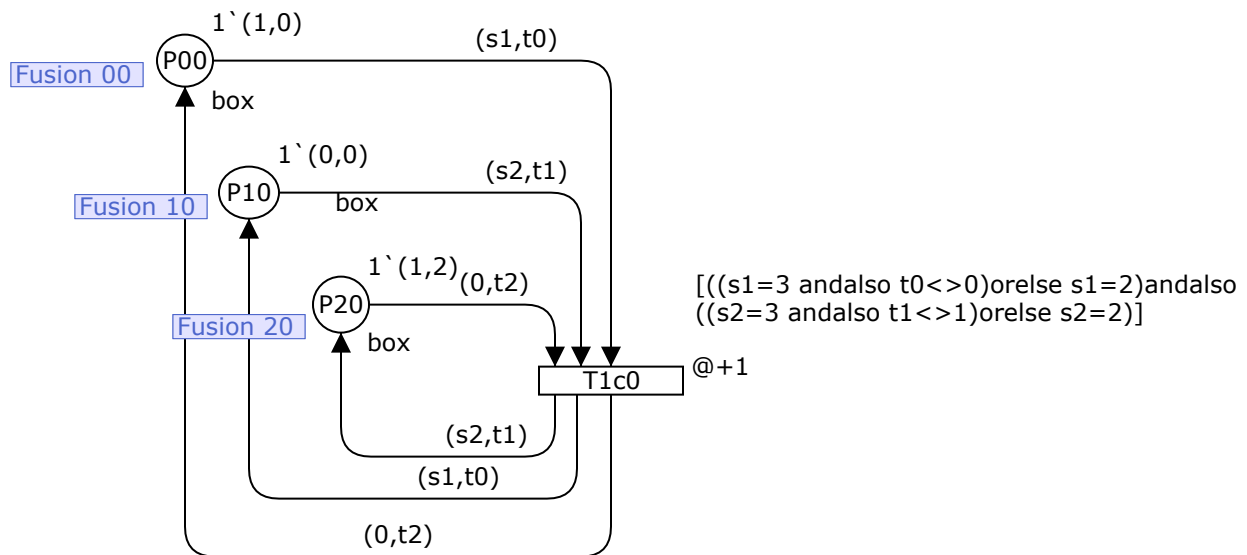


Figure 5.4: Vertical Convoy Movement in Column 0

Figure 5.4 shows a convoy movement. Place P20 is empty, and places 00 and 10 have either a requested or a replenishing item. Items in places 00 and 10 can move down to places 10 and 20. Since we do not assign any priority, we should check if a replenishing item has

arrived to its target row. We have declared a boolean expression to the transition guard for this reason. It enables the transition if the item is requested or a replenishing item has not arrived to its target row.

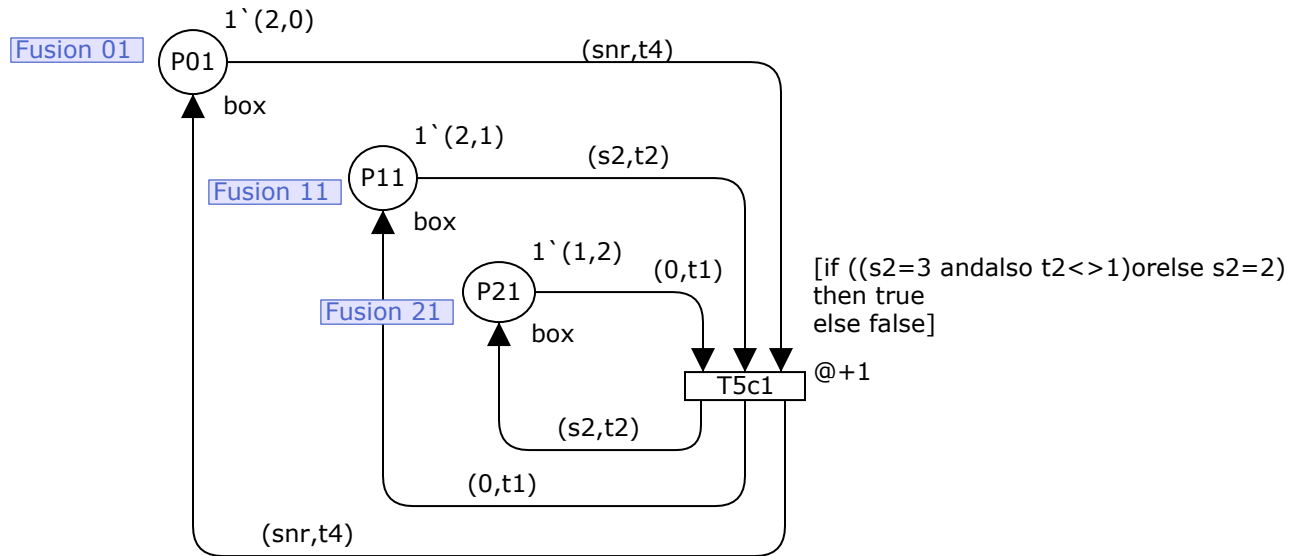


Figure 5.5: Vertical Movement in Column 1

Figure 5.5 indicates a vertical movement with one item from row 1 to row 2. Place 21 is in state empty and place 11 carries a requested or a replenishing item. The transition ensures the absence of a requested or replenishing item in row 0 by having it as an input place. The arc inscription from place 01 to the transition has *snr* color set, which corresponds to an empty or occupied unit. A transition guard evaluates to true and enables the transition if the module in row 1 has a requested item or a replenishing item has not arrived to its target row.

Figure 5.6 represents a vertical convoy movement in rows 1 and 2. Since we require that in the availability of convoys, items move with a convoy movement, the transition checks the absence of a requested or replenishing item in row 0. We also have a transition guard to confirm that the item in place 21 is either a requested item or a replenishing item that has not arrived to its target row. In this transition, the item in the last row (2nd row) leaves the system; for the release of a new request, we add a token to the place *NReq*. For a replenishing

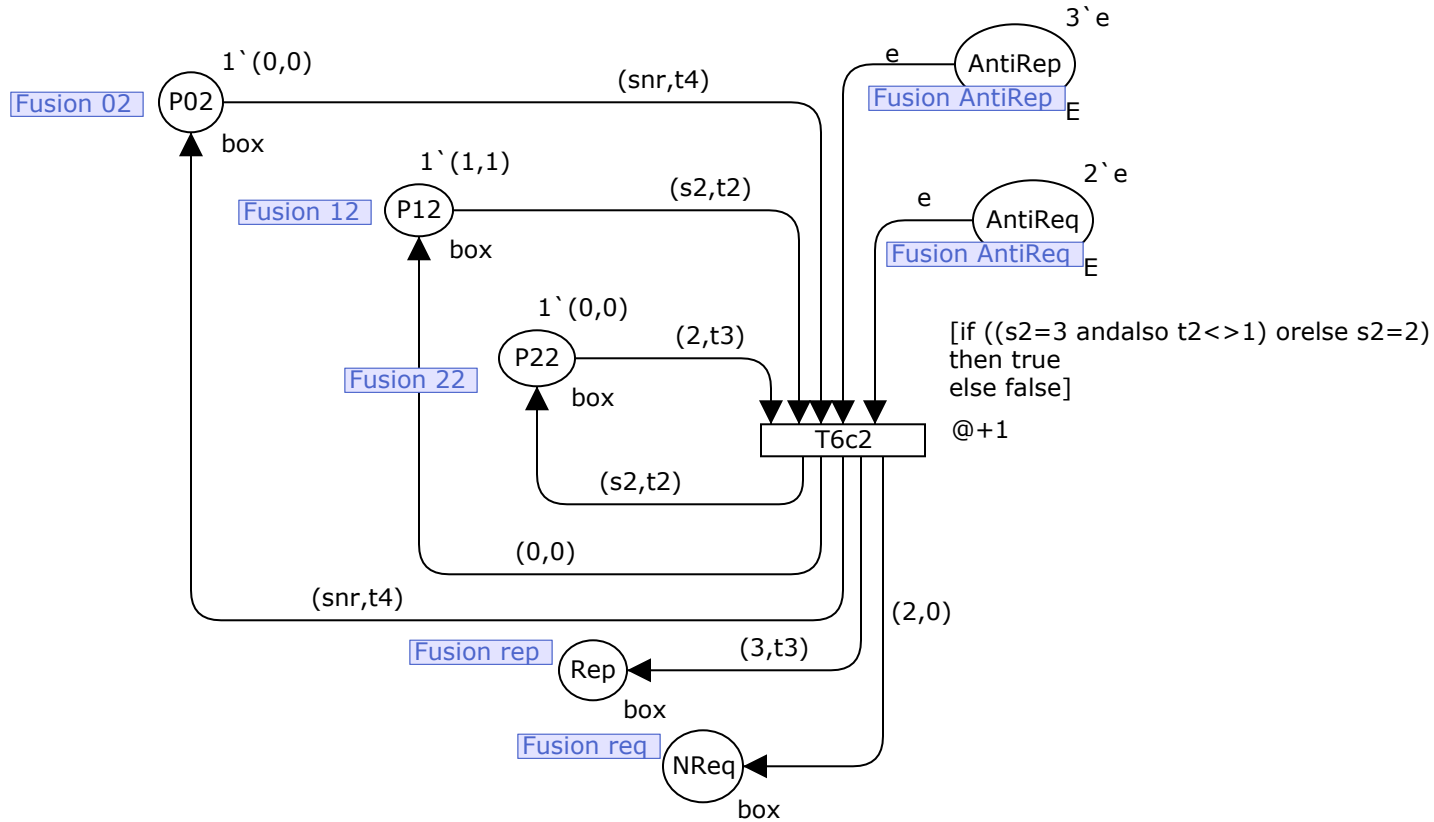


Figure 5.6: Vertical Convoy Movement in Column 2

item from the top, we add a token to the place *Rep*. To limit the capacity of these places, *AntiRep* and *AntiReq* places are input places. In this case, $WIP = 2$; therefore, *AntiReq* has 2 tokens initially. We assume an unlimited capacity for the replenishment row. However, for a finite state space, we assign a large enough value for the place *AntiRep*.

In Figure 5.7, this transition is possible when there is a requested or a replenishing item in place 00 that needs to move down. Furthermore, places 10 and 11 carry occupied items and place 12 is in state empty. To allow the priority of vertical movement, the transition checks the absence of a requested or a replenishing item in place 02. The transition is only enabled when there is an occupied or empty token in place 02. After the firing of the transition, place 10 becomes empty, and occupied items move to places 11 and 12.

In Figure 5.8, the transition is enabled when place 11 has a requested or a replenishing item that needs to move down. Place 21 has an occupied item that should allow the passage

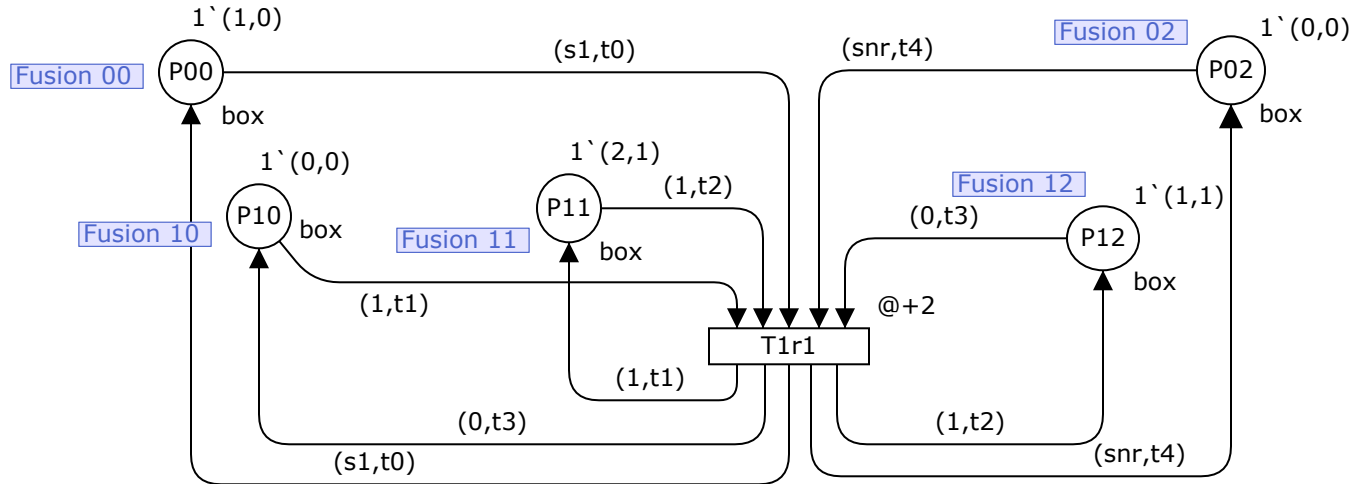


Figure 5.7: Horizontal Movement in Row 1

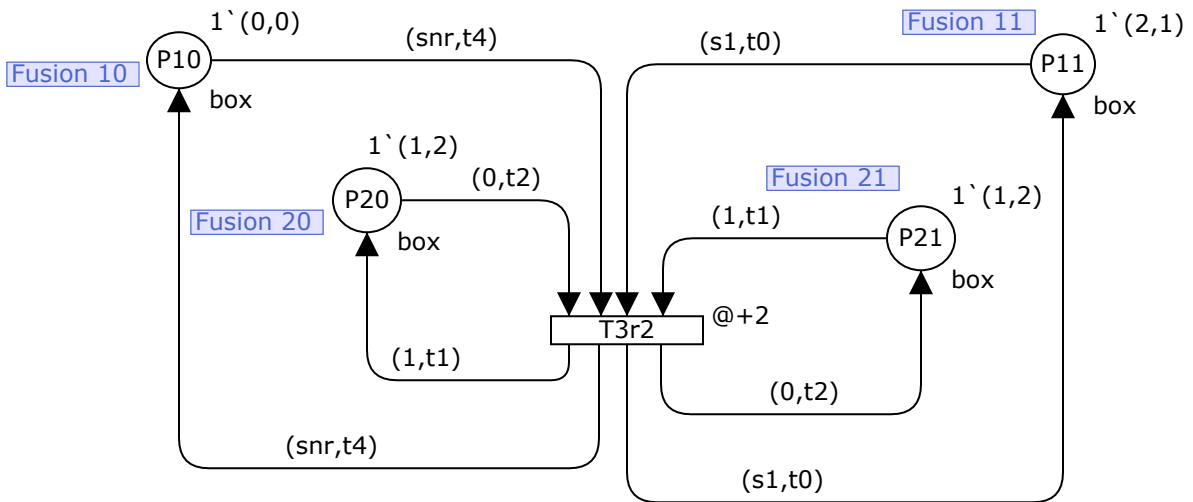


Figure 5.8: Horizontal Movement in Row 2

of the item to the north. Place 20 is in the empty state and there is no requested or replenishing item in the north neighbor that will move to place 10.

When a replenishing item arrives to its target row, it should turn into an occupied item. Figure 5.9 shows the transitions from replenishing to occupied for an item. Each transition has a guard to check the target row, which evaluates to true if the target row matches the current row. Since we do not assign a priority to the transitions, we have added guards in the vertical movement transitions. These replenishing items should not move in the vertical movement events if they have arrived to their target row.

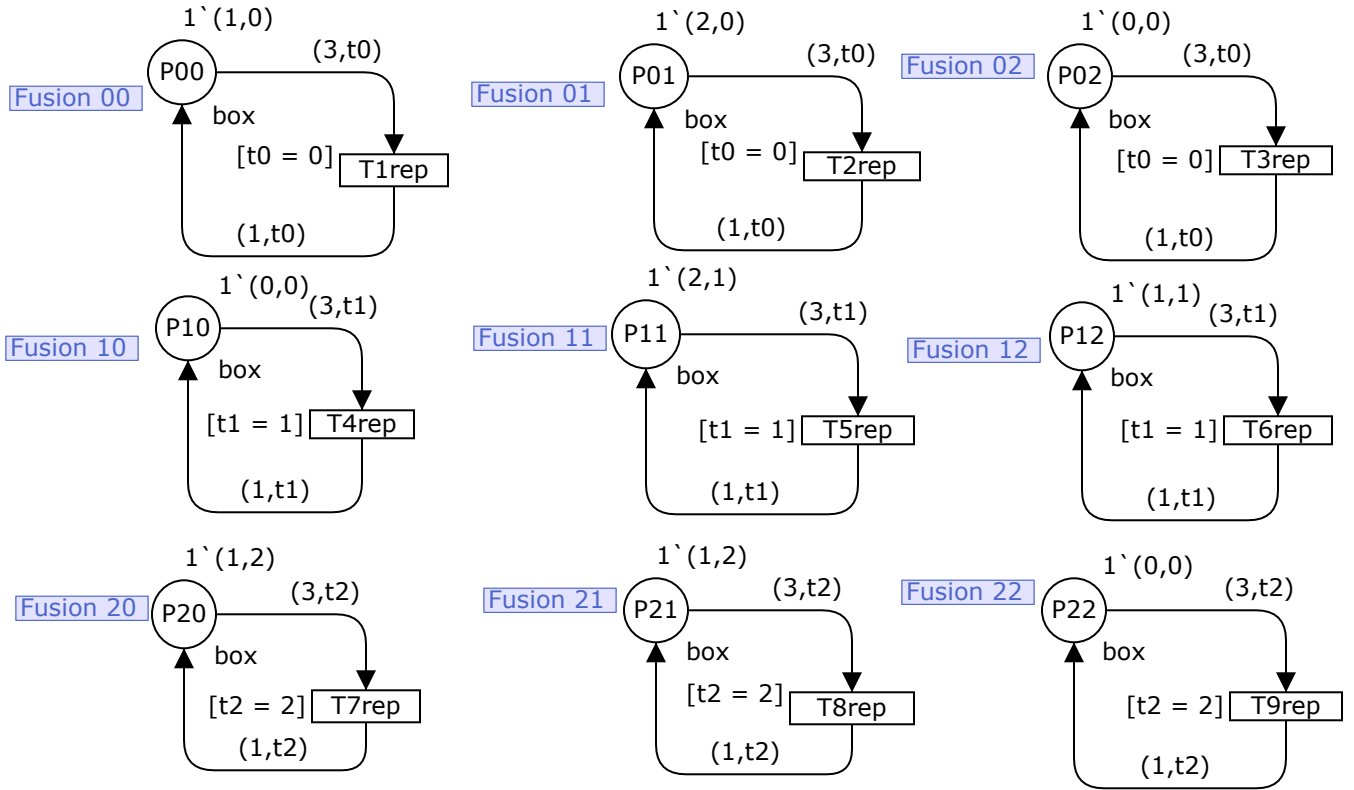


Figure 5.9: Transition from Replenishing to Occupied

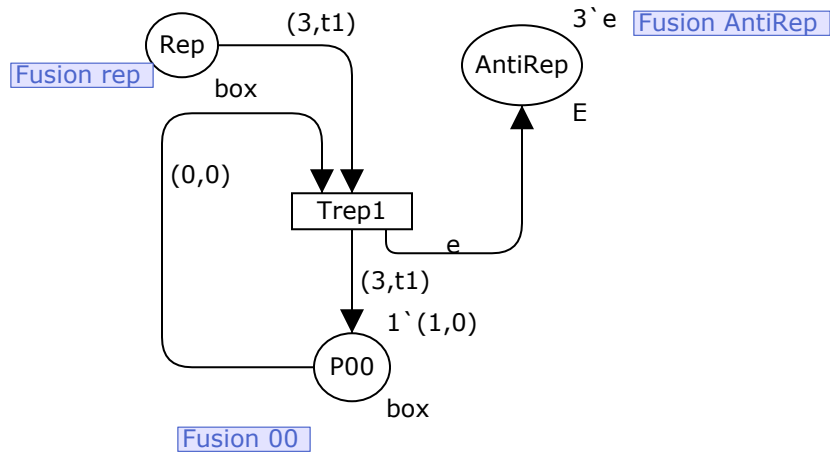


Figure 5.10: Transition for the Replenishment From the Top Row

When a requested item leaves the system, we add a token as a replenishing item to place *Rep*. place *Rep* is our replenishment queue. Figure 5.10 represents a transition for the movement from the replenishment queue to the top row. This transition is enabled if place 00 is in state empty and there is a replenishing item in place *Rep*. Upon firing of the

transition, the replenishing item enters the grid. To have a finite capacity for place *Rep*, the transition has *AntiRep* as an output place.

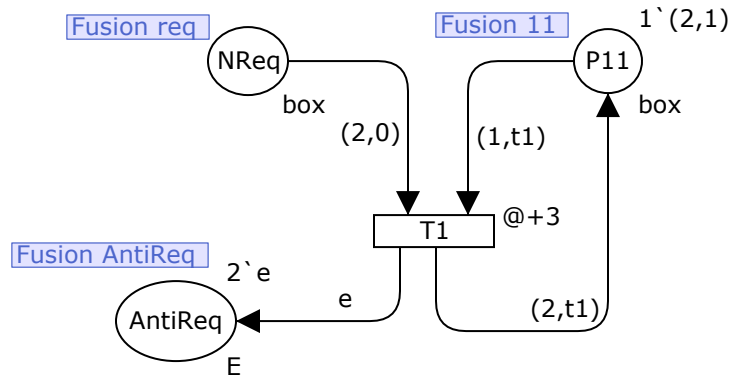


Figure 5.11: Transition for the Release of the Request

After a requested item has left the grid, another request will be released. In Figure 5.11, the transition is enabled when there is a token in place *NReq* and the corresponding place (01 for this case) has an occupied item. To have a finite capacity for place *NReq*, *AntiReq* is the output place for the transition.

We have used the LoLA Petri nets tool for the state space analysis of a 3×3 GridStore system. We have prepared the model with LoLA's modeling language with the appropriate syntax. Figure 5.12 shows the output window, and the system does not have deadlocks. It appears that LoLA unfolds the colored Petri net to have a basic Petri net, and performs the state space analysis with the unfolded Petri net.

5.5 Petri Nets Modeling of GridPick

For the structural analysis of GridPick, we have modeled the system with Petri Nets. We are interested mainly in the deadlock conditions. We have used CPN tools for the graphical user interface and simulation environment, and the package Helena for the analysis. To detect that the Petri Nets model shows the desired behavior and to validate the model, we have used the simulation ability of CPN tools. With Helena, we have detected possible deadlocks in the system. We have explored the deadlock conditions of one sided GridPick.

```

/cygdrive/c/Users/Onur/Desktop/lola
lola: 40 transitions
lola: 20 significant places
lola: net does not have deadlocks!
lola: >>>>> 21 States, 40 Edges, 21 Hash table entries

Onur@Uludag_Ailesi /cygdrive/c/Users/Onur/Desktop/lola
$ lola-deadlock /cygdrive/c/Users/Onur/Desktop/lolamodels/modelgrf1_noreprow_3x3
.lola
lola: 103 places
lola: 7254 transitions
lola: 75 significant places
lola: st:      53707  edg:    100000
lola: st:      93048  edg:    200000
lola: st:     125006  edg:    300000
lola: st:     154596  edg:    400000
lola: st:     179949  edg:    500000
lola: st:     193850  edg:    600000
lola: st:     198514  edg:    700000
lola: st:     200869  edg:    800000
lola: net does not have deadlocks!
lola: >>>>> 201614 States, 855511 Edges, 62565 Hash table entries

Onur@Uludag_Ailesi /cygdrive/c/Users/Onur/Desktop/lola
$

```

Figure 5.12: Output of LoLA Software Package for a 3×3 GridStore

We have not modeled and explored two sided GridPick's deadlock conditions due to the similar logic of the two sided model and level of complexity.

In this section, we explain the Petri nets model in detail. For the unit modules, we have defined places according to their location on the grid such as P00, P01, P11, etc. We represent the storage containers with colored tokens. There are two color sets. First one is for the state of the storage container (requested, occupied, replenishing, empty). Absence of a storage container is also represented with a token (empty). The second color set is for the row information of the storage container (target row in general). Because we have consecutive discrete events in each iteration, they are prior to one another. For each event, we have defined transitions and assigned priorities for the transitions. This is an enumeration based approach, so we have defined all possible state changes, assignments, and movement decisions with transitions.

Variables for the place domains are defined with the Helena package as follows:

- *type state : enum(e0,o1,r2,p3,bp);*
- *type targetrow : range 0 .. T;*

- *type E : enum(e);*
- *type pickerprocess : enum(pr);*
- *type ordersize : range 2 .. 5;*

State variables show possible states of the storage container as: *e0* for empty, *o1* for occupied, *r2* for requested, *p3* for replenishing, and *bp* for being picked. Target row is the row information of the storage container. “E” is used for the release of the requests. “pr” is for the worker processing. Order size defines the number of requests that will be released with the transition.

Below is an example of place definition. Domain of the place (*dom*) and the initial marking (*init*) are defined for the place. Initial marking shows the initial state or the tokens initially present in the place. In the example, the place contains a token, which is in the requested state and has row information of “1.”

```
place p10 {
    dom : state * targetrow;
    init : < (r2,1) >;
}
```

In the below figures, we have shown examples for each type of transition. Definitions on the top right of the place show the initial marking or initial state of the place. The number on the left shows the number of tokens and definition in the parenthesis represent the state of the token ($1 \cdot (e0, 0)$, for instance). Definitions on the incoming arcs from the particular place to the transition show the binding of the transition. Binding refers to the activation requirement of the transition. A specific token should be present in the incoming place ($((r2, t0)$, for instance). The outgoing token description from the transition to the places is shown on the outgoing arcs. To represent same place in several transitions, we have used *fusion* concept of CPN tools, which enables the definition of the same place in several transitions. In the package Helena, we enumerate all the transitions and use the same place

names in the enumeration. In the transition, a guard is shown if necessary on the top left side. In addition to bindings, this guard should also evaluate to true for the transition firing. On the bottom left side, a priority number is given to the transition depending on the order of the events.

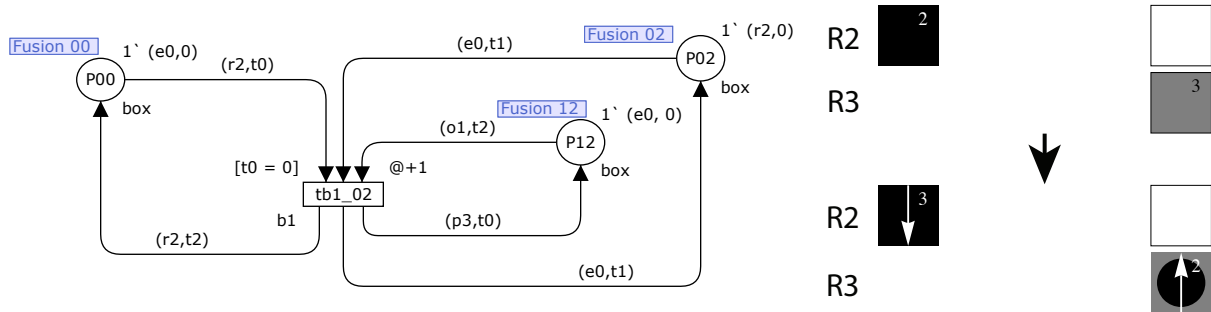


Figure 5.13: Example of Balance Rule - 1

Figure 5.13 shows an example of balance rule-1. In this case, a requested item exchanges its target row with an occupied item, which is in the below row. Graphics on the right shows the change on the system after the transition firing. Row numbers are for representation purposes and may not match Petri nets transitions.

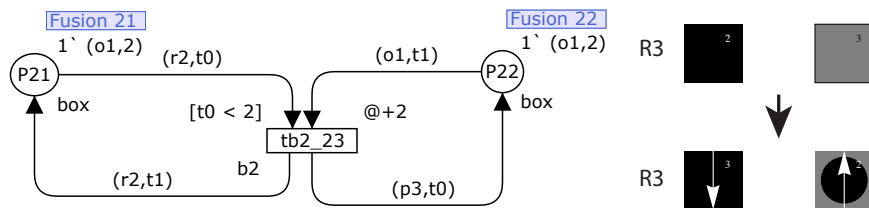


Figure 5.14: Example of Balance Rule - 2

For balance rule-2, we check if the requested item is already moved, and did not exchange its departure row before the movement (see Figure 5.14). If the guard evaluates to true, it exchanges its row information with an occupied item, and the occupied item turns into a replenishing item.

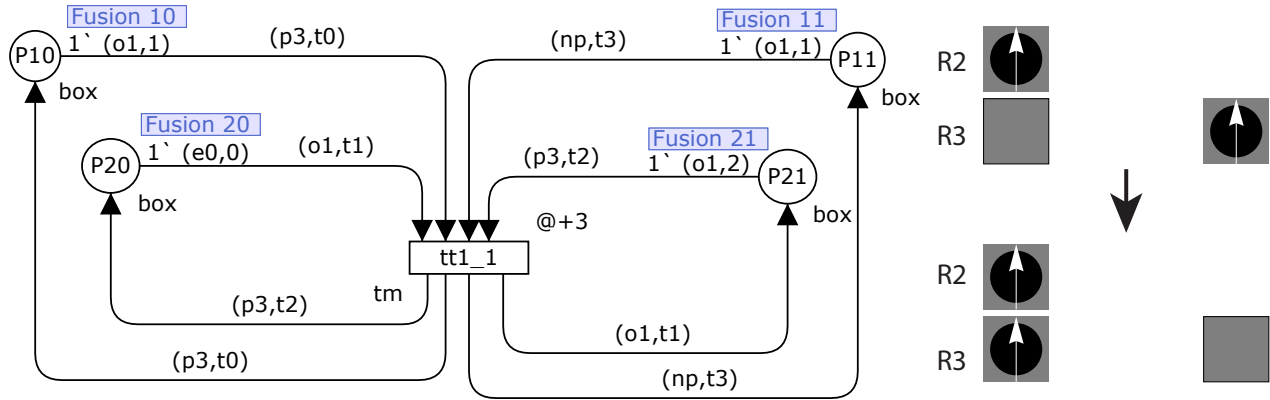


Figure 5.15: Example of the Convoy Movement Encouragement

To decrease the usage of empty cells by the replenishing items, we encourage convoys with the exchange of replenishing items. So, a replenishing item passes its balancing responsibility to an occupied item. In these transitions, we also check the northern neighbor of the replenishing item and make sure we do not interrupt a convoy (see Figure 5.15).

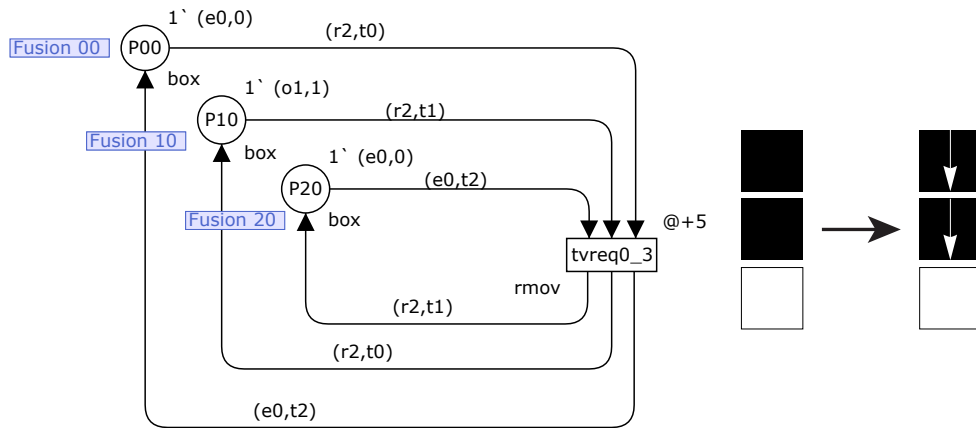


Figure 5.16: Example of the Vertical Movement for the Requested Items

Events enable convoy movements by checking the existence of requested items (see Figure 5.16). We also enable singular vertical movement by checking absence of consecutive requested items.

Events perform the vertical movement of replenishing items in a similar manner with the requested items (see figure 5.17). If places contain replenishing items (tokens with p3

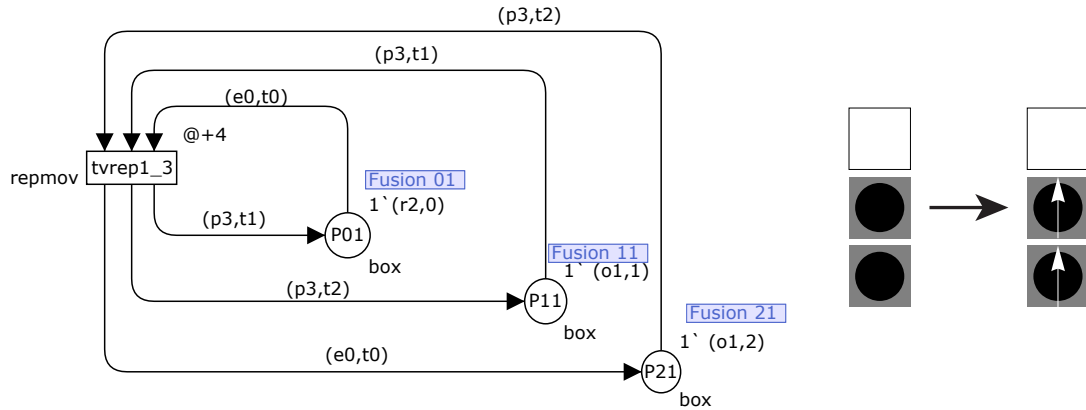


Figure 5.17: Example of the Vertical Movement for the Replenishing Items

state), and there exists a place with an empty (e_0) token, the transition is enabled and may fire.

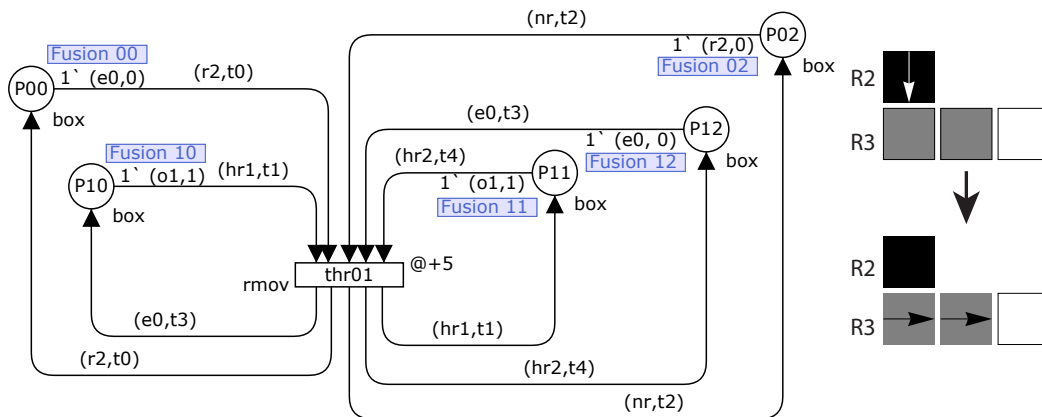


Figure 5.18: Example of the Horizontal Movement for the Requested Items

Horizontal movement aims to clear the way of the requested items. It checks the absence of a vertical movement availability to avoid assignment of additional priority among vertical and horizontal movements (see Figure 5.18). Since we have detected a deadlock case due to unavailability of replenishing item side movement in the pick face, we enabled side movement of the replenishing items by defining variables (hr_1 , hr_2 , etc.), which includes both the occupied and replenishing items.

Horizontal movement clears the way of replenishing items. Horizontally moving items have the variables (m_1 , m_2 , etc.) that include both requested and occupied items (see

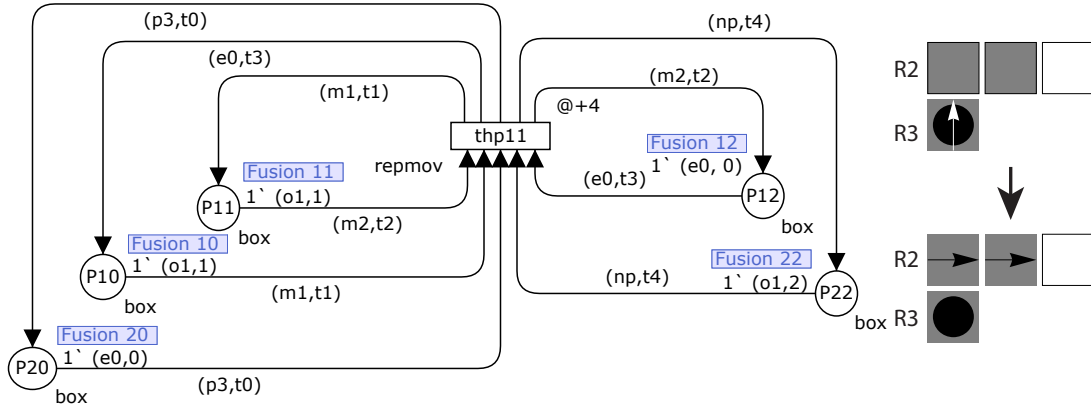


Figure 5.19: Example of the Horizontal Movement for the Replenishing Items

Figure 5.19). Horizontal movement of replenishing items is similar to the requested item movements, but it occurs prior to the requested items.

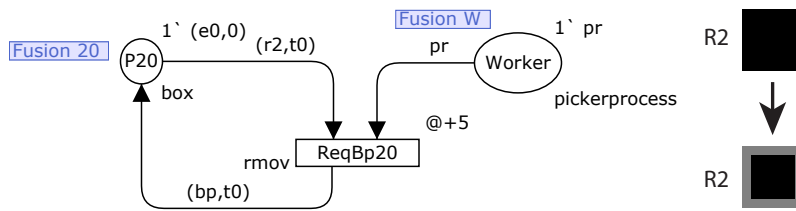


Figure 5.20: Transition Example from Requested to Being Picked Item

We have defined the worker with a place, and a token shows the processing of the worker place. So, the worker is a limited source, and we make sure worker cannot process multiple items at the same time in this way. Whenever the worker starts processing, the item goes to a state *being picked* (*bp*) from the state requested (see Figure 5.20). Therefore, the being picked item cannot move during the processing.

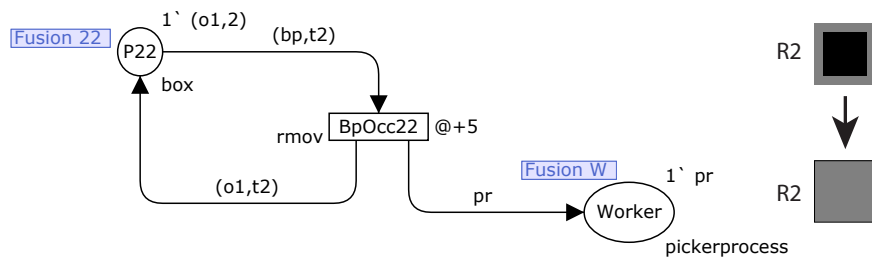


Figure 5.21: Transition Example from Being Picked to Occupied item

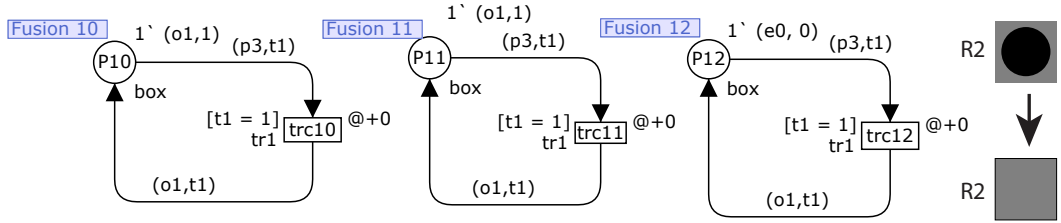


Figure 5.22: Example of the transition from Replenishing to Occupied Item

After the worker processes the order, the item goes from a state of *being picked* to the state occupied (see Figure 5.21). When the replenishing items reach their target row (it is checked with the guard), they turn into occupied item (see Figure 5.22).

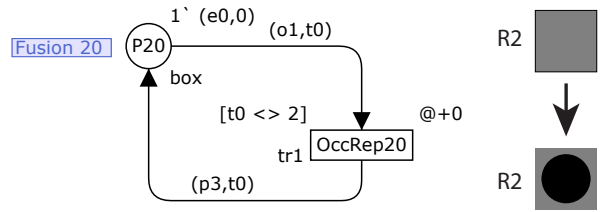


Figure 5.23: Example of the transition from Occupied to Replenishing Item

If the item turns into occupied suddenly due to an immediate picking process, it may not have the row information of current row, which is checked by the guard (see Figure 5.23). In this case, it turns into a replenishing item and travels to its target row.

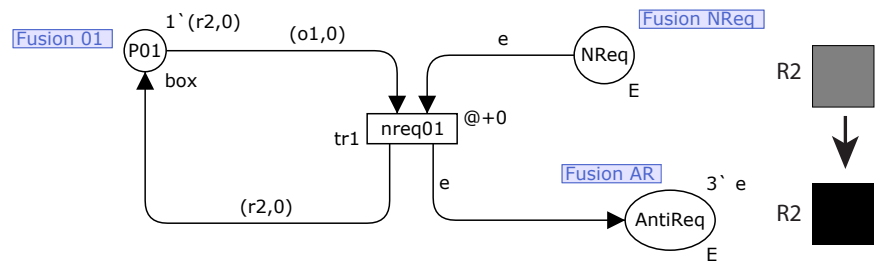


Figure 5.24: Example of the transition from Occupied to Requested Item

After the order is released, requested items are assigned by turning occupied items into requested items (see Figure 5.24). These transitions also decrease the number of tokens (e) in *NReq* place to control the number of requests for the order size.

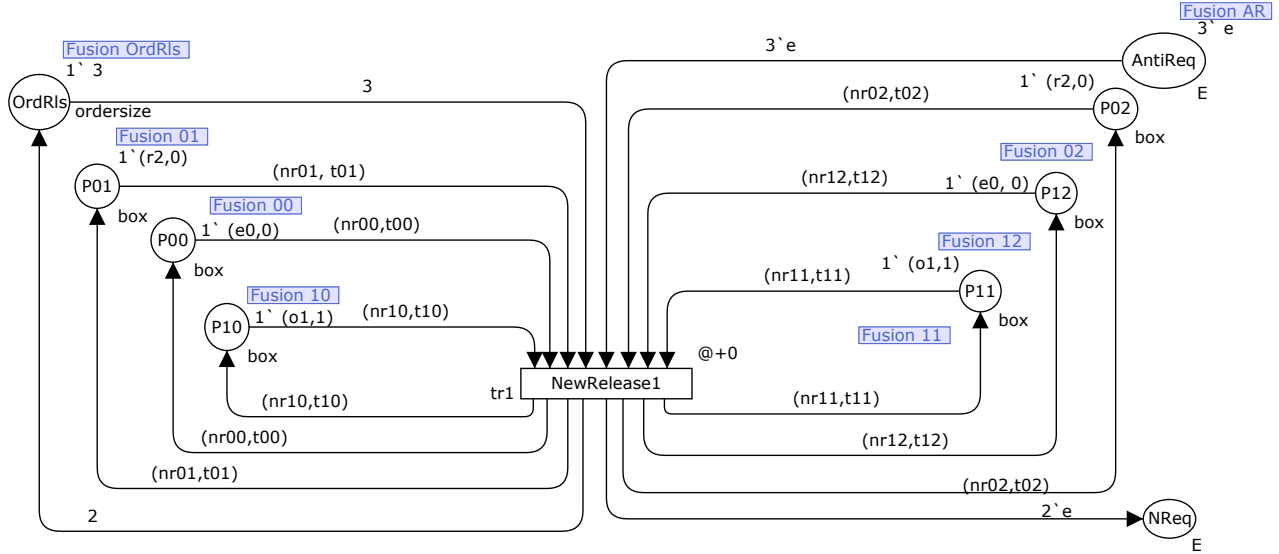


Figure 5.25: Example of the transition for the New Release of the Requested Items

The transition in Figure 5.25 mimics the order release policy. The transition checks all the positions other than the pick face for the absence of the requested items. If rows other than the pick face do not include requested items, it releases the order by assigning tokens to place $NReq$. Therefore, transitions for the new request will turn occupied items into requested items by using tokens in place $NReq$ (see Figure 5.24).

5.5.1 Contributions of the Petri Nets Model and Deadlock Conditions

We have checked the existence of deadlock cases in GridPick with the Petri Nets models. Package Helena enabled us to check this property with the following definitions.

state property not_dead :

reject deadlock;

We were able to analyze several configurations of GridPick with Petri Nets, and we have reached deadlock free models for small systems. To allow the movement of active items between rows, we require at least one empty cell ($k > 0$) in each row. Furthermore, the system is restricted to having at most $2c - 1$ requested items in the grid, and a single order cannot exceed the number of columns, c . Under these conditions we are able to prove that

Theorem 1 *GridPick systems of size 3×3 , 3×4 , 3×5 , 4×3 , 5×3 and 4×4 are deadlock free.*

Configuration	States	Deadlock States	Arcs	Compilation Time	State Space Search	Max. Memory
3×3	9,521	0	13,241	22.46 s.	0.02 s.	29.9 mb
3×4	215,840	0	340,560	43.61 s.	0.88 s.	33.1 mb
3×5	3,495,452	0	5,866,359	91.08 s.	24.05 s.	87.5 mb
4×3	83,346	0	133,083	37.53 s.	0.31 s.	31.1 mb
5×3	629,869	0	1,068,905	64.75 s.	3.61 s.	44.3 mb
4×4	4,335,866	0	7,627,308	90.24 s.	29.27 s.	98.0 mb

Table 5.3: Results of the Petri Nets Models

We run into memory storage overflow problems for larger grids such as 4×5 and 5×4 configurations (see Table 5.3).

Conjecture 1 *One sided GridPick is deadlock free for an arbitrarily large grid.*

Because the state space of a Petri Nets Model explodes, a computational proof is beyond reach. There are, however, several arguments to support the conjecture:

1. Small grids are deadlock free (Theorem 1).
2. Small, deadlock free grids embody the characteristics found in larger grids. For example, they contain boundary cells and middle cell, and are capable of convoy movement of requested and replenishing items.
3. Small grids are a sort of worst case in the sense that they have a very high percentage of requested items in the grid. For instance, in a 3×4 grid, having $2c - 1$ requested items corresponds to 7 requested items out of 9.

For a two sided GridPick model, we have the same policies used in one sided GridPick, and therefore we conjecture

Conjecture 2 *A two sided GridPick system is deadlock free for an arbitrarily large grid.*

Unfortunately, even the smallest two-sided system is beyond reasonable representation with a Petri Nets model.

Chapter 6

Conclusions and Contributions

Dynamic order picking systems are not new to industry, but they have not been explored in the material handling research literature. To the best of our knowledge, GridPick is the first research study to introduce a dynamic pick face order picking system. Dynamic order picking systems improve forward area operations by providing a higher pick density.

GridPick also provides a high density configuration without any aisle assignment, which provides space savings for distribution centers. Companies with special requirements for high density might benefit from GridPick. For instance, order picking operations under cold storage have high energy cost per unit area. Harsh business environments also require high throughput with the constraint of space. Order picking under special conditions such as large commercial green houses and food processing facilities might also be candidates for adoption.

GridPick accomplishes complex operations with a decentralized control algorithm that uses message passing and negotiation. Message passing schemes and negotiation protocols are well explored in data flow and digital networks, but they are not well studied in material handling systems design. We have built on the work of Mayer [2009] to show that very complex behavior is possible with very simple rules. As global logistics systems continue to increase in complexity capabilities models such as we have demonstrated in this dissertation will become even more important for effective control.

We have introduced Petri nets modeling for grid based storage systems to show the structural properties of the model (specifically, the deadlock-free property). Decentralized control models and material handling systems have been modeled with Petri nets in the literature; however, Petri nets modeling of material handling systems with decentralized

control has not been well addressed in the literature. Celaya et al. [2009] is one of the few exceptions. Petri nets modeling provides a formal methodology for the detection of deadlocks. Our Petri nets model can provide a general approach for deadlock detection for the future applications of grid based storage systems.

The two sided GridPick model enables order picking from two edges of the grid without additional space. With this major extension, we develop a higher level of functionality and flexibility for various demand levels. Expansion of automated material handling systems generally requires expensive equipment and significant investment cost. With the two sided extension, with the same system and physical devices, without additional space usage, the throughput increases significantly (80-95 %) with the cost of additional worker.

An even more flexible system that allows picking from all four sides is the ideal, but such a system would be much more complex even than the two sided system. We must leave that system for future research.

Bibliography

- T. Agerwala. Putting petri nets to work. *Computer*, 12(2):85–94, 1979.
- A. Alfieri, M. Cantamessa, A. Monchiero, and F. Montagna. Heuristics for puzzle-based storage systems driven by a limited set of automated guided vehicles. *Journal of Intelligent Manufacturing*, pages 1–11, 2010. published online.
- J. Ashayeri and L. F. Gelders. Warehouse design optimization. *European*, 21(3):285–294, 1985.
- P. Baker and M. Canessa. Warehouse design: a structured approach. *European J*, 193:425–436, 2009.
- J. J. Bartholdi and D. Eisenstein. Bucket brigades: A self-organizing order picking system for a warehouse. Technical report, School of Industrial and Systems Engineering, Georgia Tech, Atlanta, 1996.
- J. J. Bartholdi and S. T. Hackman. Allocating space in a forward pick area of a distribution center for small parts. *IIE Transactions*, 40:1046–1053, 2008.
- J. J. Bartholdi and S. T. Hackman. Warehouse & distribution science. Available online at <http://www.warehouse-science.com/> Release 0.95, 2011.
- F. Basile, P. Chiacchio, and J. Coppola. Colored hybrid petri nets for modeling material handling systems. In *50th IEEE Conference on Decision and Control and European Control Conference*, 2011.
- B. Berthomieu and M. Diaz. Modelling and verification of time dependent systems. *IEEE Trans. Software Eng.*, 17(3):259–273, 1991.

- J. Billington, G. R. Wheeler, and M. C. Wilbur-Ham. Protean: A high level petri net tool for the specification and verification of communication protocols. *IEEE Trans. Software Eng.*, 14(3):301–331, 1988.
- H. Brynzer, M. Johansson, and L. Medbo. A methodology for evaluation of order picking systems as a base for system design and managerial decisions. *International Journal of Operations and Production Management*, 14(3):126–139, 1994.
- D. Buchs, S. Hostettler, A. Marechal, and M. Risoldi. Alpina: A symbolic model checker. In *International Conference on Application and Theory of Petri Nets*, 2010.
- A. C. Caputo and P. M. Pelagagge. Management criteria of automated order picking systems in high rotation high volume distribution centers. *Industrial Management & Data Systems*, 106(9):1359–1383, 2006.
- C. G. Cassandras and S. Lafortune. *Introduction to discrete event systems*. Springer, 2008.
- J. R. Celaya, A. A. Desrochers, and R. J. Graves. Modeling and analysis of multi agent systems using petri nets. *Journal of Computers*, 4(10):981–996, 2009.
- F. Dallari, G. Marchet, and M. Melacini. Design of order picking system. *International*, 42(1):1–12, 2009.
- P. Davidsson, L. Henesey, L. Ramstedt, J. Törnquist, and F. Wernstedt. An analysis of agent-based approaches to transport logistics. *Transportation Research Part C*, 13:255–271, 2005.
- R. De Koster. How to assess a warehouse operation in a single tour. Technical report, RSM Erasmus University, the Netherlands, 2004.
- R. De Koster, T. Le-Duc, and K. J. Roodbergen. Design and control of warehouse order picking: A literature review. *European Journal of Operational Research*, 182:481–501, 2007.

- R. B. M. De Koster, T. Le-duc, and Y. Yu. Optimal storage rack design for a 3-dimensional compact as/rs. *International Journal of Production Research*, 46(6):1495–1514, 2008b.
- R. Dekker, R. B. M. De Koster, K. J. Roodbergen, and H. Van Kalleveen. Improving order picking response time at ankor’s warehouse. *Interfaces*, 34(4):303–313, 2004.
- Jack J Dongarra and Tom Dunigan. Message-passing performance of various computers. *Concurrency - Practice and Experience*, 9(10):915–926, 1997.
- M. Dotoli and M. P. Fanti. Modeling of an as/rs serviced by rail guided vehicles with colored petri nets: A control perspective. In *IEEE SMC*, 2002.
- S. Evangelista. *High Level Petri Nets Analysis with Helena*, volume 3536. LNCS Springer Berlin, 2005.
- G. W. Flake and E. B. Baum. Rush hour is pspace-complete, or “why you should generously tip parking lot attendants”. *Theoretical Computer Science*, 270:1–2, 2002.
- E. A. Frazelle and G. P. Sharp. Correlated assignment strategy can improve order picking operation. *Industrial Engineering*, 21(4):33–37, 1989.
- E. H. Frazelle, S. T. Hackman, U. Passy, and L. K. Platzman. *The Forward-reserve problem, in Optimization in Industry, vol. II*. Wiley, New York, NY, 1994.
- K. Furmans, F. Schönung, and K. R. Gue. Plug and work material handling systems. In *Progress in Material Handling Research: 2010*, pages 132–142, 2010.
- K. Furmans, K. R. Gue, and Z. Seibold. Optimization of failure behavior of a decentralized high density 2d storage system. In *Proceedings of The 3rd International Conference on Dynamics in Logistics*, 2012.
- M. Gardner. Mathematical games: The fantastic combinations of john conway’s new solitaire game “life”. *Scientific American*, 223:120–123, 1970.

- A. E. Gray, U. S. Karmarkar, and A. Seidman. Design and operation of an order consolidation warehouse - models and application. *European Journal of Operational Research*, 58(1):14–36, 1992.
- J. Gu, M. Goetschalckx, and L. F. McGinnis. Solving the forward-reserve allocation problem in warehouse order picking systems. *Journal of the Operational Research Society*, 61:1013–1021, 2010.
- K. R. Gue. Very high density storage systems. *IIE Transactions*, 38:93–104, 2006.
- K. R. Gue and K. Furmans. Decentralized control in a grid-based storage system. In T. Doolen and E. Van Aken, editors, *Proceedings of the 2011 Industrial Engineering Research Conference*, 2011.
- K. R. Gue and K. Kang. Staging queues in material handling and transportation systems. In *Proceedings of 2001 Winter Simulation Conference*, 2001.
- K. R. Gue and B. S. Kim. Puzzle-based storage systems. *Naval Research Logistics*, 54(5):556–567, 2007.
- K. R. Gue, K. Furmans, Z. Seibold, and O. Uludag. Gridstore: A puzzle-based storage system with decentralized control. *forthcoming in IEEE Transactions on Automation Science and Engineering*, 2013.
- S. T. Hackman and M. J. Rosenblatt. Allocating items to an automated storage and retrieval system. *IIE Transactions: Industrial Engineering Research and Development*, 22(1):7–14, 1990.
- R.W. Hall. Distance approximations for routing manual pickers in a warehouse. *IIE Transactions*, 25(4):76–87, 1993.

- S. J. He, F. Cheng, and J. Luo. Modeling and implementing of an automated warehouse via coloured timed petri nets. In *IEEE International Conference on Control and Automation*, 2007.
- R. A. Hearn. *Games, Puzzles, and Computation*. PhD thesis, Massachusetts Institute of Technology, 2006.
- S. S. Heragu, L. Du, R. J. Mantel, and P. C. Schuur. Mathematical model for warehouse design and product allocation. *International Journal of Production Research*, 43(2):327–338, 2005.
- S. S. Heragu, X. Cai, A. Krishnamurthy, and C. J. Malmberg. Analysis of autonomous vehicle storage and retrieval system by open queueing network. In *5th Annual IEEE Conference on Automation Science and Engineering*, 2009.
- J. Hopcroft, J. T. Schwartz, and M. Sharir. On the complexity of motion planning for multiple independent objects; pspace-hardness of the “warehouseman’s problem”. *International Journal of Robotics Research*, 3(4):76–88, 1984.
- S. Hsieh, J. s. Hwang, and H. C. Chou. A petri net based structure for as/rs operation modelling. *Int. J. Prod. Res.*, 36(12):3323–3346, 1998.
- Y. H. Hu, S. Y. Huang, C. Y. Chen, W. J. Hsu, A. C. Toh, C. K. Loh, and T. C. Song. Travel time analysis of a new automated storage and retrieval system. *Computers & Operations Research*, 32:1515–1544, 2005.
- M.V. Iordache, J.O. Moody, and P.J. Antsaklis. A method for deadlock prevention in discrete event systems. Technical report, University of Notre Dame, 1999.
- K. Jensen. Colored petri nets and the invariant method. *Theoretical Computer Science*, 14: 317–336, 1981.

- K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Volume 1: Basic Concepts*. Springer-Verlag, 1992.
- K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Volume 2: Analysis Methods*. Springer-Verlag, 1994.
- K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Volume 3: Practical Use*. Springer-Verlag, 1997.
- K. Jensen, L. M. Kristensen, and L. Wells. Coloured petri nets and cpn tools for modelling and validation of concurrent systems. *Int. J. Softw. Tools Technology Transfer*, 9:213–254, 2007.
- S. A. Jernigan. *Multi-tier inventory systems with space constraints*. PhD thesis, Georgia Institute of Technology, Atlanta, GA, 2004.
- M. Kamath and N. Vishwanatham. Applications of petri net based models in the modeling and analysis of fmss. In *Proceedings of the IEEE Conference on Robotics and Automation*, 1987.
- M. Kostrzewski. The procedure of warehouses designing as an integral part of the warehouses designing method and the designing software. *International Journal of Mathematical Models and Methods in Applied Sciences*, 6:535–543, 2012.
- T. Krühn, S. Falkenberg, and L. Overmeyer. Decentralized control for small scaled conveyor modules with cellular automata. In *Proceedings of the 2010 IEEE International Conference on Automation and Logistics*, 2010.
- P. H. Kuo, A. Krishnamurthy, and C. J. Malmberg. Performance modelling of autonomous vehicle storage and retrieval systems using class-based storage policies. *Int. J. Computer Applications in Technology*, 31(3/4):238–248, 2008.

- M. K. Lee. A storage assignment policy in a man on board automated storage retrieval system. *International Journal of Production Research*, 30(10):2281–2292, 1992.
- C. J. Malmberg. Conceptualizing tools for autonomous vehicle storage and retrieval systems. *International Journal of Production Research*, 40(8):1807–1822, 2002.
- S. H. Mayer. *Development of a completely decentralized control system for modular continuous conveyors*. PhD thesis, Universität Karlsruhe, 2009.
- B. Montreuil. Towards a physical internet: meeting the global logistics sustainability grand challenge. Technical report, CIRRELT, 2011.
- E. F. Moore. Machine models of self-reproduction. In *Proceedings of Symposia in Applied Mathematics*, 1962.
- T. Murata. Petri nets: Properties, analysis and applications. In *Proceedings of the IEEE*, volume 77(4), pages 541–580, April 1989.
- NAVSUP. *Warehouse Modernization and Layout Planning Guide*. Department of Navy, Naval Supply Systems Command, NAVSUP Publication 529, March 1985.
- L. Overmeyer, K. Vents, S. Falkenberg, and T. Krühn. Interfaced multidirectional small-scaled modules for intralogist. *Logistics Research*, 2:123–133, 2010.
- C. G. Petersen and G. Aase. A comparison of picking, storage, and routing policies in manual order picking. *International Journal of Production Economics*, 92:11–19, 2004.
- C. G. Petersen and R. W. Schmenner. An evaluation of routing and volume based storage policies in an order picking operation. *Decision Sciences*, 31(3):507–521, 1999.
- S. Pujari and S. Mukhopadhyay. Petri net: A tool for modeling and analyze multi-agent oriented systems. *I. J. Intelligent Systems and Applications*, 10:103–112, 2012.

- H. D. Ratliff and A. S. Rosenthal. Order picking in a rectangular warehouse: A solvable case of the traveling salesman problem. *Operations Research*, 31(3):507–521, 1983.
- K. J. Roodbergen and R. De Koster. Routing methods for warehouses with multiple cross aisles. *International Journal of Production Research*, 39(9):1865–1883, 2001.
- K. J. Roodbergen and I. F. A. Vis. A survey of literature on automated storage and retrieval systems. *European Journal of Operational Research*, 194:343–362, 2009.
- B. Rouwenhorst, B. Reuter, V. Stockrahm, G. J. van Houtum, R. J. Mantel, and W. H. M. Zijm. Warehouse design and control: Framework and literature review. *European Journal of Operational Research*, 122(3):515–533, 2000.
- Z. Sari, C. Saygin, and N. Ghouali. Travel-time models for flow-rack automated storage and retrieval systems. *International Journal of Advanced Manufacturing Technology*, 25:979–987, 2005.
- M. Sarrafzadeh and S. R. Maddila. Discrete warehouse problem. *Theoretical Computer Science*, 140:231–247, 1995.
- K. Schmidt. Lola - a low level analyser. In *International Conference on Application and Theory of Petri Nets*, 2000.
- R. Sharma and Y. Aloimonos. Coordinated motion planning: the warehouseman’s problem with constraints on free space. *IEEE Transactions on Systems, Man and Cybernetics*, 22(1):130–141, 1992.
- G. P. Sharp. Order picking: principles, practices and advanced analysis, perspectives on material handling practice. Technical report, MHIA, 1992.
- A. R. Smith. Cellular automata complexity trade-offs. *Information and Control*, 18(5):466–482, 1971.

- P. Sornkhom and Y. Permpoontanalarp. Security analysis of micali's fair contract signing protocol by using coloured petri nets. In *9th ACIS International Conference on Software Engineering, Artificial Intelligence Networking and Parallel/Distributed Computing*, 2008.
- G. D. Taylor and K. R. Gue. Retrieval time performance in puzzle-based storage systems. In A. Johnson and J. Miller, editors, *Proceedings of the 2010 Industrial Engineering Research Conference*, 2010.
- M. Ten Hompel, S. Libert, and U. Sondhof. Distributed control nodes for material flow system controls on the example of unit load conveyor and sorter facilities. *Logistics Journal*, 2:1–8, 2006.
- J. A. Tompkins, J. A. White, Y. A. Bozer, and J. M. A. Tanchoco. *Facilities Planning*. John Wiley & Sons, Inc., 2003.
- J. P. Van den Berg. A literature survey on planning and control of warehousing systems. *IIE Transactions*, 31:751–762, 1999a.
- J. P. Van den Berg, G. P. Sharp, A. J. R. M. Gademann, and Y. Pochet. Forward-reserve allocation in a warehouse with unit load replenishment. *European Journal of Operational Research*, 111(1):98–113, 1998.
- J. von Nuemann. *The Theory of Self-Producing Automata*. University of Illinois Press, Urbana, Illinois, 1966.
- J. Wang. *Petri nets for dynamic event driven system modeling*. CRC Press, 2007.
- J. Wang, S. Zhang, and F. Chen. Modeling and verification of sctp association management based on colored petri nets. In *ISECS International Colloquium on Computing, Communication, Control, and Management*, 2008.

- A. Wegrzyn, A. Karatkevich, and J. Bieganowski. Detection of deadlocks and traps in petri nets by means of the lenz's prime implicant method. *Int. J. Appl. Math. Comput. Sci.*, 14(1):113–121, 2004.
- G. Weiss, editor. *Multiagent systems: a modern approach to distributed artificial intelligence*. Cambridge, MA, USA: MIT Press, 1999.
- G. Wilfong. Motion planning in the presence of movable obstacles. In *Proceedings of 4th Symposium on Computational Geometry*, 1988.
- M. Wooldridge. *An Introduction to MultiAgent Systems*. John Wiley & Sons, Inc., 2002.
- H. Xu, M. Ayachit, and A. Reddyreddy. Formal modeling and analysis of xml firewall for service oriented systems. *Int. J. Security and Networks*, 3(3):1–13, 2008.
- C. S. Yoon and G. P. Sharp. A structured procedure for analysis and design of order pick systems. *IIE Transactions*, 28(5):379–389, 1996.
- D. V. Zizzi. What's new in the equipment field. In *International Material Handling Research Colloquium, Material Handling Institute*, 2000.
- R. Zurawski and M. Zhou. Petri nets and industrial applications: A tutorial. *IEEE Transactions on Industrial Electronics*, 41(6):567–583, 1994.

Appendix A

Appendix

A.1 Statistical Data and Confidence Intervals for the Analysis

Below tables provide the statistical information and confidence intervals for the performance measures. First number for the variable shows the k value or aspect ratio or number of copies. Second number always shows the expected order size.

A.2 Paired t Tests

Below paired t tests include comparisons for throughput, flow time, walking time, and waiting time. First number in the variable name shows the k value. The second number indicates the expected order size.

thr	Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	% Mean
	k3-4	30	0.30335	0.00621	0.00113	(0.30022, 0.30648)	0.00313	1.03%
thr	Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	% Mean
	k3-14	30	0.473242	0.002792	0.00051	(0.471837, 0.474647)	0.001405	0.30 %
thr	Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	% Mean
	k3-24	30	0.456525	0.00408	0.000745	(0.454472, 0.458578)	0.002053	0.45 %
walking	Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	% Mean
	k3-4	30	2.0146	0.0654	0.0119	(1.9817, 2.0475)	0.0329	1.63%
walking	Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	% Mean
	k3-14	30	0.6707	0.01153	0.0021	(0.66490, 0.67651)	0.00581	0.87%
walking	Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	% Mean
	k3-24	30	0.49893	0.00941	0.00172	(0.49419, 0.50367)	0.00474	0.95 %
waiting	Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	% Mean
	k3-4	30	0.53424	0.01016	0.00186	(0.52913, 0.53935)	0.00511	0.96 %
waiting	Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	% Mean
	k3-14	30	0.69261	0.0116	0.00212	(0.68677, 0.69844)	0.00583	0.84%
waiting	Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	% Mean
	k3-24	30	0.94191	0.02659	0.00485	(0.92853, 0.95529)	0.01338	1.42%
flowtime	Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	% Mean
	flowk3-4	30	9.3867	0.1058	0.0193	(9.3335, 9.4400)	0.0533	0.57%
flowtime	Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	% Mean
	flowk3-14	30	15.0662	0.2703	0.0494	(14.9302, 15.2023)	0.1361	0.90 %
flowtime	Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	% Mean
	flwk3-24	30	24.867	0.59	0.108	(24.569, 25.164)	0.297	1.19%

Table A.1: Statistical Data for the Throughput of One Sided GridPick

Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	%Mean
0.5-4	30	0.188192	0.005448	0.000995	(0.185450, 0.190933)	0.002741	1.46%
Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	%Mean
1-4	30	0.27913	0.00659	0.0012	(0.27582, 0.28245)	0.00332	1.19%
Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	%Mean
1-14	30	0.4416	0.003601	0.000657	(0.439788, 0.443412)	0.001812	0.41%
Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	%Mean
8-14	30	0.415725	0.003765	0.000687	(0.413830, 0.417620)	0.001895	0.46%
Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	%Mean
8-24	30	0.479442	0.002169	0.000396	(0.478350, 0.480533)	0.001091	0.23%

Table A.2: Statistical Data for the Aspect Ratio Configurations of One Sided GridPick

Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	%Mean
inck-4	30	0.297092	0.004702	0.000858	(0.294725, 0.299458)	0.002366	0.80%
Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	%Mean
inck-14	30	0.476475	0.00202	0.000369	(0.475459, 0.477491)	0.001016	0.21%
Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	%Mean
inck-24	30	0.468217	0.003359	0.000613	(0.466526, 0.469907)	0.00169	0.36%
Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	%Mean
deck-4	30	0.295558	0.004528	0.000827	(0.293280, 0.297837)	0.00228	0.77%
Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	%Mean
deck-14	30	0.463725	0.003401	0.000621	(0.462014, 0.465436)	0.0017	0.37%
Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	%Mean
deck-24	30	0.45188	0.0062	0.00113	(0.44876, 0.45500)	0.00312	0.69%

Table A.3: Statistical Data for the Variable k Configurations of One Sided GridPick

Paired T-Tests for Throughput Analysis of One Sided GridPick

Paired T-Test and CI: k2_2, k3_2

Paired T for k2_2 - k3_2

	N	Mean	StDev	SE Mean
k2_2	30	0.215733	0.003769	0.000688
k3_2	30	0.210083	0.004640	0.000847
Difference	30	0.00565	0.00692	0.00126

95% CI for mean difference: (0.00307, 0.00823)

T-Test of mean difference = 0 (vs not = 0): T-Value = 4.47 P-Value = 0.000

Paired T-Test and CI: k2_2, k4_2

Paired T for k2_2 - k4_2

	N	Mean	StDev	SE Mean
k2_2	30	0.215733	0.003769	0.000688
k4_2	30	0.203900	0.003603	0.000658
Difference	30	0.011833	0.005352	0.000977

95% CI for mean difference: (0.009835, 0.013832)

T-Test of mean difference = 0 (vs not = 0): T-Value = 12.11 P-Value = 0.000

Paired T-Test and CI: k3_2, k4_2

Paired T for k3_2 - k4_2

	N	Mean	StDev	SE Mean
k3_2	30	0.210083	0.004640	0.000847
k4_2	30	0.203900	0.003603	0.000658
Difference	30	0.00618	0.00576	0.00105

95% CI for mean difference: (0.00403, 0.00833)

T-Test of mean difference = 0 (vs not = 0): T-Value = 5.88 P-Value = 0.000

Paired T-Test and CI: k2_4, k3_4

Paired T for k2_4 - k3_4

	N	Mean	StDev	SE Mean
k2_4	30	0.30864	0.00587	0.00107
k3_4	30	0.30335	0.00621	0.00113
Difference	30	0.00529	0.00739	0.00135

95% CI for mean difference: (0.00253, 0.00805)

T-Test of mean difference = 0 (vs not = 0): T-Value = 3.92 P-Value = 0.000

Paired T-Test and CI: k2_4, k4_4

Paired T for k2_4 - k4_4

	N	Mean	StDev	SE Mean
k2_4	30	0.30864	0.00587	0.00107
k4_4	30	0.29876	0.00473	0.00086
Difference	30	0.00988	0.00721	0.00132

95% CI for mean difference: (0.00719, 0.01257)

T-Test of mean difference = 0 (vs not = 0): T-Value = 7.51 P-Value = 0.000

Paired T-Test and CI: k3_4, k4_4

Paired T for k3_4 - k4_4

	N	Mean	StDev	SE Mean
k3_4	30	0.30335	0.00621	0.00113
k4_4	30	0.29876	0.00473	0.00086
Difference	30	0.00459	0.00808	0.00148

95% CI for mean difference: (0.00157, 0.00761)

T-Test of mean difference = 0 (vs not = 0): T-Value = 3.11 P-Value = 0.004

Paired T-Test and CI: k2_6, k3_6

Paired T for k2_6 - k3_6

	N	Mean	StDev	SE Mean
k2_6	30	0.370592	0.005237	0.000956
k3_6	30	0.371508	0.003729	0.000681
Difference	30	-0.00092	0.00667	0.00122

95% CI for mean difference: (-0.00341, 0.00158)

T-Test of mean difference = 0 (vs not = 0): T-Value = -0.75 P-Value = 0.458

Paired T-Test and CI: k2_6, k4_6

Paired T for k2_6 - k4_6

	N	Mean	StDev	SE Mean
k2_6	30	0.370592	0.005237	0.000956
k4_6	30	0.365333	0.004632	0.000846
Difference	30	0.00526	0.00657	0.00120

95% CI for mean difference: (0.00281, 0.00771)

T-Test of mean difference = 0 (vs not = 0): T-Value = 4.39 P-Value = 0.000

Paired T-Test and CI: k3_6, k4_6

Paired T for k3_6 - k4_6

	N	Mean	StDev	SE Mean
k3_6	30	0.371508	0.003729	0.000681
k4_6	30	0.365333	0.004632	0.000846
Difference	30	0.006175	0.004673	0.000853

95% CI for mean difference: (0.004430, 0.007920)
T-Test of mean difference = 0 (vs not = 0): T-Value = 7.24 P-Value = 0.000

Paired T-Test and CI: k2_8, k3_8

Paired T for k2_8 - k3_8

	N	Mean	StDev	SE Mean
k2_8	30	0.403267	0.004584	0.000837
k3_8	30	0.416408	0.004239	0.000774
Difference	30	-0.013142	0.004986	0.000910

95% CI for mean difference: (-0.015003, -0.011280)
T-Test of mean difference = 0 (vs not = 0): T-Value = -14.44 P-Value = 0.000

Paired T-Test and CI: k2_8, k4_8

Paired T for k2_8 - k4_8

	N	Mean	StDev	SE Mean
k2_8	30	0.403267	0.004584	0.000837
k4_8	30	0.410700	0.003518	0.000642
Difference	30	-0.007433	0.005098	0.000931

95% CI for mean difference: (-0.009337, -0.005530)
T-Test of mean difference = 0 (vs not = 0): T-Value = -7.99 P-Value = 0.000

Paired T-Test and CI: k3_8, k4_8

Paired T for k3_8 - k4_8

	N	Mean	StDev	SE Mean
k3_8	30	0.416408	0.004239	0.000774
k4_8	30	0.410700	0.003518	0.000642
Difference	30	0.00571	0.00560	0.00102

95% CI for mean difference: (0.00362, 0.00780)
T-Test of mean difference = 0 (vs not = 0): T-Value = 5.58 P-Value = 0.000

Paired T-Test and CI: k2_10, k3_10

Paired T for k2_10 - k3_10

	N	Mean	StDev	SE Mean
k2_10	30	0.423633	0.004592	0.000838
k3_10	30	0.445342	0.004015	0.000733
Difference	30	-0.02171	0.00660	0.00120

95% CI for mean difference: (-0.02417, -0.01924)
T-Test of mean difference = 0 (vs not = 0): T-Value = -18.02 P-Value = 0.000

Paired T-Test and CI: k2_10, k4_10

Paired T for k2_10 - k4_10

	N	Mean	StDev	SE Mean
k2_10	30	0.423633	0.004592	0.000838
k4_10	30	0.443875	0.003180	0.000581
Difference	30	-0.02024	0.00626	0.00114

95% CI for mean difference: (-0.02258, -0.01790)

T-Test of mean difference = 0 (vs not = 0): T-Value = -17.70 P-Value = 0.000

Paired T-Test and CI: k3_10, k4_10

Paired T for k3_10 - k4_10

	N	Mean	StDev	SE Mean
k3_10	30	0.445342	0.004015	0.000733
k4_10	30	0.443875	0.003180	0.000581
Difference	30	0.001467	0.005434	0.000992

95% CI for mean difference: (-0.000562, 0.003496)

T-Test of mean difference = 0 (vs not = 0): T-Value = 1.48 P-Value = 0.150

Paired T-Test and CI: k2_12, k3_12

Paired T for k2_12 - k3_12

	N	Mean	StDev	SE Mean
k2_12	30	0.432500	0.005359	0.000978
k3_12	30	0.463808	0.002021	0.000369
Difference	30	-0.03131	0.00557	0.00102

95% CI for mean difference: (-0.03339, -0.02923)

T-Test of mean difference = 0 (vs not = 0): T-Value = -30.80 P-Value = 0.000

Paired T-Test and CI: k2_12, k4_12

Paired T for k2_12 - k4_12

	N	Mean	StDev	SE Mean
k2_12	30	0.432500	0.005359	0.000978
k4_12	30	0.462783	0.002381	0.000435
Difference	30	-0.03028	0.00593	0.00108

95% CI for mean difference: (-0.03250, -0.02807)

T-Test of mean difference = 0 (vs not = 0): T-Value = -27.97 P-Value = 0.000

Paired T-Test and CI: k3_12, k4_12

Paired T for k3_12 - k4_12

	N	Mean	StDev	SE Mean
k3_12	30	0.463808	0.002021	0.000369
k4_12	30	0.462783	0.002381	0.000435
Difference	30	0.001025	0.003501	0.000639

95% CI for mean difference: (-0.000282, 0.002332)
T-Test of mean difference = 0 (vs not = 0): T-Value = 1.60 P-Value = 0.120

Paired T-Test and CI: k2_14, k3_14

Paired T for k2_14 - k3_14

	N	Mean	StDev	SE Mean
k2_14	30	0.439400	0.004433	0.000809
k3_14	30	0.473242	0.002792	0.000510
Difference	30	-0.033842	0.004707	0.000859

95% CI for mean difference: (-0.035599, -0.032084)
T-Test of mean difference = 0 (vs not = 0): T-Value = -39.38 P-Value = 0.000

Paired T-Test and CI: k2_14, k4_14

Paired T for k2_14 - k4_14

	N	Mean	StDev	SE Mean
k2_14	30	0.439400	0.004433	0.000809
k4_14	30	0.474150	0.001779	0.000325
Difference	30	-0.034750	0.004543	0.000829

95% CI for mean difference: (-0.036446, -0.033054)
T-Test of mean difference = 0 (vs not = 0): T-Value = -41.90 P-Value = 0.000

Paired T-Test and CI: k3_14, k4_14

Paired T for k3_14 - k4_14

	N	Mean	StDev	SE Mean
k3_14	30	0.473242	0.002792	0.000510
k4_14	30	0.474150	0.001779	0.000325
Difference	30	-0.000908	0.002803	0.000512

95% CI for mean difference: (-0.001955, 0.000138)
T-Test of mean difference = 0 (vs not = 0): T-Value = -1.78 P-Value = 0.086

Paired T-Test and CI: k2_16, k3_16

Paired T for k2_16 - k3_16

	N	Mean	StDev	SE Mean
k2_16	30	0.437758	0.004541	0.000829
k3_16	30	0.475342	0.002939	0.000537
Difference	30	-0.03758	0.00580	0.00106

95% CI for mean difference: (-0.03975, -0.03542)
T-Test of mean difference = 0 (vs not = 0): T-Value = -35.50 P-Value = 0.000

Paired T-Test and CI: k2_16, k4_16

Paired T for k2_16 - k4_16

	N	Mean	StDev	SE Mean
k2_16	30	0.437758	0.004541	0.000829
k4_16	30	0.479525	0.002163	0.000395
Difference	30	-0.04177	0.00564	0.00103

95% CI for mean difference: (-0.04387, -0.03966)
T-Test of mean difference = 0 (vs not = 0): T-Value = -40.56 P-Value = 0.000

Paired T-Test and CI: k3_16, k4_16

Paired T for k3_16 - k4_16

	N	Mean	StDev	SE Mean
k3_16	30	0.475342	0.002939	0.000537
k4_16	30	0.479525	0.002163	0.000395
Difference	30	-0.004183	0.002989	0.000546

95% CI for mean difference: (-0.005300, -0.003067)
T-Test of mean difference = 0 (vs not = 0): T-Value = -7.67 P-Value = 0.000

Paired T-Test and CI: k2_18, k3_18

Paired T for k2_18 - k3_18

	N	Mean	StDev	SE Mean
k2_18	30	0.436833	0.005329	0.000973
k3_18	30	0.473258	0.003853	0.000703
Difference	30	-0.03642	0.00720	0.00132

95% CI for mean difference: (-0.03912, -0.03373)
T-Test of mean difference = 0 (vs not = 0): T-Value = -27.69 P-Value = 0.000

Paired T-Test and CI: k2_18, k4_18

Paired T for k2_18 - k4_18

	N	Mean	StDev	SE Mean
k2_18	30	0.436833	0.005329	0.000973
k4_18	30	0.479517	0.002124	0.000388
Difference	30	-0.04268	0.00590	0.00108

95% CI for mean difference: (-0.04489, -0.04048)
T-Test of mean difference = 0 (vs not = 0): T-Value = -39.63 P-Value = 0.000

Paired T-Test and CI: k3_18, k4_18

Paired T for k3_18 - k4_18

	N	Mean	StDev	SE Mean
k3_18	30	0.473258	0.003853	0.000703
k4_18	30	0.479517	0.002124	0.000388
Difference	30	-0.006258	0.004234	0.000773

95% CI for mean difference: (-0.007839, -0.004677)

T-Test of mean difference = 0 (vs not = 0): T-Value = -8.10 P-Value = 0.000

Paired T-Test and CI: k2_20, k3_20

Paired T for k2_20 - k3_20

	N	Mean	StDev	SE Mean
k2_20	30	0.434408	0.005195	0.000948
k3_20	30	0.469817	0.003821	0.000698
Difference	30	-0.03541	0.00624	0.00114

95% CI for mean difference: (-0.03774, -0.03308)

T-Test of mean difference = 0 (vs not = 0): T-Value = -31.10 P-Value = 0.000

Paired T-Test and CI: k2_20, k4_20

Paired T for k2_20 - k4_20

	N	Mean	StDev	SE Mean
k2_20	30	0.434408	0.005195	0.000948
k4_20	30	0.474792	0.003534	0.000645
Difference	30	-0.04038	0.00731	0.00134

95% CI for mean difference: (-0.04311, -0.03765)

T-Test of mean difference = 0 (vs not = 0): T-Value = -30.25 P-Value = 0.000

Paired T-Test and CI: k3_20, k4_20

Paired T for k3_20 - k4_20

	N	Mean	StDev	SE Mean
k3_20	30	0.469817	0.003821	0.000698
k4_20	30	0.474792	0.003534	0.000645
Difference	30	-0.004975	0.004682	0.000855

95% CI for mean difference: (-0.006723, -0.003227)

T-Test of mean difference = 0 (vs not = 0): T-Value = -5.82 P-Value = 0.000

Paired T-Test and CI: k2_22, k3_22

Paired T for k2_22 - k3_22

	N	Mean	StDev	SE Mean
k2_22	30	0.42982	0.00604	0.00110
k3_22	30	0.46333	0.00476	0.00087
Difference	30	-0.03352	0.00846	0.00154

95% CI for mean difference: (-0.03668, -0.03036)
T-Test of mean difference = 0 (vs not = 0): T-Value = -21.70 P-Value = 0.000

Paired T-Test and CI: k2_22, k4_22

Paired T for k2_22 - k4_22

	N	Mean	StDev	SE Mean
k2_22	30	0.42982	0.00604	0.00110
k4_22	30	0.47017	0.00480	0.00088
Difference	30	-0.04035	0.00744	0.00136

95% CI for mean difference: (-0.04313, -0.03757)
T-Test of mean difference = 0 (vs not = 0): T-Value = -29.71 P-Value = 0.000

Paired T-Test and CI: k3_22, k4_22

Paired T for k3_22 - k4_22

	N	Mean	StDev	SE Mean
k3_22	30	0.463333	0.004760	0.000869
k4_22	30	0.470167	0.004800	0.000876
Difference	30	-0.00683	0.00738	0.00135

95% CI for mean difference: (-0.00959, -0.00408)
T-Test of mean difference = 0 (vs not = 0): T-Value = -5.07 P-Value = 0.000

Paired T-Test and CI: k2_24, k3_24

Paired T for k2_24 - k3_24

	N	Mean	StDev	SE Mean
k2_24	30	0.42641	0.00587	0.00107
k3_24	30	0.45653	0.00408	0.00074
Difference	30	-0.03012	0.00686	0.00125

95% CI for mean difference: (-0.03268, -0.02755)
T-Test of mean difference = 0 (vs not = 0): T-Value = -24.04 P-Value = 0.000

Paired T-Test and CI: k2_24, k4_24

Paired T for k2_24 - k4_24

	N	Mean	StDev	SE Mean
k2_24	30	0.42641	0.00587	0.00107
k4_24	30	0.46377	0.00528	0.00096
Difference	30	-0.03736	0.00836	0.00153

95% CI for mean difference: (-0.04048, -0.03424)
T-Test of mean difference = 0 (vs not = 0): T-Value = -24.47 P-Value = 0.000

Paired T-Test and CI: k3_24, k4_24

Paired T for k3_24 - k4_24

	N	Mean	StDev	SE Mean
k3_24	30	0.456525	0.004080	0.000745
k4_24	30	0.463767	0.005276	0.000963
Difference	30	-0.00724	0.00780	0.00142

95% CI for mean difference: (-0.01015, -0.00433)
T-Test of mean difference = 0 (vs not = 0): T-Value = -5.09 P-Value = 0.000

Paired T-Test and CI: k2_26, k3_26

Paired T for k2_26 - k3_26

	N	Mean	StDev	SE Mean
k2_26	30	0.42365	0.00699	0.00128
k3_26	30	0.44973	0.00450	0.00082
Difference	30	-0.02607	0.00794	0.00145

95% CI for mean difference: (-0.02904, -0.02311)
T-Test of mean difference = 0 (vs not = 0): T-Value = -17.98 P-Value = 0.000

Paired T-Test and CI: k2_26, k4_26

Paired T for k2_26 - k4_26

	N	Mean	StDev	SE Mean
k2_26	30	0.42365	0.00699	0.00128
k4_26	30	0.45971	0.00456	0.00083
Difference	30	-0.03606	0.00776	0.00142

95% CI for mean difference: (-0.03896, -0.03316)
T-Test of mean difference = 0 (vs not = 0): T-Value = -25.44 P-Value = 0.000

Paired T-Test and CI: k3_26, k4_26

Paired T for k3_26 - k4_26

	N	Mean	StDev	SE Mean
k3_26	30	0.449725	0.004503	0.000822
k4_26	30	0.459708	0.004559	0.000832
Difference	30	-0.00998	0.00598	0.00109

95% CI for mean difference: (-0.01221, -0.00775)
T-Test of mean difference = 0 (vs not = 0): T-Value = -9.15 P-Value = 0.000

Paired T-Tests for Walking Time Analysis of One Sided GridPick

Paired T-Test and CI: k2_2, k3_2

Paired T for k2_2 - k3_2

	N	Mean	StDev	SE Mean
k2_2	30	3.3761	0.0834	0.0152
k3_2	30	3.4998	0.1071	0.0195
Difference	30	-0.1237	0.1556	0.0284

95% CI for mean difference: (-0.1819, -0.0656)

T-Test of mean difference = 0 (vs not = 0): T-Value = -4.35 P-Value = 0.000

Paired T-Test and CI: k2_4, k3_4

Paired T for k2_4 - k3_4

	N	Mean	StDev	SE Mean
k2_4	30	1.9511	0.0632	0.0115
k3_4	30	2.0146	0.0654	0.0119
Difference	30	-0.0635	0.0767	0.0140

95% CI for mean difference: (-0.0922, -0.0349)

T-Test of mean difference = 0 (vs not = 0): T-Value = -4.53 P-Value = 0.000

Paired T-Test and CI: k2_6, k3_6

Paired T for k2_6 - k3_6

	N	Mean	StDev	SE Mean
k2_6	30	1.33289	0.04214	0.00769
k3_6	30	1.38299	0.03030	0.00553
Difference	30	-0.05011	0.04950	0.00904

95% CI for mean difference: (-0.06859, -0.03162)

T-Test of mean difference = 0 (vs not = 0): T-Value = -5.54 P-Value = 0.000

Paired T-Test and CI: k2_8, k3_8

Paired T for k2_8 - k3_8

	N	Mean	StDev	SE Mean
k2_8	30	1.02971	0.02311	0.00422
k3_8	30	1.06211	0.02418	0.00441
Difference	30	-0.03240	0.02937	0.00536

95% CI for mean difference: (-0.04336, -0.02143)

T-Test of mean difference = 0 (vs not = 0): T-Value = -6.04 P-Value = 0.000

Paired T-Test and CI: k2_10, k3_10

Paired T for k2_10 - k3_10

	N	Mean	StDev	SE Mean
k2_10	30	0.83257	0.01824	0.00333
k3_10	30	0.87439	0.02098	0.00383
Difference	30	-0.04182	0.02617	0.00478

95% CI for mean difference: (-0.05159, -0.03205)

T-Test of mean difference = 0 (vs not = 0): T-Value = -8.75 P-Value = 0.000

Paired T-Test and CI: k2_12, k3_12

Paired T for k2_12 - k3_12

	N	Mean	StDev	SE Mean
k2_12	30	0.71705	0.01642	0.00300
k3_12	30	0.74818	0.01322	0.00241
Difference	30	-0.03114	0.01980	0.00361

95% CI for mean difference: (-0.03853, -0.02374)

T-Test of mean difference = 0 (vs not = 0): T-Value = -8.61 P-Value = 0.000

Paired T-Test and CI: k2_14, k3_14

Paired T for k2_14 - k3_14

	N	Mean	StDev	SE Mean
k2_14	30	0.63630	0.01106	0.00202
k3_14	30	0.67070	0.01153	0.00210
Difference	30	-0.03441	0.01603	0.00293

95% CI for mean difference: (-0.04039, -0.02842)

T-Test of mean difference = 0 (vs not = 0): T-Value = -11.76 P-Value = 0.000

Paired T-Test and CI: k2_16, k3_16

Paired T for k2_16 - k3_16

	N	Mean	StDev	SE Mean
k2_16	30	0.58461	0.00815	0.00149
k3_16	30	0.61650	0.00787	0.00144
Difference	30	-0.03189	0.01025	0.00187

95% CI for mean difference: (-0.03572, -0.02806)

T-Test of mean difference = 0 (vs not = 0): T-Value = -17.04 P-Value = 0.000

Paired T-Test and CI: k2_18, k3_18

Paired T for k2_18 - k3_18

	N	Mean	StDev	SE Mean
k2_18	30	0.54602	0.00625	0.00114
k3_18	30	0.57512	0.00846	0.00154

Difference 30 -0.02910 0.01137 0.00208

95% CI for mean difference: (-0.03335, -0.02486)

T-Test of mean difference = 0 (vs not = 0): T-Value = -14.01 P-Value = 0.000

Paired T-Test and CI: k2_20, k3_20

Paired T for k2_20 - k3_20

	N	Mean	StDev	SE Mean
k2_20	30	0.52296	0.00830	0.00151
k3_20	30	0.54564	0.01035	0.00189
Difference	30	-0.02268	0.01298	0.00237

95% CI for mean difference: (-0.02753, -0.01783)

T-Test of mean difference = 0 (vs not = 0): T-Value = -9.57 P-Value = 0.000

Paired T-Test and CI: k2_22, k3_22

Paired T for k2_22 - k3_22

	N	Mean	StDev	SE Mean
k2_22	30	0.50640	0.00653	0.00119
k3_22	30	0.52235	0.00796	0.00145
Difference	30	-0.01595	0.01058	0.00193

95% CI for mean difference: (-0.01990, -0.01200)

T-Test of mean difference = 0 (vs not = 0): T-Value = -8.26 P-Value = 0.000

Paired T-Test and CI: k2_24, k3_24

Paired T for k2_24 - k3_24

	N	Mean	StDev	SE Mean
k2_24	30	0.49219	0.00710	0.00130
k3_24	30	0.49893	0.00941	0.00172
Difference	30	-0.00674	0.01262	0.00230

95% CI for mean difference: (-0.01146, -0.00203)

T-Test of mean difference = 0 (vs not = 0): T-Value = -2.93 P-Value = 0.007

Paired T-Test and CI: k2_26, k3_26

Paired T for k2_26 - k3_26

	N	Mean	StDev	SE Mean
k2_26	30	0.48209	0.00700	0.00128
k3_26	30	0.48443	0.00627	0.00114
Difference	30	-0.00234	0.00847	0.00155

95% CI for mean difference: (-0.00551, 0.00082)

T-Test of mean difference = 0 (vs not = 0): T-Value = -1.52 P-Value = 0.140

Paired T-Test and CI: k2_2, k4_2

Paired T for k2_2 - k4_2

	N	Mean	StDev	SE Mean
k2_2	30	3.3761	0.0834	0.0152
k4_2	30	3.6503	0.0831	0.0152
Difference	30	-0.2742	0.1231	0.0225

95% CI for mean difference: (-0.3202, -0.2283)

T-Test of mean difference = 0 (vs not = 0): T-Value = -12.20 P-Value = 0.000

Paired T-Test and CI: k2_4, k4_4

Paired T for k2_4 - k4_4

	N	Mean	StDev	SE Mean
k2_4	30	1.9511	0.0632	0.0115
k4_4	30	2.0697	0.0524	0.0096
Difference	30	-0.1186	0.0795	0.0145

95% CI for mean difference: (-0.1483, -0.0889)

T-Test of mean difference = 0 (vs not = 0): T-Value = -8.17 P-Value = 0.000

Paired T-Test and CI: k2_6, k4_6

Paired T for k2_6 - k4_6

	N	Mean	StDev	SE Mean
k2_6	30	1.33289	0.04214	0.00769
k4_6	30	1.43340	0.03512	0.00641
Difference	30	-0.10051	0.05342	0.00975

95% CI for mean difference: (-0.12046, -0.08056)

T-Test of mean difference = 0 (vs not = 0): T-Value = -10.30 P-Value = 0.000

Paired T-Test and CI: k2_8, k4_8

Paired T for k2_8 - k4_8

	N	Mean	StDev	SE Mean
k2_8	30	1.02971	0.02311	0.00422
k4_8	30	1.10100	0.02166	0.00395
Difference	30	-0.07129	0.02478	0.00452

95% CI for mean difference: (-0.08054, -0.06203)

T-Test of mean difference = 0 (vs not = 0): T-Value = -15.76 P-Value = 0.000

Paired T-Test and CI: k2_10, k4_10

Paired T for k2_10 - k4_10

	N	Mean	StDev	SE Mean
k2_10	30	0.83257	0.01824	0.00333
k4_10	30	0.89311	0.01801	0.00329
Difference	30	-0.06053	0.02867	0.00523

95% CI for mean difference: (-0.07124, -0.04982)

T-Test of mean difference = 0 (vs not = 0): T-Value = -11.56 P-Value = 0.000

Paired T-Test and CI: k2_12, k4_12

Paired T for k2_12 - k4_12

	N	Mean	StDev	SE Mean
k2_12	30	0.71705	0.01642	0.00300
k4_12	30	0.76829	0.01231	0.00225
Difference	30	-0.05124	0.01800	0.00329

95% CI for mean difference: (-0.05796, -0.04452)

T-Test of mean difference = 0 (vs not = 0): T-Value = -15.59 P-Value = 0.000

Paired T-Test and CI: k2_14, k4_14

Paired T for k2_14 - k4_14

	N	Mean	StDev	SE Mean
k2_14	30	0.63630	0.01106	0.00202
k4_14	30	0.68626	0.00952	0.00174
Difference	30	-0.04996	0.01366	0.00249

95% CI for mean difference: (-0.05506, -0.04486)

T-Test of mean difference = 0 (vs not = 0): T-Value = -20.03 P-Value = 0.000

Paired T-Test and CI: k2_16, k4_16

Paired T for k2_16 - k4_16

	N	Mean	StDev	SE Mean
k2_16	30	0.58461	0.00815	0.00149
k4_16	30	0.62653	0.00960	0.00175
Difference	30	-0.04193	0.00975	0.00178

95% CI for mean difference: (-0.04557, -0.03829)

T-Test of mean difference = 0 (vs not = 0): T-Value = -23.55 P-Value = 0.000

Paired T-Test and CI: k2_18, k4_18

Paired T for k2_18 - k4_18

	N	Mean	StDev	SE Mean
k2_18	30	0.54602	0.00625	0.00114
k4_18	30	0.58192	0.01007	0.00184

Difference 30 -0.03590 0.01245 0.00227

95% CI for mean difference: (-0.04055, -0.03125)

T-Test of mean difference = 0 (vs not = 0): T-Value = -15.80 P-Value = 0.000

Paired T-Test and CI: k2_20, k4_20

Paired T for k2_20 - k4_20

	N	Mean	StDev	SE Mean
k2_20	30	0.52296	0.00830	0.00151
k4_20	30	0.54807	0.00920	0.00168
Difference	30	-0.02510	0.01003	0.00183

95% CI for mean difference: (-0.02885, -0.02136)

T-Test of mean difference = 0 (vs not = 0): T-Value = -13.71 P-Value = 0.000

Paired T-Test and CI: k2_22, k4_22

Paired T for k2_22 - k4_22

	N	Mean	StDev	SE Mean
k2_22	30	0.50640	0.00653	0.00119
k4_22	30	0.51359	0.01232	0.00225
Difference	30	-0.00720	0.01268	0.00231

95% CI for mean difference: (-0.01193, -0.00246)

T-Test of mean difference = 0 (vs not = 0): T-Value = -3.11 P-Value = 0.004

Paired T-Test and CI: k2_24, k4_24

Paired T for k2_24 - k4_24

	N	Mean	StDev	SE Mean
k2_24	30	0.49219	0.00710	0.00130
k4_24	30	0.48512	0.01170	0.00214
Difference	30	0.00707	0.01324	0.00242

95% CI for mean difference: (0.00212, 0.01202)

T-Test of mean difference = 0 (vs not = 0): T-Value = 2.92 P-Value = 0.007

Paired T-Test and CI: k2_26, k4_26

Paired T for k2_26 - k4_26

	N	Mean	StDev	SE Mean
k2_26	30	0.48209	0.00700	0.00128
k4_26	30	0.46453	0.01345	0.00246
Difference	30	0.01756	0.01463	0.00267

95% CI for mean difference: (0.01210, 0.02302)

T-Test of mean difference = 0 (vs not = 0): T-Value = 6.58 P-Value = 0.000

Paired T-Test and CI: k3_2, k4_2

Paired T for k3_2 - k4_2

	N	Mean	StDev	SE Mean
k3_2	30	3.4998	0.1071	0.0195
k4_2	30	3.6503	0.0831	0.0152
Difference	30	-0.1505	0.1315	0.0240

95% CI for mean difference: (-0.1996, -0.1014)

T-Test of mean difference = 0 (vs not = 0): T-Value = -6.27 P-Value = 0.000

Paired T-Test and CI: k3_4, k4_4

Paired T for k3_4 - k4_4

	N	Mean	StDev	SE Mean
k3_4	30	2.0146	0.0654	0.0119
k4_4	30	2.0697	0.0524	0.0096
Difference	30	-0.0551	0.0865	0.0158

95% CI for mean difference: (-0.0874, -0.0228)

T-Test of mean difference = 0 (vs not = 0): T-Value = -3.49 P-Value = 0.002

Paired T-Test and CI: k3_6, k4_6

Paired T for k3_6 - k4_6

	N	Mean	StDev	SE Mean
k3_6	30	1.38299	0.03030	0.00553
k4_6	30	1.43340	0.03512	0.00641
Difference	30	-0.05040	0.03930	0.00718

95% CI for mean difference: (-0.06508, -0.03573)

T-Test of mean difference = 0 (vs not = 0): T-Value = -7.02 P-Value = 0.000

Paired T-Test and CI: k3_8, k4_8

Paired T for k3_8 - k4_8

	N	Mean	StDev	SE Mean
k3_8	30	1.06211	0.02418	0.00441
k4_8	30	1.10100	0.02166	0.00395
Difference	30	-0.03889	0.03472	0.00634

95% CI for mean difference: (-0.05186, -0.02592)

T-Test of mean difference = 0 (vs not = 0): T-Value = -6.13 P-Value = 0.000

Paired T-Test and CI: k3_10, k4_10

Paired T for k3_10 - k4_10

	N	Mean	StDev	SE Mean
k3_10	30	0.87439	0.02098	0.00383
k4_10	30	0.89311	0.01801	0.00329
Difference	30	-0.01871	0.02866	0.00523

95% CI for mean difference: (-0.02941, -0.00801)

T-Test of mean difference = 0 (vs not = 0): T-Value = -3.58 P-Value = 0.001

Paired T-Test and CI: k3_12, k4_12

Paired T for k3_12 - k4_12

	N	Mean	StDev	SE Mean
k3_12	30	0.74818	0.01322	0.00241
k4_12	30	0.76829	0.01231	0.00225
Difference	30	-0.02011	0.02039	0.00372

95% CI for mean difference: (-0.02772, -0.01249)

T-Test of mean difference = 0 (vs not = 0): T-Value = -5.40 P-Value = 0.000

Paired T-Test and CI: k3_14, k4_14

Paired T for k3_14 - k4_14

	N	Mean	StDev	SE Mean
k3_14	30	0.67070	0.01153	0.00210
k4_14	30	0.68626	0.00952	0.00174
Difference	30	-0.01555	0.01239	0.00226

95% CI for mean difference: (-0.02018, -0.01092)

T-Test of mean difference = 0 (vs not = 0): T-Value = -6.87 P-Value = 0.000

Paired T-Test and CI: k3_16, k4_16

Paired T for k3_16 - k4_16

	N	Mean	StDev	SE Mean
k3_16	30	0.61650	0.00787	0.00144
k4_16	30	0.62653	0.00960	0.00175
Difference	30	-0.01004	0.01427	0.00260

95% CI for mean difference: (-0.01537, -0.00471)

T-Test of mean difference = 0 (vs not = 0): T-Value = -3.85 P-Value = 0.001

Paired T-Test and CI: k3_18, k4_18

Paired T for k3_18 - k4_18

	N	Mean	StDev	SE Mean
k3_18	30	0.57512	0.00846	0.00154

k4_18	30	0.58192	0.01007	0.00184
Difference	30	-0.00680	0.01383	0.00253

95% CI for mean difference: (-0.01196, -0.00163)

T-Test of mean difference = 0 (vs not = 0): T-Value = -2.69 P-Value = 0.012

Paired T-Test and CI: k3_20, k4_20

Paired T for k3_20 - k4_20

	N	Mean	StDev	SE Mean
k3_20	30	0.54564	0.01035	0.00189
k4_20	30	0.54807	0.00920	0.00168
Difference	30	-0.00242	0.01450	0.00265

95% CI for mean difference: (-0.00784, 0.00299)

T-Test of mean difference = 0 (vs not = 0): T-Value = -0.92 P-Value = 0.367

Paired T-Test and CI: k3_22, k4_22

Paired T for k3_22 - k4_22

	N	Mean	StDev	SE Mean
k3_22	30	0.52235	0.00796	0.00145
k4_22	30	0.51359	0.01232	0.00225
Difference	30	0.00876	0.01695	0.00309

95% CI for mean difference: (0.00243, 0.01509)

T-Test of mean difference = 0 (vs not = 0): T-Value = 2.83 P-Value = 0.008

Paired T-Test and CI: k3_24, k4_24

Paired T for k3_24 - k4_24

	N	Mean	StDev	SE Mean
k3_24	30	0.49893	0.00941	0.00172
k4_24	30	0.48512	0.01170	0.00214
Difference	30	0.01381	0.01508	0.00275

95% CI for mean difference: (0.00818, 0.01944)

T-Test of mean difference = 0 (vs not = 0): T-Value = 5.02 P-Value = 0.000

Paired T-Test and CI: k3_26, k4_26

Paired T for k3_26 - k4_26

	N	Mean	StDev	SE Mean
k3_26	30	0.48443	0.00627	0.00114
k4_26	30	0.46453	0.01345	0.00246
Difference	30	0.01990	0.01439	0.00263

95% CI for mean difference: (0.01453, 0.02528)

T-Test of mean difference = 0 (vs not = 0): T-Value = 7.58 P-Value = 0.000

Paired T-Tests for Waiting Time Analysis of One Sided GridPick

Paired T-Test and CI: k2_2, k3_2

Paired T for k2_2 - k3_2

	N	Mean	StDev	SE Mean
k2_2	30	0.51272	0.00976	0.00178
k3_2	30	0.51390	0.01114	0.00203
Difference	30	-0.00118	0.01686	0.00308

95% CI for mean difference: (-0.00748, 0.00511)

T-Test of mean difference = 0 (vs not = 0): T-Value = -0.38 P-Value = 0.704

Paired T-Test and CI: k2_4, k3_4

Paired T for k2_4 - k3_4

	N	Mean	StDev	SE Mean
k2_4	30	0.54087	0.01049	0.00192
k3_4	30	0.53424	0.01016	0.00186
Difference	30	0.00663	0.01646	0.00301

95% CI for mean difference: (0.00048, 0.01278)

T-Test of mean difference = 0 (vs not = 0): T-Value = 2.21 P-Value = 0.035

Paired T-Test and CI: k2_6, k3_6

Paired T for k2_6 - k3_6

	N	Mean	StDev	SE Mean
k2_6	30	0.61647	0.02133	0.00390
k3_6	30	0.55963	0.00761	0.00139
Difference	30	0.05685	0.02337	0.00427

95% CI for mean difference: (0.04812, 0.06557)

T-Test of mean difference = 0 (vs not = 0): T-Value = 13.33 P-Value = 0.000

Paired T-Test and CI: k2_8, k3_8

Paired T for k2_8 - k3_8

	N	Mean	StDev	SE Mean
k2_8	30	0.70082	0.02245	0.00410
k3_8	30	0.58989	0.00814	0.00149
Difference	30	0.11093	0.02581	0.00471

95% CI for mean difference: (0.10129, 0.12056)

T-Test of mean difference = 0 (vs not = 0): T-Value = 23.54 P-Value = 0.000

Paired T-Test and CI: k2_10, k3_10

Paired T for k2_10 - k3_10

	N	Mean	StDev	SE Mean
k2_10	30	0.77807	0.02508	0.00458
k3_10	30	0.62158	0.00611	0.00112
Difference	30	0.15650	0.02651	0.00484

95% CI for mean difference: (0.14660, 0.16640)

T-Test of mean difference = 0 (vs not = 0): T-Value = 32.33 P-Value = 0.000

Paired T-Test and CI: k2_12, k3_12

Paired T for k2_12 - k3_12

	N	Mean	StDev	SE Mean
k2_12	30	0.84566	0.03046	0.00556
k3_12	30	0.65810	0.01177	0.00215
Difference	30	0.18757	0.03162	0.00577

95% CI for mean difference: (0.17576, 0.19937)

T-Test of mean difference = 0 (vs not = 0): T-Value = 32.49 P-Value = 0.000

Paired T-Test and CI: k2_14, k3_14

Paired T for k2_14 - k3_14

	N	Mean	StDev	SE Mean
k2_14	30	0.88997	0.02221	0.00406
k3_14	30	0.69261	0.01160	0.00212
Difference	30	0.19736	0.02434	0.00444

95% CI for mean difference: (0.18827, 0.20645)

T-Test of mean difference = 0 (vs not = 0): T-Value = 44.41 P-Value = 0.000

Paired T-Test and CI: k2_16, k3_16

Paired T for k2_16 - k3_16

	N	Mean	StDev	SE Mean
k2_16	30	0.95015	0.02601	0.00475
k3_16	30	0.73747	0.01289	0.00235
Difference	30	0.21269	0.03093	0.00565

95% CI for mean difference: (0.20114, 0.22423)

T-Test of mean difference = 0 (vs not = 0): T-Value = 37.67 P-Value = 0.000

Paired T-Test and CI: k2_18, k3_18

Paired T for k2_18 - k3_18

	N	Mean	StDev	SE Mean
k2_18	30	0.99330	0.02902	0.00530

k3_18	30	0.78819	0.02059	0.00376
Difference	30	0.20510	0.03994	0.00729

95% CI for mean difference: (0.19019, 0.22002)

T-Test of mean difference = 0 (vs not = 0): T-Value = 28.12 P-Value = 0.000

Paired T-Test and CI: k2_20, k3_20

Paired T for k2_20 - k3_20

	N	Mean	StDev	SE Mean
k2_20	30	1.02935	0.03066	0.00560
k3_20	30	0.83299	0.02318	0.00423
Difference	30	0.19636	0.03685	0.00673

95% CI for mean difference: (0.18260, 0.21012)

T-Test of mean difference = 0 (vs not = 0): T-Value = 29.18 P-Value = 0.000

Paired T-Test and CI: k2_22, k3_22

Paired T for k2_22 - k3_22

	N	Mean	StDev	SE Mean
k2_22	30	1.07110	0.03500	0.00639
k3_22	30	0.88588	0.02748	0.00502
Difference	30	0.18522	0.04902	0.00895

95% CI for mean difference: (0.16691, 0.20352)

T-Test of mean difference = 0 (vs not = 0): T-Value = 20.69 P-Value = 0.000

Paired T-Test and CI: k2_24, k3_24

Paired T for k2_24 - k3_24

	N	Mean	StDev	SE Mean
k2_24	30	1.10417	0.03507	0.00640
k3_24	30	0.94191	0.02659	0.00485
Difference	30	0.16226	0.04460	0.00814

95% CI for mean difference: (0.14561, 0.17891)

T-Test of mean difference = 0 (vs not = 0): T-Value = 19.93 P-Value = 0.000

Paired T-Test and CI: k2_26, k3_26

Paired T for k2_26 - k3_26

	N	Mean	StDev	SE Mean
k2_26	30	1.12908	0.03990	0.00729
k3_26	30	0.98981	0.02515	0.00459
Difference	30	0.13926	0.04429	0.00809

95% CI for mean difference: (0.12272, 0.15580)

T-Test of mean difference = 0 (vs not = 0): T-Value = 17.22 P-Value = 0.000

Paired T-Test and CI: k2_2, k4_2

Paired T for k2_2 - k4_2

	N	Mean	StDev	SE Mean
k2_2	30	0.51272	0.00976	0.00178
k4_2	30	0.50739	0.01099	0.00201
Difference	30	0.00533	0.01472	0.00269

95% CI for mean difference: (-0.00017, 0.01083)

T-Test of mean difference = 0 (vs not = 0): T-Value = 1.98 P-Value = 0.057

Paired T-Test and CI: k2_4, k4_4

Paired T for k2_4 - k4_4

	N	Mean	StDev	SE Mean
k2_4	30	0.54087	0.01049	0.00192
k4_4	30	0.52916	0.00692	0.00126
Difference	30	0.01171	0.01279	0.00234

95% CI for mean difference: (0.00693, 0.01649)

T-Test of mean difference = 0 (vs not = 0): T-Value = 5.01 P-Value = 0.000

Paired T-Test and CI: k2_6, k4_6

Paired T for k2_6 - k4_6

	N	Mean	StDev	SE Mean
k2_6	30	0.61647	0.02133	0.00390
k4_6	30	0.55462	0.00832	0.00152
Difference	30	0.06185	0.02339	0.00427

95% CI for mean difference: (0.05311, 0.07058)

T-Test of mean difference = 0 (vs not = 0): T-Value = 14.48 P-Value = 0.000

Paired T-Test and CI: k2_8, k4_8

Paired T for k2_8 - k4_8

	N	Mean	StDev	SE Mean
k2_8	30	0.70082	0.02245	0.00410
k4_8	30	0.58437	0.00699	0.00128
Difference	30	0.11644	0.02238	0.00409

95% CI for mean difference: (0.10809, 0.12480)

T-Test of mean difference = 0 (vs not = 0): T-Value = 28.49 P-Value = 0.000

Paired T-Test and CI: k2_10, k4_10

Paired T for k2_10 - k4_10

	N	Mean	StDev	SE Mean
k2_10	30	0.77807	0.02508	0.00458
k4_10	30	0.61016	0.00875	0.00160
Difference	30	0.16791	0.02675	0.00488

95% CI for mean difference: (0.15792, 0.17790)

T-Test of mean difference = 0 (vs not = 0): T-Value = 34.38 P-Value = 0.000

Paired T-Test and CI: k2_12, k4_12

Paired T for k2_12 - k4_12

	N	Mean	StDev	SE Mean
k2_12	30	0.84566	0.03046	0.00556
k4_12	30	0.64279	0.00855	0.00156
Difference	30	0.20288	0.03137	0.00573

95% CI for mean difference: (0.19117, 0.21459)

T-Test of mean difference = 0 (vs not = 0): T-Value = 35.43 P-Value = 0.000

Paired T-Test and CI: k2_14, k4_14

Paired T for k2_14 - k4_14

	N	Mean	StDev	SE Mean
k2_14	30	0.88997	0.02221	0.00406
k4_14	30	0.67302	0.00637	0.00116
Difference	30	0.21695	0.02398	0.00438

95% CI for mean difference: (0.20800, 0.22591)

T-Test of mean difference = 0 (vs not = 0): T-Value = 49.55 P-Value = 0.000

Paired T-Test and CI: k2_16, k4_16

Paired T for k2_16 - k4_16

	N	Mean	StDev	SE Mean
k2_16	30	0.95015	0.02601	0.00475
k4_16	30	0.70906	0.01133	0.00207
Difference	30	0.24109	0.02663	0.00486

95% CI for mean difference: (0.23114, 0.25103)

T-Test of mean difference = 0 (vs not = 0): T-Value = 49.59 P-Value = 0.000

Paired T-Test and CI: k2_18, k4_18

Paired T for k2_18 - k4_18

	N	Mean	StDev	SE Mean
k2_18	30	0.99330	0.02902	0.00530

k4_18 30 0.75362 0.01473 0.00269
Difference 30 0.23968 0.03432 0.00627

95% CI for mean difference: (0.22686, 0.25250)
T-Test of mean difference = 0 (vs not = 0): T-Value = 38.25 P-Value = 0.000

Paired T-Test and CI: k2_20, k4_20

Paired T for k2_20 - k4_20

	N	Mean	StDev	SE Mean
k2_20	30	1.02935	0.03066	0.00560
k4_20	30	0.80854	0.02041	0.00373
Difference	30	0.22081	0.04144	0.00757

95% CI for mean difference: (0.20533, 0.23628)
T-Test of mean difference = 0 (vs not = 0): T-Value = 29.19 P-Value = 0.000

Paired T-Test and CI: k2_22, k4_22

Paired T for k2_22 - k4_22

	N	Mean	StDev	SE Mean
k2_22	30	1.07110	0.03500	0.00639
k4_22	30	0.86354	0.03144	0.00574
Difference	30	0.20756	0.04483	0.00818

95% CI for mean difference: (0.19082, 0.22430)
T-Test of mean difference = 0 (vs not = 0): T-Value = 25.36 P-Value = 0.000

Paired T-Test and CI: k2_24, k4_24

Paired T for k2_24 - k4_24

	N	Mean	StDev	SE Mean
k2_24	30	1.10417	0.03507	0.00640
k4_24	30	0.92137	0.03286	0.00600
Difference	30	0.18280	0.05194	0.00948

95% CI for mean difference: (0.16341, 0.20220)
T-Test of mean difference = 0 (vs not = 0): T-Value = 19.28 P-Value = 0.000

Paired T-Test and CI: k2_26, k4_26

Paired T for k2_26 - k4_26

	N	Mean	StDev	SE Mean
k2_26	30	1.12908	0.03990	0.00729
k4_26	30	0.96134	0.03078	0.00562
Difference	30	0.16774	0.04912	0.00897

95% CI for mean difference: (0.14940, 0.18608)

T-Test of mean difference = 0 (vs not = 0): T-Value = 18.70 P-Value = 0.000

11/19/2013 12:10:24 AM

Paired T-Test and CI: k3_2, k4_2

Paired T for k3_2 - k4_2

	N	Mean	StDev	SE Mean
k3_2	30	0.51390	0.01114	0.00203
k4_2	30	0.50739	0.01099	0.00201
Difference	30	0.00651	0.01880	0.00343

95% CI for mean difference: (-0.00051, 0.01353)

T-Test of mean difference = 0 (vs not = 0): T-Value = 1.90 P-Value = 0.068

Paired T-Test and CI: k3_4, k4_4

Paired T for k3_4 - k4_4

	N	Mean	StDev	SE Mean
k3_4	30	0.53424	0.01016	0.00186
k4_4	30	0.52916	0.00692	0.00126
Difference	30	0.00508	0.01236	0.00226

95% CI for mean difference: (0.00046, 0.00970)

T-Test of mean difference = 0 (vs not = 0): T-Value = 2.25 P-Value = 0.032

Paired T-Test and CI: k3_6, k4_6

Paired T for k3_6 - k4_6

	N	Mean	StDev	SE Mean
k3_6	30	0.55963	0.00761	0.00139
k4_6	30	0.55462	0.00832	0.00152
Difference	30	0.00500	0.01139	0.00208

95% CI for mean difference: (0.00075, 0.00925)

T-Test of mean difference = 0 (vs not = 0): T-Value = 2.41 P-Value = 0.023

Paired T-Test and CI: k3_8, k4_8

Paired T for k3_8 - k4_8

	N	Mean	StDev	SE Mean
k3_8	30	0.58989	0.00814	0.00149
k4_8	30	0.58437	0.00699	0.00128
Difference	30	0.00552	0.01071	0.00196

95% CI for mean difference: (0.00152, 0.00952)

T-Test of mean difference = 0 (vs not = 0): T-Value = 2.82 P-Value = 0.009

Paired T-Test and CI: k3_10, k4_10

Paired T for k3_10 - k4_10

	N	Mean	StDev	SE Mean
k3_10	30	0.62158	0.00611	0.00112
k4_10	30	0.61016	0.00875	0.00160
Difference	30	0.01141	0.01139	0.00208

95% CI for mean difference: (0.00716, 0.01567)

T-Test of mean difference = 0 (vs not = 0): T-Value = 5.49 P-Value = 0.000

Paired T-Test and CI: k3_12, k4_12

Paired T for k3_12 - k4_12

	N	Mean	StDev	SE Mean
k3_12	30	0.65810	0.01177	0.00215
k4_12	30	0.64279	0.00855	0.00156
Difference	30	0.01531	0.01577	0.00288

95% CI for mean difference: (0.00942, 0.02120)

T-Test of mean difference = 0 (vs not = 0): T-Value = 5.32 P-Value = 0.000

Paired T-Test and CI: k3_14, k4_14

Paired T for k3_14 - k4_14

	N	Mean	StDev	SE Mean
k3_14	30	0.69261	0.01160	0.00212
k4_14	30	0.67302	0.00637	0.00116
Difference	30	0.01959	0.01327	0.00242

95% CI for mean difference: (0.01463, 0.02454)

T-Test of mean difference = 0 (vs not = 0): T-Value = 8.08 P-Value = 0.000

Paired T-Test and CI: k3_16, k4_16

Paired T for k3_16 - k4_16

	N	Mean	StDev	SE Mean
k3_16	30	0.73747	0.01289	0.00235
k4_16	30	0.70906	0.01133	0.00207
Difference	30	0.02840	0.01493	0.00273

95% CI for mean difference: (0.02283, 0.03398)

T-Test of mean difference = 0 (vs not = 0): T-Value = 10.42 P-Value = 0.000

Paired T-Test and CI: k3_18, k4_18

Paired T for k3_18 - k4_18

	N	Mean	StDev	SE Mean
k3_18	30	0.78819	0.02059	0.00376
k4_18	30	0.75362	0.01473	0.00269
Difference	30	0.03458	0.02407	0.00439

95% CI for mean difference: (0.02559, 0.04356)
T-Test of mean difference = 0 (vs not = 0): T-Value = 7.87 P-Value = 0.000

Paired T-Test and CI: k3_20, k4_20

Paired T for k3_20 - k4_20

	N	Mean	StDev	SE Mean
k3_20	30	0.83299	0.02318	0.00423
k4_20	30	0.80854	0.02041	0.00373
Difference	30	0.02445	0.02957	0.00540

95% CI for mean difference: (0.01341, 0.03549)
T-Test of mean difference = 0 (vs not = 0): T-Value = 4.53 P-Value = 0.000

Paired T-Test and CI: k3_22, k4_22

Paired T for k3_22 - k4_22

	N	Mean	StDev	SE Mean
k3_22	30	0.88588	0.02748	0.00502
k4_22	30	0.86354	0.03144	0.00574
Difference	30	0.02234	0.04646	0.00848

95% CI for mean difference: (0.00499, 0.03969)
T-Test of mean difference = 0 (vs not = 0): T-Value = 2.63 P-Value = 0.013

Paired T-Test and CI: k3_24, k4_24

Paired T for k3_24 - k4_24

	N	Mean	StDev	SE Mean
k3_24	30	0.94191	0.02659	0.00485
k4_24	30	0.92137	0.03286	0.00600
Difference	30	0.02054	0.04782	0.00873

95% CI for mean difference: (0.00269, 0.03840)
T-Test of mean difference = 0 (vs not = 0): T-Value = 2.35 P-Value = 0.026

Paired T-Test and CI: k3_26, k4_26

Paired T for k3_26 - k4_26

	N	Mean	StDev	SE Mean
k3_26	30	0.98981	0.02515	0.00459
k4_26	30	0.96134	0.03078	0.00562
Difference	30	0.02848	0.03870	0.00707

95% CI for mean difference: (0.01403, 0.04293)
T-Test of mean difference = 0 (vs not = 0): T-Value = 4.03 P-Value = 0.000

Paired T-Tests for Flow Time Analysis of One Sided GridPick

Paired T-Test and CI: flowk2_2, flowk3_2

Paired T for flowk2_2 - flowk3_2

	N	Mean	StDev	SE Mean
flowk2_2	30	8.8280	0.1294	0.0236
flowk3_2	30	9.1492	0.1524	0.0278
Difference	30	-0.3211	0.2003	0.0366

95% CI for mean difference: (-0.3959, -0.2464)

T-Test of mean difference = 0 (vs not = 0): T-Value = -8.78 P-Value = 0.000

Paired T-Test and CI: flowk2_4, flowk3_4

Paired T for flowk2_4 - flowk3_4

	N	Mean	StDev	SE Mean
flowk2_4	30	9.0903	0.1076	0.0196
flowk3_4	30	9.3867	0.1058	0.0193
Difference	30	-0.2964	0.1740	0.0318

95% CI for mean difference: (-0.3614, -0.2315)

T-Test of mean difference = 0 (vs not = 0): T-Value = -9.33 P-Value = 0.000

Paired T-Test and CI: flowk2_6, flowk3_6

Paired T for flowk2_6 - flowk3_6

	N	Mean	StDev	SE Mean
flowk2_6	30	9.8721	0.1734	0.0317
flowk3_6	30	10.1296	0.1407	0.0257
Difference	30	-0.2575	0.2158	0.0394

95% CI for mean difference: (-0.3381, -0.1770)

T-Test of mean difference = 0 (vs not = 0): T-Value = -6.54 P-Value = 0.000

Paired T-Test and CI: flowk2_8, flowk3_8

Paired T for flowk2_8 - flowk3_8

	N	Mean	StDev	SE Mean
flowk2_8	30	11.0970	0.1922	0.0351
flowk3_8	30	11.1240	0.1471	0.0269
Difference	30	-0.0270	0.2560	0.0467

95% CI for mean difference: (-0.1226, 0.0686)

T-Test of mean difference = 0 (vs not = 0): T-Value = -0.58 P-Value = 0.568

Paired T-Test and CI: flowk2_10, flowk3_10

Paired T for flowk2_10 - flowk3_10

	N	Mean	StDev	SE Mean
flowk2_10	30	12.7355	0.2756	0.0503
flowk3_10	30	12.3304	0.2394	0.0437
Difference	30	0.4051	0.3165	0.0578

95% CI for mean difference: (0.2870, 0.5233)

T-Test of mean difference = 0 (vs not = 0): T-Value = 7.01 P-Value = 0.000

Paired T-Test and CI: flowk2_12, flowk3_12

Paired T for flowk2_12 - flowk3_12

	N	Mean	StDev	SE Mean
flowk2_12	30	14.3682	0.4173	0.0762
flowk3_12	30	13.7338	0.2607	0.0476
Difference	30	0.6344	0.4586	0.0837

95% CI for mean difference: (0.4631, 0.8056)

T-Test of mean difference = 0 (vs not = 0): T-Value = 7.58 P-Value = 0.000

Paired T-Test and CI: flowk2_14, flowk3_14

Paired T for flowk2_14 - flowk3_14

	N	Mean	StDev	SE Mean
flowk2_14	30	16.2227	0.2997	0.0547
flowk3_14	30	15.0662	0.2703	0.0494
Difference	30	1.1565	0.4891	0.0893

95% CI for mean difference: (0.9739, 1.3391)

T-Test of mean difference = 0 (vs not = 0): T-Value = 12.95 P-Value = 0.000

Paired T-Test and CI: flowk2_16, flowk3_16

Paired T for flowk2_16 - flowk3_16

	N	Mean	StDev	SE Mean
flowk2_16	30	18.2717	0.4967	0.0907
flowk3_16	30	16.7265	0.3086	0.0563
Difference	30	1.5452	0.5050	0.0922

95% CI for mean difference: (1.3566, 1.7337)

T-Test of mean difference = 0 (vs not = 0): T-Value = 16.76 P-Value = 0.000

Paired T-Test and CI: flowk2_18, flowk3_18

Paired T for flowk2_18 - flowk3_18

	N	Mean	StDev	SE Mean
flowk2_18	30	20.431	0.562	0.103

flowk3_18	30	18.667	0.437	0.080
Difference	30	1.764	0.637	0.116

95% CI for mean difference: (1.526, 2.001)

T-Test of mean difference = 0 (vs not = 0): T-Value = 15.17 P-Value = 0.000

Paired T-Test and CI: flowk2_20, flowk3_20

Paired T for flowk2_20 - flowk3_20

	N	Mean	StDev	SE Mean
flowk2_20	30	22.161	0.486	0.089
flowk3_20	30	20.738	0.580	0.106
Difference	30	1.423	0.792	0.145

95% CI for mean difference: (1.127, 1.719)

T-Test of mean difference = 0 (vs not = 0): T-Value = 9.84 P-Value = 0.000

Paired T-Test and CI: flowk2_22, flowk3_22

Paired T for flowk2_22 - flowk3_22

	N	Mean	StDev	SE Mean
flowk2_22	30	23.920	0.720	0.132
flowk3_22	30	22.899	0.741	0.135
Difference	30	1.021	1.143	0.209

95% CI for mean difference: (0.595, 1.448)

T-Test of mean difference = 0 (vs not = 0): T-Value = 4.89 P-Value = 0.000

Paired T-Test and CI: flowk2_24, flowk3_24

Paired T for flowk2_24 - flowk3_24

	N	Mean	StDev	SE Mean
flowk2_24	30	25.607	0.640	0.117
flowk3_24	30	24.867	0.590	0.108
Difference	30	0.741	0.896	0.164

95% CI for mean difference: (0.406, 1.075)

T-Test of mean difference = 0 (vs not = 0): T-Value = 4.53 P-Value = 0.000

Paired T-Test and CI: flowk2_26, flowk3_26

Paired T for flowk2_26 - flowk3_26

	N	Mean	StDev	SE Mean
flowk2_26	30	26.974	0.896	0.164
flowk3_26	30	26.742	0.702	0.128
Difference	30	0.231	1.285	0.235

95% CI for mean difference: (-0.249, 0.711)

T-Test of mean difference = 0 (vs not = 0): T-Value = 0.99 P-Value = 0.332

Paired T-Test and CI: flowk2_2, flowk4_2

Paired T for flowk2_2 - flowk4_2

	N	Mean	StDev	SE Mean
flowk2_2	30	8.8280	0.1294	0.0236
flowk4_2	30	9.3838	0.1558	0.0285
Difference	30	-0.5557	0.1997	0.0365

95% CI for mean difference: (-0.6303, -0.4812)

T-Test of mean difference = 0 (vs not = 0): T-Value = -15.24 P-Value = 0.000

Paired T-Test and CI: flowk2_4, flowk4_4

Paired T for flowk2_4 - flowk4_4

	N	Mean	StDev	SE Mean
flowk2_4	30	9.0903	0.1076	0.0196
flowk4_4	30	9.5913	0.1557	0.0284
Difference	30	-0.5010	0.1631	0.0298

95% CI for mean difference: (-0.5619, -0.4401)

T-Test of mean difference = 0 (vs not = 0): T-Value = -16.83 P-Value = 0.000

Paired T-Test and CI: flowk2_6, flowk4_6

Paired T for flowk2_6 - flowk4_6

	N	Mean	StDev	SE Mean
flowk2_6	30	9.8721	0.1734	0.0317
flowk4_6	30	10.2660	0.1484	0.0271
Difference	30	-0.3940	0.2349	0.0429

95% CI for mean difference: (-0.4817, -0.3063)

T-Test of mean difference = 0 (vs not = 0): T-Value = -9.19 P-Value = 0.000

Paired T-Test and CI: flowk2_8, flowk4_8

Paired T for flowk2_8 - flowk4_8

	N	Mean	StDev	SE Mean
flowk2_8	30	11.0970	0.1922	0.0351
flowk4_8	30	11.2906	0.1570	0.0287
Difference	30	-0.1936	0.2409	0.0440

95% CI for mean difference: (-0.2836, -0.1037)

T-Test of mean difference = 0 (vs not = 0): T-Value = -4.40 P-Value = 0.000

Paired T-Test and CI: flowk2_10, flowk4_10

Paired T for flowk2_10 - flowk4_10

	N	Mean	StDev	SE Mean
flowk2_10	30	12.7355	0.2756	0.0503
flowk4_10	30	12.5043	0.2027	0.0370
Difference	30	0.2312	0.3081	0.0563

95% CI for mean difference: (0.1162, 0.3463)

T-Test of mean difference = 0 (vs not = 0): T-Value = 4.11 P-Value = 0.000

Paired T-Test and CI: flowk2_12, flowk4_12

Paired T for flowk2_12 - flowk4_12

	N	Mean	StDev	SE Mean
flowk2_12	30	14.3682	0.4173	0.0762
flowk4_12	30	13.8693	0.2120	0.0387
Difference	30	0.4989	0.5267	0.0962

95% CI for mean difference: (0.3023, 0.6956)

T-Test of mean difference = 0 (vs not = 0): T-Value = 5.19 P-Value = 0.000

Paired T-Test and CI: flowk2_14, flowk4_14

Paired T for flowk2_14 - flowk4_14

	N	Mean	StDev	SE Mean
flowk2_14	30	16.2227	0.2997	0.0547
flowk4_14	30	15.1082	0.2509	0.0458
Difference	30	1.1145	0.3880	0.0708

95% CI for mean difference: (0.9696, 1.2593)

T-Test of mean difference = 0 (vs not = 0): T-Value = 15.73 P-Value = 0.000

Paired T-Test and CI: flowk2_16, flowk4_16

Paired T for flowk2_16 - flowk4_16

	N	Mean	StDev	SE Mean
flowk2_16	30	18.2717	0.4967	0.0907
flowk4_16	30	16.4867	0.3267	0.0597
Difference	30	1.785	0.562	0.103

95% CI for mean difference: (1.575, 1.995)

T-Test of mean difference = 0 (vs not = 0): T-Value = 17.39 P-Value = 0.000

Paired T-Test and CI: flowk2_18, flowk4_18

Paired T for flowk2_18 - flowk4_18

	N	Mean	StDev	SE Mean
flowk2_18	30	20.431	0.562	0.103
flowk4_18	30	18.195	0.421	0.077

Difference 30 2.236 0.755 0.138

95% CI for mean difference: (1.954, 2.518)

T-Test of mean difference = 0 (vs not = 0): T-Value = 16.23 P-Value = 0.000

Paired T-Test and CI: flowk2_20, flowk4_20

Paired T for flowk2_20 - flowk4_20

	N	Mean	StDev	SE Mean
flowk2_20	30	22.161	0.486	0.089
flowk4_20	30	20.303	0.648	0.118
Difference	30	1.858	0.732	0.134

95% CI for mean difference: (1.585, 2.131)

T-Test of mean difference = 0 (vs not = 0): T-Value = 13.91 P-Value = 0.000

Paired T-Test and CI: flowk2_22, flowk4_22

Paired T for flowk2_22 - flowk4_22

	N	Mean	StDev	SE Mean
flowk2_22	30	23.920	0.720	0.132
flowk4_22	30	22.641	0.978	0.179
Difference	30	1.279	1.113	0.203

95% CI for mean difference: (0.864, 1.695)

T-Test of mean difference = 0 (vs not = 0): T-Value = 6.30 P-Value = 0.000

Paired T-Test and CI: flowk2_24, flowk4_24

Paired T for flowk2_24 - flowk4_24

	N	Mean	StDev	SE Mean
flowk2_24	30	25.607	0.640	0.117
flowk4_24	30	24.813	0.834	0.152
Difference	30	0.795	1.154	0.211

95% CI for mean difference: (0.364, 1.226)

T-Test of mean difference = 0 (vs not = 0): T-Value = 3.77 P-Value = 0.001

Paired T-Test and CI: flowk2_26, flowk4_26

Paired T for flowk2_26 - flowk4_26

	N	Mean	StDev	SE Mean
flowk2_26	30	26.974	0.896	0.164
flowk4_26	30	26.513	0.998	0.182
Difference	30	0.461	1.292	0.236

95% CI for mean difference: (-0.022, 0.943)

T-Test of mean difference = 0 (vs not = 0): T-Value = 1.95 P-Value = 0.061

Paired T-Test and CI: flowk3_2, flowk4_2

Paired T for flowk3_2 - flowk4_2

	N	Mean	StDev	SE Mean
flowk3_2	30	9.1492	0.1524	0.0278
flowk4_2	30	9.3838	0.1558	0.0285
Difference	30	-0.2346	0.1911	0.0349

95% CI for mean difference: (-0.3059, -0.1632)

T-Test of mean difference = 0 (vs not = 0): T-Value = -6.72 P-Value = 0.000

Paired T-Test and CI: flowk3_4, flowk4_4

Paired T for flowk3_4 - flowk4_4

	N	Mean	StDev	SE Mean
flowk3_4	30	9.3867	0.1058	0.0193
flowk4_4	30	9.5913	0.1557	0.0284
Difference	30	-0.2046	0.1917	0.0350

95% CI for mean difference: (-0.2762, -0.1330)

T-Test of mean difference = 0 (vs not = 0): T-Value = -5.85 P-Value = 0.000

Paired T-Test and CI: flowk3_6, flowk4_6

Paired T for flowk3_6 - flowk4_6

	N	Mean	StDev	SE Mean
flowk3_6	30	10.1296	0.1407	0.0257
flowk4_6	30	10.2660	0.1484	0.0271
Difference	30	-0.1364	0.1926	0.0352

95% CI for mean difference: (-0.2083, -0.0645)

T-Test of mean difference = 0 (vs not = 0): T-Value = -3.88 P-Value = 0.001

Paired T-Test and CI: flowk3_8, flowk4_8

Paired T for flowk3_8 - flowk4_8

	N	Mean	StDev	SE Mean
flowk3_8	30	11.1240	0.1471	0.0269
flowk4_8	30	11.2906	0.1570	0.0287
Difference	30	-0.1666	0.2312	0.0422

95% CI for mean difference: (-0.2530, -0.0803)

T-Test of mean difference = 0 (vs not = 0): T-Value = -3.95 P-Value = 0.000

Paired T-Test and CI: flowk3_10, flowk4_10

Paired T for flowk3_10 - flowk4_10

	N	Mean	StDev	SE Mean
flowk3_10	30	12.3304	0.2394	0.0437
flowk4_10	30	12.5043	0.2027	0.0370
Difference	30	-0.1739	0.3179	0.0580

95% CI for mean difference: (-0.2926, -0.0552)

T-Test of mean difference = 0 (vs not = 0): T-Value = -3.00 P-Value = 0.006

Paired T-Test and CI: flowk3_12, flowk4_12

Paired T for flowk3_12 - flowk4_12

	N	Mean	StDev	SE Mean
flowk3_12	30	13.7338	0.2607	0.0476
flowk4_12	30	13.8693	0.2120	0.0387
Difference	30	-0.1354	0.3886	0.0710

95% CI for mean difference: (-0.2806, 0.0097)

T-Test of mean difference = 0 (vs not = 0): T-Value = -1.91 P-Value = 0.066

Paired T-Test and CI: flowk3_14, flowk4_14

Paired T for flowk3_14 - flowk4_14

	N	Mean	StDev	SE Mean
flowk3_14	30	15.0662	0.2703	0.0494
flowk4_14	30	15.1082	0.2509	0.0458
Difference	30	-0.0420	0.3570	0.0652

95% CI for mean difference: (-0.1753, 0.0913)

T-Test of mean difference = 0 (vs not = 0): T-Value = -0.64 P-Value = 0.524

Paired T-Test and CI: flowk3_16, flowk4_16

Paired T for flowk3_16 - flowk4_16

	N	Mean	StDev	SE Mean
flowk3_16	30	16.7265	0.3086	0.0563
flowk4_16	30	16.4867	0.3267	0.0597
Difference	30	0.2398	0.4706	0.0859

95% CI for mean difference: (0.0641, 0.4155)

T-Test of mean difference = 0 (vs not = 0): T-Value = 2.79 P-Value = 0.009

Paired T-Test and CI: flowk3_18, flowk4_18

Paired T for flowk3_18 - flowk4_18

	N	Mean	StDev	SE Mean
flowk3_18	30	18.6671	0.4374	0.0799
flowk4_18	30	18.1948	0.4213	0.0769
Difference	30	0.472	0.674	0.123

95% CI for mean difference: (0.221, 0.724)
T-Test of mean difference = 0 (vs not = 0): T-Value = 3.84 P-Value = 0.001

Paired T-Test and CI: flowk3_20, flowk4_20

Paired T for flowk3_20 - flowk4_20

	N	Mean	StDev	SE Mean
flowk3_20	30	20.738	0.580	0.106
flowk4_20	30	20.303	0.648	0.118
Difference	30	0.435	0.850	0.155

95% CI for mean difference: (0.118, 0.752)
T-Test of mean difference = 0 (vs not = 0): T-Value = 2.80 P-Value = 0.009

Paired T-Test and CI: flowk3_22, flowk4_22

Paired T for flowk3_22 - flowk4_22

	N	Mean	StDev	SE Mean
flowk3_22	30	22.899	0.741	0.135
flowk4_22	30	22.641	0.978	0.179
Difference	30	0.258	1.279	0.234

95% CI for mean difference: (-0.220, 0.735)
T-Test of mean difference = 0 (vs not = 0): T-Value = 1.10 P-Value = 0.279

Paired T-Test and CI: flowk3_24, flowk4_24

Paired T for flowk3_24 - flowk4_24

	N	Mean	StDev	SE Mean
flowk3_24	30	24.867	0.590	0.108
flowk4_24	30	24.813	0.834	0.152
Difference	30	0.054	0.998	0.182

95% CI for mean difference: (-0.319, 0.426)
T-Test of mean difference = 0 (vs not = 0): T-Value = 0.29 P-Value = 0.770

Paired T-Test and CI: flowk3_26, flowk4_26

Paired T for flowk3_26 - flowk4_26

	N	Mean	StDev	SE Mean
flowk3_26	30	26.742	0.702	0.128
flowk4_26	30	26.513	0.998	0.182
Difference	30	0.229	1.245	0.227

95% CI for mean difference: (-0.236, 0.694)
T-Test of mean difference = 0 (vs not = 0): T-Value = 1.01 P-Value = 0.322

Paired T-Tests for Throughput Analysis of Two Sided GridPick

Paired T-Test and CI: k2_2, k3_2

Paired T for k2_2 - k3_2

	N	Mean	StDev	SE Mean
k2_2	30	0.42033	0.00501	0.00091
k3_2	30	0.41150	0.00640	0.00117
Difference	30	0.00883	0.00861	0.00157

95% CI for mean difference: (0.00562, 0.01205)

T-Test of mean difference = 0 (vs not = 0): T-Value = 5.62 P-Value = 0.000

Paired T-Test and CI: k2_4, k3_4

Paired T for k2_4 - k3_4

	N	Mean	StDev	SE Mean
k2_4	30	0.55485	0.00635	0.00116
k3_4	30	0.58971	0.00707	0.00129
Difference	30	-0.03486	0.00982	0.00179

95% CI for mean difference: (-0.03853, -0.03119)

T-Test of mean difference = 0 (vs not = 0): T-Value = -19.43 P-Value = 0.000

Paired T-Test and CI: k2_6, k3_6

Paired T for k2_6 - k3_6

	N	Mean	StDev	SE Mean
k2_6	30	0.62818	0.00779	0.00142
k3_6	30	0.70188	0.00594	0.00108
Difference	30	-0.07370	0.00865	0.00158

95% CI for mean difference: (-0.07693, -0.07047)

T-Test of mean difference = 0 (vs not = 0): T-Value = -46.67 P-Value = 0.000

Paired T-Test and CI: k2_8, k3_8

Paired T for k2_8 - k3_8

	N	Mean	StDev	SE Mean
k2_8	30	0.67145	0.00902	0.00165
k3_8	30	0.77295	0.00932	0.00170
Difference	30	-0.10150	0.01117	0.00204

95% CI for mean difference: (-0.10567, -0.09733)

T-Test of mean difference = 0 (vs not = 0): T-Value = -49.75 P-Value = 0.000

Paired T-Test and CI: k2_10, k3_10

Paired T for k2_10 - k3_10

	N	Mean	StDev	SE Mean
k2_10	30	0.70488	0.00927	0.00169
k3_10	30	0.82299	0.00762	0.00139
Difference	30	-0.11812	0.01005	0.00184

95% CI for mean difference: (-0.12187, -0.11436)

T-Test of mean difference = 0 (vs not = 0): T-Value = -64.36 P-Value = 0.000

Paired T-Test and CI: k2_12, k3_12

Paired T for k2_12 - k3_12

	N	Mean	StDev	SE Mean
k2_12	30	0.72367	0.00903	0.00165
k3_12	30	0.85464	0.00636	0.00116
Difference	30	-0.13097	0.00952	0.00174

95% CI for mean difference: (-0.13452, -0.12741)

T-Test of mean difference = 0 (vs not = 0): T-Value = -75.38 P-Value = 0.000

Paired T-Test and CI: k2_14, k3_14

Paired T for k2_14 - k3_14

	N	Mean	StDev	SE Mean
k2_14	30	0.73726	0.01003	0.00183
k3_14	30	0.86712	0.00608	0.00111
Difference	30	-0.12987	0.01082	0.00197

95% CI for mean difference: (-0.13391, -0.12583)

T-Test of mean difference = 0 (vs not = 0): T-Value = -65.76 P-Value = 0.000

Paired T-Test and CI: k2_16, k3_16

Paired T for k2_16 - k3_16

	N	Mean	StDev	SE Mean
k2_16	30	0.74321	0.00924	0.00169
k3_16	30	0.86382	0.00571	0.00104
Difference	30	-0.12061	0.01059	0.00193

95% CI for mean difference: (-0.12456, -0.11665)

T-Test of mean difference = 0 (vs not = 0): T-Value = -62.37 P-Value = 0.000

Paired T-Test and CI: k2_18, k3_18

Paired T for k2_18 - k3_18

	N	Mean	StDev	SE Mean
k2_18	30	0.74322	0.01196	0.00218
k3_18	30	0.85160	0.00540	0.00099

Difference 30 -0.10838 0.01443 0.00264

95% CI for mean difference: (-0.11377, -0.10299)

T-Test of mean difference = 0 (vs not = 0): T-Value = -41.13 P-Value = 0.000

Paired T-Test and CI: k2_20, k3_20

Paired T for k2_20 - k3_20

	N	Mean	StDev	SE Mean
k2_20	30	0.74372	0.00778	0.00142
k3_20	30	0.84083	0.00701	0.00128
Difference	30	-0.09711	0.00978	0.00179

95% CI for mean difference: (-0.10076, -0.09346)

T-Test of mean difference = 0 (vs not = 0): T-Value = -54.37 P-Value = 0.000

Paired T-Test and CI: k2_22, k3_22

Paired T for k2_22 - k3_22

	N	Mean	StDev	SE Mean
k2_22	30	0.74129	0.00755	0.00138
k3_22	30	0.83047	0.00747	0.00136
Difference	30	-0.08918	0.00992	0.00181

95% CI for mean difference: (-0.09289, -0.08548)

T-Test of mean difference = 0 (vs not = 0): T-Value = -49.25 P-Value = 0.000

Paired T-Test and CI: k2_24, k3_24

Paired T for k2_24 - k3_24

	N	Mean	StDev	SE Mean
k2_24	30	0.73611	0.00902	0.00165
k3_24	30	0.82238	0.00656	0.00120
Difference	30	-0.08627	0.00952	0.00174

95% CI for mean difference: (-0.08983, -0.08272)

T-Test of mean difference = 0 (vs not = 0): T-Value = -49.62 P-Value = 0.000

Paired T-Test and CI: k2_26, k3_26

Paired T for k2_26 - k3_26

	N	Mean	StDev	SE Mean
k2_26	30	0.72976	0.00804	0.00147
k3_26	30	0.81566	0.00669	0.00122
Difference	30	-0.08590	0.00976	0.00178

95% CI for mean difference: (-0.08954, -0.08226)

T-Test of mean difference = 0 (vs not = 0): T-Value = -48.22 P-Value = 0.000

Paired T-Test and CI: k2_2, k4_2

Paired T for k2_2 - k4_2

	N	Mean	StDev	SE Mean
k2_2	30	0.420333	0.005011	0.000915
k4_2	30	0.400158	0.005422	0.000990
Difference	30	0.02017	0.00756	0.00138

95% CI for mean difference: (0.01735, 0.02300)

T-Test of mean difference = 0 (vs not = 0): T-Value = 14.62 P-Value = 0.000

Paired T-Test and CI: k2_4, k4_4

Paired T for k2_4 - k4_4

	N	Mean	StDev	SE Mean
k2_4	30	0.55485	0.00635	0.00116
k4_4	30	0.57429	0.00667	0.00122
Difference	30	-0.01944	0.00923	0.00168

95% CI for mean difference: (-0.02289, -0.01600)

T-Test of mean difference = 0 (vs not = 0): T-Value = -11.54 P-Value = 0.000

Paired T-Test and CI: k2_6, k4_6

Paired T for k2_6 - k4_6

	N	Mean	StDev	SE Mean
k2_6	30	0.62818	0.00779	0.00142
k4_6	30	0.69890	0.00775	0.00141
Difference	30	-0.07072	0.01038	0.00189

95% CI for mean difference: (-0.07459, -0.06684)

T-Test of mean difference = 0 (vs not = 0): T-Value = -37.33 P-Value = 0.000

Paired T-Test and CI: k2_8, k4_8

Paired T for k2_8 - k4_8

	N	Mean	StDev	SE Mean
k2_8	30	0.67145	0.00902	0.00165
k4_8	30	0.77603	0.00586	0.00107
Difference	30	-0.10457	0.01179	0.00215

95% CI for mean difference: (-0.10898, -0.10017)

T-Test of mean difference = 0 (vs not = 0): T-Value = -48.59 P-Value = 0.000

Paired T-Test and CI: k2_10, k4_10

Paired T for k2_10 - k4_10

	N	Mean	StDev	SE Mean
k2_10	30	0.70488	0.00927	0.00169
k4_10	30	0.82148	0.00436	0.00080
Difference	30	-0.11661	0.01009	0.00184

95% CI for mean difference: (-0.12038, -0.11284)

T-Test of mean difference = 0 (vs not = 0): T-Value = -63.29 P-Value = 0.000

Paired T-Test and CI: k2_12, k4_12

Paired T for k2_12 - k4_12

	N	Mean	StDev	SE Mean
k2_12	30	0.72367	0.00903	0.00165
k4_12	30	0.84933	0.00428	0.00078
Difference	30	-0.12566	0.00860	0.00157

95% CI for mean difference: (-0.12887, -0.12245)

T-Test of mean difference = 0 (vs not = 0): T-Value = -80.01 P-Value = 0.000

Paired T-Test and CI: k2_14, k4_14

Paired T for k2_14 - k4_14

	N	Mean	StDev	SE Mean
k2_14	30	0.73726	0.01003	0.00183
k4_14	30	0.86583	0.00389	0.00071
Difference	30	-0.12857	0.01042	0.00190

95% CI for mean difference: (-0.13246, -0.12469)

T-Test of mean difference = 0 (vs not = 0): T-Value = -67.61 P-Value = 0.000

Paired T-Test and CI: k2_16, k4_16

Paired T for k2_16 - k4_16

	N	Mean	StDev	SE Mean
k2_16	30	0.74321	0.00924	0.00169
k4_16	30	0.86697	0.00322	0.00059
Difference	30	-0.12376	0.00931	0.00170

95% CI for mean difference: (-0.12724, -0.12028)

T-Test of mean difference = 0 (vs not = 0): T-Value = -72.79 P-Value = 0.000

Paired T-Test and CI: k2_18, k4_18

Paired T for k2_18 - k4_18

	N	Mean	StDev	SE Mean
k2_18	30	0.74322	0.01196	0.00218
k4_18	30	0.85641	0.00456	0.00083

Difference 30 -0.11319 0.01245 0.00227

95% CI for mean difference: (-0.11784, -0.10854)

T-Test of mean difference = 0 (vs not = 0): T-Value = -49.79 P-Value = 0.000

Paired T-Test and CI: k2_20, k4_20

Paired T for k2_20 - k4_20

	N	Mean	StDev	SE Mean
k2_20	30	0.74372	0.00778	0.00142
k4_20	30	0.84600	0.00419	0.00077
Difference	30	-0.10228	0.00851	0.00155

95% CI for mean difference: (-0.10546, -0.09911)

T-Test of mean difference = 0 (vs not = 0): T-Value = -65.86 P-Value = 0.000

Paired T-Test and CI: k2_22, k4_22

Paired T for k2_22 - k4_22

	N	Mean	StDev	SE Mean
k2_22	30	0.74129	0.00755	0.00138
k4_22	30	0.83518	0.00532	0.00097
Difference	30	-0.09389	0.00989	0.00180

95% CI for mean difference: (-0.09758, -0.09020)

T-Test of mean difference = 0 (vs not = 0): T-Value = -52.02 P-Value = 0.000

Paired T-Test and CI: k2_24, k4_24

Paired T for k2_24 - k4_24

	N	Mean	StDev	SE Mean
k2_24	30	0.73611	0.00902	0.00165
k4_24	30	0.83297	0.00602	0.00110
Difference	30	-0.09686	0.00964	0.00176

95% CI for mean difference: (-0.10046, -0.09326)

T-Test of mean difference = 0 (vs not = 0): T-Value = -55.03 P-Value = 0.000

Paired T-Test and CI: k2_26, k4_26

Paired T for k2_26 - k4_26

	N	Mean	StDev	SE Mean
k2_26	30	0.72976	0.00804	0.00147
k4_26	30	0.82958	0.00555	0.00101
Difference	30	-0.09982	0.00951	0.00174

95% CI for mean difference: (-0.10337, -0.09628)

T-Test of mean difference = 0 (vs not = 0): T-Value = -57.52 P-Value = 0.000

Paired T-Test and CI: k3_2, k4_2

Paired T for k3_2 - k4_2

	N	Mean	StDev	SE Mean
k3_2	30	0.41150	0.00640	0.00117
k4_2	30	0.40016	0.00542	0.00099
Difference	30	0.01134	0.00741	0.00135

95% CI for mean difference: (0.00858, 0.01411)

T-Test of mean difference = 0 (vs not = 0): T-Value = 8.39 P-Value = 0.000

Paired T-Test and CI: k3_4, k4_4

Paired T for k3_4 - k4_4

	N	Mean	StDev	SE Mean
k3_4	30	0.58971	0.00707	0.00129
k4_4	30	0.57429	0.00667	0.00122
Difference	30	0.01542	0.00991	0.00181

95% CI for mean difference: (0.01172, 0.01912)

T-Test of mean difference = 0 (vs not = 0): T-Value = 8.52 P-Value = 0.000

Paired T-Test and CI: k3_6, k4_6

Paired T for k3_6 - k4_6

	N	Mean	StDev	SE Mean
k3_6	30	0.70188	0.00594	0.00108
k4_6	30	0.69890	0.00775	0.00141
Difference	30	0.00298	0.01108	0.00202

95% CI for mean difference: (-0.00115, 0.00712)

T-Test of mean difference = 0 (vs not = 0): T-Value = 1.48 P-Value = 0.151

Paired T-Test and CI: k3_8, k4_8

Paired T for k3_8 - k4_8

	N	Mean	StDev	SE Mean
k3_8	30	0.77295	0.00932	0.00170
k4_8	30	0.77603	0.00586	0.00107
Difference	30	-0.00307	0.01242	0.00227

95% CI for mean difference: (-0.00771, 0.00156)

T-Test of mean difference = 0 (vs not = 0): T-Value = -1.36 P-Value = 0.185

Paired T-Test and CI: k3_10, k4_10

Paired T for k3_10 - k4_10

	N	Mean	StDev	SE Mean
k3_10	30	0.82299	0.00762	0.00139
k4_10	30	0.82148	0.00436	0.00080
Difference	30	0.00151	0.00824	0.00150

95% CI for mean difference: (-0.00157, 0.00459)

T-Test of mean difference = 0 (vs not = 0): T-Value = 1.00 P-Value = 0.324

Paired T-Test and CI: k3_12, k4_12

Paired T for k3_12 - k4_12

	N	Mean	StDev	SE Mean
k3_12	30	0.85464	0.00636	0.00116
k4_12	30	0.84933	0.00428	0.00078
Difference	30	0.00531	0.00618	0.00113

95% CI for mean difference: (0.00300, 0.00762)

T-Test of mean difference = 0 (vs not = 0): T-Value = 4.70 P-Value = 0.000

Paired T-Test and CI: k3_14, k4_14

Paired T for k3_14 - k4_14

	N	Mean	StDev	SE Mean
k3_14	30	0.86712	0.00608	0.00111
k4_14	30	0.86583	0.00389	0.00071
Difference	30	0.00129	0.00621	0.00113

95% CI for mean difference: (-0.00103, 0.00361)

T-Test of mean difference = 0 (vs not = 0): T-Value = 1.14 P-Value = 0.264

Paired T-Test and CI: k3_16, k4_16

Paired T for k3_16 - k4_16

	N	Mean	StDev	SE Mean
k3_16	30	0.86382	0.00571	0.00104
k4_16	30	0.86697	0.00322	0.00059
Difference	30	-0.00315	0.00742	0.00136

95% CI for mean difference: (-0.00592, -0.00038)

T-Test of mean difference = 0 (vs not = 0): T-Value = -2.32 P-Value = 0.027

Paired T-Test and CI: k3_18, k4_18

Paired T for k3_18 - k4_18

	N	Mean	StDev	SE Mean
k3_18	30	0.851600	0.005398	0.000986
k4_18	30	0.856408	0.004560	0.000833

Difference 30 -0.00481 0.00760 0.00139

95% CI for mean difference: (-0.00764, -0.00197)

T-Test of mean difference = 0 (vs not = 0): T-Value = -3.47 P-Value = 0.002

Paired T-Test and CI: k3_20, k4_20

Paired T for k3_20 - k4_20

	N	Mean	StDev	SE Mean
k3_20	30	0.84083	0.00701	0.00128
k4_20	30	0.84600	0.00419	0.00077
Difference	30	-0.00517	0.00679	0.00124

95% CI for mean difference: (-0.00771, -0.00264)

T-Test of mean difference = 0 (vs not = 0): T-Value = -4.18 P-Value = 0.000

Paired T-Test and CI: k3_22, k4_22

Paired T for k3_22 - k4_22

	N	Mean	StDev	SE Mean
k3_22	30	0.83047	0.00747	0.00136
k4_22	30	0.83518	0.00532	0.00097
Difference	30	-0.00471	0.00775	0.00141

95% CI for mean difference: (-0.00760, -0.00182)

T-Test of mean difference = 0 (vs not = 0): T-Value = -3.33 P-Value = 0.002

Paired T-Test and CI: k3_24, k4_24

Paired T for k3_24 - k4_24

	N	Mean	StDev	SE Mean
k3_24	30	0.82238	0.00656	0.00120
k4_24	30	0.83297	0.00602	0.00110
Difference	30	-0.01058	0.00909	0.00166

95% CI for mean difference: (-0.01398, -0.00719)

T-Test of mean difference = 0 (vs not = 0): T-Value = -6.38 P-Value = 0.000

Paired T-Test and CI: k3_26, k4_26

Paired T for k3_26 - k4_26

	N	Mean	StDev	SE Mean
k3_26	30	0.81566	0.00669	0.00122
k4_26	30	0.82958	0.00555	0.00101
Difference	30	-0.01393	0.00785	0.00143

95% CI for mean difference: (-0.01685, -0.01100)

T-Test of mean difference = 0 (vs not = 0): T-Value = -9.72 P-Value = 0.000

Paired T-Tests for Walking Time Analysis of Two Sided GridPick

Paired T-Test and CI: k2_2, k3_2

Paired T for k2_2 - k3_2

	N	Mean	StDev	SE Mean
k2_2	30	3.4129	0.0567	0.0104
k3_2	30	3.5420	0.0792	0.0145
Difference	30	-0.1291	0.0955	0.0174

95% CI for mean difference: (-0.1647, -0.0934)

T-Test of mean difference = 0 (vs not = 0): T-Value = -7.40 P-Value = 0.000

Paired T-Test and CI: k2_4, k3_4

Paired T for k2_4 - k3_4

	N	Mean	StDev	SE Mean
k2_4	30	1.94832	0.03748	0.00684
k3_4	30	2.01959	0.04034	0.00736
Difference	30	-0.07127	0.05047	0.00921

95% CI for mean difference: (-0.09012, -0.05243)

T-Test of mean difference = 0 (vs not = 0): T-Value = -7.73 P-Value = 0.000

Paired T-Test and CI: k2_6, k3_6

Paired T for k2_6 - k3_6

	N	Mean	StDev	SE Mean
k2_6	30	1.33859	0.02629	0.00480
k3_6	30	1.39584	0.02126	0.00388
Difference	30	-0.05725	0.03201	0.00584

95% CI for mean difference: (-0.06921, -0.04530)

T-Test of mean difference = 0 (vs not = 0): T-Value = -9.80 P-Value = 0.000

Paired T-Test and CI: k2_8, k3_8

Paired T for k2_8 - k3_8

	N	Mean	StDev	SE Mean
k2_8	30	1.02705	0.02058	0.00376
k3_8	30	1.07391	0.02078	0.00379
Difference	30	-0.04686	0.03150	0.00575

95% CI for mean difference: (-0.05863, -0.03510)

T-Test of mean difference = 0 (vs not = 0): T-Value = -8.15 P-Value = 0.000

Paired T-Test and CI: k2_10, k3_10

Paired T for k2_10 - k3_10

	N	Mean	StDev	SE Mean
k2_10	30	0.83770	0.01584	0.00289
k3_10	30	0.87696	0.01366	0.00249
Difference	30	-0.03926	0.02427	0.00443

95% CI for mean difference: (-0.04832, -0.03020)

T-Test of mean difference = 0 (vs not = 0): T-Value = -8.86 P-Value = 0.000

Paired T-Test and CI: k2_12, k3_12

Paired T for k2_12 - k3_12

	N	Mean	StDev	SE Mean
k2_12	30	0.71549	0.01223	0.00223
k3_12	30	0.75057	0.01311	0.00239
Difference	30	-0.03508	0.01473	0.00269

95% CI for mean difference: (-0.04058, -0.02958)

T-Test of mean difference = 0 (vs not = 0): T-Value = -13.05 P-Value = 0.000

Paired T-Test and CI: k2_14, k3_14

Paired T for k2_14 - k3_14

	N	Mean	StDev	SE Mean
k2_14	30	0.62930	0.00918	0.00168
k3_14	30	0.67007	0.00926	0.00169
Difference	30	-0.04077	0.01210	0.00221

95% CI for mean difference: (-0.04529, -0.03625)

T-Test of mean difference = 0 (vs not = 0): T-Value = -18.45 P-Value = 0.000

Paired T-Test and CI: k2_16, k3_16

Paired T for k2_16 - k3_16

	N	Mean	StDev	SE Mean
k2_16	30	0.57152	0.00986	0.00180
k3_16	30	0.62445	0.00880	0.00161
Difference	30	-0.05293	0.01062	0.00194

95% CI for mean difference: (-0.05689, -0.04896)

T-Test of mean difference = 0 (vs not = 0): T-Value = -27.31 P-Value = 0.000

Paired T-Test and CI: k2_18, k3_18

Paired T for k2_18 - k3_18

	N	Mean	StDev	SE Mean
k2_18	30	0.53021	0.00883	0.00161
k3_18	30	0.59471	0.00860	0.00157

Difference 30 -0.06450 0.01369 0.00250

95% CI for mean difference: (-0.06961, -0.05938)

T-Test of mean difference = 0 (vs not = 0): T-Value = -25.80 P-Value = 0.000

Paired T-Test and CI: k2_20, k3_20

Paired T for k2_20 - k3_20

	N	Mean	StDev	SE Mean
k2_20	30	0.50310	0.00814	0.00149
k3_20	30	0.56640	0.00886	0.00162
Difference	30	-0.06329	0.01218	0.00222

95% CI for mean difference: (-0.06784, -0.05874)

T-Test of mean difference = 0 (vs not = 0): T-Value = -28.45 P-Value = 0.000

Paired T-Test and CI: k2_22, k3_22

Paired T for k2_22 - k3_22

	N	Mean	StDev	SE Mean
k2_22	30	0.48330	0.00708	0.00129
k3_22	30	0.54409	0.00799	0.00146
Difference	30	-0.06079	0.01193	0.00218

95% CI for mean difference: (-0.06525, -0.05634)

T-Test of mean difference = 0 (vs not = 0): T-Value = -27.90 P-Value = 0.000

Paired T-Test and CI: k2_24, k3_24

Paired T for k2_24 - k3_24

	N	Mean	StDev	SE Mean
k2_24	30	0.46783	0.00645	0.00118
k3_24	30	0.52103	0.00987	0.00180
Difference	30	-0.05320	0.01060	0.00194

95% CI for mean difference: (-0.05716, -0.04924)

T-Test of mean difference = 0 (vs not = 0): T-Value = -27.49 P-Value = 0.000

Paired T-Test and CI: k2_26, k3_26

Paired T for k2_26 - k3_26

	N	Mean	StDev	SE Mean
k2_26	30	0.45887	0.00563	0.00103
k3_26	30	0.50086	0.00943	0.00172
Difference	30	-0.04199	0.00971	0.00177

95% CI for mean difference: (-0.04562, -0.03837)

T-Test of mean difference = 0 (vs not = 0): T-Value = -23.70 P-Value = 0.000

Paired T-Test and CI: k2_2, k4_2

Paired T for k2_2 - k4_2

	N	Mean	StDev	SE Mean
k2_2	30	3.4129	0.0567	0.0104
k4_2	30	3.6913	0.0656	0.0120
Difference	30	-0.2784	0.0860	0.0157

95% CI for mean difference: (-0.3105, -0.2463)

T-Test of mean difference = 0 (vs not = 0): T-Value = -17.74 P-Value = 0.000

Paired T-Test and CI: k2_4, k4_4

Paired T for k2_4 - k4_4

	N	Mean	StDev	SE Mean
k2_4	30	1.94832	0.03748	0.00684
k4_4	30	2.12875	0.03945	0.00720
Difference	30	-0.1804	0.0559	0.0102

95% CI for mean difference: (-0.2013, -0.1596)

T-Test of mean difference = 0 (vs not = 0): T-Value = -17.68 P-Value = 0.000

Paired T-Test and CI: k2_6, k4_6

Paired T for k2_6 - k4_6

	N	Mean	StDev	SE Mean
k2_6	30	1.33859	0.02629	0.00480
k4_6	30	1.45707	0.03341	0.00610
Difference	30	-0.11848	0.04040	0.00738

95% CI for mean difference: (-0.13357, -0.10339)

T-Test of mean difference = 0 (vs not = 0): T-Value = -16.06 P-Value = 0.000

Paired T-Test and CI: k2_8, k4_8

Paired T for k2_8 - k4_8

	N	Mean	StDev	SE Mean
k2_8	30	1.02705	0.02058	0.00376
k4_8	30	1.12108	0.02084	0.00380
Difference	30	-0.09404	0.02760	0.00504

95% CI for mean difference: (-0.10434, -0.08373)

T-Test of mean difference = 0 (vs not = 0): T-Value = -18.66 P-Value = 0.000

Paired T-Test and CI: k2_10, k4_10

Paired T for k2_10 - k4_10

	N	Mean	StDev	SE Mean
k2_10	30	0.83770	0.01584	0.00289
k4_10	30	0.93095	0.01252	0.00229
Difference	30	-0.09325	0.02022	0.00369

95% CI for mean difference: (-0.10080, -0.08570)

T-Test of mean difference = 0 (vs not = 0): T-Value = -25.26 P-Value = 0.000

Paired T-Test and CI: k2_12, k4_12

Paired T for k2_12 - k4_12

	N	Mean	StDev	SE Mean
k2_12	30	0.71549	0.01223	0.00223
k4_12	30	0.80553	0.01237	0.00226
Difference	30	-0.09004	0.01818	0.00332

95% CI for mean difference: (-0.09683, -0.08325)

T-Test of mean difference = 0 (vs not = 0): T-Value = -27.12 P-Value = 0.000

Paired T-Test and CI: k2_14, k4_14

Paired T for k2_14 - k4_14

	N	Mean	StDev	SE Mean
k2_14	30	0.62930	0.00918	0.00168
k4_14	30	0.71898	0.00821	0.00150
Difference	30	-0.08968	0.01321	0.00241

95% CI for mean difference: (-0.09461, -0.08475)

T-Test of mean difference = 0 (vs not = 0): T-Value = -37.19 P-Value = 0.000

Paired T-Test and CI: k2_16, k4_16

Paired T for k2_16 - k4_16

	N	Mean	StDev	SE Mean
k2_16	30	0.57152	0.00986	0.00180
k4_16	30	0.65889	0.00724	0.00132
Difference	30	-0.08737	0.01033	0.00189

95% CI for mean difference: (-0.09122, -0.08351)

T-Test of mean difference = 0 (vs not = 0): T-Value = -46.34 P-Value = 0.000

Paired T-Test and CI: k2_18, k4_18

Paired T for k2_18 - k4_18

	N	Mean	StDev	SE Mean
k2_18	30	0.53021	0.00883	0.00161

k4_18	30	0.62607	0.00570	0.00104
Difference	30	-0.09586	0.00926	0.00169

95% CI for mean difference: (-0.09932, -0.09240)

T-Test of mean difference = 0 (vs not = 0): T-Value = -56.68 P-Value = 0.000

Paired T-Test and CI: k2_20, k4_20

Paired T for k2_20 - k4_20

	N	Mean	StDev	SE Mean
k2_20	30	0.50310	0.00814	0.00149
k4_20	30	0.59798	0.00856	0.00156
Difference	30	-0.09487	0.00963	0.00176

95% CI for mean difference: (-0.09847, -0.09128)

T-Test of mean difference = 0 (vs not = 0): T-Value = -53.94 P-Value = 0.000

Paired T-Test and CI: k2_22, k4_22

Paired T for k2_22 - k4_22

	N	Mean	StDev	SE Mean
k2_22	30	0.48330	0.00708	0.00129
k4_22	30	0.57543	0.00876	0.00160
Difference	30	-0.09213	0.01231	0.00225

95% CI for mean difference: (-0.09673, -0.08753)

T-Test of mean difference = 0 (vs not = 0): T-Value = -40.99 P-Value = 0.000

Paired T-Test and CI: k2_24, k4_24

Paired T for k2_24 - k4_24

	N	Mean	StDev	SE Mean
k2_24	30	0.46783	0.00645	0.00118
k4_24	30	0.54065	0.01192	0.00218
Difference	30	-0.07282	0.01339	0.00244

95% CI for mean difference: (-0.07782, -0.06782)

T-Test of mean difference = 0 (vs not = 0): T-Value = -29.80 P-Value = 0.000

Paired T-Test and CI: k2_26, k4_26

Paired T for k2_26 - k4_26

	N	Mean	StDev	SE Mean
k2_26	30	0.45887	0.00563	0.00103
k4_26	30	0.51819	0.01017	0.00186
Difference	30	-0.05933	0.01267	0.00231

95% CI for mean difference: (-0.06406, -0.05460)

T-Test of mean difference = 0 (vs not = 0): T-Value = -25.65 P-Value = 0.000

Paired T-Test and CI: k3_2, k4_2

Paired T for k3_2 - k4_2

	N	Mean	StDev	SE Mean
k3_2	30	3.5420	0.0792	0.0145
k4_2	30	3.6913	0.0656	0.0120
Difference	30	-0.1494	0.0925	0.0169

95% CI for mean difference: (-0.1839, -0.1148)

T-Test of mean difference = 0 (vs not = 0): T-Value = -8.85 P-Value = 0.000

Paired T-Test and CI: k3_4, k4_4

Paired T for k3_4 - k4_4

	N	Mean	StDev	SE Mean
k3_4	30	2.01959	0.04034	0.00736
k4_4	30	2.12875	0.03945	0.00720
Difference	30	-0.1092	0.0576	0.0105

95% CI for mean difference: (-0.1307, -0.0877)

T-Test of mean difference = 0 (vs not = 0): T-Value = -10.38 P-Value = 0.000

Paired T-Test and CI: k3_6, k4_6

Paired T for k3_6 - k4_6

	N	Mean	StDev	SE Mean
k3_6	30	1.39584	0.02126	0.00388
k4_6	30	1.45707	0.03341	0.00610
Difference	30	-0.06123	0.04232	0.00773

95% CI for mean difference: (-0.07703, -0.04543)

T-Test of mean difference = 0 (vs not = 0): T-Value = -7.92 P-Value = 0.000

Paired T-Test and CI: k3_8, k4_8

Paired T for k3_8 - k4_8

	N	Mean	StDev	SE Mean
k3_8	30	1.07391	0.02078	0.00379
k4_8	30	1.12108	0.02084	0.00380
Difference	30	-0.04717	0.03199	0.00584

95% CI for mean difference: (-0.05912, -0.03523)

T-Test of mean difference = 0 (vs not = 0): T-Value = -8.08 P-Value = 0.000

Paired T-Test and CI: k3_10, k4_10

Paired T for k3_10 - k4_10

	N	Mean	StDev	SE Mean
k3_10	30	0.87696	0.01366	0.00249
k4_10	30	0.93095	0.01252	0.00229
Difference	30	-0.05399	0.01991	0.00363

95% CI for mean difference: (-0.06142, -0.04656)

T-Test of mean difference = 0 (vs not = 0): T-Value = -14.86 P-Value = 0.000

Paired T-Test and CI: k3_12, k4_12

Paired T for k3_12 - k4_12

	N	Mean	StDev	SE Mean
k3_12	30	0.75057	0.01311	0.00239
k4_12	30	0.80553	0.01237	0.00226
Difference	30	-0.05496	0.01617	0.00295

95% CI for mean difference: (-0.06100, -0.04892)

T-Test of mean difference = 0 (vs not = 0): T-Value = -18.61 P-Value = 0.000

Paired T-Test and CI: k3_14, k4_14

Paired T for k3_14 - k4_14

	N	Mean	StDev	SE Mean
k3_14	30	0.67007	0.00926	0.00169
k4_14	30	0.71898	0.00821	0.00150
Difference	30	-0.04891	0.01307	0.00239

95% CI for mean difference: (-0.05379, -0.04403)

T-Test of mean difference = 0 (vs not = 0): T-Value = -20.50 P-Value = 0.000

Paired T-Test and CI: k3_16, k4_16

Paired T for k3_16 - k4_16

	N	Mean	StDev	SE Mean
k3_16	30	0.62445	0.00880	0.00161
k4_16	30	0.65889	0.00724	0.00132
Difference	30	-0.03444	0.01109	0.00203

95% CI for mean difference: (-0.03858, -0.03030)

T-Test of mean difference = 0 (vs not = 0): T-Value = -17.01 P-Value = 0.000

Paired T-Test and CI: k3_18, k4_18

Paired T for k3_18 - k4_18

	N	Mean	StDev	SE Mean
k3_18	30	0.59471	0.00860	0.00157

k4_18	30	0.62607	0.00570	0.00104
Difference	30	-0.03136	0.01050	0.00192

95% CI for mean difference: (-0.03528, -0.02744)

T-Test of mean difference = 0 (vs not = 0): T-Value = -16.36 P-Value = 0.000

Paired T-Test and CI: k3_20, k4_20

Paired T for k3_20 - k4_20

	N	Mean	StDev	SE Mean
k3_20	30	0.56640	0.00886	0.00162
k4_20	30	0.59798	0.00856	0.00156
Difference	30	-0.03158	0.01368	0.00250

95% CI for mean difference: (-0.03669, -0.02647)

T-Test of mean difference = 0 (vs not = 0): T-Value = -12.64 P-Value = 0.000

Paired T-Test and CI: k3_22, k4_22

Paired T for k3_22 - k4_22

	N	Mean	StDev	SE Mean
k3_22	30	0.54409	0.00799	0.00146
k4_22	30	0.57543	0.00876	0.00160
Difference	30	-0.03134	0.01151	0.00210

95% CI for mean difference: (-0.03564, -0.02704)

T-Test of mean difference = 0 (vs not = 0): T-Value = -14.91 P-Value = 0.000

Paired T-Test and CI: k3_24, k4_24

Paired T for k3_24 - k4_24

	N	Mean	StDev	SE Mean
k3_24	30	0.52103	0.00987	0.00180
k4_24	30	0.54065	0.01192	0.00218
Difference	30	-0.01962	0.01463	0.00267

95% CI for mean difference: (-0.02508, -0.01416)

T-Test of mean difference = 0 (vs not = 0): T-Value = -7.35 P-Value = 0.000

Paired T-Test and CI: k3_26, k4_26

Paired T for k3_26 - k4_26

	N	Mean	StDev	SE Mean
k3_26	30	0.50086	0.00943	0.00172
k4_26	30	0.51819	0.01017	0.00186
Difference	30	-0.01734	0.01428	0.00261

95% CI for mean difference: (-0.02267, -0.01200)

T-Test of mean difference = 0 (vs not = 0): T-Value = -6.65 P-Value = 0.000

Paired T-Tests for Waiting Time Analysis of Two Sided GridPick

Paired T-Test and CI: k2_2, k3_2

Paired T for k2_2 - k3_2

	N	Mean	StDev	SE Mean
k2_2	30	0.59780	0.02299	0.00420
k3_2	30	0.57108	0.01107	0.00202
Difference	30	0.02671	0.02409	0.00440

95% CI for mean difference: (0.01772, 0.03571)

T-Test of mean difference = 0 (vs not = 0): T-Value = 6.07 P-Value = 0.000

Paired T-Test and CI: k2_4, k3_4

Paired T for k2_4 - k3_4

	N	Mean	StDev	SE Mean
k2_4	30	0.90791	0.04042	0.00738
k3_4	30	0.62326	0.00861	0.00157
Difference	30	0.28465	0.03893	0.00711

95% CI for mean difference: (0.27011, 0.29918)

T-Test of mean difference = 0 (vs not = 0): T-Value = 40.05 P-Value = 0.000

Paired T-Test and CI: k2_6, k3_6

Paired T for k2_6 - k3_6

	N	Mean	StDev	SE Mean
k2_6	30	1.09605	0.02945	0.00538
k3_6	30	0.70445	0.01614	0.00295
Difference	30	0.39161	0.03693	0.00674

95% CI for mean difference: (0.37782, 0.40540)

T-Test of mean difference = 0 (vs not = 0): T-Value = 58.07 P-Value = 0.000

Paired T-Test and CI: k2_8, k3_8

Paired T for k2_8 - k3_8

	N	Mean	StDev	SE Mean
k2_8	30	1.20218	0.03974	0.00726
k3_8	30	0.76430	0.01946	0.00355
Difference	30	0.43788	0.03704	0.00676

95% CI for mean difference: (0.42405, 0.45172)

T-Test of mean difference = 0 (vs not = 0): T-Value = 64.74 P-Value = 0.000

Paired T-Test and CI: k2_10, k3_10

Paired T for k2_10 - k3_10

	N	Mean	StDev	SE Mean
k2_10	30	1.25317	0.03459	0.00631
k3_10	30	0.80372	0.02061	0.00376
Difference	30	0.44946	0.03154	0.00576

95% CI for mean difference: (0.43768, 0.46123)

T-Test of mean difference = 0 (vs not = 0): T-Value = 78.06 P-Value = 0.000

Paired T-Test and CI: k2_12, k3_12

Paired T for k2_12 - k3_12

	N	Mean	StDev	SE Mean
k2_12	30	1.29856	0.03623	0.00662
k3_12	30	0.83880	0.01342	0.00245
Difference	30	0.45975	0.03709	0.00677

95% CI for mean difference: (0.44590, 0.47360)

T-Test of mean difference = 0 (vs not = 0): T-Value = 67.89 P-Value = 0.000

Paired T-Test and CI: k2_14, k3_14

Paired T for k2_14 - k3_14

	N	Mean	StDev	SE Mean
k2_14	30	1.33224	0.03733	0.00682
k3_14	30	0.88443	0.01484	0.00271
Difference	30	0.44780	0.04144	0.00757

95% CI for mean difference: (0.43233, 0.46327)

T-Test of mean difference = 0 (vs not = 0): T-Value = 59.19 P-Value = 0.000

Paired T-Test and CI: k2_16, k3_16

Paired T for k2_16 - k3_16

	N	Mean	StDev	SE Mean
k2_16	30	1.36972	0.03636	0.00664
k3_16	30	0.93811	0.01677	0.00306
Difference	30	0.43161	0.04386	0.00801

95% CI for mean difference: (0.41523, 0.44799)

T-Test of mean difference = 0 (vs not = 0): T-Value = 53.89 P-Value = 0.000

Paired T-Test and CI: k2_18, k3_18

Paired T for k2_18 - k3_18

	N	Mean	StDev	SE Mean
k2_18	30	1.39741	0.03296	0.00602
k3_18	30	1.00080	0.01889	0.00345

Difference 30 0.39661 0.03866 0.00706

95% CI for mean difference: (0.38217, 0.41105)

T-Test of mean difference = 0 (vs not = 0): T-Value = 56.19 P-Value = 0.000

Paired T-Test and CI: k2_20, k3_20

Paired T for k2_20 - k3_20

	N	Mean	StDev	SE Mean
k2_20	30	1.43438	0.03477	0.00635
k3_20	30	1.06153	0.02059	0.00376
Difference	30	0.37285	0.03434	0.00627

95% CI for mean difference: (0.36003, 0.38567)

T-Test of mean difference = 0 (vs not = 0): T-Value = 59.47 P-Value = 0.000

Paired T-Test and CI: k2_22, k3_22

Paired T for k2_22 - k3_22

	N	Mean	StDev	SE Mean
k2_22	30	1.45794	0.02901	0.00530
k3_22	30	1.11471	0.02180	0.00398
Difference	30	0.34323	0.03112	0.00568

95% CI for mean difference: (0.33161, 0.35485)

T-Test of mean difference = 0 (vs not = 0): T-Value = 60.40 P-Value = 0.000

Paired T-Test and CI: k2_24, k3_24

Paired T for k2_24 - k3_24

	N	Mean	StDev	SE Mean
k2_24	30	1.48791	0.03737	0.00682
k3_24	30	1.16180	0.01778	0.00325
Difference	30	0.32611	0.04094	0.00747

95% CI for mean difference: (0.31082, 0.34140)

T-Test of mean difference = 0 (vs not = 0): T-Value = 43.63 P-Value = 0.000

Paired T-Test and CI: k2_26, k3_26

Paired T for k2_26 - k3_26

	N	Mean	StDev	SE Mean
k2_26	30	1.52608	0.03357	0.00613
k3_26	30	1.20297	0.02118	0.00387
Difference	30	0.32311	0.03701	0.00676

95% CI for mean difference: (0.30929, 0.33693)

T-Test of mean difference = 0 (vs not = 0): T-Value = 47.82 P-Value = 0.000

Paired T-Test and CI: k2_2, k4_2

Paired T for k2_2 - k4_2

	N	Mean	StDev	SE Mean
k2_2	30	0.59780	0.02299	0.00420
k4_2	30	0.56000	0.01002	0.00183
Difference	30	0.03780	0.02636	0.00481

95% CI for mean difference: (0.02795, 0.04764)

T-Test of mean difference = 0 (vs not = 0): T-Value = 7.85 P-Value = 0.000

Paired T-Test and CI: k2_4, k4_4

Paired T for k2_4 - k4_4

	N	Mean	StDev	SE Mean
k2_4	30	0.90791	0.04042	0.00738
k4_4	30	0.60537	0.00947	0.00173
Difference	30	0.30254	0.04038	0.00737

95% CI for mean difference: (0.28746, 0.31762)

T-Test of mean difference = 0 (vs not = 0): T-Value = 41.04 P-Value = 0.000

Paired T-Test and CI: k2_6, k4_6

Paired T for k2_6 - k4_6

	N	Mean	StDev	SE Mean
k2_6	30	1.09605	0.02945	0.00538
k4_6	30	0.65558	0.01076	0.00196
Difference	30	0.44047	0.03499	0.00639

95% CI for mean difference: (0.42741, 0.45354)

T-Test of mean difference = 0 (vs not = 0): T-Value = 68.95 P-Value = 0.000

Paired T-Test and CI: k2_8, k4_8

Paired T for k2_8 - k4_8

	N	Mean	StDev	SE Mean
k2_8	30	1.20218	0.03974	0.00726
k4_8	30	0.70673	0.00760	0.00139
Difference	30	0.49546	0.03983	0.00727

95% CI for mean difference: (0.48058, 0.51033)

T-Test of mean difference = 0 (vs not = 0): T-Value = 68.13 P-Value = 0.000

Paired T-Test and CI: k2_10, k4_10

Paired T for k2_10 - k4_10

	N	Mean	StDev	SE Mean
k2_10	30	1.25317	0.03459	0.00631
k4_10	30	0.75409	0.00935	0.00171
Difference	30	0.49908	0.03720	0.00679

95% CI for mean difference: (0.48519, 0.51297)

T-Test of mean difference = 0 (vs not = 0): T-Value = 73.47 P-Value = 0.000

Paired T-Test and CI: k2_12, k4_12

Paired T for k2_12 - k4_12

	N	Mean	StDev	SE Mean
k2_12	30	1.29856	0.03623	0.00662
k4_12	30	0.79962	0.00986	0.00180
Difference	30	0.49894	0.03566	0.00651

95% CI for mean difference: (0.48562, 0.51226)

T-Test of mean difference = 0 (vs not = 0): T-Value = 76.63 P-Value = 0.000

Paired T-Test and CI: k2_14, k4_14

Paired T for k2_14 - k4_14

	N	Mean	StDev	SE Mean
k2_14	30	1.33224	0.03733	0.00682
k4_14	30	0.84124	0.00836	0.00153
Difference	30	0.49099	0.03865	0.00706

95% CI for mean difference: (0.47656, 0.50542)

T-Test of mean difference = 0 (vs not = 0): T-Value = 69.58 P-Value = 0.000

Paired T-Test and CI: k2_16, k4_16

Paired T for k2_16 - k4_16

	N	Mean	StDev	SE Mean
k2_16	30	1.36972	0.03636	0.00664
k4_16	30	0.89833	0.01036	0.00189
Difference	30	0.47139	0.03834	0.00700

95% CI for mean difference: (0.45707, 0.48570)

T-Test of mean difference = 0 (vs not = 0): T-Value = 67.34 P-Value = 0.000

Paired T-Test and CI: k2_18, k4_18

Paired T for k2_18 - k4_18

	N	Mean	StDev	SE Mean
k2_18	30	1.39741	0.03296	0.00602
k4_18	30	0.95951	0.01261	0.00230

Difference 30 0.43789 0.03396 0.00620

95% CI for mean difference: (0.42521, 0.45057)

T-Test of mean difference = 0 (vs not = 0): T-Value = 70.63 P-Value = 0.000

Paired T-Test and CI: k2_20, k4_20

Paired T for k2_20 - k4_20

	N	Mean	StDev	SE Mean
k2_20	30	1.43438	0.03477	0.00635
k4_20	30	1.01657	0.01389	0.00254
Difference	30	0.41781	0.03963	0.00724

95% CI for mean difference: (0.40301, 0.43261)

T-Test of mean difference = 0 (vs not = 0): T-Value = 57.74 P-Value = 0.000

Paired T-Test and CI: k2_22, k4_22

Paired T for k2_22 - k4_22

	N	Mean	StDev	SE Mean
k2_22	30	1.45794	0.02901	0.00530
k4_22	30	1.06977	0.01643	0.00300
Difference	30	0.38817	0.03242	0.00592

95% CI for mean difference: (0.37607, 0.40028)

T-Test of mean difference = 0 (vs not = 0): T-Value = 65.59 P-Value = 0.000

Paired T-Test and CI: k2_24, k4_24

Paired T for k2_24 - k4_24

	N	Mean	StDev	SE Mean
k2_24	30	1.48791	0.03737	0.00682
k4_24	30	1.11140	0.01929	0.00352
Difference	30	0.37652	0.03962	0.00723

95% CI for mean difference: (0.36172, 0.39131)

T-Test of mean difference = 0 (vs not = 0): T-Value = 52.05 P-Value = 0.000

Paired T-Test and CI: k2_26, k4_26

Paired T for k2_26 - k4_26

	N	Mean	StDev	SE Mean
k2_26	30	1.52608	0.03357	0.00613
k4_26	30	1.14327	0.01657	0.00303
Difference	30	0.38281	0.04020	0.00734

95% CI for mean difference: (0.36780, 0.39782)

T-Test of mean difference = 0 (vs not = 0): T-Value = 52.16 P-Value = 0.000

Paired T-Test and CI: k3_2, k4_2

Paired T for k3_2 - k4_2

	N	Mean	StDev	SE Mean
k3_2	30	0.57108	0.01107	0.00202
k4_2	30	0.56000	0.01002	0.00183
Difference	30	0.01108	0.01476	0.00269

95% CI for mean difference: (0.00557, 0.01659)

T-Test of mean difference = 0 (vs not = 0): T-Value = 4.11 P-Value = 0.000

Paired T-Test and CI: k3_4, k4_4

Paired T for k3_4 - k4_4

	N	Mean	StDev	SE Mean
k3_4	30	0.62326	0.00861	0.00157
k4_4	30	0.60537	0.00947	0.00173
Difference	30	0.01789	0.01391	0.00254

95% CI for mean difference: (0.01270, 0.02309)

T-Test of mean difference = 0 (vs not = 0): T-Value = 7.05 P-Value = 0.000

Paired T-Test and CI: k3_6, k4_6

Paired T for k3_6 - k4_6

	N	Mean	StDev	SE Mean
k3_6	30	0.70445	0.01614	0.00295
k4_6	30	0.65558	0.01076	0.00196
Difference	30	0.04886	0.01838	0.00336

95% CI for mean difference: (0.04200, 0.05573)

T-Test of mean difference = 0 (vs not = 0): T-Value = 14.56 P-Value = 0.000

Paired T-Test and CI: k3_8, k4_8

Paired T for k3_8 - k4_8

	N	Mean	StDev	SE Mean
k3_8	30	0.76430	0.01946	0.00355
k4_8	30	0.70673	0.00760	0.00139
Difference	30	0.05757	0.01987	0.00363

95% CI for mean difference: (0.05015, 0.06499)

T-Test of mean difference = 0 (vs not = 0): T-Value = 15.87 P-Value = 0.000

Paired T-Test and CI: k3_10, k4_10

Paired T for k3_10 - k4_10

	N	Mean	StDev	SE Mean
k3_10	30	0.80372	0.02061	0.00376
k4_10	30	0.75409	0.00935	0.00171
Difference	30	0.04963	0.02481	0.00453

95% CI for mean difference: (0.04036, 0.05889)

T-Test of mean difference = 0 (vs not = 0): T-Value = 10.95 P-Value = 0.000

Paired T-Test and CI: k3_12, k4_12

Paired T for k3_12 - k4_12

	N	Mean	StDev	SE Mean
k3_12	30	0.83880	0.01342	0.00245
k4_12	30	0.79962	0.00986	0.00180
Difference	30	0.03919	0.01463	0.00267

95% CI for mean difference: (0.03373, 0.04465)

T-Test of mean difference = 0 (vs not = 0): T-Value = 14.67 P-Value = 0.000

Paired T-Test and CI: k3_14, k4_14

Paired T for k3_14 - k4_14

	N	Mean	StDev	SE Mean
k3_14	30	0.88443	0.01484	0.00271
k4_14	30	0.84124	0.00836	0.00153
Difference	30	0.04319	0.01928	0.00352

95% CI for mean difference: (0.03599, 0.05039)

T-Test of mean difference = 0 (vs not = 0): T-Value = 12.27 P-Value = 0.000

Paired T-Test and CI: k3_16, k4_16

Paired T for k3_16 - k4_16

	N	Mean	StDev	SE Mean
k3_16	30	0.93811	0.01677	0.00306
k4_16	30	0.89833	0.01036	0.00189
Difference	30	0.03978	0.02166	0.00395

95% CI for mean difference: (0.03169, 0.04787)

T-Test of mean difference = 0 (vs not = 0): T-Value = 10.06 P-Value = 0.000

Paired T-Test and CI: k3_18, k4_18

Paired T for k3_18 - k4_18

	N	Mean	StDev	SE Mean
k3_18	30	1.00080	0.01889	0.00345

k4_18 30 0.95951 0.01261 0.00230
Difference 30 0.04128 0.01901 0.00347

95% CI for mean difference: (0.03418, 0.04838)
T-Test of mean difference = 0 (vs not = 0): T-Value = 11.90 P-Value = 0.000

Paired T-Test and CI: k3_20, k4_20

Paired T for k3_20 - k4_20

	N	Mean	StDev	SE Mean
k3_20	30	1.06153	0.02059	0.00376
k4_20	30	1.01657	0.01389	0.00254
Difference	30	0.04497	0.02547	0.00465

95% CI for mean difference: (0.03545, 0.05448)
T-Test of mean difference = 0 (vs not = 0): T-Value = 9.67 P-Value = 0.000

Paired T-Test and CI: k3_22, k4_22

Paired T for k3_22 - k4_22

	N	Mean	StDev	SE Mean
k3_22	30	1.11471	0.02180	0.00398
k4_22	30	1.06977	0.01643	0.00300
Difference	30	0.04494	0.02668	0.00487

95% CI for mean difference: (0.03498, 0.05491)
T-Test of mean difference = 0 (vs not = 0): T-Value = 9.23 P-Value = 0.000

Paired T-Test and CI: k3_24, k4_24

Paired T for k3_24 - k4_24

	N	Mean	StDev	SE Mean
k3_24	30	1.16180	0.01778	0.00325
k4_24	30	1.11140	0.01929	0.00352
Difference	30	0.05040	0.02629	0.00480

95% CI for mean difference: (0.04059, 0.06022)
T-Test of mean difference = 0 (vs not = 0): T-Value = 10.50 P-Value = 0.000

Paired T-Test and CI: k3_26, k4_26

Paired T for k3_26 - k4_26

	N	Mean	StDev	SE Mean
k3_26	30	1.20297	0.02118	0.00387
k4_26	30	1.14327	0.01657	0.00303
Difference	30	0.05970	0.02556	0.00467

95% CI for mean difference: (0.05016, 0.06924)
T-Test of mean difference = 0 (vs not = 0): T-Value = 12.79 P-Value = 0.000

Paired T-Tests for Flow Time Analysis of Two Sided GridPick

Paired T-Test and CI: k2_2, k3_2

Paired T for k2_2 - k3_2

	N	Mean	StDev	SE Mean
k2_2	30	8.8554	0.0905	0.0165
k3_2	30	9.2549	0.1227	0.0224
Difference	30	-0.3994	0.1495	0.0273

95% CI for mean difference: (-0.4553, -0.3436)

T-Test of mean difference = 0 (vs not = 0): T-Value = -14.63 P-Value = 0.000

Paired T-Test and CI: k2_4, k3_4

Paired T for k2_4 - k3_4

	N	Mean	StDev	SE Mean
k2_4	30	9.6373	0.1464	0.0267
k3_4	30	9.5073	0.0985	0.0180
Difference	30	0.1301	0.1497	0.0273

95% CI for mean difference: (0.0742, 0.1860)

T-Test of mean difference = 0 (vs not = 0): T-Value = 4.76 P-Value = 0.000

Paired T-Test and CI: k2_6, k3_6

Paired T for k2_6 - k3_6

	N	Mean	StDev	SE Mean
k2_6	30	11.1027	0.2074	0.0379
k3_6	30	10.4570	0.0985	0.0180
Difference	30	0.6457	0.2356	0.0430

95% CI for mean difference: (0.5577, 0.7337)

T-Test of mean difference = 0 (vs not = 0): T-Value = 15.01 P-Value = 0.000

Paired T-Test and CI: k2_8, k3_8

Paired T for k2_8 - k3_8

	N	Mean	StDev	SE Mean
k2_8	30	12.9113	0.2371	0.0433
k3_8	30	11.8248	0.1587	0.0290
Difference	30	1.0865	0.3057	0.0558

95% CI for mean difference: (0.9724, 1.2007)

T-Test of mean difference = 0 (vs not = 0): T-Value = 19.47 P-Value = 0.000

Paired T-Test and CI: k2_10, k3_10

Paired T for k2_10 - k3_10

	N	Mean	StDev	SE Mean
k2_10	30	14.9464	0.3529	0.0644
k3_10	30	13.3110	0.2021	0.0369
Difference	30	1.6353	0.4462	0.0815

95% CI for mean difference: (1.4687, 1.8019)

T-Test of mean difference = 0 (vs not = 0): T-Value = 20.08 P-Value = 0.000

Paired T-Test and CI: k2_12, k3_12

Paired T for k2_12 - k3_12

	N	Mean	StDev	SE Mean
k2_12	30	17.0255	0.4330	0.0790
k3_12	30	14.9656	0.2726	0.0498
Difference	30	2.060	0.598	0.109

95% CI for mean difference: (1.837, 2.283)

T-Test of mean difference = 0 (vs not = 0): T-Value = 18.87 P-Value = 0.000

Paired T-Test and CI: k2_14, k3_14

Paired T for k2_14 - k3_14

	N	Mean	StDev	SE Mean
k2_14	30	19.3070	0.4906	0.0896
k3_14	30	16.9712	0.3079	0.0562
Difference	30	2.3358	0.5469	0.0998

95% CI for mean difference: (2.1316, 2.5400)

T-Test of mean difference = 0 (vs not = 0): T-Value = 23.39 P-Value = 0.000

Paired T-Test and CI: k2_16, k3_16

Paired T for k2_16 - k3_16

	N	Mean	StDev	SE Mean
k2_16	30	21.802	0.598	0.109
k3_16	30	19.292	0.330	0.060
Difference	30	2.510	0.620	0.113

95% CI for mean difference: (2.278, 2.741)

T-Test of mean difference = 0 (vs not = 0): T-Value = 22.16 P-Value = 0.000

Paired T-Test and CI: k2_18, k3_18

Paired T for k2_18 - k3_18

	N	Mean	StDev	SE Mean
k2_18	30	24.4767	0.4664	0.0851

k3_18	30	21.7137	0.5168	0.0944
Difference	30	2.763	0.697	0.127

95% CI for mean difference: (2.503, 3.023)

T-Test of mean difference = 0 (vs not = 0): T-Value = 21.72 P-Value = 0.000

Paired T-Test and CI: k2_20, k3_20

Paired T for k2_20 - k3_20

	N	Mean	StDev	SE Mean
k2_20	30	26.725	0.616	0.112
k3_20	30	24.056	0.476	0.087
Difference	30	2.669	0.777	0.142

95% CI for mean difference: (2.379, 2.959)

T-Test of mean difference = 0 (vs not = 0): T-Value = 18.83 P-Value = 0.000

Paired T-Test and CI: k2_22, k3_22

Paired T for k2_22 - k3_22

	N	Mean	StDev	SE Mean
k2_22	30	29.029	0.606	0.111
k3_22	30	26.262	0.525	0.096
Difference	30	2.767	0.763	0.139

95% CI for mean difference: (2.482, 3.052)

T-Test of mean difference = 0 (vs not = 0): T-Value = 19.87 P-Value = 0.000

Paired T-Test and CI: k2_24, k3_24

Paired T for k2_24 - k3_24

	N	Mean	StDev	SE Mean
k2_24	30	31.097	0.599	0.109
k3_24	30	28.201	0.496	0.091
Difference	30	2.896	0.735	0.134

95% CI for mean difference: (2.622, 3.171)

T-Test of mean difference = 0 (vs not = 0): T-Value = 21.58 P-Value = 0.000

Paired T-Test and CI: k2_26, k3_26

Paired T for k2_26 - k3_26

	N	Mean	StDev	SE Mean
k2_26	30	32.7086	0.5453	0.0996
k3_26	30	29.9577	0.4692	0.0857
Difference	30	2.751	0.652	0.119

95% CI for mean difference: (2.507, 2.994)

T-Test of mean difference = 0 (vs not = 0): T-Value = 23.11 P-Value = 0.000

Paired T-Test and CI: k2_2, k4_2

Paired T for k2_2 - k4_2

	N	Mean	StDev	SE Mean
k2_2	30	8.8554	0.0905	0.0165
k4_2	30	9.5474	0.0981	0.0179
Difference	30	-0.6920	0.1490	0.0272

95% CI for mean difference: (-0.7476, -0.6363)

T-Test of mean difference = 0 (vs not = 0): T-Value = -25.44 P-Value = 0.000

Paired T-Test and CI: k2_4, k4_4

Paired T for k2_4 - k4_4

	N	Mean	StDev	SE Mean
k2_4	30	9.6373	0.1464	0.0267
k4_4	30	9.8493	0.1033	0.0189
Difference	30	-0.2119	0.1768	0.0323

95% CI for mean difference: (-0.2780, -0.1459)

T-Test of mean difference = 0 (vs not = 0): T-Value = -6.56 P-Value = 0.000

Paired T-Test and CI: k2_6, k4_6

Paired T for k2_6 - k4_6

	N	Mean	StDev	SE Mean
k2_6	30	11.1027	0.2074	0.0379
k4_6	30	10.8899	0.1429	0.0261
Difference	30	0.2127	0.2607	0.0476

95% CI for mean difference: (0.1154, 0.3101)

T-Test of mean difference = 0 (vs not = 0): T-Value = 4.47 P-Value = 0.000

Paired T-Test and CI: k2_8, k4_8

Paired T for k2_8 - k4_8

	N	Mean	StDev	SE Mean
k2_8	30	12.9113	0.2371	0.0433
k4_8	30	12.2487	0.1912	0.0349
Difference	30	0.6626	0.3032	0.0554

95% CI for mean difference: (0.5494, 0.7758)

T-Test of mean difference = 0 (vs not = 0): T-Value = 11.97 P-Value = 0.000

Paired T-Test and CI: k2_10, k4_10

Paired T for k2_10 - k4_10

	N	Mean	StDev	SE Mean
k2_10	30	14.9464	0.3529	0.0644
k4_10	30	13.8193	0.1428	0.0261
Difference	30	1.1271	0.3995	0.0729

95% CI for mean difference: (0.9779, 1.2763)

T-Test of mean difference = 0 (vs not = 0): T-Value = 15.45 P-Value = 0.000

Paired T-Test and CI: k2_12, k4_12

Paired T for k2_12 - k4_12

	N	Mean	StDev	SE Mean
k2_12	30	17.0255	0.4330	0.0790
k4_12	30	15.4762	0.2569	0.0469
Difference	30	1.549	0.551	0.101

95% CI for mean difference: (1.343, 1.755)

T-Test of mean difference = 0 (vs not = 0): T-Value = 15.39 P-Value = 0.000

Paired T-Test and CI: k2_14, k4_14

Paired T for k2_14 - k4_14

	N	Mean	StDev	SE Mean
k2_14	30	19.3070	0.4906	0.0896
k4_14	30	17.1632	0.2373	0.0433
Difference	30	2.144	0.550	0.100

95% CI for mean difference: (1.938, 2.349)

T-Test of mean difference = 0 (vs not = 0): T-Value = 21.33 P-Value = 0.000

Paired T-Test and CI: k2_16, k4_16

Paired T for k2_16 - k4_16

	N	Mean	StDev	SE Mean
k2_16	30	21.802	0.598	0.109
k4_16	30	19.238	0.435	0.079
Difference	30	2.563	0.644	0.117

95% CI for mean difference: (2.323, 2.804)

T-Test of mean difference = 0 (vs not = 0): T-Value = 21.82 P-Value = 0.000

Paired T-Test and CI: k2_18, k4_18

Paired T for k2_18 - k4_18

	N	Mean	StDev	SE Mean
k2_18	30	24.4767	0.4664	0.0851

k4_18	30	21.4976	0.3810	0.0696
Difference	30	2.979	0.599	0.109

95% CI for mean difference: (2.756, 3.203)

T-Test of mean difference = 0 (vs not = 0): T-Value = 27.25 P-Value = 0.000

Paired T-Test and CI: k2_20, k4_20

Paired T for k2_20 - k4_20

	N	Mean	StDev	SE Mean
k2_20	30	26.725	0.616	0.112
k4_20	30	23.632	0.477	0.087
Difference	30	3.093	0.642	0.117

95% CI for mean difference: (2.853, 3.333)

T-Test of mean difference = 0 (vs not = 0): T-Value = 26.40 P-Value = 0.000

Paired T-Test and CI: k2_22, k4_22

Paired T for k2_22 - k4_22

	N	Mean	StDev	SE Mean
k2_22	30	29.029	0.606	0.111
k4_22	30	25.725	0.613	0.112
Difference	30	3.304	0.741	0.135

95% CI for mean difference: (3.027, 3.580)

T-Test of mean difference = 0 (vs not = 0): T-Value = 24.41 P-Value = 0.000

Paired T-Test and CI: k2_24, k4_24

Paired T for k2_24 - k4_24

	N	Mean	StDev	SE Mean
k2_24	30	31.097	0.599	0.109
k4_24	30	27.836	0.574	0.105
Difference	30	3.261	0.864	0.158

95% CI for mean difference: (2.939, 3.584)

T-Test of mean difference = 0 (vs not = 0): T-Value = 20.68 P-Value = 0.000

Paired T-Test and CI: k2_26, k4_26

Paired T for k2_26 - k4_26

	N	Mean	StDev	SE Mean
k2_26	30	32.7086	0.5453	0.0996
k4_26	30	29.5559	0.4123	0.0753
Difference	30	3.153	0.777	0.142

95% CI for mean difference: (2.863, 3.443)

T-Test of mean difference = 0 (vs not = 0): T-Value = 22.23 P-Value = 0.000

Paired T-Test and CI: k3_2, k4_2

Paired T for k3_2 - k4_2

	N	Mean	StDev	SE Mean
k3_2	30	9.2549	0.1227	0.0224
k4_2	30	9.5474	0.0981	0.0179
Difference	30	-0.2925	0.1373	0.0251

95% CI for mean difference: (-0.3438, -0.2412)

T-Test of mean difference = 0 (vs not = 0): T-Value = -11.67 P-Value = 0.000

Paired T-Test and CI: k3_4, k4_4

Paired T for k3_4 - k4_4

	N	Mean	StDev	SE Mean
k3_4	30	9.5073	0.0985	0.0180
k4_4	30	9.8493	0.1033	0.0189
Difference	30	-0.3420	0.1057	0.0193

95% CI for mean difference: (-0.3815, -0.3025)

T-Test of mean difference = 0 (vs not = 0): T-Value = -17.72 P-Value = 0.000

Paired T-Test and CI: k3_6, k4_6

Paired T for k3_6 - k4_6

	N	Mean	StDev	SE Mean
k3_6	30	10.4570	0.0985	0.0180
k4_6	30	10.8899	0.1429	0.0261
Difference	30	-0.4330	0.1962	0.0358

95% CI for mean difference: (-0.5062, -0.3597)

T-Test of mean difference = 0 (vs not = 0): T-Value = -12.09 P-Value = 0.000

Paired T-Test and CI: k3_8, k4_8

Paired T for k3_8 - k4_8

	N	Mean	StDev	SE Mean
k3_8	30	11.8248	0.1587	0.0290
k4_8	30	12.2487	0.1912	0.0349
Difference	30	-0.4239	0.2127	0.0388

95% CI for mean difference: (-0.5034, -0.3445)

T-Test of mean difference = 0 (vs not = 0): T-Value = -10.92 P-Value = 0.000

Paired T-Test and CI: k3_10, k4_10

Paired T for k3_10 - k4_10

	N	Mean	StDev	SE Mean
k3_10	30	13.3110	0.2021	0.0369
k4_10	30	13.8193	0.1428	0.0261
Difference	30	-0.5082	0.2356	0.0430

95% CI for mean difference: (-0.5962, -0.4202)

T-Test of mean difference = 0 (vs not = 0): T-Value = -11.82 P-Value = 0.000

Paired T-Test and CI: k3_12, k4_12

Paired T for k3_12 - k4_12

	N	Mean	StDev	SE Mean
k3_12	30	14.9656	0.2726	0.0498
k4_12	30	15.4762	0.2569	0.0469
Difference	30	-0.5106	0.3563	0.0650

95% CI for mean difference: (-0.6436, -0.3775)

T-Test of mean difference = 0 (vs not = 0): T-Value = -7.85 P-Value = 0.000

Paired T-Test and CI: k3_14, k4_14

Paired T for k3_14 - k4_14

	N	Mean	StDev	SE Mean
k3_14	30	16.9712	0.3079	0.0562
k4_14	30	17.1632	0.2373	0.0433
Difference	30	-0.1920	0.3716	0.0678

95% CI for mean difference: (-0.3308, -0.0533)

T-Test of mean difference = 0 (vs not = 0): T-Value = -2.83 P-Value = 0.008

Paired T-Test and CI: k3_16, k4_16

Paired T for k3_16 - k4_16

	N	Mean	StDev	SE Mean
k3_16	30	19.2920	0.3305	0.0603
k4_16	30	19.2382	0.4349	0.0794
Difference	30	0.054	0.621	0.113

95% CI for mean difference: (-0.178, 0.286)

T-Test of mean difference = 0 (vs not = 0): T-Value = 0.47 P-Value = 0.639

Paired T-Test and CI: k3_18, k4_18

Paired T for k3_18 - k4_18

	N	Mean	StDev	SE Mean
k3_18	30	21.7137	0.5168	0.0944
k4_18	30	21.4976	0.3810	0.0696

Difference 30 0.216 0.646 0.118

95% CI for mean difference: (-0.025, 0.457)

T-Test of mean difference = 0 (vs not = 0): T-Value = 1.83 P-Value = 0.077

Paired T-Test and CI: k3_20, k4_20

Paired T for k3_20 - k4_20

	N	Mean	StDev	SE Mean
k3_20	30	24.0556	0.4762	0.0869
k4_20	30	23.6319	0.4768	0.0870
Difference	30	0.424	0.643	0.117

95% CI for mean difference: (0.184, 0.664)

T-Test of mean difference = 0 (vs not = 0): T-Value = 3.61 P-Value = 0.001

Paired T-Test and CI: k3_22, k4_22

Paired T for k3_22 - k4_22

	N	Mean	StDev	SE Mean
k3_22	30	26.262	0.525	0.096
k4_22	30	25.725	0.613	0.112
Difference	30	0.537	0.821	0.150

95% CI for mean difference: (0.230, 0.843)

T-Test of mean difference = 0 (vs not = 0): T-Value = 3.58 P-Value = 0.001

Paired T-Test and CI: k3_24, k4_24

Paired T for k3_24 - k4_24

	N	Mean	StDev	SE Mean
k3_24	30	28.201	0.496	0.091
k4_24	30	27.836	0.574	0.105
Difference	30	0.365	0.736	0.134

95% CI for mean difference: (0.090, 0.640)

T-Test of mean difference = 0 (vs not = 0): T-Value = 2.71 P-Value = 0.011

Paired T-Test and CI: k3_26, k4_26

Paired T for k3_26 - k4_26

	N	Mean	StDev	SE Mean
k3_26	30	29.9577	0.4692	0.0857
k4_26	30	29.5559	0.4123	0.0753
Difference	30	0.402	0.582	0.106

95% CI for mean difference: (0.184, 0.619)

T-Test of mean difference = 0 (vs not = 0): T-Value = 3.78 P-Value = 0.001

Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	%Mean
twoC-4	30	0.238525	0.004381	0.0008	(0.236320, 0.240730)	0.0022	0.92%
Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	%Mean
twoC-14	30	0.4646	0.003923	0.000716	(0.462626, 0.466574)	0.002	0.425%
Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	%Mean
twoC-24	30	0.464042	0.003673	0.000671	(0.462193, 0.465890)	0.0018	0.40%
Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	%Mean
threeC-4	30	0.16548	0.0057	0.00104	(0.16262, 0.16835)	0.0029	1.73%
Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	%Mean
threeC-14	30	0.39621	0.00724	0.00132	(0.39256, 0.39985)	0.00364	0.92%
Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	%Mean
threeC-24	30	0.455117	0.004885	0.000892	(0.452658, 0.457575)	0.0025	0.54%

Table A.4: Statistical Data for the Multi Copy Configurations of One Sided GridPick

throughput	Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	%Mean
	k3-4	30	0.58971	0.00707	0.00129	(0.58615, 0.59327)	0.00356	0.60%
throughput	Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	%Mean
	k3-14	30	0.86712	0.00608	0.00111	(0.86406, 0.87019)	0.00307	0.354%
throughput	Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	%Mean
	k3-24	30	0.82238	0.00656	0.0012	(0.81908, 0.82569)	0.00331	0.4025%
walking	Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	%Mean
	k3-4	30	2.01959	0.04034	0.00736	(1.99929, 2.03989)	0.0203	1.01%
walking	Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	%Mean
	k3-14	30	0.67007	0.00926	0.00169	(0.66541, 0.67473)	0.00466	0.70%
walking	Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	%Mean
	k3-24	30	0.52103	0.00987	0.0018	(0.51606, 0.52599)	0.00496	0.95%
waiting	Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	%Mean
	k3-4	30	0.62326	0.00861	0.00157	(0.61893, 0.62759)	0.00433	0.69%
waiting	Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	%Mean
	k3-14	30	0.88443	0.01484	0.00271	(0.87697, 0.89190)	0.00747	0.844%
waiting	Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	%Mean
	k3-24	30	1.1618	0.01778	0.00325	(1.15285, 1.17075)	0.00895	0.77%
flow	Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	%Mean
	k3-4	30	9.5073	0.0985	0.018	(9.4577, 9.5568)	0.0495	0.52%
flow	Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	%Mean
	k3-14	30	16.9712	0.3079	0.0562	(16.8162, 17.1261)	0.1549	0.91%
flow	Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	%Mean
	k3-24	30	28.2008	0.4958	0.0905	(27.9513, 28.4503)	0.2495	0.88%

Table A.5: Statistical Data for the Throughput of Two Sided GridPick

Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	%Mean
0.5-4	30	0.32997	0.00673	0.00123	(0.32658, 0.33336)	0.00339	1.03%
Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	%Mean
1-4	30	0.49481	0.00806	0.00147	(0.49075, 0.49887)	0.00406	0.82%
Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	%Mean
1-14	30	0.78439	0.0077	0.00141	(0.78052, 0.78827)	0.0039	0.49%
Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	%Mean
8-14	30	0.78484	0.00553	0.00101	(0.78206, 0.78762)	0.00278	0.35%
Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	%Mean
8-24	30	0.886667	0.00519	0.000947	(0.884055, 0.889278)	0.0026	0.29%

Table A.6: Statistical Data for the Aspect Ratio Configurations of Two Sided GridPick

Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	%Mean
aedges-4	30	0.57165	0.00625	0.00114	(0.56851, 0.57479)	0.0031	0.55%
Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	%Mean
aedges-14	30	0.84426	0.00785	0.00143	(0.84031, 0.84821)	0.004	0.47%
Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	%Mean
aedges-24	30	0.83459	0.00606	0.00111	(0.83154, 0.83764)	0.00305	0.37%

Table A.7: Statistical Data for the Variable k Configurations of Two Sided GridPick

Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	%Mean
w1-4	30	0.289017	0.005042	0.00092	(0.286480, 0.291554)	0.002537	0.88%
Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	%Mean
w1-14	30	0.43895	0.002542	0.000464	(0.437671, 0.440229)	0.0013	0.29%
Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	%Mean
w1-24	30	0.420525	0.004904	0.000895	(0.418057, 0.422993)	0.0025	0.59%
Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	%Mean
w2-4	30	0.285275	0.003877	0.000708	(0.283324, 0.287226)	0.002	0.68%
Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	%Mean
w2-14	30	0.426883	0.002909	0.000531	(0.425420, 0.428347)	0.0015	0.34%
Variable	N	Mean	StDev	SE Mean	99% CI	Halfwidth	%Mean
w2-24	30	0.412442	0.004344	0.000793	(0.410255, 0.414628)	0.0022	0.53%

Table A.8: Statistical Data for the Two Workers Comparison of Two Sided GridPick