9-1-2008

Development of deterministic collision-avoidance algorithms for routing automated guided vehicles

Arun S. Pai

Rochester Institute of Technology

Development of Deterministic Collision-avoidance Algorithms for Routing Automated Guided Vehicles

A Thesis

Submitted in partial fulfillment of the

requirements for the degree of

Master of Science in Industrial Engineering

in the

Department of Industrial & Systems Engineering

Kate Gleason College of Engineering

by

Arun S. Pai

M.S., Mechanical Engineering, Rochester Institute of Technology, 2002

September 2008

DEPARTMENT OF INDUSTRIAL AND SYSTEMS ENGINEERING

KATE GLEASON COLLEGE OF ENGINEERING

ROCHESTER INSTITUTE OF TECHNOLOGY

ROCHESTER, NEW YORK

CERTIFICATE OF APPROVAL

M.S. DEGREE THESIS

The M.S. Degree Thesis of Arun S. Pai

has been examined and approved by the

thesis committee as satisfactory for the

thesis requirement for the

Master of Science degree

Approved by:

_____

Dr. Michael E. Kuhl, Thesis Advisor

_____

Dr. Moises Sudit, Thesis Advisor

Table of Contents

Abstract:

A manufacturing job spends a small portion of its total flow time being processed on machines, and during the remaining time, either it is in a queue or being transported from one work center to another. In a fully automated material-handling environment, automated guided vehicles (AGV) perform the function of transporting the jobs between workstations, and high operational costs are involved in these material-handling activities. Consequently, the AGV route schedule dictates subsequent work-center scheduling.

For an AGV job transportation schedule to be effective, the issue of collisions amongst AGV during travel needs to be addressed. Such collisions cause stalemate situations that potentially disrupt the flow of materials in the job shop, adding to the non-value time of job processing, and thus, increase the material handling and inventory holding costs. The current research goal was to develop a methodology that could effectively and efficiently derive optimal AGV routes for a given set of transportation requests, considering the issue of collisions amongst AGV during travel.

As part of the solution approach in the proposed work, an integer linear program was formulated in Phase I with the capability of optimally predicting the AGV routes for a deterministic set of transportation requests. Collision avoidance constraints were developed in this model. The model was programmed using OPL / Visual Basic, and the program feasibility were experimentally analyzed for different problem domain specifications. Due to the complexity and combinatorial nature of the formulation in Phase I, computationally it was expected to be NP-Hard. Hence, to improve the computation prediction capability (estimation of upper bounds), it was required that in Phase II, heuristics be developed to relax the computational complexity of the original

problem. In Phase III, experimental techniques were used to compute the lower and upper bounds of the original problem. The performances of the different heuristics were compared using experimental analysis.

1 Introduction and Problem Motivation

1.1 Introduction

Material handling is an important aspect of any production system. Material handling systems have been prevalent since the beginning of mass production, either as manual systems, mechanical systems (forklifts, conveyors), or in more recent years as fully automated systems (Automated Guided Vehicle Systems [AGVS] or Automatic Storage & Retrieval Systems [AS/RS]). Due to the high operational costs attributed to material handling activities, for years organizations have been looking for ways to minimize the time spent on material handling, and for ways to optimize material handling operations. With technological advancements in use of automated equipment such as conveyors and automated guided vehicles (AGV), companies now have improved material handling alternatives. Particularly, due to the routing flexibility associated with AGVs, their applications have spanned from use in distribution centers, warehouses, and terminals to large-scale manufacturing in assembly facilities.

The following sections in the chapter present an overview of an AGV system, discuss the advantages of using such a system, outline the important decision variables in an AGVS design, and finally detail the motivation for pursuing this research work.

1.2 Description of an AGVS

An AGVS is an advanced material-handling system that involves one or more driverless vehicles, each following a physical or virtual guide path under the control of a computer. Two primary control systems are in use today. In the first type, the control lies in a simple central computer inside the vehicle itself. In the second type of control, the vehicles have minimal intelligence and an off-board computer controls the vehicles. The different components of an AGVS include:

- The vehicle that consists of the frame, batteries, on-board charging unit, electrical system, drive unit, steering, safety system, communication unit and the work platform;
- The guide path and guidance systems; and
- The floor and system controls.

1.2.1 Advantages of an AGVS

The advantages of an AGVS include reliability, automatic operation, flexibility in adapting to changes in material flow, reduced labor and increased productivity, and automated interfaces with other systems. Unlike conveyors or other material handling systems, AGV are small in size and only move along the aisles. Hence, an AGVS offers additional benefits of reduced space requirements. The real-time control of material handling that the AGVS offers helps in identification of the parts, the routes they travel and the vehicles they travel in, resulting in a lower WIP inventory, reduced tardiness, lower inventory costs and better response to demands (Hammond, 1986). Thus, an AGVS can improve the working environment, reduces product damage, and provides better inventory control and quality.

1.2.2 Critical Variables in an AGVS design

The system performance of an AGVS directly affects the performance of the whole facility. To realize an AGVS full potential for flexibility, careful planning and control of the system design and operation is essential. The enormity in design and hardware requirements of an AGVS necessitates a number of variables to be considered at each level of the decision-making process. The relevant issues can be divided into the following main categories: guide-path design, estimating the required number of vehicles, vehicle scheduling, idle-vehicle positioning, battery management, vehicle routing, and conflict resolution. These issues relate to different levels of the decision-making process.

The guide-path design problem can be seen as a problem at the strategic level. The decision at this stage has a strong effect on decisions at other levels. Issues at the tactical level include estimating the number of vehicles, scheduling vehicle, positioning idle vehicles, and managing battery-charging scheme. Finally, the vehicle routing and conflict resolution problems are addressed at the operational level. During the design, implementation, and control of an AGVS, interactions and iterations between the different decision levels should be accounted for. An overview of these decision levels is outlined in Table 1.

Table 1: Important Decision-Variables in an AGVS Design

No.	Decision level	Type of decisions
1	Process focus	Type and number of vehicles to be present in a system.
2	Equipment considerations	Type of vehicle steering control, routing method, traffic management, load transfer mechanisms at load and unload points; and monitoring of the AGVS.
3	Facility considerations	Optimal number of workstations, optimal number of machines per workstation, optimal number of buffers; and optimal number of kits.
4	Workstation considerations	Workstation layout; and even distribution of processing time over all the stations in the system.
5	Task-related considerations	Transportation times based on considerations of number of job types, lot size descriptions and intensity of flow collisions.
6	Travel-related considerations	Type of flow path, track layout, zoning considerations, dedication of the AGV, staging areas, load and unload times; and travel speeds of the AGVs.
7	Schedule-related considerations	Job dispatching rules, sequencing of moves requests, and intersection problems.

1.3 Problem Motivation

The motivation for this work is derived from insight into two separate yet similar considerations – the primary motivation is the lack of a comprehensive methodology for collision prevention and optimality in an AGV system, while the secondary motivation is glitch in handling AGV deadlock situations in a commercial simulation software (ARENA) used for modeling of AGV systems. The motivations are discussed in detail in the following sections.

1.3.1 Primary Motivation: Methodology for Collision Prevention and Optimality

Of all the decision variables summarized in Table 1, operational issues related to vehicle travel and scheduling have been the major focus areas of research in recent times. This is because of the mathematical complexities involved in their optimization.

AGV routing decisions are iteratively dependant on effective flow path designs. So, AGV flow path decision considerations have been the major focus of investigations. As the AGV travel layouts improved, focus shifted towards developing vehicle routing algorithms that could conclusively address the issue of prevention of collisions amongst AGV during travel, and effectively generating optimal vehicle routes relative to minimizing transportation time. The rule-based strategies that have been developed since are not able to provide near-global optimal solutions in computationally efficient times. Moreover, these techniques can be applied effectively only to small-scale problem domains. Therefore, there is room for improvement in AGV routing methodologies that include vehicle collision prevention, thus, stimulating the need for a comprehensive and complete model for solving the problem. This is the primary motivation for this study.

1.3.2 Secondary Motivation: Commercial Simulation Software for Modeling of AGV systems

Current commercial (off the shelf) simulation software such as ARENA can model material handling systems. There is however a glitch in using such packages for AGV routing. Although ARENA uses a collision detection strategy in such situations, studies indicate that if proper control is not specified for the transporter, the software terminates on detection of a collision or deadlock situation amongst vehicles. To illustrate this, a case study is discussed in the following sub-section.

The results of the study (outlined in Appendix A.1) were a secondary motivation for the author to conduct research in AGV routing. It is hoped that from this work an optimal AGV routing methodology with collision prevention can be developed, that could be integrated with simulation systems such as ARENA in future research.

1.4 Outline of the Thesis Work

This thesis document is organized as follows: Chapter 2 contains the Problem Statement. Chapter 3 is a thorough review of the relevant literature. Chapter 4 discusses the scope of work and the approach used to solve the problem (model formulation and the design of the different heuristics). Chapter 5 summarizes the results inclusive of optimality runs, experimental designs and statistical analysis. The conclusions of the research work are presented in Chapter 6.

2 Problem Statement

A manufacturing job can spend on an average only 5% of its total flow time being processed on machines, and during the remaining time, either it is in a queue or being transported from one work center to another (Han et al., 1989). In a fully automated material-handling environment, AGVs perform the function of transporting the jobs between workstations, and high operational costs are involved in these material-handling activities. AGV scheduling influences subsequent work-center scheduling, and dictates the overall production schedule. Therefore, there is a potential loss of overall system performance, and an increase in the material handling and inventory holding costs, if the AGV routing is not effective.

In an AGV routing system based on a deterministic approach, all tasks are known prior to the planning period. The complete AGV routes can be developed before vehicles carry them out. Such a problem is similar to the pick-up and delivery problem with time windows (PDPTW), which often has travel time minimization or minimization of the number of vehicles as objectives. The PDPTW problem is known to be NP-hard (Dumas et al., 1991), so it is infeasible to develop an algorithm to solve this type of problem in polynomial time. Due to this reason, heuristics are the most appropriate approach to cope with this type of problem. Past researches like Gaskins et al. (1987); Savelsbergh et al. (1995); Seo et al. (1995); Bilge et al. (1995) and Ulusoy et al. (1997) have focused on developing approaches that identify imminent collisions through forward sensing and their aversion through vehicle backtracking or rerouting.

In summary, current approaches are based on forward sensing instead of a collision prevention scheme. They employ the concept of shortest travel distance, thereby arriving at schedules that are likely far away from an optimal route. A better approach that can predict near

optimal routes should include constraints that totally prevent AGV collisions in the route scheduling model. Collisions cause stalemate situations that potentially disrupt the flow of materials in the job shop, adding to the non-value time of job processing, and thus, increase the material handling and inventory holding costs.

Hence, the research goal is to develop a methodology that can derive optimal AGV routes for a given set of transportation requests, considering the issue of collisions amongst AGV during travel. Secondly, computationally the program should solve problems in "practical feasible time" (based on application). Developing a methodology that can effectively and efficiently address these issues is the overall goal of this work.

The problem statement can be summarized as developing a route planning system that:

- Works effectively for any type of AGV guide path layout;
- Can optimally derive vehicle routes for a deterministic set of transportation requests;
- Accounts for collision avoidance amongst vehicles during travel; and,
- Arrives at solutions in a practically feasible (based on application) computation time.

As part of the solution approach in the proposed work, an integer linear program is formulated in Phase I with the capability of optimally predicting the AGV routes for a deterministic set of transportation requests. Collision avoidance constraints are developed in this model. The model is programmed using Visual Basic / CPLEX 9.0 / OPL 3.7, and the program feasibility is experimentally analyzed for different problem domain specifications. Due to the complexity and combinatorial nature of the formulation in Phase I, computationally the mathematical model is expected to be NP-Hard. Hence, to improve the efficiency of the computation, in Phase II, heuristics are developed to relax the computational complexity of the original problem. In Phase III, experimental techniques are used to compute the lower and upper

bounds of the original problem. The performances of the different heuristics are compared using experimental analysis.

3 Literature Review

This chapter presents a review of available AGV literature in the areas of vehicle dispatching, vehicle routing and traffic control.

3.1 AGV Routing Based on Guide Path Layout

The following sub-sections detail the available literature for the different types of guide paths used in AGV systems.

3.1.1 Uni-Directional Guide Path

The earliest research on AGVS modeling in a warehouse / manufacturing set-up addressed process and travel-related issues such as vehicle fleet sizing and uni-directional guide path layout. The primary focus was to maximize space utilization, and since in such a set-up, most of the floor space is taken by the storage facility, the AGV flow paths were restricted to a series of narrow aisles, and hence, it was mathematically and computationally infeasible to model a bi-directional multi-AGV routing layout. The seminal paper in this area was by Maxwell et al. (1982). This work computed the minimum number of AGVs required in a time-independent environment to efficiently transfer material from one facility to another. However, since time was essentially ignored, the authors assumed no congestion or blocking in the system. This paper led to many extensions and research in related issues and is cited extensively.

Gaskins et al. (1987); Hodgson et al. (1987); Kaspi et al. (1990); and Goetz et al. (1990) incorporated the time element in the AGVS modeling to determine the directional flow on a uni-directional AGV path. The objective of these studies was to minimize the total distance traveled, given a known set of requests between pair of locations. The assumptions made in these studies, however, restricted the robustness of the models. Gaskins et al. (1987) only considered the movement of loaded vehicles and assumed that the flow path movement of the AGV was

restricted to certain areas such as aisles. The model could not be generalized to include factors such as the travel of the unloaded vehicles, vehicle blocking and congestion. Similarly, Hodgson et al. (1987) used Markov decision processes to develop AGVS dispatching rules in a time-dependent environment. Such a procedure was however, increasingly difficult to model even for a simple AGVS due to the large number of states involved. This further necessitated several constraints to be set in for the semi-Markov problem to be tractable. Kaspi et al. (1990) solved the optimal flow path design problem in a uni-directional network using a branch and bound technique, while Goetz et al. (1990) developed an algorithm to minimize the total AGV travel distance in a uni-directional layout. It was observed that for larger problem sizes, both of the above models were difficult to solve.

Due to computational and mathematical complexities involved in the routing algorithms, in the mid-nineties; focus shifted towards iterative, phased approaches in algorithm development. Seo et al. (1995) were the initial researchers to design a two-stage method to solve the routing problem; wherein the first stage uses a binary integer program and considers the AGV loaded travel in its objective. The second stage takes into account the empty travel of the AGV, and is used only if an optimal closed solution is not obtained in the first stage. A branch and bound heuristic was used to solve the problem. The process was found effective to solve a problem with nine workstations. The method was computationally efficient, however, its efficiency in case of larger real world problems were not examined. Moreover, the model was not robust enough to be effectively scaled-up for bi-directional networks.

It was evident from the results of the above studies that even the most efficient uni-directional system would increase the travel times between some pair of locations. Hence, the very objective to minimize the total travel time for the AGV in a collision –free environment

would not be accomplished. This necessitated a need to research the possible advantages and complications in a bi-directional flow path.

3.1.2 Bi-Directional Guide Path

The earliest research on bi-directional AGVS modeling by Egbelu et al (1986) discussed the different types of bi-directional travel guide paths. The study was based on an assumption that all flow within an aisle can be in either direction; however, at any moment in time all flow within a single aisle is in the same direction. The results of the simulation study conducted by the authors demonstrated that a bi-directional system required fewer vehicles to achieve the same workload as compared to a uni-directional system over the same layout. Similarly, Zeng et al. (1991) investigated a less restrictive bi-directional option, in which vehicles were allowed to travel along the same aisle in opposite directions, as long as a collision situation is not detected. Their solution introduced a time element, and was based on an extension of the petri-net approach. However, their algorithm could not suggest an alternative route in cases where a collision is detected. So, the mid-nineties saw a shift in research focus towards collision detection. Krishnamurthy et al. (1993) used a column generation collision detection technique to develop a conflict free routing algorithm for AGV. They considered a bi-directional network and sub-divided the problem into a master problem and a sub problem. The authors attempted several empirical solution techniques to solve the sub problem and the link between the sub and the master problems. With technological advancements, AGV collision avoidance rather than collision detection became research goals for a comprehensive optimal routing system. Dowsland et al. (1994) were the initial researchers to focus on this concept. They formulated the AGV flow path as a graph network and considered the different cases where collision might

occur at a given node. They used delays and deviations along the spur of a node to determine the collision avoidance. The authors, however, recommended a need for future study into further validations of their results using a more comprehensive set of simulations. In the late nineties, Endo et al. (1998) proposed a petri-net approach to solve the motion-planning problem for multiple AGV. In a separate work, Endo et al. (2000) developed a genetic algorithm to tackle the collision avoidance problem. However, both these methods were applicable only to the small sized problems.

3.2 Optimization Methods for AGV Conflict resolution and Route Planning

It is difficult to manage an AGV system efficiently, and it is an issue in itself. This issue includes several sub issues such as AGV scheduling, idle vehicle positioning, vehicle routing and conflict resolution. In actuality, online scheduling and dispatching systems are much more popular than offline scheduling due to the stochastic nature of AGV systems, and perform better than simple dispatching systems (Yang et al., 1999). However, there is no secret that modeling them is much more complex and computationally cumbersome. In an offline (Pre-Planned) scheduling system, all tasks are known prior to the planning period. The complete AGV routes can be developed before vehicles carry them out. Past researches in this area like Gaskins et al. (1987); Savelsbergh et al. (1995); Seo et al. (1995); Bilge et al. (1995) and Ulusoy et al. (1997) have focused on developing approaches that identify imminent collisions through forward sensing and their aversion through vehicle backtracking or rerouting. In summary, current approaches are based on forward sensing instead of a collision prevention scheme. They employ the concept of shortest travel distance, thereby arriving at schedules that are necessarily far away

from an optimal route. A better approach that can predict near optimal routes should include constraints totally prevent AGV collisions in the route scheduling model.

Currently, conflict-free routing in AGV systems is established by means of one of the following three approaches: (i) the problem elimination through the adoption of a segmented path flow or tandem queue configuration (Egbelu et al., 1986); (ii) the identification of imminent collisions through forward sensing and their aversion through vehicle backtracking and/or rerouting (Zeng et al., 1991) and (Hsieh et al, 1998); or (iii) the imposition of zone control and extensive route pre-planning, typically based on deterministic timing of the vehicle traveling and docking stages (Ho, 2000). Among these three approaches, the segmented path flow-based approach presents the highest robustness to the system randomness, but at the cost of restricted vehicle routings and the need for complicated handling operations. These conflict-free routing systems make use of dispatching rules for initial vehicle assignment. Egbelu et al. (1984) were the first researchers to characterize dispatching rules for AGV using simulation. They considered two categories of rules (the work center initiated and the vehicle initiated) for vehicle dispatching decisions. The rules for the work center initiated tasks included nearest vehicle (NV), farthest vehicle (FV) and least utilized vehicle (LUV). For vehicle initiated tasks, the rules included modified first come first serve (MFCFS), shortest travel time/distance, longest travel time/ distance and maximum outgoing queue size (MOQS). However, their study focused more on dispatching rules in a unidirectional network.

Kim et al. (1993) used the concept of time windows to analyze the problem of routing a single AGV from source to destination in the shortest time duration. They maintained a table of scheduled arrival and departure times for all other AGV for each node. The authors then defined a time window to be the duration between the entry and exit at a node, such that each time

window was uniquely reserved for a vehicle with no other AGV allowed to cross that node during this time period. At around the same time, Seifert et al. (1995) evaluated AGV routing strategies using hierarchical simulation. Beyond the static deterministic approach that simply follows the shortest travel distance, the authors also introduced a dynamic vehicle routing strategy based on hierarchical simulation. They developed a generic concept in which the current status of the system was used to embed sub simulations at each decision epoch to mimic the future operation of the system. One of the major recommendations from the study was to generate a sufficient number of alternative vehicle paths so that the critical bottlenecks could be bypassed dynamically. Ho (2000) introduced the concept of a "dynamic-zone strategy" to prevent vehicle collision. This strategy allows reassignment of a vehicle to a zone at any point of time, and uses a zone adjustment procedure to change the area of a zone according to a current production demand. A zone assistance procedure enables the vehicles to balance their workload amongst each other at all instances of time. The strategy developed by Ho was based on the assumption that the AGV system has a single-loop guide path. In addition, it is assumed that an AGV can transport only one load at a time. In the present work, these issues have been addressed conclusively.

4 Scope and Methodology

The following sections in this chapter detail the research scope, describe the mathematical models formulated to solve the problem, and discuss the various heuristic algorithm approaches developed to obtain computationally practical feasible solutions.

4.1 Scope

This research work focuses on optimization of routing AGVs based on prevention of collision amongst the vehicles.

The research goal is to develop an optimally feasible and computationally practical AGV routing system. The objective of the work is to determine an optimal transportation route and timing for the different AGVs in the system, for any given "deterministic" set of transportation requests.

The research vision is to develop a strong mathematical background (in the current work) that could be used in the future to expand the proposed methodology in development of optimally feasible solutions for a "dynamic" set of transportation requests. However, these expanded studies are beyond the scope of the current work.

The research scope can be summarized as:

- Comprehensive mathematical representation of the variables in an AGV routing system, independent of vehicle guide path layout;
- Incorporation of distinctive collision prevention constraints in the model;
- Formulation and programming of different heuristic procedures for problem solution in practically feasible time;

- Selection of the best heuristic or combination of heuristics that could be used across different AGV routing systems; and
- Dynamic simulation of the vehicle routes is out-of-scope of this work.

4.2 Solution Approach

The mathematical model is designed to incorporate the following routing elements:

- Allow AGVs to move on multiple paths yet have the ability to select the best path; and
- Allow the AGVs to move on paths with intersections, yet avoid collisions in a way that best meets the needs of the facility.

The development of the solution methodology involves four (4) phases:

Phase I: Formulation of The Mathematical Model

In this phase, different solution formulation models are developed, progressively reiterated and evaluated in terms of comprehensively capturing the problem domain and research objectives. Based on the evaluation, an integer linear program (IP) is developed and selected with the capability of optimally predicting the AGV routes for a deterministic set of transportation requests. The model is based on inclusion of positioning constraints, motion feasibility constraints, collision prevention constraints, target constraints, and integrality constraints; that are developed to cover all aspects of AGV routing. The model is programmed using OPL 3.7 and C++, while CPLEX 9.0 is used as the solver. Numerical examples are presented in this phase to support model justification.

Phase II: Heuristic Development

Due to the complexity and combinatorial nature of the formulation in Phase I, computationally it is NP-Hard. Hence, to improve the efficiency of the computation, in Phase II, heuristic algorithms are developed that relax the computational complexity of the original

problem. In this phase, different heuristic procedures are designed, programmed and tested to compute the upper bounds of the original formulation.

Phase III: Heuristic Selection

Experimental designs for different sized systems are conducted in this phase in order to compare the performance of the different heuristics. For small-sized systems, the heuristic results are compared to optimal solutions; while for large-sized systems, the bounds obtained from different heuristic procedures are compared. Statistical techniques are used in the analysis. The goal of this phase is to arrive at a robust heuristic / combination of heuristic procedures that can be used across any sized routing system.

Phase IV: Conclusions and Scope for Future Work

The results from this work are summarized in this phase with segmented recommendations for solution expansion to dynamic transportation request problem.

The conclusions and results drawn from each phase are progressively used to contribute to the goal of subsequent phases, and to the overall research objectives.

4.2.1 Phase I: Formulation of The Mathematical Model

The AGV route-planning problem is defined as follows: A grid layout represents the shop floor / distribution / warehouse center. Figure 1 shows a sample grid layout, with the pre-designed AGV guide path represented by the red line.

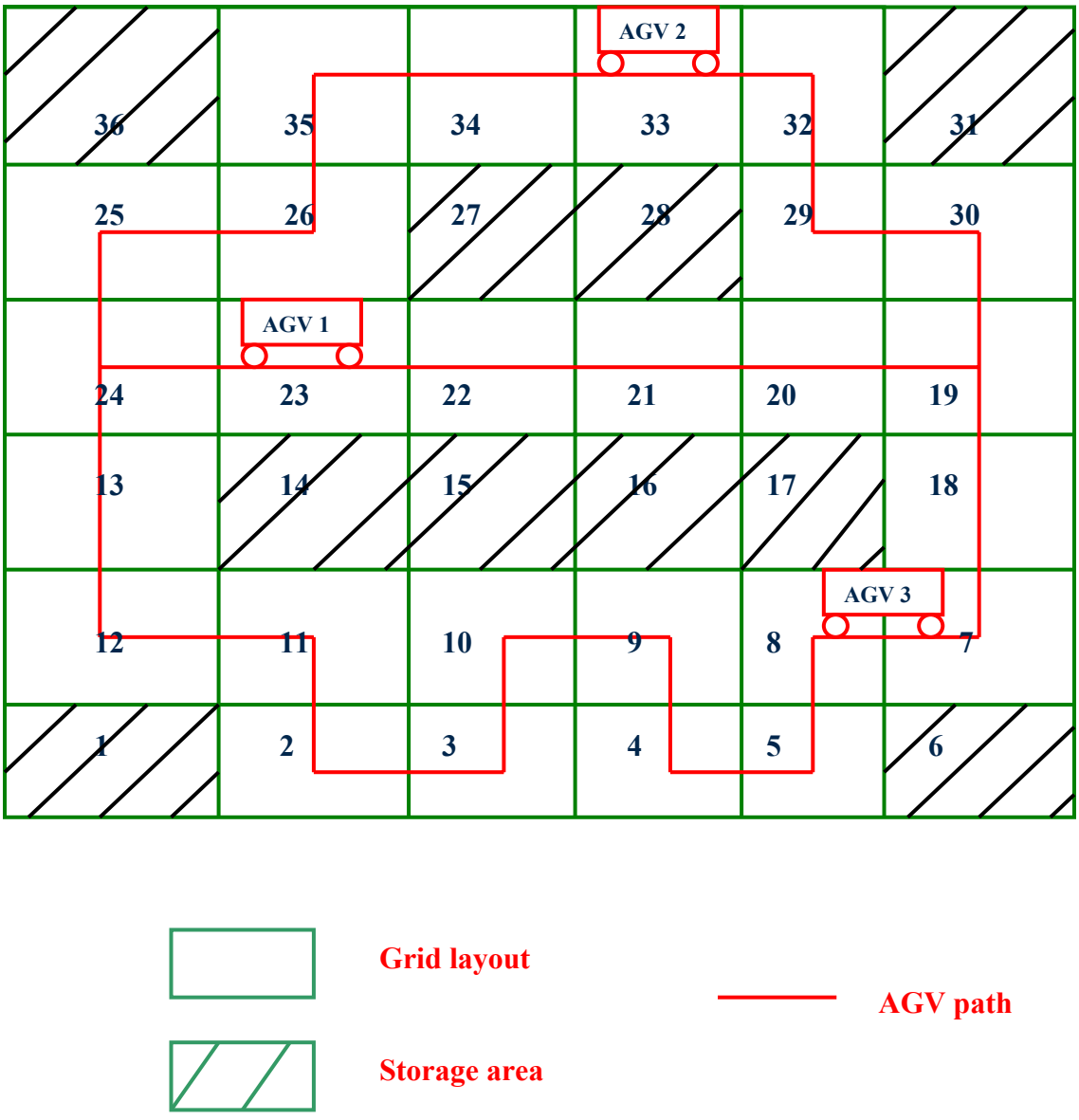


Figure 1: Example of an AGV Grid Layout

In the sample grid layout presented in Figure 1, there are 36 grids in all. The grids that are shaded represent the storage areas, and as such, the AGVs are not programmed to move into these locations. The guide path is shown by the red solid line, and is indicative of a feasible point for an AGV from any particular grid. For example, as shown in Figure 1, AGV 1 is located at Grid 23 at initial time instant 0. At time instant 1, the program constraints the motion of AGV 1 to either Grid 24 or 22 or to stay at Grid 23 itself. There are pre-defined target grids which have pre-defined destination nodes. The AGVs have capacities to carry load. For the mathematical model, it is assumed that each AGV can at any time carry at most one load.

4.2.1.1 Model Assumptions

The following are the assumptions made in the formulation:

1. All the grids are rectangular in shape, and equal in dimensions.
2. All the AGV's considered are capable of bi-directional travel.
3. The velocity for each AGV is constant, with the transportation time proportional to the grid dimensions.
4. Every AGV has the same ability of transportation and carrying capacity.
5. Every AGV carries / delivers only one unit load at a time.

4.2.1.2 Nomenclature

The variables (other than the decision variables) that define the problem domain are presented in this section. Section 4.2.1.3 discusses the decision variables used in the formulation.

J = Number of AGVs in the system

T = Total time required to complete all deliveries in the system

I = Number of grid elements in the system grid

F_i = Set of feasible moves from grid i

H = Set of targets

r_i = Destination node for the target at node i such that $i \in H$

4.2.1.3 Decision variables

Three (3) decision variables are used to completely define the problem. All the X_{ijt} capture the presence of an AGV j at a grid i at any particular time t; Y_{ijt} define the instance t when a target $i \in H$ is picked up by an AGV j; and Z_{ijt} are activated when the target $i \in H$ is delivered by the AGV j to the destination. These decision variables are further explained below.

$$X_{ijt} = \begin{cases} 1 & \text{if an AGV } j \text{ visits grid } i \text{ at time } t \\ 0 & \text{otherwise} \end{cases} \qquad \text{for } \forall i, \forall j, \forall t$$

$$Y_{ijt} = \begin{cases} 1 & \text{if an AGV } j \text{ picks up the target at grid } i \text{ at time } t \\ 0 & \text{otherwise} \end{cases} \qquad \text{for } i \in H, \forall j, \forall t$$

$$Z_{ijt} = \begin{cases} 1 & \text{if the target at grid } i \text{ is dropped to the destination at time t} \\ 0 & \text{otherwise} \end{cases} \qquad \text{for } i \in H, \forall j, \forall t$$

4.2.1.4 Objective Function

The objective of the formulation is to minimize the total transportation time required for target delivery. The mathematical model can be represented as:

Minimize w

Subject to all the constraints detailed in Section 4.2.1.5 Table 2.

w is constrained by Equation 11 in Section 4.2.1.5 Table 2.

4.2.1.5 Mathematical Model – Constraints

The AGV routing path is developed with the objective that there are no vehicle collisions at any point of time, each AGV maintains motion feasibility, and the whole process minimizes the load transportation time in the application set-up. The model constraints detailed in Table 2 incorporate each of these aspects, and can be broadly categorized as:

- **AGV Positioning Constraints:** Equation 1 defines the position of an AGV at any particular instance of time, while Equation 5 constraints an AGV to be present at a grid where it picks up the target at that instance of time. Similarly, Equation 9 synchronizes target deliverance to the requirement of AGV presence at the delivery point, at that particular time.
- **AGV Motion Feasibility Constraint:** Equation 3 constraints subsequent AGV moves based on the feasibility moves on the guide path from any particular grid. In this constraint, the model extends feasibility of moves for applicability in any type of AGV guide path layout.
- **Collision Prevention Constraints:** Equations 2 and 10 are the collision prevention constraints that are responsible for the uniqueness of the model.
- **Target Constraints:** Equations 4, 6, 7 (or equivalently a combination of 7A and 7B) and 8 define the requirements of pick-up and delivery of targets.
- **Integer Constraints:** Equations 12 to 14 are binary constraints for the X's, Y's and Z's.

The following section discusses in detail each of these constraints.

Table 2: Constraints in the Mathematical Model

Equation		*Description*	*Equation No.*
$\sum_{i=1}^{I} X_{ijt} = 1$	for $\forall j, \forall t$	***Any time an AGV will occupy only one grid***	**(1)**
$\sum_{j=1}^{J} X_{ijt} \leq 1$	for $\forall i, \forall t$	***Any time a grid would be occupied by a maximum of one AGV***	**(2)**
$\sum_{k \epsilon F_i} X_{kjt+1} - \quad X_{ijt} \geq 0$	for $\forall i, \forall j, \forall t$	***Continuity & feasibility (i.e.) For each successive time instance, an AGV should either stay at grid i or move to the next feasible grid***	**(3)**
$\sum_{j=1}^{J} \sum_{t=1}^{T} Y_{ijt} = 1$	for $i \in H$	***A target is picked by exactly one AGV (all time)***	**(4)**
$Y_{ijt} \leq X_{ijt}$	for $i \in H, \forall j, \forall t$	***An AGV picking up a load at grid I at time t must be located at grid I at time t***	**(5)**
$\sum_{j=1}^{J} \sum_{t=1}^{T} Z_{ijt} = 1$	for $i \in H$	***A target that has been picked from grid i would be delivered to the destination exactly once***	**(6)**
$\sum_{t=1}^{T} tZ_{ijt} > \sum_{t=1}^{T} tY_{ijt}$	for $i \in H, \forall j$	***The time at which the target is delivered to the destination exceeds the time when the target is picked up***	**(7)**

Table 2 (continued): Constraints in the Mathematical Model

Constraint 7 can also be modeled as a combination of Constraints 7A and 7B:			
$\sum_{t=t+1}^{T} Z_{ijt} > Y_{ijt}$	for $i \in H$, $\forall j$, $\forall t$	***The time at which the target is delivered to the destination exceeds the time when the target is picked up***	**(7A)**
$Y_{ijT} = 0$	for $i \in H$, $\forall j$		**(7B)**
$\sum_{i \in H} \sum_{k=1}^{t} Z_{ijk} + C_j \geq \sum_{i \in H} \sum_{k=1}^{t} Y_{ijk}$	for $\forall j$, $\forall t$	***At any time, all the AGV's should have picked total number of loads ≤ their capacities***	**(8)**
$Z_{ijt} \leq X_{r_i jt}$	for $i \in H$, $\forall j$, $\forall t$	***The target is delivered only when the loaded AGV is at the destination***	**(9)**
$X_{kjt} + X_{ilt} + X_{ijt+1} + X_{klt+1} \leq 3$	for $\forall i$, $\forall j$, $\forall t$ and $k \in F_i$ and $l \in J$ such that $l \neq j$	***If two AGV's occupy adjacent grids at any time instant, they cannot switch their positions in the next time instant***	**(10)**
$w \geq \sum_{j=1}^{J} \sum_{t=1}^{T} t Z_{ijt}$	for $i \in H$	***Constraint to minimize the total transportation time required for target delivery***	**(11)**
$X_{ijt} = 0$ or 1	for $\forall i$, $\forall j$, $\forall t$	***Integer constraints***	**(12)**
$Y_{ijt} = 0$ or 1	for $i \in H$, $\forall j$, $\forall t$	***Integer constraints***	**(13)**
$Z_{ijt} = 0$ or 1	for $i \in H$, $\forall j$, $\forall t$	***Integer constraints***	**(14)**

4.2.1.6 Solving Approach

The mathematical formulation was programmed using Optimization Programming Language (OPL), and tested to provide an initial gauge of the computational capacity (solving time) and computational accuracy (confirmation that all essential collision-prevention constraints had been captured). It was found that for systems up to 5*5 grid layouts, the model provides an optimal solution in a reasonable amount of time. However, for systems beyond a 25-grid layout, the problem becomes NP-hard and development of heuristic algorithm(s) becomes essential (described in following sections).

4.2.2 Phase II: Heuristic Development

A problem is assigned to the NP (nondeterministic polynomial time) class if it is verifiable in polynomial time by a nondeterministic turing machine. A problem is said to be NP-hard if an algorithm for solving it can be translated into one for solving any other NP-problem. Due to the complexity and combinatorial nature of the mathematical formulation in Phase I, computationally it was found to be NP-Hard. Hence, to improve the efficiency of the computation, in Phase II, a heuristic solution of the formulation was mathematically required.

The heuristics functions can be divided into two groups of admissible and non-admissible heuristics. An admissible heuristic is one that never overestimates the optimal cost, i.e. a lower bound. In this work, since the IP is a minimization problem, the non-admissible heuristic would provide an upper bound to the original problem. All the heuristics are computed by solving deterministic relaxations of the original formulation.

The heuristic methods proposed in this work are developed using the dispatching strategies characterized by Egbelu et al. (1984) as a basis. In this work, Egbelu's rules have been

modified to account for specific characteristics of the problem under consideration. The proposed heuristic methodology for the AGV routing is divided into two portions:

1. Development of heuristic priority rules to construct a flexible routing mechanism; and
2. Implementation of the routing system in a flexible software structure.

Multiple heuristics were proposed in initial feasibility studies, however, based on screening trials on computational accuracy (comparison to optimal solution for small-sized systems), the following 4 heuristic approaches were selected for subsequent experimental study purposes:

- The Greedy Approach;
- The Nearest Neighbor Approach;
- The Least Utilized AGV Approach; and
- The Modified Greedy Approach

The mathematical logic for each of the above heuristics is detailed in the following sections.

4.2.2.1 The Greedy Approach (G)

In order to reduce the computational time taken to solve the original formulation, the model is solved in "h" iterations, where h equals the number of targets. This reduces the complexity of the problem solved at each iteration. Each iteration involves solving a modified version of the original problem, where only one target-AGV combination is selected by the OPL program such that the selected target is delivered the earliest amongst all targets. The general steps in the Greedy Approach are described below:

Step 1: For a set of j AGV and h targets, the OPL program decides the initial assignment of an AGV to a target with the earliest possible delivery time.

Step 2: In the next iteration, the locations of the AGV from Step 1 solution are fixed for

the time instances till the target is delivered. The program is solved to arrive at the next AGV-target combination.

Step 3: Steps 1 and 2 are repeated for the remaining targets, and the program stopped when all the targets have been delivered.

The mathematical formulation of the greedy approach is discussed below:

Minimize w

Subject to

$$\sum_{i=1}^{I} X_{ijt} = 1 \qquad \text{for } \forall j, \forall t \qquad \text{---------- (1)}$$

$$\sum_{j=1}^{J} X_{ijt} \leq 1 \qquad \text{for } \forall i, \forall t \qquad \text{---------- (2)}$$

$$\sum_{k \in F_i} X_{kjt+1} - X_{ijt} \geq 0 \qquad \text{for } \forall i, \forall j, \forall t \qquad \text{---------- (3)}$$

$$\sum_{i \in H} \sum_{j=1}^{J} \sum_{t=1}^{T} Y_{ijt} = 1 \qquad \textbf{\textit{At the most 1 target is picked in each iteration}} \qquad \text{---------- (4)}$$

$$Y_{ijt} \leq X_{ijt} \qquad \text{for } i \in H, \forall j, \forall t \qquad \text{---------- (5)}$$

$$\sum_{i \in H} \sum_{j=1}^{J} \sum_{t=1}^{T} Z_{ijt} = 1 \qquad \textbf{\textit{The target that is picked in the iteration is delivered once}} \qquad \text{---------- (6)}$$

$$\sum_{t=t+11}^{T} Z_{ijt} > Y_{ijt} \qquad \text{for } i \in H, \forall j, \forall t \qquad \text{---------- (7A)}$$

$Y_{ijT} = 0$ for $i \in H$, $\forall j$ ---------- **(7B)**

$$\sum_{i \in H} \sum_{k=1}^{t} Z_{ijk} + C_j \geq \sum_{i \in H} \sum_{k=1}^{t} Y_{ijk}$$ for $\forall j$, $\forall t$ ---------- **(8)**

$Z_{ijt} \leq X_{r_i jt}$ for $i \in H$, $\forall j$, $\forall t$ ---------- **(9)**

$X_{kjt} + X_{ilt} + X_{ijt+1} + X_{klt+1} \leq 3$ for $\forall i$, $\forall j$, $\forall t$ and $k \in F_i$ and $l \in J$ such that $l \neq j$ ---------- **(10)**

$X_{ijt} = 0$ or 1 for $\forall i$, $\forall j$, $\forall t$ ---------- **(11)**

$Y_{ijt} = 0$ or 1 for $i \in H$, $\forall j$, $\forall t$ ---------- **(12)**

$Z_{ijt} = 0$ or 1 for $i \in H$, $\forall j$, $\forall t$ ---------- **(13)**

$$w \geq \sum_{j=1}^{J} \sum_{t=1}^{T} t Z_{ijt}$$ for $i \in H$ ---------- **(14)**

4.2.2.2 The Nearest Neighbor Approach (NN)

In order to reduce the computational time taken to solve the original formulation, the model is solved in "h" iterations, where h equals the number of targets. Each of the iterations involves two (2) sub-iterations. In the first sub-iteration, the program selects an AGV-Target combination, such that the target is picked at the earliest possible time. In summary, amongst all targets, the target that can be picked-up the earliest is assigned to the AGV that can perform this

function. In the second sub-iteration, this target is delivered to its destination in the shortest transportation time. For the following iterations, the positions of the AGV from the previous iterations are fixed, and the sub-iterations are repeated. The general steps in the Nearest Neighbor Approach are described below:

Step 1A: For a set of j AGV and h targets, the OPL program selects an AGV-Target combination with the earliest possible pick-up time.

Step 1B: The selected target from Step IA is delivered to its destination by the assigned AGV in the shortest delivery time.

Step 2: For the next iteration, the locations of the AGV from previous iterations (Step 1B solutions) are fixed for the time instances till the target is delivered. Steps 1A and 1B are repeated for the remaining targets, and the program stopped when all the targets have been delivered.

The mathematical formulations of the nearest neighbor approach is discussed below:

Sub-Iteration 1:

Minimize w

Subject to

$$\sum_{i=1}^{I} X_{ijt} = 1 \qquad \text{for } \forall j, \forall t \qquad \text{---------- (1)}$$

$$\sum_{j=1}^{J} X_{ijt} \leq 1 \qquad \text{for } \forall i, \forall t \qquad \text{---------- (2)}$$

$$\sum_{k \varepsilon F_i} X_{kjt+1} - \; X_{ijt} \geq 0 \qquad \text{for } \forall i, \forall j, \forall t \qquad \text{---------- (3)}$$

$$\sum_{i \varepsilon H} \sum_{j=1}^{J} \sum_{t=1}^{T} Y_{ijt} = 1 \qquad \textbf{\textit{At the most 1 target is picked}} \qquad \text{---------- (4)}$$

$$Y_{ijt} \leq X_{ijt} \qquad \text{for } i \in H, \forall j, \forall t \qquad \text{---------- (5)}$$

$$X_{ijt} = 0 \text{ or } 1 \qquad \text{for } \forall i, \forall j, \forall t \qquad \text{---------- (6)}$$

$$Y_{ijt} = 0 \text{ or } 1 \qquad \text{for } i \in H, \forall j, \forall t \qquad \text{---------- (7)}$$

$$\text{w} \geq \sum_{j=1}^{J} \sum_{t=1}^{T} t Y_{ijt} \qquad \text{for } i \in H \qquad \text{---------- (8)}$$

Sub-Iteration 2:

Constraints 1 to 14 discussed below would be included in the formulation, with the "j" and the "h" replaced by the ones selected by the OPL program from Step IB.

Minimize w

Subject to

$$\sum_{i=1}^{I} X_{ijt} = 1 \qquad \text{for } \forall j, \forall t \qquad \text{---------- (1)}$$

$$\sum_{j=1}^{J} X_{ijt} \le 1 \qquad \text{for } \forall i, \forall t \qquad \text{---------- (2)}$$

$$\sum_{k \in F_i} X_{kjt+1} - X_{ijt} \ge 0 \qquad \text{for } \forall i, \forall j, \forall t \qquad \text{---------- (3)}$$

$$\sum_{i \in H} \sum_{j=1}^{J} \sum_{t=1}^{T} Y_{ijt} = 1 \qquad \textbf{\textit{At the most 1 target is picked in each iteration}} \qquad \text{---------- (4)}$$

$$Y_{ijt} \le X_{ijt} \qquad \text{for } i \in H, \forall j, \forall t \qquad \text{---------- (5)}$$

$$\sum_{i \in H} \sum_{j=1}^{J} \sum_{t=1}^{T} Z_{ijt} = 1 \qquad \textbf{\textit{The target that is picked in the iteration is delivered once}} \qquad \text{---------- (6)}$$

$$\sum_{t=t+11}^{T} Z_{ijt} > Y_{ijt} \qquad \text{for } i \in H, \forall j, \forall t \qquad \text{---------- (7A)}$$

$$Y_{ijT} = 0 \qquad \text{for } i \in H, \forall j \qquad \text{---------- (7B)}$$

$$\sum_{i \in H} \sum_{k=1}^{t} Z_{ijk} + C_j \ge \sum_{i \in H} \sum_{k=1}^{t} Y_{ijk} \qquad \text{for } \forall j, \forall t \qquad \text{---------- (8)}$$

$Z_{ijt} \leq X_{r_i jt}$ for $i \in H$, $\forall j$, $\forall t$ ---------- **(9)**

$X_{kjt} + X_{ilt} + X_{ijt+1} + X_{klt+1} \leq 3$ for $\forall i$, $\forall j$, $\forall t$ and $k \in F_i$ and $l \in J$ such that $l \neq j$ ---------- **(10)**

$X_{ijt} = 0$ or 1 for $\forall i$, $\forall j$, $\forall t$ ---------- **(11)**

$Y_{ijt} = 0$ or 1 for $i \in H$, $\forall j$, $\forall t$ ---------- **(12)**

$Z_{ijt} = 0$ or 1 for $i \in H$, $\forall j$, $\forall t$ ---------- **(13)**

$w \geq \sum_{j=1}^{J} \sum_{t=1}^{T} t Z_{ijt}$ for $i \in H$ ---------- **(14)**

4.2.2.3 The Least Utilized AGV Approach (LUA)

In this heuristic approach, the AGV that has been least utilized (or in specificity the one that has been idle for the most amount of time) is assigned a target closest to the AGV. In the next sub-iteration, the AGV delivers this target to its destination in the shortest delivery time. The model accounts for collision constraints during the execution of this sub-iteration. For the following iterations, the positions of the utilized AGV from the previous iterations are fixed, and the sub-iterations are repeated. The number of iterations that the program undergoes equals the number of targets "h". The initial assignment is the same as the Nearest Neighbor rule. The general steps in the Least Utilized AGV Approach are described below:

Step 1A: For a set of j AGV and h targets, the OPL program selects the least utilized AGV (one that has been idle the most since its last delivery).

Step 1B: The target closest to the selected AGV is assigned to that vehicle.

Step 1C: The AGV delivers the target to its destination in the shortest delivery time.

Step 2: For the next iteration, the locations of the utilized AGV from previous iterations (Step 1C solutions) are fixed for the time instances till the target is delivered. Steps 1A, 1B and 1C are repeated for the remaining targets, and the program stopped when all the targets have been delivered.

The mathematical formulations are similar to the Nearest Neighbor approach, with the sole exception being in the least utilized AGV selection.

4.2.2.4 Modified Greedy Approach (MG)

The MG heuristic is similar to the Greedy heuristic except that in iterative AGV selection (subsequent requests), all the "j" AGVs are considered in the selection process. The general steps in the Modified Greedy Approach are described below:

Step 1: For a set of j AGV and h targets, the OPL program decides the initial assignment of an AGV to a target with the earliest possible delivery time.

Step 2: In the next iteration, the locations of the AGV from Step 1 solution are fixed for the time instances till the target is delivered. The program is solved to arrive at the next AGV-target combination.

Step 3: Steps 1 and 2 are repeated for the remaining targets (difference from the Greedy Approach is that in this step all the j AGV go back in the selection process), and the program stopped when all the targets have been delivered.

The mathematical formulations are similar to the Greedy approach, with the sole exception being in the Step 3 selection.

5 Experiments and Results

The following sections in this chapter discuss the experimental designs used for the problem analysis, and the results from the different research phases.

5.1 Results and Discussion

The following sections discuss the results from the different research phases.

5.1.1 Phase I – Formulation of the Mathematical Model

Numerical examples are presented and discussed in this phase to support model justification and to demonstrate optimality of the original mathematical formulation (presented in Section 4.2.1). The data and discussion of results from these examples are presented in the following sections.

5.1.2 Demonstrating Model Optimality

The AGV routing path is developed with the objective that there are no vehicle collisions at any point of time, each AGV maintains motion feasibility, and the whole process minimizes the load transportation time in the application set-up. AGV positioning and motion feasibility constraints discussed in Section 4.2.1.5 are the "necessity" constraints that define and restrict vehicle position and motion to feasible grids. The constraints of specific interest for demonstrating formulation optimality are the collision prevention and the target constraints, and an example is presented in this section with a step-by-step assessment of model optimality. For demonstration purposes, we consider a 4*4 grid with 2 AGV and 4 target locations. A schematic of the system is shown in Figure 2, and the problem data is outlined in Table 3.

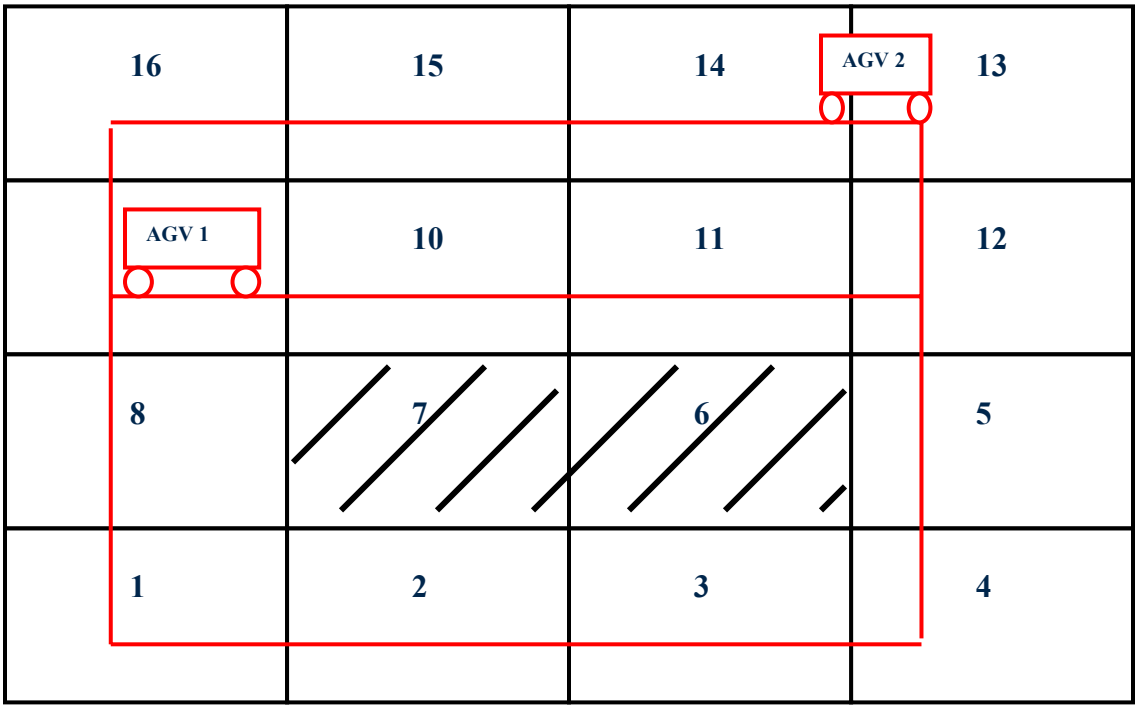


Figure 2: Model Optimality Example – System Layout

Table 3: Model Optimality - Sample Problem Specifications

Target Location	Destination Location
1	16
2	9
3	8
8	11

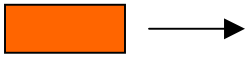
We first solved the model without constraints 2, 5, 7A, 7B, 9, and 10 (refer Section 4.2.1.4). The skipped constraints are the collision prevention and the target constraints. The OPL program outputted a solution with transportation time for combination of all jobs as 1 unit. This clearly

indicated that the solution was infeasible, and many new constraints need to be captured and added to the model. So, as a next step, constraints 5 and 9 were added. These additional constraints limit the pickup or delivery of a target only when the AGV is physically present at that location. On solving this model, the solution obtained is tabulated in Table 4.

Table 4: Model Optimality – Without Collision Prevention and Target Constraints

Assignments:		
Time	**AGV # (Location)**	
	1	**2**
1	9	13
2	16	12
3	9	11
4	8	12

Target Pick-Up



Target Delivery

The results from Table 4 show that all the targets have been delivered even before they are picked up. This necessitates the addition of a constraint that restricts target delivery only after pickup. The model was re-solved with the addition of constraints 7A and 7B. The results of the simulation are tabulated in Table 5.

Table 5: Model Optimality – Without Collision Prevention Constraints

Assignments:		
Time	**AGV # (Location)**	
	1	**2**
1	9	13
2	8	12
3	1	5
4	2	4
5	1	3
6	8	2
7	9	1
8	8	8
9	9	1
10	10	8
11	11	9
12	10	16

Target Pick-Up **Target Delivery**

From Table 4, we observe that at time 8, both the AGV are at the same grid. Hence, for proving model optimality, it is essential that the formulation include collision prevention constraints. The optimal solution is tabulated in Table 6.

Table 6: Model Optimality – Complete Formulation

Assignments:		
Time	**AGV # (Location)**	
	1	**2**
1	9	13
2	8	12
3	1	5
4	2	4
5	1	3
6	8	2
7	9	1
8	9	8
9	8	1
10	9	8
11	10	9
12	11	16

5.2 Phase II - Heuristic Development – Preliminary Analysis

The following sections compare the functionality of the different heuristic approaches (detailed in Section 4.2.2) using examples having similar set of input parameters. To illustrate these approaches, we consider four (4) randomly selected sample problems, with their domains described in Table 7. The results of this exercise are discussed in Table 8.

Table 7: Heuristic Approaches - Sample Examples – Data

Problem No.	Grid Size	No. of P/D	No. of AGV
1	25	8	2
2	25	6	2
3	36	10	4
4	64	7	4

For the two (2) 25-grid problems, the same layout was selected. For output purposes, average % of loaded travel for all the AGV combined is also considered. This measure in conjunction with the total transportation time was used to gauge the heuristic performance.

Table 8: Heuristic Approaches - Sample Examples – Results

Problem No.	Heuristic Approach	Transportation Time	% Average AGV Loaded Travel
1	**Greedy**	27	72.22
	Nearest Neighbor	23	84.78
	Least Utilized AGV	23	80.43
	Modified Greedy	25	78.57
2	**Greedy**	23	63.04
	Nearest Neighbor	23	50.00
	Least Utilized AGV	23	50.00
	Modified Greedy	23	63.04
3	**Greedy**	28	38.39
	Nearest Neighbor	25	58.00
	Least Utilized AGV	25	57.00
	Modified Greedy	26	44.31
4	**Greedy**	23	67.39
	Nearest Neighbor	23	64.13
	Least Utilized AGV	23	61.96
	Modified Greedy	23	67.39

Based on the results in Table 8, the following were the selected observations and next steps:

- Each heuristic has its own unique mechanism of selecting the initial assignments. It is not possible to observe trends in % AGV loaded travel or the transportation time by looking at a few sample examples.
- Based on the preliminary heuristic studies, it was decided to also investigate a fifth heuristic that would be a combination of the 4 heuristics that are listed.
- Detailed experimental designs were required to differentiate between the heuristic performances, and to compare them with the optimal routes. The following sections discuss the results from these studies.

5.3 Phase III: Design of Experiments for Heuristic Comparisons / Selection

As discussed in the problem statement (Chapter 2), this work develops optimization approaches for effectively and efficiently routing AGV in material-handling applications. This section specifically discusses the methods used in the comparison of the developed optimization heuristic algorithms. Experimental design methods are used for data gathering purposes, and statistical techniques are used for solution analysis.

The goal of the experimental designs was to determine a heuristic approach that is robust across systems, one that can be used in any problem domain irrespective of the number of AGV or the number of pick-up/delivery (P/D) stations. Two (2) sets of experiments were conducted in this phase in order to evaluate and compare the performance of the different heuristics. Each experimental study was a four (4)-step process as discussed by Richardson et al. (2005). These steps are described below:

Step 1: The controlling factors were identified and their levels decided. The factor levels were bracketed to encompass a practically applicable data range.

Step 2: The second step in the experimental study was to develop a hypothesis based on scientific reasoning to guide decisions on the type and amount of data required to detect a significant difference. The other activities in this step included deciding the significance level for testing and the sample size needed for the design to have adequate power to detect a practically meaningful difference. For practical purposes, a significance level (p-value) of 0.05 was set. In order to increase the power of the statistical tests, it was important to decide on the sample size.

Step 3: In the third step of each experimental study, an appropriate statistical test for the data analysis was determined.

Step 4: In the final step of the process, the results of the statistical testing were interpreted to identify statistically significant differences.

The experimental studies are presented in the following sections, and specifically investigate the results from the experimental designs used in the comparison of the heuristic algorithms. The statistical differences in the heuristic performance (as measured by its offset from the optimal solution for small systems, and measured as upper bounds for larger systems) are analyzed as a function of the system layout, the number of vehicles in the system, and the number of P/D under consideration.

5.3.1 Experiment Set # 1: Non-Independent Responses

5.3.1.1 Objective: To determine if there was a difference between the 4 heuristics.

5.3.1.2 Design Considerations:

- In order to account for variability across systems, guide path layout was selected as a "Blocking" factor.
- Responses (solutions from the different heuristics) were non-independent.
- 48 experiments per heuristic – each heuristic tested for same set of AGV starting position & pickup – delivery (P/D) locations.
- No randomization introduced in the AGV starting position & P/D locations.

5.3.1.3 Factors Evaluated:

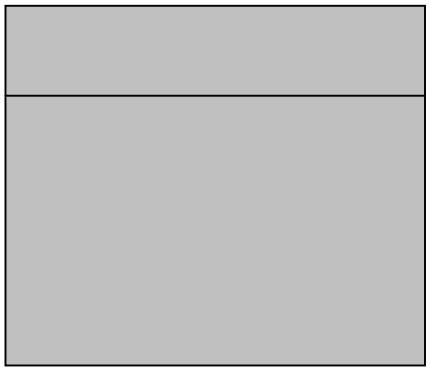
- Number of AGV in the system (Factor A) – 2 levels (2 & 3 AGVs).
- Number of pickup/delivery (P/D) stations (Factor B) – 2 levels (4 & 8 P/D).
- Type of Heuristic used to estimate the upper bound (Factor C) – 5 heuristics.
- In order to account for variability across system guide path layouts, a fourth factor called as "System" is used as a blocking factor (Factor D) – 3 levels. We are only interested in the main effects of this factor, and would essentially ignore its interaction effects with other factors.
- 4 replicates (different AGV starting position & P/D location) per system.

5.3.1.4 Layout:

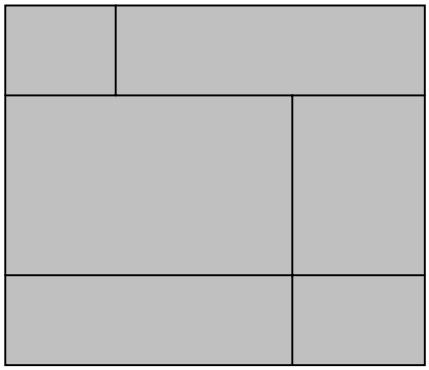
The selection of appropriate system layouts (defined by the guide path design) was critical in the validity of the experimental design. The methodology used by Beamon et al., (1998) for layout selection was used as a guideline for this study.

For the experimental design, three material handling system layouts were considered. These were denoted as System S1, System S2 and System S3. All the three systems allowed for bi-directional travel. System S1 is a single loop; System S2 is a single loop containing one cutover, and System S3 is a single loop containing two horizontal and two vertical cut-overs. The system layouts are shown in Figure 3. Based on the layout and routing optimality considerations, grid sizes 9, 16 and 25 were used for the small systems. These grid sizes corresponded to systems S1 through S3, respectively.

System S1



System S2



System S3

Figure 3: System Layout Selected for "Non-Independent Responses" Experimental Design

5.3.1.5 Analysis & Results

The experimental data was analyzed using the methodology outlined by Hothorn et al. (2004). The authors introduce a sound and flexible theoretical framework for the comparison of candidate algorithms and algorithm selection for arbitrary learning problems. A matched pairs or dependent samples design is used for the comparison of algorithms since the performance of all K algorithms is evaluated using the same random samples. In this work, the problem of interest is to test whether the 5 algorithms under investigation perform equally well against the alternative that at least one of them outperforms all other candidates. In a dependent K samples design, the test statistic (Equation 15 below) developed by Hothorn et al., (2004) equals:

$$t^* = \frac{\sum_{k}\left(B^{-1}\sum_{b} p_{kb} - (BK)^{-1}\sum_{k,b} p_{kb} \right)^2}{\sum_{k,b}\left(p_{kb} - K^{-1}\sum_{k} p_{kb} - (B)^{-1}\sum_{b} p_{kb} + (BK)^{-1}\sum_{k,b} p_{kb} \right)^2} \qquad \text{(Equation 15)}$$

where,

B = Number of random samples = 48

K = Number of algorithms = 5

p_{kb} = Response of the k^{th} algorithm for the b^{th} sample

K1 = The Nearest Neighbor Approach

K2 = Greedy Approach

K3 = The Least Utilized AGV Approach

K4 = Modified Greedy Approach

K5 = Combination of the other 4 heuristics

The t-statistic is used to construct a permutation test, where the distribution of t is obtained by permuting each of the algorithms for each of the samples $(b = 1,...., B)$ independently.

The null (H_o) and alternative (H_a) hypothesis for the study are summarized below:

H_o: The performances of the algorithms are equal.

H_a: The performances of the algorithms are not equal.

5.3.1.6 Results and Discussion

The algorithm performance is measured in terms of the deviation of its solution from the optimal. The experimental data used for analysis and the results calculations are summarized in Appendix A.3.1, while the final results of the study are summarized below.

$$t^* = \frac{\sum_k \left(B^{-1} \sum_b p_{kb} - (BK)^{-1} \sum_{k,b} p_{kb} \right)^2}{\sum_{k,b} \left(p_{kb} - K^{-1} \sum_k p_{kb} - (B)^{-1} \sum_b p_{kb} + (BK)^{-1} \sum_{k,b} p_{kb} \right)^2} \qquad \text{(Equation 15)}$$

$$t^* = \frac{1.5056}{299.9832} = 0.0050$$

For the performance measure based on cross-validation, the value of the test statistic is t* = 0.0050 which corresponds to a conditional P-value of less than 0.001, thus, the null hypothesis can be rejected at significance level of 0.05. Thus, statistically, there is difference in the performance of the 5 algorithms.

As the next step in the study, it is of special interest to identify the algorithms that caused the rejection of the global equality null hypothesis, and to provide a means for classifying the

difference. For this purpose, a matched pair wise comparison for the different algorithm sets were performed. The heuristics were pair wise compared based on the actual transportation time computed in the experiments. The studentized t-statistic used to compare the performance distributions is as follows:

$$t = \sqrt{B}\,\frac{\bar{d}}{\sqrt{(B-1)^{-1}\sum_{b}\left(d_b - \bar{d}\right)^2}}$$

where,

B = Number of random samples

$d_b = p_{1b} - p_{2b}\,(b = 1,....,B)$ for the observations p_{1b} and p_{2b} of algorithms a_1 and a_2 respectively

d = Average of all the differences d_b

The experimental data used for analysis and the results calculations are summarized in Appendix A.3.2. The results of the comparisons between the 5 heuristics – Nearest Neighbor (NN), Greedy (G), Least Utilized AGV (LUA), Modified Greedy (MG) and Combination (C) are detailed in Table 8 below:

Table 9: Experimental Set # 1: Match Pair Heuristic Comparisons for Dependent Samples

	T-value	P-value (two-tail test)	Statistically Significant Difference (at α = 0.05)
NN ~ G	1.9459	0.0577	No
NN ~ LUA	-1.9437	0.0579	No
NN ~ MG	1.7901	0.0799	No
NN ~ C	5.0444	< 0.0001	Yes
G ~ LUA	-2.9786	0.0046	Yes
G ~ MG	-0.4435	0.6594	No
G ~ C	4.4309	< 0.0001	Yes
LUA ~ MG	2.6059	0.0122	Yes
LUA ~ C	5.6874	< 0.0001	Yes
MG ~ C	3.9654	0.0002	Yes

Based on the results in Table 9, the heuristics can be grouped as follows in order of performance (best to worst) as shown in Figure 4.

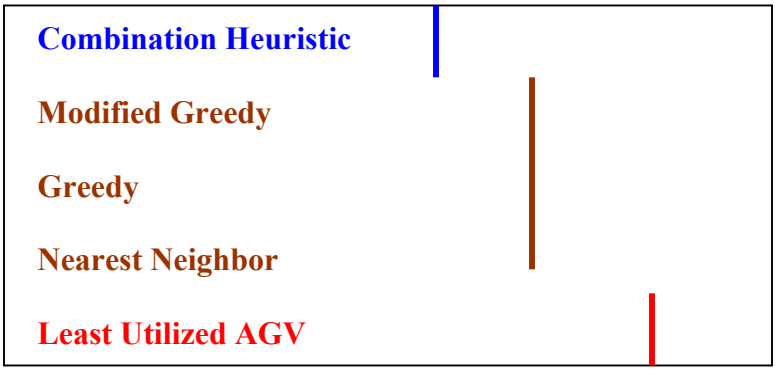


Figure 4: Experimental Set # 1: Grouping of the Heuristics in Order of Performance (Best to Worst)

The performance of the heuristics in comparison to the optimal solution for the experiments is presented in Figure 5 below.

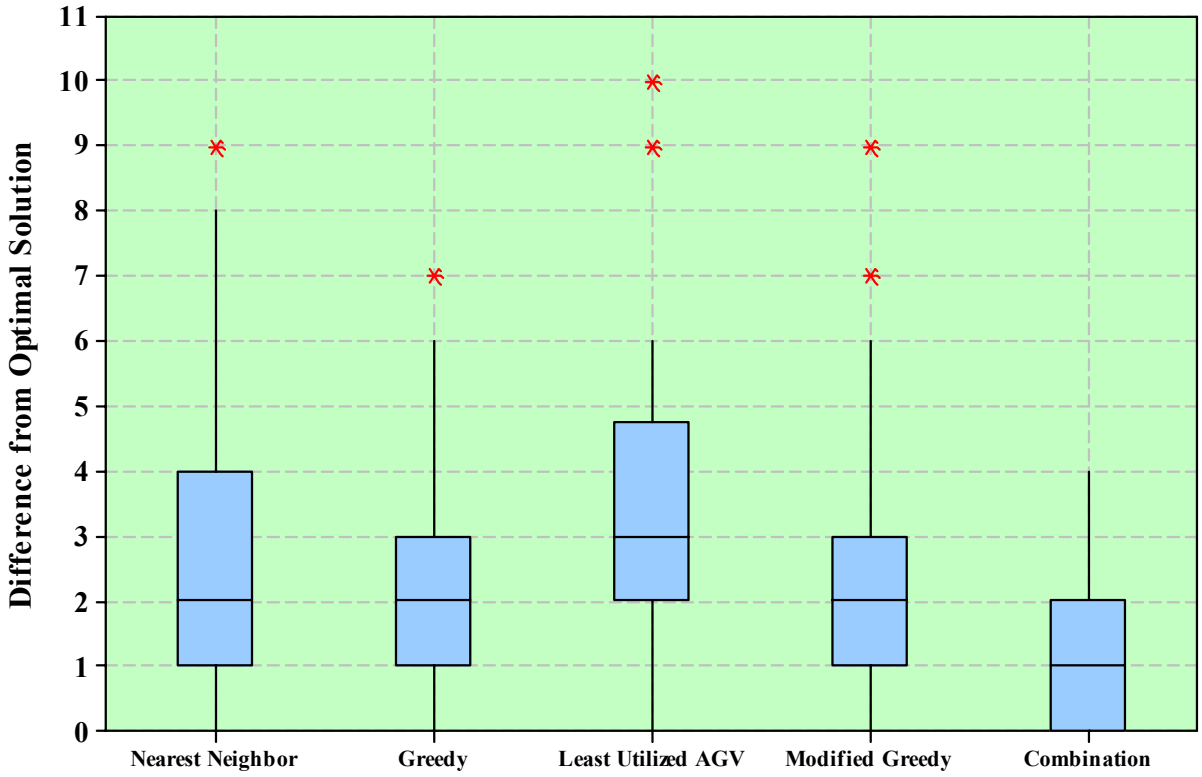


Figure 5: Experimental Set # 1: Box-plot of Heuristic Comparisons for Dependent Sample Test

As a second step, an experimental design study for an independent sample test was conducted to corroborate the above findings.

5.3.2 Experiment Set # 2: Independent Responses

5.3.2.1 Objective: To rank the heuristics in the order of performance.

5.3.2.2 Design Considerations:

- Randomization introduced in the guide path layout, AGV starting position & P/D locations.
- Responses (solutions from the different heuristics) were independent.
- 36 experiments per heuristic (144 in all tested in the study design) – each of the 4 heuristic (combination heuristic not included) tested for a randomized set of AGV starting position & P/D locations.
- 5*5 grid selected for ease of comparison with the optimal solution.

5.3.2.3 Factors Evaluated:

- Number of AGV in the system (Factor A) – 2 levels (2 & 3 AGVs).
- Number of pickup/delivery (P/D) stations (Factor B) – 2 levels (4 & 8 P/D).

5.3.2.4 Layout:

The selection of appropriate system layouts (defined by the guide path design) was random, and the factor combinations were replicated to the extent that the probability of results convergence was high.

5.3.2.5 Analysis

In this set of experimental studies, the heuristics was tested for different unique random combinations of guide path layout, AGV starting position & P/D locations. For each level of AGV & P/D combination, nine (9) sets of unique experiments (with 3 unique guide path layouts) were conducted for each heuristic. The response selected was the difference of the heuristic solution (upper bound for each experimental run) with the optimal solution. Due to the randomly generated runs, the responses from the different heuristic runs were independent.

ANOVA and Tukey's HSD (Honestly Significantly Different) test is applied to all pair wise differences between means in order to determine which specific algorithm(s) are statistically different from each other.

In the Tukey's test, all the algorithm means for the treatment runs are ranked in order of magnitude; group with lowest mean gets a ranking of 1. The pair wise differences between means, starting with the largest mean compared to the smallest mean, are tabulated between each group pair and divided by the standard error. This value is compared to a Studentized range critical value, and if it is larger than the critical value (which would be calculated on the basis of risk of 5%), then the expression between that group pair is considered to be statistically different.

5.3.2.6 Results and Discussion

The performances of the heuristics (defined as difference from the optimal solution) for the random experiments are presented in Figures 6 through 8 and in Table 10. The experimental design used for analysis is detailed in Appendix A.3.3. From the ANOVA results in Figure 5 and the piecewise comparisons in Figure 6, we observe that there is a statistically significant difference between the "Least Utilized AGV" and "Combination" algorithms. This is in agreement with the findings from Experiment Set # 1.

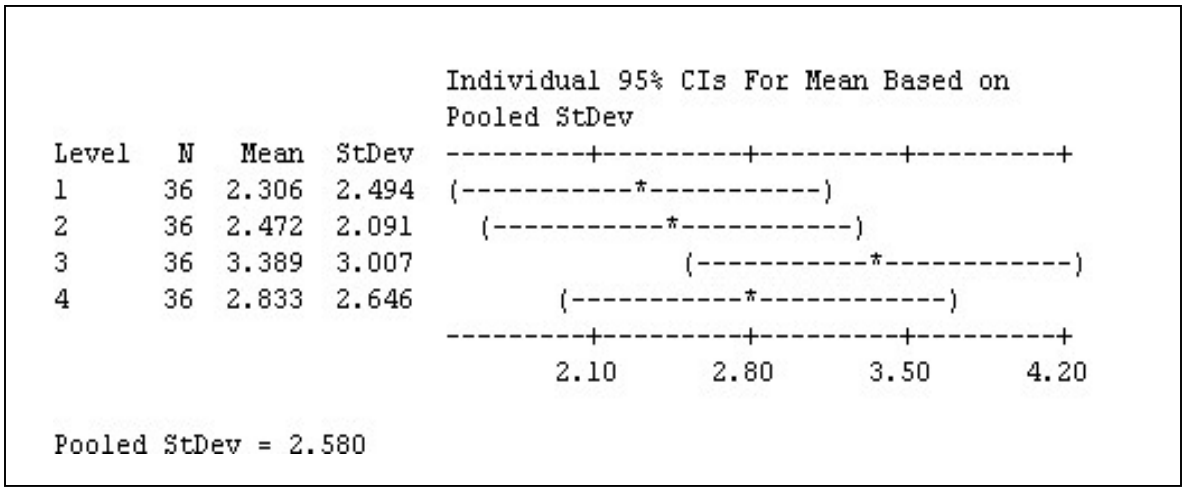


Figure 6: Experimental Set # 2: ANOVA Results

where,

$K = 1$ = Greedy Algorithm;

$K = 2$ = Nearest Neighbor Algorithm;

$K = 3$ = Least Utilized Algorithm; and

$K = 4$ = Modified Greedy Algorithm.

```
Tukey 95% Simultaneous Confidence Intervals
All Pairwise Comparisons among Levels of H

Individual confidence level = 98.97%


H = 1 subtracted from:

H   Lower  Center  Upper  -------+---------+---------+---------+--
2  -1.416   0.167  1.749          (---------*----------)
3  -0.499   1.083  2.666                (---------*----------)
4  -1.055   0.528  2.110             (----------*---------)
                         -------+---------+---------+---------+--
                             -1.5       0.0       1.5       3.0


H = 2 subtracted from:

H   Lower  Center  Upper  -------+---------+---------+---------+--
3  -0.666   0.917  2.499              (---------*----------)
4  -1.222   0.361  1.944           (---------*----------)
                         -------+---------+---------+---------+--
                             -1.5       0.0       1.5       3.0


H = 3 subtracted from:

H   Lower  Center  Upper  -------+---------+---------+---------+--
4  -2.138  -0.556  1.027     (----------*----------)
                         -------+---------+---------+---------+--
                             -1.5       0.0       1.5       3.0
```

Figure 7: Experiment Set # 2: All Piecewise Comparisons among Levels of Heuristics (designated as "K")

Table 10: Experiment Set # 2: Two-Sample T-Test Heuristic Comparisons for Independent Sample Runs

	T-value	P-value	Statistically Significant Difference (at $\alpha = 0.05$)
NN ~ G	0.31	0.760	No
NN ~ LUA	-1.50	0.138	No
NN ~ MG	-0.64	0.523	No
NN ~ C	0.77	0.442	No
G ~ LUA	-1.66	0.101	No
G ~ MG	-0.87	0.387	No
G ~ C	0.37	0.709	No
LUA ~ MG	0.83	0.408	No
LUA ~ C	2.17	0.034	Yes
MG ~ C	1.34	0.185	No

Combining the Tukey test and t-test results, for the independent sample experimental study, the heuristics can be grouped in order of performance (best to worst) as shown in Figure 8.

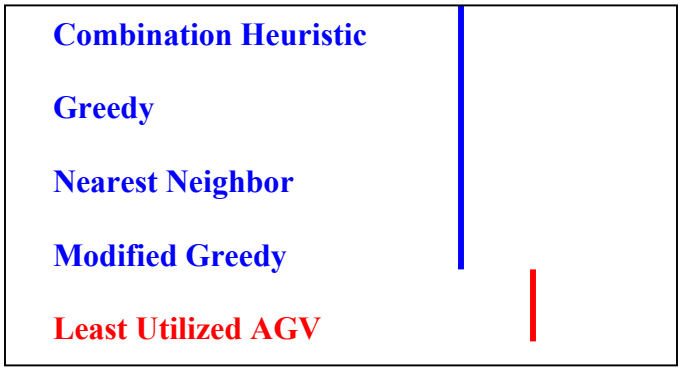


Figure 8: Experimental Set # 2: Grouping of the Heuristics in Order of Performance (Best to Worst)

The performance of the heuristics in comparison to the optimal solution for the experiments is presented in Figure 9 below.

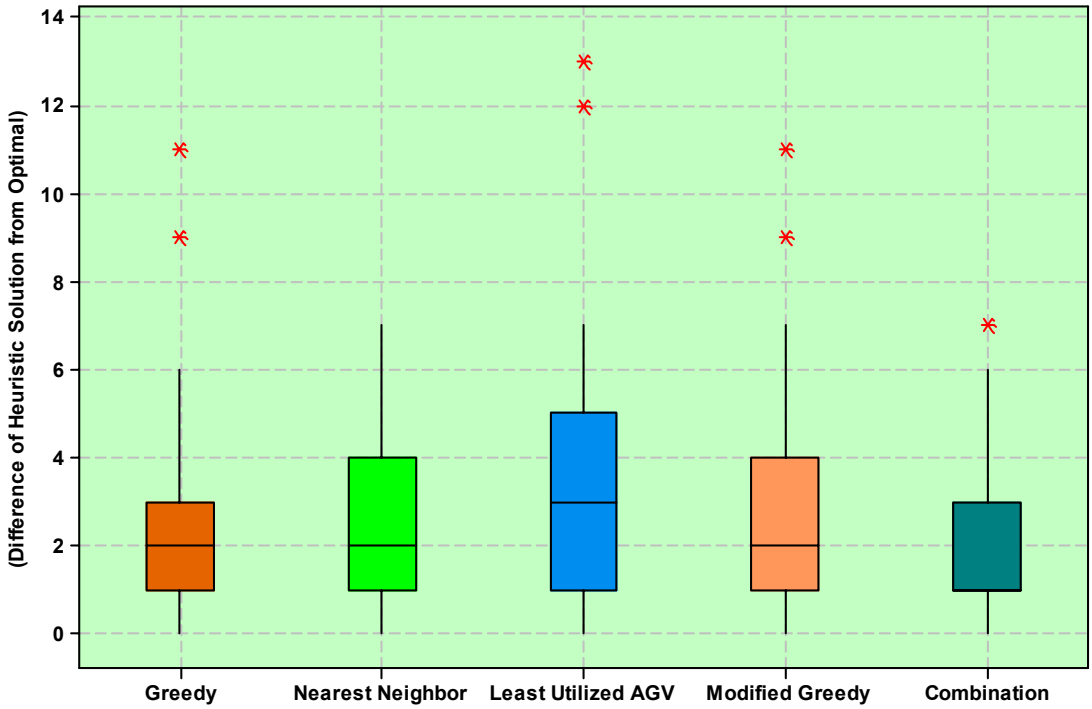


Figure 9: Experiment Set # 2: Box-plot of Heuristic Comparisons for Independent Sample Test

5.3.3 Summary of the Experimental Designs

The followings are the conclusions from the two (2) sets of experimental studies:

- Based on analysis of the results from experimental set # 1 (non-independent responses), the "Combination" heuristic is statistically best performer compared to the other 4 heuristics. The "Least Utilized AGV" heuristic is statistically least performer compared to the other 4 heuristics.
- Based on analysis of the results from experimental set # 2 (independent responses), the "Least Utilized AGV" heuristic is statistically least performer compared to the other 4 heuristics. Amongst the other 4 heuristics, there is no one heuristic that statistically outperforms the others.
- Based on the two sets of experiments, not one heuristic stands out in terms of performance. Depending on application, problem layout and domain size, any of the 4 heuristics (Greedy, Nearest Neighbor, Least Utilized AGV, Modified Greedy) could stand out from the rest. However, statistically the combination of the above 4 heuristics is the best performing algorithm.

5.4 Discussion of Software Program developed for the Problem Formulation

The mathematical models have been programmed using Visual Basic and/or Optimization Programming Language (OPL). CPLEX has been used as the solver for the analysis. The Visual Basic and OPL codes for the optimal model are detailed in Appendix A.4.

A front-end has been developed in Visual Basic that enables the user to input the variables to the system, inclusive of number of AGVs; number of grids, number of targets and number of targets. The form includes provision for entering a random number generating "seed". Based on the seed input, the program randomizes the AGV initial positions and the pickup-delivery combination for the targets. The random assignments are displayed in the front-end, the program converts the code into a linear program which gets inputted into the CPLEX program for analysis. A schematic of the front-end is shown in Figure 10.

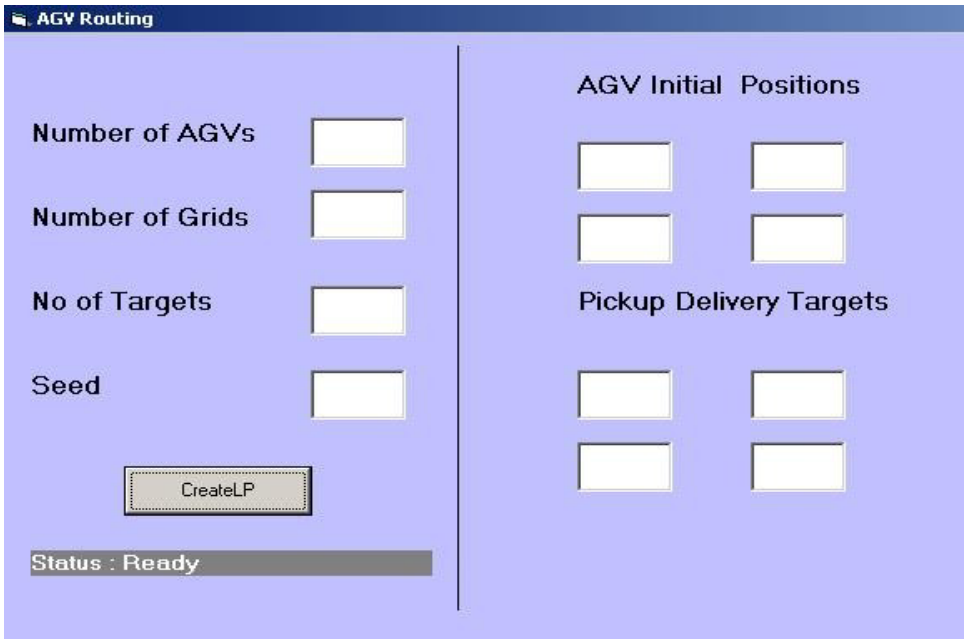


Figure 10: Visual Basic front-end for the Problem Formulation

6 Conclusions

In this research work, the variables in an AGV routing system have been comprehensively represented mathematically. The developed route planning mathematical model works effectively for any type of AGV guide path layout, and can optimally derive vehicle routes for a deterministic set of transportation requests. Distinctive collision prevention constraints have been incorporated in the mathematical model. Different heuristic algorithms have been developed, and their performance evaluated using independent and non-independent experimental designs. Statistical techniques have been used in the data analysis.

The following are the specific conclusions from this research work:

- Mathematical representation of the variables in an AGV routing system (independent of vehicle guide path layout) has been comprehensively and successfully designed in the developed formulation model. The design includes incorporation of distinctive collision prevention constraints in the model.
- The developed formulation is a NP-hard integer program that can be used for optimally solving small-sized problems (up to 5*5 grids) in practically feasible time.
- Approximate solutions (upper bounds) using multiple candidate heuristic approaches have been proposed, evaluated and experimentally compared to solve computationally infeasible capacity problems.
- The heuristics have been grouped based on their order of performance for both dependent sample tests (heuristic individual values used as responses), as well as for independent sample tests (difference of heuristic from optimal solution used as responses).
- Based on the experiments, not one heuristic stands out in terms of performance. Depending on application, problem layout and domain size, any of the 4 heuristics

(Greedy, Nearest Neighbor, Least Utilized AGV, Modified Greedy) could stand out from the rest. However, statistically the combination of the above 4 heuristics is the best performing algorithm. Computationally, the combination algorithm solving time is comparable to the 4 individual heuristic solvers.

To summarize, the developed model and recommended combination algorithm provide a complete platform for efficiently and effectively obtaining collision-free AGV routing solutions for a deterministic request system. This represents the significant contribution of this work in the field of AGV route planning.

Bibliography

Bilge Ü., Ulusoy G., A time window approach to simultaneous scheduling of machines and material handling system in an FMS, *Operations Research,* v 43, No.6, 1995, pp 1058 - 1070.

Dumas Y., Desrosiers J., Soumis F., The pickup and delivery problem with time windows, *European Journal of Operational Research,* v 54, 1991, pp 7 - 22.

Dowsland K.A., Greaves A.M., Collision avoidance in bi-directional AGV systems, *Journal of the Operational Research Society*, v 45, Jul 1994, pp 817 – 826.

Egbelu P.J, Tanchoco J.M.A, Characterization of Automated Guided Vehicle Dispatching Rules, *International Journal of Production Research*, v 22, No.3, 1984, pp 359 – 374.

Egbelu P.J., Tanchoco J.M.A., Potentials for bi-directional guide path for automated guided vehicle based systems, *International Journal of Production Research*, v 24, 1986, pp 1075 – 1097.

Faraji M., Batta R., Forming cells to eliminate vehicle interference and system locking in an AGVS, *International Journal of Production Research*, v 32, Sep 1994, pp 2219 – 2241.

Gaskins R.J., Tanchoco J.M.A., Flow path design for automated guided vehicle systems, *International Journal of Production Research*, v 25, n 5, 1987, pp 667 – 676.

Hammond G., AGVS at work, *IFS Publications Ltd., United Kingdom,* 1986.

Han, M-H., and McGinnis, L.F., Control of Material Handling Transporter in Automated Manufacturing, *IIE Transactions*, 21 (2), 1989, pp 184-190.

Hodgson T.J., King R.E., Moteith S.K., Schultz S.R., Developing control rules for an AGVS using Markov Decision processes, *Material Flow*, v 4, 1987, pp 85 –96.

Hsieh S., Kang M.-Y., Developing AGVS petri net control models from flowpath nets, *Journal of Manufacturing Systems*, v 17, No.4, 1998, pp 237 - 250.

Kim J., Klein C.M., Noble J.S., Location of pickup and delivery points for AGVs, *Proceedings of the Industrial Engineering research Conference*, 1993, pp 390 – 394.

Krishnamurthy N.N, Batta R., Karwan M.H., Developing conflict-free routes for automated guided vehicles, *Operations Research*, v 41, n 6, 1993, pp 1077 – 1090.

Maxwell W.L., Muckstadt J.A., Design of automated guided vehicle systems, *IIE Transactions*, v 14, n 2, 1982, pp 114 – 124.

Meersmans P., Optimization of container handling systems, *Dissertation, Erasmus University Rotterdam*, 2002

Savelsbergh M.W.P., Sol, M., The general pickup and delivery problem, *Transportation Science*, v 29, No.1, 1995, pp 17 - 29.

Seifert R.W., Kay M.G., Wilson J.R., Evaluation of AGV routing strategies using hierarchical simulation, *Proceedings of the Winter Simulation Conference*, 1995.

Ulgen O.M., Kedia P., Using simulation in design of a cellular assembly plant with automatic guided vehicles, *Proceedings of the Winter Simulation Conference*, 1990.

Ulusoy G., Sivrikaya-Serifoglu F., Bilge Ü., A genetic algorithm approach to the simultaneous scheduling of machines and automated guided vehicles, *Computers Operations Research*, v 24, No.4, 1997, pp 335 - 351.

Yang J., Jaillet P, Mahmassani H., Real-Time multi-vehicle truckload pickup and delivery problems, *Transportation Science*, v 38, No.2, 2002, pp 135 - 148.

Zeng L., Wang H.P.B., Jin S., Conflict detection of automated guided vehicles: a Petri net approach, *International Journal of Production Research*, v 29, No.5, 1995, pp 865 - 879.

Appendices

A.1 Appendix 1: Secondary Motivation for the Research Work - AGV Modeling using ARENA – A Case Study

A manufacturing subsystem consisting of four workstations: two assembly stations, one inspection station, and one packing station were modeled. Parts for model assemblies – Model A & Model B supplied by two upstream manufactories are assembled using the same subsystem. The parts are transported between any two workstations using 3 AGV. The first upstream factory sends parts for Model A to the subsystem and the inter-arrival times are exponentially distributed with a mean of 6 hours. The second upstream factory sends parts for Model B to the subsystem one arrival at a time, and the inter-arrival time follows a normal distribution with the mean of 1.5 hours and the standard deviation of 20 minutes. The parts for both the models are inspected upon arrival. The inspection station inspects the parts for any damage during shipping. If the parts are confirmed good, it is sent to the corresponding assembly station, otherwise the parts for the model are discarded. The inspection times are given in the table.

The assembly process consists of two assembly workstations that put the respective models together. Model A has to be first assembled on the assembly station 2, and then on assembly station 1. Model B has to be first assembled on the assembly station 1 and then on assembly station 2. Assembly station 1 has only two buffer spaces. If there is no room in the assembly station 1 buffer space, the parts are routed to another backup assembly station. The processing time for the backup assembly station is the same for model A and B and is distributed uniformly between 1.5 to 3 hours. Model A requires TRIA (60,90,135) minutes on Assembly 1 & TRIA (30,45,135) minutes on Assembly 2. Model B requires TRIA (30,45,75) minutes on Assembly 1 & TRIA (60,75,120) minutes on Assembly 2. After the assembly stations have

processed the parts, they are sent to the inspection station. If a computer is confirmed as a good product, the computer will be sent to the packing department and then departs the system. If the computer is considered as a non-confirmed product, it will be discarded. If the computer goes through the backup assembly station, it is inspected on-line and goes directly go to the packing station. Model A requires TRIA (0.4,0.6,0.7) hours on pre-assembly inspection, NORM (1.25, 0.15) hours on packing & TRIA (0.3,0.6,0.7) hours on post-assembly inspection. Model B requires TRIA (0.6,0.9,0.5) hours on pre-assembly inspection, NORM (0.9, 0.12) hours on packing & TRIA (0.5,0.6,0.9) hours on post-assembly inspection. Figure A1 shows the network layout.

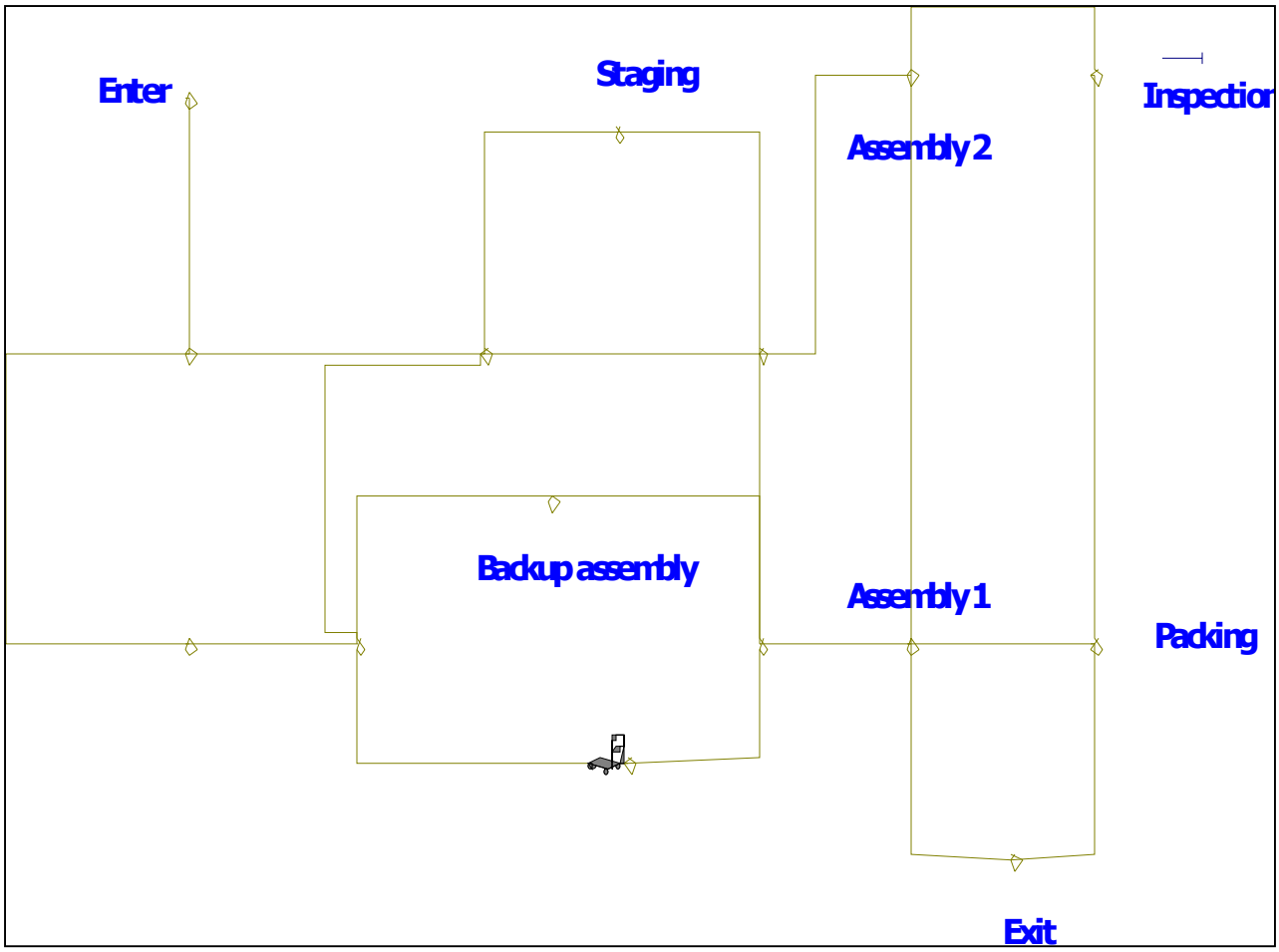


Figure A1: AGV network layout of the manufacturing set-up

In the first replication of simulation run, as soon as a collision was detected, the ARENA program terminated.

A.2 Appendix 2: Potential Model Extension problems for Future Work

6.1.1 A.2.1 Time at which each load becomes available for pick-up were known

The problem formulation can be modified to include equation A.1 as an additional constraint:

$$\sum_{t=1}^{T} tY_{ijt} \geq a_i \qquad \text{for i} \in \text{H}, \ \forall \text{j} \qquad \textbf{\textit{A target can be picked only after the load becomes available for pick-up}} \qquad \text{(Equation A.1)}$$

6.1.2 A.2.2 Delivery due time of each load to its destination were known

The problem formulation can be modified to include equation A.2 as an additional constraint:

$$\sum_{j=1}^{J} \sum_{t=1}^{T} tZ_{ijt} \leq d_i \qquad \text{for i} \in \text{H} \qquad \textbf{\textit{A target has to be delivered before its due time}} \qquad \text{(Equation A.2)}$$

A few sample problems were solved to demonstrate the applicability potential of developed mathematical formulation for the above model extension problems.

6.1.3 A.2.3 Model Extension – Due Time Constraint

Constraints added to the original formulation include due time requirements, which restricts the target to be delivered on or after its due time. The feasibility of the approach is demonstrated using a 6*6 grid with 4 targets and 3 AGV. The sample problem specifications are outlined in Table A1.

Table A1: Model Extension - Due Time Constraint - Sample Problem Specifications

Target Location	Destination Location	Due Time
24	33	9
34	26	14
10	22	14
20	12	18

The OPL program statistics and the model solution are presented in Table A2 below.

Table A2: Model Extension - Due Time Constraint - Sample Problem - Routing Solution

OPL Computation Time	1125 secs	**No. of Iterations**	5,438,246
Assignments:			

Time	**AGV # (Location)**		
	1	**2**	**3**
1	23	33	8
2	23	32	5
3	24	29	4
4	25	30	9
5	26	19	10
6	35	20	3
7	34	21	2
8	33	22	11
9	34	23	12
10	35	24	13
11	26	25	24
12	25	24	23
13	24	13	22
14	25	12	23

Target Pick-Up **Target Delivery**

6.1.4 A.2.4 Model Extension – Pick-up Constraint

Constraints added to the original formulation include load pick-up availability, which says that the target can be picked up only after a certain time. The feasibility of the approach is demonstrated using a 4*4 grid with 4 targets and 2 AGV. The sample problem specifications are outlined in Table A3.

Table A3: Model Extension – Pick-Up Constraint - Sample Problem Specifications

Target Location	Destination Location	Pick-Up Availability - Time
11	9	1
5	2	1
15	13	4
4	10	5

The OPL program statistics and the model solution are presented in Table A4 below.

Table A4: Model Extension – Pick-Up Constraint - Sample Problem - Routing Solution

OPL Computation Time	82 secs	
Assignments:		
Time	**AGV # (Location)**	
	1	**2**
1	10	5
2	11	4
3	10	3
4	9	2
5	16	3
6	15	4
7	14	5
8	13	12
9	34	11
10	35	10

Target Pick-Up **Target Delivery**

A.3 Appendix 3: Experimental Studies – Raw Data & Calculations

6.1.5 A.3.1 Experimental Set # 1: Non-Independent Responses

Table A5: Experimental Set # 1 – Data Analysis

Obs.	k	b	p_{kb}	$\sum_{k} p_{kb}$	$\mathbf{K}^{-1}\sum_{k} p_{kb}$	$\sum_{b} p_{kb}$	$\mathbf{B}^{-1}\sum_{b} p_{kb}$	$(\mathbf{BK})^{-1}\sum_{k,b} p_{kb}$
1	1	1	2	10	2.0000	131	2.7292	2.3167
2	2	1	2	10	2.0000	103	2.1458	2.3167
3	3	1	2	10	2.0000	146	3.0417	2.3167
4	4	1	2	10	2.0000	107	2.2292	2.3167
5	5	1	2	10	2.0000	69	1.4375	2.3167
6	1	2	1	6	1.2000	131	2.7292	2.3167
7	2	2	1	6	1.2000	103	2.1458	2.3167
8	3	2	2	6	1.2000	146	3.0417	2.3167
9	4	2	1	6	1.2000	107	2.2292	2.3167
10	5	2	1	6	1.2000	69	1.4375	2.3167
11	1	3	2	12	2.4000	131	2.7292	2.3167
12	2	3	3	12	2.4000	103	2.1458	2.3167
13	3	3	2	12	2.4000	146	3.0417	2.3167
14	4	3	3	12	2.4000	107	2.2292	2.3167
15	5	3	2	12	2.4000	69	1.4375	2.3167
16	1	4	8	23	4.6000	131	2.7292	2.3167
17	2	4	1	23	4.6000	103	2.1458	2.3167
18	3	4	10	23	4.6000	146	3.0417	2.3167
19	4	4	3	23	4.6000	107	2.2292	2.3167
20	5	4	1	23	4.6000	69	1.4375	2.3167
21	1	5	6	22	4.4000	131	2.7292	2.3167
22	2	5	4	22	4.4000	103	2.1458	2.3167
23	3	5	4	22	4.4000	146	3.0417	2.3167
24	4	5	4	22	4.4000	107	2.2292	2.3167
25	5	5	4	22	4.4000	69	1.4375	2.3167

Table A5: Experimental Set # 1 – Data Analysis

Obs.	k	b	p_{kb}	$\sum_{k} p_{kb}$	$K^{-1}\sum_{k} p_{kb}$	$\sum_{b} p_{kb}$	$B^{-1}\sum_{b} p_{kb}$	$(BK)^{-1}\sum_{k,b} p_{kb}$
26	1	6	3	25	5.0000	131	2.7292	2.3167
27	2	6	7	25	5.0000	103	2.1458	2.3167
28	3	6	3	25	5.0000	146	3.0417	2.3167
29	4	6	9	25	5.0000	107	2.2292	2.3167
30	5	6	3	25	5.0000	69	1.4375	2.3167
31	1	7	3	15	3.0000	131	2.7292	2.3167
32	2	7	3	15	3.0000	103	2.1458	2.3167
33	3	7	3	15	3.0000	146	3.0417	2.3167
34	4	7	3	15	3.0000	107	2.2292	2.3167
35	5	7	3	15	3.0000	69	1.4375	2.3167
36	1	8	3	14	2.8000	131	2.7292	2.3167
37	2	8	2	14	2.8000	103	2.1458	2.3167
38	3	8	3	14	2.8000	146	3.0417	2.3167
39	4	8	2	14	2.8000	107	2.2292	2.3167
40	5	8	2	14	2.8000	69	1.4375	2.3167
41	1	9	0	0	0.0000	131	2.7292	2.3167
42	2	9	0	0	0.0000	103	2.1458	2.3167
43	3	9	0	0	0.0000	146	3.0417	2.3167
44	4	9	0	0	0.0000	107	2.2292	2.3167
45	5	9	0	0	0.0000	69	1.4375	2.3167
46	1	10	3	15	3.0000	131	2.7292	2.3167
47	2	10	3	15	3.0000	103	2.1458	2.3167
48	3	10	3	15	3.0000	146	3.0417	2.3167
49	4	10	3	15	3.0000	107	2.2292	2.3167
50	5	10	3	15	3.0000	69	1.4375	2.3167
51	1	11	2	8	1.6000	131	2.7292	2.3167
52	2	11	2	8	1.6000	103	2.1458	2.3167
53	3	11	2	8	1.6000	146	3.0417	2.3167
54	4	11	1	8	1.6000	107	2.2292	2.3167

Table A5: Experimental Set # 1 – Data Analysis

Obs.	k	b	p_{kb}	$\sum_{k} p_{kb}$	$K^{-1}\sum_{k} p_{kb}$	$\sum_{b} p_{kb}$	$B^{-1}\sum_{b} p_{kb}$	$(BK)^{-1}\sum_{k,b} p_{kb}$
55	5	11	1	8	1.6000	69	1.4375	2.3167
56	1	12	2	19	3.8000	131	2.7292	2.3167
57	2	12	6	19	3.8000	103	2.1458	2.3167
58	3	12	6	19	3.8000	146	3.0417	2.3167
59	4	12	3	19	3.8000	107	2.2292	2.3167
60	5	12	2	19	3.8000	69	1.4375	2.3167
61	1	13	0	0	0.0000	131	2.7292	2.3167
62	2	13	0	0	0.0000	103	2.1458	2.3167
63	3	13	0	0	0.0000	146	3.0417	2.3167
64	4	13	0	0	0.0000	107	2.2292	2.3167
65	5	13	0	0	0.0000	69	1.4375	2.3167
66	1	14	0	3	0.6000	131	2.7292	2.3167
67	2	14	1	3	0.6000	103	2.1458	2.3167
68	3	14	0	3	0.6000	146	3.0417	2.3167
69	4	14	2	3	0.6000	107	2.2292	2.3167
70	5	14	0	3	0.6000	69	1.4375	2.3167
71	1	15	0	1	0.2000	131	2.7292	2.3167
72	2	15	0	1	0.2000	103	2.1458	2.3167
73	3	15	0	1	0.2000	146	3.0417	2.3167
74	4	15	1	1	0.2000	107	2.2292	2.3167
75	5	15	0	1	0.2000	69	1.4375	2.3167
76	1	16	3	10	2.0000	131	2.7292	2.3167
77	2	16	1	10	2.0000	103	2.1458	2.3167
78	3	16	3	10	2.0000	146	3.0417	2.3167
79	4	16	2	10	2.0000	107	2.2292	2.3167
80	5	16	1	10	2.0000	69	1.4375	2.3167
81	1	17	3	11	2.2000	131	2.7292	2.3167
82	2	17	4	11	2.2000	103	2.1458	2.3167
83	3	17	4	11	2.2000	146	3.0417	2.3167

Table A5: Experimental Set # 1 – Data Analysis

Obs.	k	b	p_{kb}	$\sum_{k} p_{kb}$	$K^{-1}\sum_{k} p_{kb}$	$\sum_{b} p_{kb}$	$B^{-1}\sum_{b} p_{kb}$	$(BK)^{-1}\sum_{k,b} p_{kb}$
84	4	17	0	11	2.2000	107	2.2292	2.3167
85	5	17	0	11	2.2000	69	1.4375	2.3167
86	1	18	5	19	3.8000	131	2.7292	2.3167
87	2	18	1	19	3.8000	103	2.1458	2.3167
88	3	18	5	19	3.8000	146	3.0417	2.3167
89	4	18	7	19	3.8000	107	2.2292	2.3167
90	5	18	1	19	3.8000	69	1.4375	2.3167
91	1	19	6	25	5.0000	131	2.7292	2.3167
92	2	19	4	25	5.0000	103	2.1458	2.3167
93	3	19	6	25	5.0000	146	3.0417	2.3167
94	4	19	5	25	5.0000	107	2.2292	2.3167
95	5	19	4	25	5.0000	69	1.4375	2.3167
96	1	20	4	15	3.0000	131	2.7292	2.3167
97	2	20	2	15	3.0000	103	2.1458	2.3167
98	3	20	5	15	3.0000	146	3.0417	2.3167
99	4	20	2	15	3.0000	107	2.2292	2.3167
100	5	20	2	15	3.0000	69	1.4375	2.3167
101	1	21	1	8	1.6000	131	2.7292	2.3167
102	2	21	1	8	1.6000	103	2.1458	2.3167
103	3	21	3	8	1.6000	146	3.0417	2.3167
104	4	21	2	8	1.6000	107	2.2292	2.3167
105	5	21	1	8	1.6000	69	1.4375	2.3167
106	1	22	4	17	3.4000	131	2.7292	2.3167
107	2	22	5	17	3.4000	103	2.1458	2.3167
108	3	22	4	17	3.4000	146	3.0417	2.3167
109	4	22	2	17	3.4000	107	2.2292	2.3167
110	5	22	2	17	3.4000	69	1.4375	2.3167
111	1	23	2	4	0.8000	131	2.7292	2.3167
112	2	23	1	4	0.8000	103	2.1458	2.3167

Table A5: Experimental Set # 1 – Data Analysis

Obs.	k	b	p_{kb}	$\sum_{k} p_{kb}$	$K^{-1}\sum_{k} p_{kb}$	$\sum_{b} p_{kb}$	$B^{-1}\sum_{b} p_{kb}$	$(BK)^{-1}\sum_{k,b} p_{kb}$
113	3	23	1	4	0.8000	146	3.0417	2.3167
114	4	23	0	4	0.8000	107	2.2292	2.3167
115	5	23	0	4	0.8000	69	1.4375	2.3167
116	1	24	6	21	4.2000	131	2.7292	2.3167
117	2	24	3	21	4.2000	103	2.1458	2.3167
118	3	24	4	21	4.2000	146	3.0417	2.3167
119	4	24	5	21	4.2000	107	2.2292	2.3167
120	5	24	3	21	4.2000	69	1.4375	2.3167
121	1	25	0	1	0.2000	131	2.7292	2.3167
122	2	25	0	1	0.2000	103	2.1458	2.3167
123	3	25	1	1	0.2000	146	3.0417	2.3167
124	4	25	0	1	0.2000	107	2.2292	2.3167
125	5	25	0	1	0.2000	69	1.4375	2.3167
126	1	26	5	12	2.4000	131	2.7292	2.3167
127	2	26	0	12	2.4000	103	2.1458	2.3167
128	3	26	5	12	2.4000	146	3.0417	2.3167
129	4	26	2	12	2.4000	107	2.2292	2.3167
130	5	26	0	12	2.4000	69	1.4375	2.3167
131	1	27	0	2	0.4000	131	2.7292	2.3167
132	2	27	1	2	0.4000	103	2.1458	2.3167
133	3	27	0	2	0.4000	146	3.0417	2.3167
134	4	27	1	2	0.4000	107	2.2292	2.3167
135	5	27	0	2	0.4000	69	1.4375	2.3167
136	1	28	6	20	4.0000	131	2.7292	2.3167
137	2	28	4	20	4.0000	103	2.1458	2.3167
138	3	28	6	20	4.0000	146	3.0417	2.3167
139	4	28	2	20	4.0000	107	2.2292	2.3167
140	5	28	2	20	4.0000	69	1.4375	2.3167
141	1	29	9	29	5.8000	131	2.7292	2.3167

Table A5: Experimental Set # 1 – Data Analysis

Obs.	k	b	p_{kb}	$\sum_{k} p_{kb}$	$K^{-1}\sum_{k} p_{kb}$	$\sum_{b} p_{kb}$	$B^{-1}\sum_{b} p_{kb}$	$(BK)^{-1}\sum_{k,b} p_{kb}$
142	2	29	3	29	5.8000	103	2.1458	2.3167
143	3	29	9	29	5.8000	146	3.0417	2.3167
144	4	29	5	29	5.8000	107	2.2292	2.3167
145	5	29	3	29	5.8000	69	1.4375	2.3167
146	1	30	4	20	4.0000	131	2.7292	2.3167
147	2	30	4	20	4.0000	103	2.1458	2.3167
148	3	30	4	20	4.0000	146	3.0417	2.3167
149	4	30	4	20	4.0000	107	2.2292	2.3167
150	5	30	4	20	4.0000	69	1.4375	2.3167
151	1	31	0	0	0.0000	131	2.7292	2.3167
152	2	31	0	0	0.0000	103	2.1458	2.3167
153	3	31	0	0	0.0000	146	3.0417	2.3167
154	4	31	0	0	0.0000	107	2.2292	2.3167
155	5	31	0	0	0.0000	69	1.4375	2.3167
156	1	32	2	9	1.8000	131	2.7292	2.3167
157	2	32	3	9	1.8000	103	2.1458	2.3167
158	3	32	2	9	1.8000	146	3.0417	2.3167
159	4	32	1	9	1.8000	107	2.2292	2.3167
160	5	32	1	9	1.8000	69	1.4375	2.3167
161	1	33	2	14	2.8000	131	2.7292	2.3167
162	2	33	3	14	2.8000	103	2.1458	2.3167
163	3	33	4	14	2.8000	146	3.0417	2.3167
164	4	33	3	14	2.8000	107	2.2292	2.3167
165	5	33	2	14	2.8000	69	1.4375	2.3167
166	1	34	4	17	3.4000	131	2.7292	2.3167
167	2	34	2	17	3.4000	103	2.1458	2.3167
168	3	34	3	17	3.4000	146	3.0417	2.3167
169	4	34	6	17	3.4000	107	2.2292	2.3167
170	5	34	2	17	3.4000	69	1.4375	2.3167

Table A5: Experimental Set # 1 – Data Analysis

Obs.	k	b	p_{kb}	$\sum_{k} p_{kb}$	$K^{-1}\sum_{k} p_{kb}$	$\sum_{b} p_{kb}$	$B^{-1}\sum_{b} p_{kb}$	$(BK)^{-1}\sum_{k,b} p_{kb}$
171	1	35	5	14	2.8000	131	2.7292	2.3167
172	2	35	2	14	2.8000	103	2.1458	2.3167
173	3	35	5	14	2.8000	146	3.0417	2.3167
174	4	35	1	14	2.8000	107	2.2292	2.3167
175	5	35	1	14	2.8000	69	1.4375	2.3167
176	1	36	2	14	2.8000	131	2.7292	2.3167
177	2	36	3	14	2.8000	103	2.1458	2.3167
178	3	36	5	14	2.8000	146	3.0417	2.3167
179	4	36	2	14	2.8000	107	2.2292	2.3167
180	5	36	2	14	2.8000	69	1.4375	2.3167
181	1	37	0	2	0.4000	131	2.7292	2.3167
182	2	37	2	2	0.4000	103	2.1458	2.3167
183	3	37	0	2	0.4000	146	3.0417	2.3167
184	4	37	0	2	0.4000	107	2.2292	2.3167
185	5	37	0	2	0.4000	69	1.4375	2.3167
186	1	38	0	0	0.0000	131	2.7292	2.3167
187	2	38	0	0	0.0000	103	2.1458	2.3167
188	3	38	0	0	0.0000	146	3.0417	2.3167
189	4	38	0	0	0.0000	107	2.2292	2.3167
190	5	38	0	0	0.0000	69	1.4375	2.3167
191	1	39	0	2	0.4000	131	2.7292	2.3167
192	2	39	1	2	0.4000	103	2.1458	2.3167
193	3	39	0	2	0.4000	146	3.0417	2.3167
194	4	39	1	2	0.4000	107	2.2292	2.3167
195	5	39	0	2	0.4000	69	1.4375	2.3167
196	1	40	2	8	1.6000	131	2.7292	2.3167
197	2	40	1	8	1.6000	103	2.1458	2.3167
198	3	40	2	8	1.6000	146	3.0417	2.3167
199	4	40	2	8	1.6000	107	2.2292	2.3167

Table A5: Experimental Set # 1 – Data Analysis

Obs.	k	b	p_{kb}	$\sum_{k} p_{kb}$	$K^{-1}\sum_{k} p_{kb}$	$\sum_{b} p_{kb}$	$B^{-1}\sum_{b} p_{kb}$	$(BK)^{-1}\sum_{k,b} p_{kb}$
200	5	40	1	8	1.6000	69	1.4375	2.3167
201	1	41	6	18	3.6000	131	2.7292	2.3167
202	2	41	2	18	3.6000	103	2.1458	2.3167
203	3	41	6	18	3.6000	146	3.0417	2.3167
204	4	41	2	18	3.6000	107	2.2292	2.3167
205	5	41	2	18	3.6000	69	1.4375	2.3167
206	1	42	1	7	1.4000	131	2.7292	2.3167
207	2	42	2	7	1.4000	103	2.1458	2.3167
208	3	42	2	7	1.4000	146	3.0417	2.3167
209	4	42	1	7	1.4000	107	2.2292	2.3167
210	5	42	1	7	1.4000	69	1.4375	2.3167
211	1	43	3	13	2.6000	131	2.7292	2.3167
212	2	43	2	13	2.6000	103	2.1458	2.3167
213	3	43	4	13	2.6000	146	3.0417	2.3167
214	4	43	2	13	2.6000	107	2.2292	2.3167
215	5	43	2	13	2.6000	69	1.4375	2.3167
216	1	44	3	15	3.0000	131	2.7292	2.3167
217	2	44	3	15	3.0000	103	2.1458	2.3167
218	3	44	3	15	3.0000	146	3.0417	2.3167
219	4	44	3	15	3.0000	107	2.2292	2.3167
220	5	44	3	15	3.0000	69	1.4375	2.3167
221	1	45	1	1	0.2000	131	2.7292	2.3167
222	2	45	0	1	0.2000	103	2.1458	2.3167
223	3	45	0	1	0.2000	146	3.0417	2.3167
224	4	45	0	1	0.2000	107	2.2292	2.3167
225	5	45	0	1	0.2000	69	1.4375	2.3167
226	1	46	3	13	2.6000	131	2.7292	2.3167
227	2	46	3	13	2.6000	103	2.1458	2.3167
228	3	46	2	13	2.6000	146	3.0417	2.3167

Table A5: Experimental Set # 1 – Data Analysis

Obs.	k	b	p_{kb}	$\sum_{k} p_{kb}$	$\mathbf{K^{-1}}\sum_{k} p_{kb}$	$\sum_{b} p_{kb}$	$\mathbf{B^{-1}}\sum_{b} p_{kb}$	$\mathbf{(BK)^{-1}}\sum_{k,b} p_{kb}$
229	4	46	3	13	2.6000	107	2.2292	2.3167
230	5	46	2	13	2.6000	69	1.4375	2.3167
231	1	47	2	8	1.6000	131	2.7292	2.3167
232	2	47	2	8	1.6000	103	2.1458	2.3167
233	3	47	2	8	1.6000	146	3.0417	2.3167
234	4	47	1	8	1.6000	107	2.2292	2.3167
235	5	47	1	8	1.6000	69	1.4375	2.3167
236	1	48	2	15	3.0000	131	2.7292	2.3167
237	2	48	3	15	3.0000	103	2.1458	2.3167
238	3	48	5	15	3.0000	146	3.0417	2.3167
239	4	48	3	15	3.0000	107	2.2292	2.3167
240	5	48	2	15	3.0000	69	1.4375	2.3167

6.1.6 A.3.2 Experimental Set # 1B: Non-Independent Responses – Matched Pair Heuristic Comparison – Raw Data

Table A6: Nearest Neighbor vs. Greedy Algorithms				
Expt #	**Nearest Neighbor**	**Greedy**	**Difference**	**(db - d bar)^2**
1	11	11	0.0000	0.3651
2	14	14	0.0000	0.3651
3	16	17	-1.0000	2.5735
4	20	13	7.0000	40.9063
5	29	27	2.0000	1.9483
6	23	27	-4.0000	21.1987
7	9	9	0.0000	0.3651
8	11	10	1.0000	0.1567
9	9	9	0.0000	0.3651
10	12	12	0.0000	0.3651
11	19	19	0.0000	0.3651
12	20	24	-4.0000	21.1987
13	6	6	0.0000	0.3651
14	13	14	-1.0000	2.5735
15	16	16	0.0000	0.3651
16	16	14	2.0000	1.9483
17	27	28	-1.0000	2.5735
18	33	29	4.0000	11.5315
19	12	10	2.0000	1.9483
20	15	13	2.0000	1.9483
21	14	14	0.0000	0.3651
22	13	13	0.0000	0.3651
23	20	19	1.0000	0.1567
24	22	19	3.0000	5.7399
25	9	9	0.0000	0.3651
26	17	12	5.0000	19.3231

Table A6 (continued): Nearest Neighbor vs. Greedy Algorithms				
Expt #	**Nearest Neighbor**	**Greedy**	**Difference**	**(db - d bar)^2**
27	10	11	-1.0000	2.5735
28	20	18	2.0000	1.9483
29	30	24	6.0000	29.1147
30	27	27	0.0000	0.3651
31	9	9	0.0000	0.3651
32	14	15	-1.0000	2.5735
33	13	14	-1.0000	2.5735
34	13	11	2.0000	1.9483
35	20	17	3.0000	5.7399
36	20	21	-1.0000	2.5735
37	8	10	-2.0000	6.7819
38	12	12	0.0000	0.3651
39	17	18	-1.0000	2.5735
40	12	11	1.0000	0.1567
41	29	25	4.0000	11.5315
42	13	14	-1.0000	2.5735
43	10	9	1.0000	0.1567
44	16	16	0.0000	0.3651
45	13	12	1.0000	0.1567
46	13	13	0.0000	0.3651
47	19	19	0.0000	0.3651
48	20	21	-1.0000	2.5735
		Average d bar	**0.6042**	
		Sum		**217.4792**
		$(B-1)^{-1}$		**0.0213**
		Denominator		**2.1511**
		Numerator		**4.1858**
		t value		**1.9459**
		P-VALUE		**0.0577**

Table A7: Nearest Neighbor vs. Least Utilized AGV Algorithms				
Expt #	**Nearest Neighbor**	**Least Utilized AGV**	**Difference**	**(db - d bar)^2**
1	11	11	0	0.0977
2	14	15	-1	0.4727
3	16	16	0	0.0977
4	20	22	-2	2.8477
5	29	27	2	5.3477
6	23	23	0	0.0977
7	9	9	0	0.0977
8	11	11	0	0.0977
9	9	9	0	0.0977
10	12	12	0	0.0977
11	19	19	0	0.0977
12	20	24	-4	13.5977
13	6	6	0	0.0977
14	13	13	0	0.0977
15	16	16	0	0.0977
16	16	16	0	0.0977
17	27	28	-1	0.4727
18	33	33	0	0.0977
19	12	12	0	0.0977
20	15	16	-1	0.4727
21	14	16	-2	2.8477
22	13	13	0	0.0977
23	20	20	0	0.0977
24	22	20	2	5.3477
25	9	10	-1	0.4727
26	17	17	0	0.0977
27	10	10	0	0.0977
28	20	20	0	0.0977
29	30	30	0	0.0977

Table A7 (continued): Nearest Neighbor vs. Least Utilized AGV Algorithms				
Expt #	**Nearest Neighbor**	**Least Utilized AGV**	**Difference**	**(db - d bar)^2**
30	27	27	0	0.0977
31	9	9	0	0.0977
32	14	14	0	0.0977
33	13	15	-2	2.8477
34	13	12	1	1.7227
35	20	20	0	0.0977
36	20	23	-3	7.2227
37	8	8	0	0.0977
38	12	12	0	0.0977
39	17	17	0	0.0977
40	12	12	0	0.0977
41	29	29	0	0.0977
42	13	14	-1	0.4727
43	10	11	-1	0.4727
44	16	16	0	0.0977
45	13	12	1	1.7227
46	13	12	1	1.7227
47	19	19	0	0.0977
48	20	23	-3	7.2227
		Average d bar	**-0.3125**	
		Sum		**58.3125**
		$(B-1)^{-1}$		**0.0213**
		Denominator		**1.1139**
		Numerator		**-2.1651**
		t value		**-1.9437**
		P-VALUE		**0.0579**

Table A8: Nearest Neighbor vs. Modified Greedy Algorithms				
Expt #	**Nearest Neighbor**	**Modified Greedy**	**Difference**	**(db - d bar)^2**
1	11	11	0	0.2500
2	14	14	0	0.2500
3	16	17	-1	2.2500
4	20	15	5	20.2500
5	29	27	2	2.2500
6	23	29	-6	42.2500
7	9	9	0	0.2500
8	11	10	1	0.2500
9	9	9	0	0.2500
10	12	12	0	0.2500
11	19	18	1	0.2500
12	20	21	-1	2.2500
13	6	6	0	0.2500
14	13	15	-2	6.2500
15	16	17	-1	2.2500
16	16	15	1	0.2500
17	27	24	3	6.2500
18	33	35	-2	6.2500
19	12	11	1	0.2500
20	15	13	2	2.2500
21	14	15	-1	2.2500
22	13	11	2	2.2500
23	20	18	2	2.2500
24	22	21	1	0.2500
25	9	9	0	0.2500
26	17	14	3	6.2500
27	10	11	-1	2.2500
28	20	16	4	12.2500
29	30	26	4	12.2500

Table A8 (continued): Nearest Neighbor vs. Modified Greedy Algorithms				
Expt #	**Nearest Neighbor**	**Modified Greedy**	**Difference**	**(db - d bar)^2**
30	27	27	0	0.2500
31	9	9	0	0.2500
32	14	13	1	0.2500
33	13	14	-1	2.2500
34	13	15	-2	6.2500
35	20	16	4	12.2500
36	20	20	0	0.2500
37	8	8	0	0.2500
38	12	12	0	0.2500
39	17	18	-1	2.2500
40	12	12	0	0.2500
41	29	25	4	12.2500
42	13	13	0	0.2500
43	10	9	1	0.2500
44	16	16	0	0.2500
45	13	12	1	0.2500
46	13	13	0	0.2500
47	19	18	1	0.2500
48	20	21	-1	2.2500
		Average d bar	0.5000	
		Sum		176.0000
		$(B-1)^{-1}$		0.0213
		Denominator		1.9351
		Numerator		3.4641
		t value		1.7901
		P-VALUE		0.0799

Table A9: Nearest Neighbor vs. Combination Heuristic Algorithms				
Expt #	**Nearest Neighbor**	**Combination Heuristic**	**Difference**	**(db - d bar)^2**
1	11	11	0	1.6685
2	14	14	0	1.6685
3	16	16	0	1.6685
4	20	13	7	32.5847
5	29	27	2	0.5017
6	23	23	0	1.6685
7	9	9	0	1.6685
8	11	10	1	0.0851
9	9	9	0	1.6685
10	12	12	0	1.6685
11	19	18	1	0.0851
12	20	20	0	1.6685
13	6	6	0	1.6685
14	13	13	0	1.6685
15	16	16	0	1.6685
16	16	14	2	0.5017
17	27	24	3	2.9183
18	33	29	4	7.3349
19	12	10	2	0.5017
20	15	13	2	0.5017
21	14	14	0	1.6685
22	13	11	2	0.5017
23	20	18	2	0.5017
24	22	19	3	2.9183
25	9	9	0	1.6685
26	17	12	5	13.7515
27	10	10	0	1.6685
28	20	16	4	7.3349
29	30	24	6	22.1681

Table A9 (continued): Nearest Neighbor vs. Combination Heuristic Algorithms				
Expt #	**Nearest Neighbor**	**Combination Heuristic**	**Difference**	**(db - d bar)^2**
30	27	27	0	1.6685
31	9	9	0	1.6685
32	14	13	1	0.0851
33	13	13	0	1.6685
34	13	11	2	0.5017
35	20	16	4	7.3349
36	20	20	0	1.6685
37	8	8	0	1.6685
38	12	12	0	1.6685
39	17	17	0	1.6685
40	12	11	1	0.0851
41	29	25	4	7.3349
42	13	13	0	1.6685
43	10	9	1	0.0851
44	16	16	0	1.6685
45	13	12	1	0.0851
46	13	12	1	0.0851
47	19	18	1	0.0851
48	20	20	0	1.6685
		Average d bar	1.2917	
		Sum		147.9167
		$(B-1)^{-1}$		0.0213
		Denominator		1.7740
		Numerator		8.9489
		t value		5.0444
		P-VALUE		< 0.0001

Table A10: Greedy vs. Least Utilized AGV Algorithms

Expt #	Greedy	Least Utilized AGV	Difference	(db - d bar)^2
1	11	11	0	0.8403
2	14	15	-1	0.0069
3	17	16	1	3.6737
4	13	22	-9	65.3397
5	27	27	0	0.8403
6	27	23	4	24.1739
7	9	9	0	0.8403
8	10	11	-1	0.0069
9	9	9	0	0.8403
10	12	12	0	0.8403
11	19	19	0	0.8403
12	24	24	0	0.8403
13	6	6	0	0.8403
14	14	13	1	3.6737
15	16	16	0	0.8403
16	14	16	-2	1.1735
17	28	28	0	0.8403
18	29	33	-4	9.5067
19	10	12	-2	1.1735
20	13	16	-3	4.3401
21	14	16	-2	1.1735
22	13	13	0	0.8403
23	19	20	-1	0.0069
24	19	20	-1	0.0069
25	9	10	-1	0.0069
26	12	17	-5	16.6733
27	11	10	1	3.6737
28	18	20	-2	1.1735
29	24	30	-6	25.8399

Table A10 (continued): Greedy vs. Least Utilized AGV Algorithms				
Expt #	**Greedy**	**Least Utilized AGV**	**Difference**	**(db - d bar)^2**
30	27	27	0	0.8403
31	9	9	0	0.8403
32	15	14	1	3.6737
33	14	15	-1	0.0069
34	11	12	-1	0.0069
35	17	20	-3	4.3401
36	21	23	-2	1.1735
37	10	8	2	8.5071
38	12	12	0	0.8403
39	18	17	1	3.6737
40	11	12	-1	0.0069
41	25	29	-4	9.5067
42	14	14	0	0.8403
43	9	11	-2	1.1735
44	16	16	0	0.8403
45	12	12	0	0.8403
46	13	12	1	3.6737
47	19	19	0	0.8403
48	21	23	-2	1.1735
		Average d bar	-0.9167	
		Sum		213.6667
		$(B-1)^{-1}$		0.0213
		Denominator		2.1322
		Numerator		-6.3509
		t value		-2.9786
		P-VALUE		0.0046

Table A11: Greedy vs. Modified Greedy Algorithms

Expt #	Greedy	Modified Greedy	Difference	(db - d bar)^2
1	11	11	0	0.0109
2	14	14	0	0.0109
3	17	17	0	0.0109
4	13	15	-2	3.5941
5	27	27	0	0.0109
6	27	29	-2	3.5941
7	9	9	0	0.0109
8	10	10	0	0.0109
9	9	9	0	0.0109
10	12	12	0	0.0109
11	19	18	1	1.2193
12	24	21	3	9.6361
13	6	6	0	0.0109
14	14	15	-1	0.8025
15	16	17	-1	0.8025
16	14	15	-1	0.8025
17	28	24	4	16.8445
18	29	35	-6	34.7605
19	10	11	-1	0.8025
20	13	13	0	0.0109
21	14	15	-1	0.8025
22	13	11	2	4.4277
23	19	18	1	1.2193
24	19	21	-2	3.5941
25	9	9	0	0.0109
26	12	14	-2	3.5941
27	11	11	0	0.0109
28	18	16	2	4.4277

Table A11 (continued): Greedy vs. Modified Greedy Algorithms

29	24	26	-2	3.5941
30	27	27	0	0.0109
31	9	9	0	0.0109
32	15	13	2	4.4277
33	14	14	0	0.0109
34	11	15	-4	15.1773
35	17	16	1	1.2193
36	21	20	1	1.2193
37	10	8	2	4.4277
38	12	12	0	0.0109
39	18	18	0	0.0109
40	11	12	-1	0.8025
41	25	25	0	0.0109
42	14	13	1	1.2193
43	9	9	0	0.0109
44	16	16	0	0.0109
45	12	12	0	0.0109
46	13	13	0	0.0109
47	19	18	1	1.2193
48	21	21	0	0.0109
		Average d bar	-0.1042	
		Sum		124.4792
		$(B-1)^{-1}$		0.0213
		Denominator		1.6274
		Numerator		-0.7217
		t value		-0.4435
		P-VALUE		0.6594

Table A12: Greedy vs. Combination Heuristic Algorithms

Expt #	Greedy	Combination Heuristic	Difference	(db - d bar)^2
1	11	11	0	0.4727
2	14	14	0	0.4727
3	17	16	1	0.0977
4	13	13	0	0.4727
5	27	27	0	0.4727
6	27	23	4	10.9727
7	9	9	0	0.4727
8	10	10	0	0.4727
9	9	9	0	0.4727
10	12	12	0	0.4727
11	19	18	1	0.0977
12	24	20	4	10.9727
13	6	6	0	0.4727
14	14	13	1	0.0977
15	16	16	0	0.4727
16	14	14	0	0.4727
17	28	24	4	10.9727
18	29	29	0	0.4727
19	10	10	0	0.4727
20	13	13	0	0.4727
21	14	14	0	0.4727
22	13	11	2	1.7227
23	19	18	1	0.0977
24	19	19	0	0.4727
25	9	9	0	0.4727
26	12	12	0	0.4727
27	11	10	1	0.0977
28	18	16	2	1.7227

Table A12 (continued): Greedy vs. Combination Heuristic Algorithms				
29	24	24	0	0.4727
30	27	27	0	0.4727
31	9	9	0	0.4727
32	15	13	2	1.7227
33	14	13	1	0.0977
34	11	11	0	0.4727
35	17	16	1	0.0977
36	21	20	1	0.0977
37	10	8	2	1.7227
38	12	12	0	0.4727
39	18	17	1	0.0977
40	11	11	0	0.4727
41	25	25	0	0.4727
42	14	13	1	0.0977
43	9	9	0	0.4727
44	16	16	0	0.4727
45	12	12	0	0.4727
46	13	12	1	0.0977
47	19	18	1	0.0977
48	21	20	1	0.0977
		Average d bar	0.6875	
		Sum		54.3125
		(B-1)		0.0213
		Denominator		1.0750
		Numerator		4.7631
		t value		
		P-VALUE		0.0122

Table A13: Least Utilized AGV vs. Modified Greedy Algorithms

Expt #	Least Utilized AGV	Modified Greedy	Difference	(db - d bar)^2
1	11	11	0	0.6602
2	15	14	1	0.0352
3	16	17	-1	3.2852
4	22	15	7	38.2852
5	27	27	0	0.6602
6	23	29	-6	46.4102
7	9	9	0	0.6602
8	11	10	1	0.0352
9	9	9	0	0.6602
10	12	12	0	0.6602
11	19	18	1	0.0352
12	24	21	3	4.7852
13	6	6	0	0.6602
14	13	15	-2	7.9102
15	16	17	-1	3.2852
16	16	15	1	0.0352
17	28	24	4	10.1602
18	33	35	-2	7.9102
19	12	11	1	0.0352
20	16	13	3	4.7852
21	16	15	1	0.0352
22	13	11	2	1.4102
23	20	18	2	1.4102
24	20	21	-1	3.2852
25	10	9	1	0.0352
26	17	14	3	4.7852
27	10	11	-1	3.2852
28	20	16	4	10.1602

Table A13 (continued): Least Utilized AGV vs. Modified Greedy Algorithms				
29	30	26	4	10.1602
30	27	27	0	0.6602
31	9	9	0	0.6602
32	14	13	1	0.0352
33	15	14	1	0.0352
34	12	15	-3	14.5352
35	20	16	4	10.1602
36	23	20	3	4.7852
37	8	8	0	0.6602
38	12	12	0	0.6602
39	17	18	-1	3.2852
40	12	12	0	0.6602
41	29	25	4	10.1602
42	14	13	1	0.0352
43	11	9	2	1.4102
44	16	16	0	0.6602
45	12	12	0	0.6602
46	12	13	-1	3.2852
47	19	18	1	0.0352
48	23	21	2	1.4102
		Average d bar	0.8125	
		Sum		219.3125
		$(B-1)^{-1}$		0.0213
		Denominator		2.1601
		Numerator		5.6292
		t value		2.6059
		P-VALUE		0.0122

Table A14: Least Utilized AGV vs. Combination Heuristic Algorithms

Expt #	Least Utilized AGV	Combination Heuristic	Difference	(db - d bar)^2
1	11	11	0	2.5735
2	15	14	1	0.3651
3	16	16	0	2.5735
4	22	13	9	54.6979
5	27	27	0	2.5735
6	23	23	0	2.5735
7	9	9	0	2.5735
8	11	10	1	0.3651
9	9	9	0	2.5735
10	12	12	0	2.5735
11	19	18	1	0.3651
12	24	20	4	5.7399
13	6	6	0	2.5735
14	13	13	0	2.5735
15	16	16	0	2.5735
16	16	14	2	0.1567
17	28	24	4	5.7399
18	33	29	4	5.7399
19	12	10	2	0.1567
20	16	13	3	1.9483
21	16	14	2	0.1567
22	13	11	2	0.1567
23	20	18	2	0.1567
24	20	19	1	0.3651
25	10	9	1	0.3651
26	17	12	5	11.5315
27	10	10	0	2.5735
28	20	16	4	5.7399

Table A14 (continued): Least Utilized AGV vs. Combination Heuristic Algorithms				
29	30	24	6	19.3231
30	27	27	0	2.5735
31	9	9	0	2.5735
32	14	13	1	0.3651
33	15	13	2	0.1567
34	12	11	1	0.3651
35	20	16	4	5.7399
36	23	20	3	1.9483
37	8	8	0	2.5735
38	12	12	0	2.5735
39	17	17	0	2.5735
40	12	11	1	0.3651
41	29	25	4	5.7399
42	14	13	1	0.3651
43	11	9	2	0.1567
44	16	16	0	2.5735
45	12	12	0	2.5735
46	12	12	0	2.5735
47	19	18	1	0.3651
48	23	20	3	1.9483
		Average d bar	1.6042	
		Sum		179.4792
		$(B-1)^{-1}$		0.0213
		Denominator		1.9542
		Numerator		11.1140
		t value		5.6874
		P-VALUE		0.0122

Table A15: Modified Greedy vs. Combination Heuristic Algorithms

Expt #	Modified Greedy	Combination Heuristic	Difference	(db - d bar)^2
1	11	11	0	0.6268
2	14	14	0	0.6268
3	17	16	1	0.0434
4	15	13	2	1.4600
5	27	27	0	0.6268
6	29	23	6	27.1264
7	9	9	0	0.6268
8	10	10	0	0.6268
9	9	9	0	0.6268
10	12	12	0	0.6268
11	18	18	0	0.6268
12	21	20	1	0.0434
13	6	6	0	0.6268
14	15	13	2	1.4600
15	17	16	1	0.0434
16	15	14	1	0.0434
17	24	24	0	0.6268
18	35	29	6	27.1264
19	11	10	1	0.0434
20	13	13	0	0.6268
21	15	14	1	0.0434
22	11	11	0	0.6268
23	18	18	0	0.6268
24	21	19	2	1.4600
25	9	9	0	0.6268
26	14	12	2	1.4600
27	11	10	1	0.0434
28	16	16	0	0.6268

Table A15 (continued): Modified Greedy vs. Combination Heuristic Algorithms				
29	26	24	2	1.4600
30	27	27	0	0.6268
31	9	9	0	0.6268
32	13	13	0	0.6268
33	14	13	1	0.0434
34	15	11	4	10.2932
35	16	16	0	0.6268
36	20	20	0	0.6268
37	8	8	0	0.6268
38	12	12	0	0.6268
39	18	17	1	0.0434
40	12	11	1	0.0434
41	25	25	0	0.6268
42	13	13	0	0.6268
43	9	9	0	0.6268
44	16	16	0	0.6268
45	12	12	0	0.6268
46	13	12	1	0.0434
47	18	18	0	0.6268
48	21	20	1	0.0434
		Average d bar	0.7917	
		Sum		89.9167
		$(B-1)^{-1}$		0.0213
		Denominator		1.3832
		Numerator		5.4848
		t value		3.9654
		P-VALUE		0.0122

6.1.7 A.3.3 Experimental Set # 2: Independent Responses

Table A16: Experimental Design for Independent Response Test

No. of AGV	No. of Targets	Heuristic	Upper Bound	Optimal	Response	AGV Initial Positions	Target Nos.	Destination Nos.
2	4	G	12	11	1	8,2	1,4,3,7	6,2,8,4
2	4	G	9	9	0	8,2	1,4,3,7	6,2,8,4
2	4	G	14	13	1	8,2	1,4,3,7	6,2,8,4
2	4	G	13	13	0	9,13	2,11,16,1	14,5,8,3
2	4	G	18	16	2	9,13	2,11,16,1	14,5,8,3
2	4	G	16	15	1	9,13	2,11,16,1	14,5,8,3
2	4	G	21	20	1	11,16	19,6,3,22	14,10,24,15
2	4	G	14	14	0	11,16	19,6,3,22	14,10,24,15
2	4	G	15	15	0	11,16	19,6,3,22	14,10,24,15
2	8	G	24	21	3	4,6	7,2,4,3,1,8,6,9	8,3,6,6,7,4,7,3
2	8	G	32	26	6	4,6	7,2,4,3,1,8,6,9	8,3,6,6,7,4,7,3
2	8	G	19	18	1	4,6	7,2,4,3,1,8,6,9	8,3,6,6,7,4,7,3
2	8	G	27	24	3	2,5	1,11,16,12,3,8,13,2	10,14,5,8,15,13,3,11
2	8	G	29	26	3	2,5	1,11,16,12,3,8,13,2	10,14,5,8,15,13,3,11
2	8	G	34	31	3	2,5	1,11,16,12,3,8,13,2	10,14,5,8,15,13,3,11
2	8	G	23	23	0	9,14	18,20,6,1,24,5,9,25	17,23,19,2,9,20,7,4
2	8	G	26	21	5	9,14	18,20,6,1,24,5,9,25	17,23,19,2,9,20,7,4

Table A16: Experimental Design for Independent Response Test

No. of AGV	No. of Targets	Heuristic	Upper Bound	Optimal	Response	AGV Initial Positions	Target Nos.	Destination Nos.
2	8	G	28	19	9	9,14	18,20,6,1,24,5,9,25	17,23,19,2,9,20,7,4
4	4	G	14	14	0	2,8,9	7,4,1,2	3,2,6,9
4	4	G	13	12	1	2,8,9	7,4,1,2	3,2,6,9
4	4	G	17	15	2	2,8,9	7,4,1,2	3,2,6,9
4	4	G	18	17	1	10,2,5	11,15,5,4	9,13,2,10
4	4	G	12	12	0	10,2,5	11,15,5,4	9,13,2,10
4	4	G	19	14	5	10,2,5	11,15,5,4	9,13,2,10
4	4	G	21	20	1	7,24,10	11,6,5,19	22,14,3,9
4	4	G	14	12	2	7,24,10	11,6,5,19	22,14,3,9
4	4	G	15	12	3	7,24,10	11,6,5,19	22,14,3,9
4	8	G	23	21	2	1,6,8	7,2,4,3,1,8,6,9	6,8,3,7,3,1,8,4
4	8	G	38	27	11	1,6,8	7,2,4,3,1,8,6,9	6,8,3,7,3,1,8,4
4	8	G	26	21	5	1,6,8	7,2,4,3,1,8,6,9	6,8,3,7,3,1,8,4
4	8	G	29	28	1	9,11,3	2,14,9,5,1,13,3,16	15,8,12,15,11,3,15,8
4	8	G	17	17	0	9,11,3	2,14,9,5,1,13,3,16	15,8,12,15,11,3,15,8
4	8	G	32	30	2	9,11,3	2,14,9,5,1,13,3,16	15,8,12,15,11,3,15,8
4	8	G	24	21	3	19,5,25	1,11,8,14,24,17,19,10	14,16,23,7,19,2,14,16
4	8	G	29	26	3	19,5,25	1,11,8,14,24,17,19,10	14,16,23,7,19,2,14,16
4	8	G	26	24	2	19,5,25	1,11,8,14,24,17,19,10	14,16,23,7,19,2,14,16

Table A16: Experimental Design for Independent Response Test

No. of AGV	No. of Targets	Heuristic	Upper Bound	Optimal	Response	AGV Initial Positions	Target Nos.	Destination Nos.
2	4	NN	11	11	0	6,8	1,4,9,7	2,3,8,8
2	4	NN	15	12	3	6,8	1,4,9,7	2,3,8,8
2	4	NN	14	13	1	6,8	1,4,9,7	2,3,8,8
2	4	NN	9	8	1	10,2	15,12,8,4	10,2,13,11
2	4	NN	17	14	3	10,2	15,12,8,4	10,2,13,11
2	4	NN	18	13	5	10,2	15,12,8,4	10,2,13,11
2	4	NN	16	15	1	19,6	1,8,15,24	22,24,11,7
2	4	NN	14	14	0	19,6	1,8,15,24	22,24,11,7
2	4	NN	20	17	3	19,6	1,8,15,24	22,24,11,7
2	8	NN	25	19	6	7,3	1,2,3,4,6,7,8,9	8,4,6,7,8,1,3,2
2	8	NN	18	14	4	7,3	1,2,3,4,6,7,8,9	8,4,6,7,8,1,3,2
2	8	NN	16	16	0	7,3	1,2,3,4,6,7,8,9	8,4,6,7,8,1,3,2
2	8	NN	19	17	2	5,9	1,2,3,8,10,12,13,15	5,10,8,5,3,16,9,5
2	8	NN	27	21	6	5,9	1,2,3,8,10,12,13,15	5,10,8,5,3,16,9,5
2	8	NN	25	23	2	5,9	1,2,3,8,10,12,13,15	5,10,8,5,3,16,9,5
2	8	NN	23	22	1	7,19	1,5,9,10,11,16,25,23	7,11,16,18,6,9,11,14
2	8	NN	19	16	3	7,19	1,5,9,10,11,16,25,23	7,11,16,18,6,9,11,14
2	8	NN	24	22	2	7,19	1,5,9,10,11,16,25,23	7,11,16,18,6,9,11,14
4	4	NN	21	21	0	1,4,2	7,8,3,6	1,9,8,4

Table A16: Experimental Design for Independent Response Test

No. of AGV	No. of Targets	Heuristic	Upper Bound	Optimal	Response	AGV Initial Positions	Target Nos.	Destination Nos.
4	4	NN	8	8	0	1,4,2	7,8,3,6	1,9,8,4
4	4	NN	19	17	2	1,4,2	7,8,3,6	1,9,8,4
4	4	NN	12	12	0	8,14,5	1,2,3,4	10,13,12,8
4	4	NN	15	13	2	8,14,5	1,2,3,4	10,13,12,8
4	4	NN	16	13	3	8,14,5	1,2,3,4	10,13,12,8
4	4	NN	20	14	6	9,4,6	3,11,17,22	21,6,7,15
4	4	NN	22	18	4	9,4,6	3,11,17,22	21,6,7,15
4	4	NN	18	17	1	9,4,6	3,11,17,22	21,6,7,15
4	8	NN	23	22	1	6,7,4	1,2,3,4,6,7,8,9	4,7,6,6,3,4,1,2
4	8	NN	34	27	7	6,7,4	1,2,3,4,6,7,8,9	4,7,6,6,3,4,1,2
4	8	NN	32	28	4	6,7,4	1,2,3,4,6,7,8,9	4,7,6,6,3,4,1,2
4	8	NN	28	28	0	2,8,15	1,2,3,8,10,12,13,15	5,10,8,5,3,16,9,5
4	8	NN	35	29	6	2,8,15	1,2,3,8,10,12,13,15	5,10,8,5,3,16,9,5
4	8	NN	26	25	1	2,8,15	1,2,3,8,10,12,13,15	5,10,8,5,3,16,9,5
4	8	NN	29	26	3	9,10,23	1,5,9,10,11,16,25,23	7,11,16,18,6,9,11,14
4	8	NN	21	20	1	9,10,23	1,5,9,10,11,16,25,23	7,11,16,18,6,9,11,14
4	8	NN	30	25	5	9,10,23	1,5,9,10,11,16,25,23	7,11,16,18,6,9,11,14
2	4	LUA	15	14	1	6,4	1,7,3,9	8,4,6,2
2	4	LUA	18	17	1	6,4	1,7,3,9	8,4,6,2

Table A16: Experimental Design for Independent Response Test

No. of AGV	No. of Targets	Heuristic	Upper Bound	Optimal	Response	AGV Initial Positions	Target Nos.	Destination Nos.
2	4	LUA	19	16	3	6,4	1,7,3,9	8,4,6,2
2	4	LUA	12	8	4	3,12	1,16,4,13	15,2,14,3
2	4	LUA	7	7	0	3,12	1,16,4,13	15,2,14,3
2	4	LUA	14	12	2	3,12	1,16,4,13	15,2,14,3
2	4	LUA	16	14	2	9,25	19,16,10,4	17,21,3,15
2	4	LUA	12	12	0	9,25	19,16,10,4	17,21,3,15
2	4	LUA	18	11	7	9,25	19,16,10,4	17,21,3,15
2	8	LUA	23	18	5	7,8	1,2,3,4,6,7,8,9	4,7,6,6,3,4,1,2
2	8	LUA	18	15	3	7,8	1,2,3,4,6,7,8,9	4,7,6,6,3,4,1,2
2	8	LUA	17	16	1	7,8	1,2,3,4,6,7,8,9	4,7,6,6,3,4,1,2
2	8	LUA	31	25	6	8,16	1,2,3,8,10,12,13,15	5,10,8,5,3,16,9,5
2	8	LUA	26	23	3	8,16	1,2,3,8,10,12,13,15	5,10,8,5,3,16,9,5
2	8	LUA	19	12	7	8,16	1,2,3,8,10,12,13,15	5,10,8,5,3,16,9,5
2	8	LUA	23	21	2	10,6	2,8,10,15,19,14,24,25	20,15,17,19,7,22,6,11
2	8	LUA	27	21	6	10,6	2,8,10,15,19,14,24,25	20,15,17,19,7,22,6,11
2	8	LUA	25	20	5	10,6	2,8,10,15,19,14,24,25	20,15,17,19,7,22,6,11
4	4	LUA	18	18	0	4,8,2	1,7,3,9	8,4,6,2
4	4	LUA	11	11	0	4,8,2	1,7,3,9	8,4,6,2
4	4	LUA	15	13	2	4,8,2	1,7,3,9	8,4,6,2

Table A16: Experimental Design for Independent Response Test

No. of AGV	No. of Targets	Heuristic	Upper Bound	Optimal	Response	AGV Initial Positions	Target Nos.	Destination Nos.
4	4	LUA	23	20	3	10,2,5	14,15,8,4	2,12,13,9
4	4	LUA	22	16	6	10,2,5	14,15,8,4	2,12,13,9
4	4	LUA	17	15	2	10,2,5	14,15,8,4	2,12,13,9
4	4	LUA	19	18	1	15,18,3	7,11,24,2	2,4,6,20
4	4	LUA	21	19	2	15,18,3	7,11,24,2	2,4,6,20
4	4	LUA	22	21	1	15,18,3	7,11,24,2	2,4,6,20
4	8	LUA	26	23	3	7,8,9	7,2,4,3,1,8,6,9	6,8,3,7,3,1,8,4
4	8	LUA	34	22	12	7,8,9	7,2,4,3,1,8,6,9	6,8,3,7,3,1,8,4
4	8	LUA	41	28	13	7,8,9	7,2,4,3,1,8,6,9	6,8,3,7,3,1,8,4
4	8	LUA	19	18	1	10,4,1	1,2,3,8,10,12,13,15	5,10,8,5,3,16,9,5
4	8	LUA	28	24	4	10,4,1	1,2,3,8,10,12,13,15	5,10,8,5,3,16,9,5
4	8	LUA	27	23	4	10,4,1	1,2,3,8,10,12,13,15	5,10,8,5,3,16,9,5
4	8	LUA	24	22	2	15,18,3	1,11,8,14,24,17,19,10	14,16,23,7,19,2,14,16
4	8	LUA	32	29	3	15,18,3	1,11,8,14,24,17,19,10	14,16,23,7,19,2,14,16
4	8	LUA	29	24	5	15,18,3	1,11,8,14,24,17,19,10	14,16,23,7,19,2,14,16
2	4	MG	6	6	0	4,7	1,4,3,7	6,2,8,4
2	4	MG	8	7	1	4,7	1,4,3,7	6,2,8,4
2	4	MG	14	12	2	4,7	1,4,3,7	6,2,8,4
2	4	MG	18	16	2	1,5	2,11,16,1	14,5,8,3

Table A16: Experimental Design for Independent Response Test

No. of AGV	No. of Targets	Heuristic	Upper Bound	Optimal	Response	AGV Initial Positions	Target Nos.	Destination Nos.
2	4	MG	12	12	0	1,5	2,11,16,1	14,5,8,3
2	4	MG	15	14	1	1,5	2,11,16,1	14,5,8,3
2	4	MG	17	15	2	8,24	19,6,3,22	14,10,24,15
2	4	MG	19	14	5	8,24	19,6,3,22	14,10,24,15
2	4	MG	11	11	0	8,24	19,6,3,22	14,10,24,15
2	8	MG	19	18	1	7,3	8,3,6,9,7,4,1,2	7,2,4,3,1,8,6,9
2	8	MG	25	21	4	7,3	8,3,6,9,7,4,1,2	7,2,4,3,1,8,6,9
2	8	MG	12	11	1	7,3	8,3,6,9,7,4,1,2	7,2,4,3,1,8,6,9
2	8	MG	34	29	5	11,14	1,2,3,8,10,12,13,15	5,10,8,5,3,16,9,5
2	8	MG	26	25	1	11,14	1,2,3,8,10,12,13,15	5,10,8,5,3,16,9,5
2	8	MG	31	24	7	11,14	1,2,3,8,10,12,13,15	5,10,8,5,3,16,9,5
2	8	MG	18	17	1	9,1	7,2,4,3,1,8,6,9	8,3,6,6,7,4,7,3
2	8	MG	24	23	1	9,1	7,2,4,3,1,8,6,9	8,3,6,6,7,4,7,3
2	8	MG	23	22	1	9,1	7,2,4,3,1,8,6,9	8,3,6,6,7,4,7,3
4	4	MG	14	14	0	9,4,2	7,8,3,6	1,9,8,4
4	4	MG	16	15	1	9,4,2	7,8,3,6	1,9,8,4
4	4	MG	23	21	2	9,4,2	7,8,3,6	1,9,8,4
4	4	MG	18	14	4	16,15,14	1,2,3,4	10,13,12,8
4	4	MG	25	23	2	16,15,14	1,2,3,4	10,13,12,8

Table A16: Experimental Design for Independent Response Test

No. of AGV	No. of Targets	Heuristic	Upper Bound	Optimal	Response	AGV Initial Positions	Target Nos.	Destination Nos.
4	4	MG	22	21	1	16,15,14	1,2,3,4	10,13,12,8
4	4	MG	27	25	2	14,8,15	3,11,17,22	21,6,7,15
4	4	MG	29	25	4	14,8,15	3,11,17,22	21,6,7,15
4	4	MG	16	16	0	14,8,15	3,11,17,22	21,6,7,15
4	8	MG	32	21	11	2,4,3	7,2,4,3,1,8,6,9	6,8,3,7,3,1,8,4
4	8	MG	26	22	4	2,4,3	7,2,4,3,1,8,6,9	6,8,3,7,3,1,8,4
4	8	MG	28	24	4	2,4,3	7,2,4,3,1,8,6,9	6,8,3,7,3,1,8,4
4	8	MG	34	27	7	9,11,3	2,14,9,5,1,13,3,16	15,8,12,15,11,3,15,8
4	8	MG	40	34	6	9,11,3	2,14,9,5,1,13,3,16	15,8,12,15,11,3,15,8
4	8	MG	23	20	3	9,11,3	2,14,9,5,1,13,3,16	15,8,12,15,11,3,15,8
4	8	MG	28	19	9	11,8,15	1,5,9,10,11,16,25,23	7,11,16,18,6,9,11,14
4	8	MG	27	23	4	11,8,15	1,5,9,10,11,16,25,23	7,11,16,18,6,9,11,14
4	8	MG	28	25	3	11,8,15	1,5,9,10,11,16,25,23	7,11,16,18,6,9,11,14
2	4	C	12	11	1	8,2	1,4,3,7	6,2,8,4
2	4	C	9	9	0	8,2	1,4,3,7	6,2,8,4
2	4	C	13	13	0	8,2	1,4,3,7	6,2,8,4
2	4	C	11	11	0	6,8	1,4,9,7	2,3,8,8
2	4	C	14	12	2	6,8	1,4,9,7	2,3,8,8
2	4	C	14	13	1	6,8	1,4,9,7	2,3,8,8

Table A16: Experimental Design for Independent Response Test

No. of AGV	No. of Targets	Heuristic	Upper Bound	Optimal	Response	AGV Initial Positions	Target Nos.	Destination Nos.
2	4	C	15	14	1	6,4	1,7,3,9	8,4,6,2
2	4	C	18	17	1	6,4	1,7,3,9	8,4,6,2
2	4	C	17	16	1	6,4	1,7,3,9	8,4,6,2
4	4	C	18	17	1	10,2,5	11,15,5,4	9,13,2,10
4	4	C	12	12	0	10,2,5	11,15,5,4	9,13,2,10
4	4	C	17	14	3	10,2,5	11,15,5,4	9,13,2,10
4	4	C	12	12	0	8,14,5	1,2,3,4	10,13,12,8
4	4	C	14	13	1	8,14,5	1,2,3,4	10,13,12,8
4	4	C	15	13	2	8,14,5	1,2,3,4	10,13,12,8
4	4	C	17	14	3	16,15,14	1,2,3,4	10,13,12,8
4	4	C	25	23	2	16,15,14	1,2,3,4	10,13,12,8
4	4	C	22	21	1	16,15,14	1,2,3,4	10,13,12,8
2	8	C	23	22	1	7,19	1,5,9,10,11,16,25,23	7,11,16,18,6,9,11,14
2	8	C	19	16	3	7,19	1,5,9,10,11,16,25,23	7,11,16,18,6,9,11,14
2	8	C	23	22	1	7,19	1,5,9,10,11,16,25,23	7,11,16,18,6,9,11,14
2	8	C	23	21	2	10,6	2,8,10,15,19,14,24,25	20,15,17,19,7,22,6,11
2	8	C	26	21	5	10,6	2,8,10,15,19,14,24,25	20,15,17,19,7,22,6,11
2	8	C	23	20	3	10,6	2,8,10,15,19,14,24,25	20,15,17,19,7,22,6,11
2	8	C	18	17	1	9,1	7,2,4,3,1,8,6,9	8,3,6,6,7,4,7,3

Table A16: Experimental Design for Independent Response Test

No. of AGV	No. of Targets	Heuristic	Upper Bound	Optimal	Response	AGV Initial Positions	Target Nos.	Destination Nos.
2	8	C	24	23	1	9,1	7,2,4,3,1,8,6,9	8,3,6,6,7,4,7,3
2	8	C	23	22	1	9,1	7,2,4,3,1,8,6,9	8,3,6,6,7,4,7,3
4	8	C	27	21	6	2,4,3	7,2,4,3,1,8,6,9	6,8,3,7,3,1,8,4
4	8	C	26	22	4	2,4,3	7,2,4,3,1,8,6,9	6,8,3,7,3,1,8,4
4	8	C	27	24	3	2,4,3	7,2,4,3,1,8,6,9	6,8,3,7,3,1,8,4
4	8	C	22	21	1	1,6,8	7,2,4,3,1,8,6,9	6,8,3,7,3,1,8,4
4	8	C	32	27	5	1,6,8	7,2,4,3,1,8,6,9	6,8,3,7,3,1,8,4
4	8	C	25	21	4	1,6,8	7,2,4,3,1,8,6,9	6,8,3,7,3,1,8,4

A.4 Appendix 4: Software Code for Optimal Problem Formulation

A.4.1 Sample Visual Basic Code

```
Attribute VB_Name = "SRMS_LP_INPUT"
Option Explicit
Dim iAGV_LP, iTargets, iAGVInitial  As Integer


Sub Create_AGV_LP()

   FileName = "c:\ASPAGV\ASPAGV.lp"
   If Dir(FileName) <> "" Then Kill (FileName)

   iAGV_LP = FreeFile
  Open FileName For Append As #iAGV_LP

  Print #iAGV_LP, "Min "
  'Write objective function
  WriteObjFn

  Print #iAGV_LP, "S.T."

  'Write constraints
  WriteInUseCon

  Print #iAGV_LP, "binary"
  WriteBinary

  Print #iAGV_LP, "End"
  Close #iAGV_LP

End Sub
'WriteObjFn():
'Writes the objective function for the week optimization.

Sub WriteObjFn()


   OutputString = ""
   count = 0
       For i = 1 To nbgrid
     For k = 1 To nbgrid
     If TargetDeliveryCombinationArray(i, k) = 1 Then
          count = count + 1
   OutputString = OutputString & " + w" & count
```

```
   End If
   Next k
   Next i
    Print #iAGV_LP, OutputString

End Sub

Sub WriteBinary()
  For i = 1 To nbgrid
    For j = 1 To nbAGV
      For t = 1 To nbTime
      OutputString = ""
      OutputString = "location(" & i & "," & j & "," & t & ")"
      Print #iAGV_LP, OutputString
      OutputString = "pickup(" & i & "," & j & "," & t & ")"
      Print #iAGV_LP, OutputString
      OutputString = "delivery(" & i & "," & j & "," & t & ")"
      Print #iAGV_LP, OutputString
      Next t
    Next j
  Next i

End Sub

Sub WriteInUseCon()

 'Constraint 1
  For j = 1 To nbAGV
    For t = 1 To nbTime
    OutputString = "c1" & j & t & ":"
     For i = 1 To nbgrid - 1
      OutputString = OutputString & "location(" & i & "," & j & "," & t & ") + "
     Next i
     OutputString = OutputString & "location(" & nbgrid & "," & j & "," & t & ")= 1 "
     Print #iAGV_LP, OutputString
    Next t
  Next j

  'Constraint 2
  For i = 1 To nbgrid
    For t = 1 To nbTime
    OutputString = "c2" & i & t & ":"
     For j = 1 To nbAGV - 1
      OutputString = OutputString & "location(" & i & "," & j & "," & t & ") + "
     Next j
     OutputString = OutputString & "location(" & i & "," & nbAGV & "," & t & ")<= 1 "
```

```
   Print #iAGV_LP, OutputString
  Next t
Next i

LoadBaseCombinationArray
'Constraint 3
 For i = 1 To nbgrid
  For j = 1 To nbAGV
   For t = 1 To nbTime - 1
   OutputString = "c3" & i & j & t & ":"
     For k = 1 To nbgrid
      If BaseCombinationArray(i, k) = 1 Then
      OutputString = OutputString & "location(" & k & "," & j & "," & t + 1 & ") + "
      End If
    Next k
   OutputString = OutputString & "dummy - location(" & i & "," & j & "," & t & ")>= 0 "
   Print #iAGV_LP, OutputString
  Next t
 Next j
Next i


'Constraint 4
 For k = 1 To nbgrid

  For i = 1 To nbgrid
  If TargetDeliveryCombinationArray(i, k) = 1 Then
  OutputString = "c4" & k & i & ":"
  For j = 1 To nbAGV
  For t = 1 To nbTime
        OutputString = OutputString & "pickup(" & i & "," & j & "," & t & ") + "
    Next t
   Next j
    OutputString = OutputString & "dummy = 1 "
   Print #iAGV_LP, OutputString
   End If
    Next i
   Next k

   'Constraint 5
    For i = 1 To nbgrid
    For k = 1 To nbgrid
    If TargetDeliveryCombinationArray(i, k) = 1 Then
    For j = 1 To nbAGV
       For t = 1 To nbTime
     OutputString = "c5" & i & j & k & t & ":"
```

```
   OutputString = OutputString & "pickup(" & i & "," & j & "," & t & ") - location(" & i & ","
  & j & "," & t & ") < = 0"

 Print #iAGV_LP, OutputString
 Next t
 Next j
 End If
 Next k
Next i

 'Constraint 6
 For k = 1 To nbgrid

For i = 1 To nbgrid
If TargetDeliveryCombinationArray(i, k) = 1 Then
OutputString = "c6" & k & i & ":"
For j = 1 To nbAGV
For t = 1 To nbTime
      OutputString = OutputString & "delivery(" & k & "," & j & "," & t & ") + "
   Next t
  Next j
   OutputString = OutputString & "dummy = 1 "
 Print #iAGV_LP, OutputString
 End If
  Next i
 Next k

 'Constraint 7A
 Dim count As Integer
 For j = 1 To nbAGV
 For k = 1 To nbgrid
  For i = 1 To nbgrid
     For t = 1 To nbTime - 1
     If TargetDeliveryCombinationArray(i, k) = 1 Then
 count = t + 1
 If count <= nbTime Then
 OutputString = "c7a" & i & j & k & t & count & ":"
 For c = count To nbTime
 OutputString = OutputString & "delivery(" & i & "," & j & "," & c & ") + "
 Next c
 OutputString = OutputString & "dummy - pickup(" & i & "," & j & "," & t & ") >= 0"
 Print #iAGV_LP, OutputString
 End If
 End If
 Next t
 Next i
```

```
    Next k
    Next j

    ' Constraint 7B
        For j = 1 To nbAGV
    For k = 1 To nbgrid
     For i = 1 To nbgrid
   If TargetDeliveryCombinationArray(i, k) = 1 Then
   OutputString = "c7B" & i & j & k & ":"
    OutputString = OutputString & "pickup(" & i & "," & j & "," & nbTime & ") = 0"
    Print #iAGV_LP, OutputString
    End If
    Next i
    Next k
    Next j

    ' Constraint 8
    For j = 1 To nbAGV
    For t = 1 To nbTime
    OutputString = "c8" & j & t & ":"
    For k = 1 To nbgrid
     For i = 1 To nbgrid
   If TargetDeliveryCombinationArray(i, k) = 1 Then
   For c = 1 To nbTime
   If c <= t Then
   OutputString = OutputString & "pickup(" & i & "," & j & "," & c & ") - delivery(" & i & "," &
  j & "," & c & ") + "
    End If
    Next c
    End If
    Next i
    Next k
    OutputString = OutputString & "dummy <= 1"
    Print #iAGV_LP, OutputString
    Next t
    Next j

    'Constraint 9
    For i = 1 To nbgrid
     For k = 1 To nbgrid
     If TargetDeliveryCombinationArray(i, k) = 1 Then
     For j = 1 To nbAGV
        For t = 1 To nbTime
      OutputString = "c9" & i & j & k & t & ":"
          OutputString = OutputString & "delivery(" & k & "," & j & "," & t & ") - location(" & k
& "," & j & "," & t & ") < = 0"
```

```
 Print #iAGV_LP, OutputString
 Next t
 Next j
 End If
 Next k
Next i

' Constraint 10
For i = 1 To nbgrid
For k = 1 To nbgrid
   If BaseCombinationArray(i, k) = 1 Then
   For j = 1 To nbAGV
   For l = 1 To nbAGV
   If l <> j Then
   For t = 1 To nbTime
   OutputString = "c10" & i & j & k & t & l & ":"
      OutputString = OutputString & "location(" & k & "," & j & "," & t & ") + location(" & i
& "," & l & "," & t & ")+ location(" & i & "," & j & "," & t + 1 & ")+ location(" & k & "," & l &
"," & t + 1 & ") < = 3"
Print #iAGV_LP, OutputString
Next t
End If
Next l
Next j
End If
Next k
Next i

'Constraint 11
count = 0
   For i = 1 To nbgrid
  For k = 1 To nbgrid
  If TargetDeliveryCombinationArray(i, k) = 1 Then
  OutputString = "c11" & k & i & ":"
  count = count + 1
  For j = 1 To nbAGV
For t = 1 To nbTime
OutputString = OutputString & t & " delivery(" & i & "," & j & "," & t & ") + "
Next t
Next j
OutputString = OutputString & "dummy - w" & count & " = 0"
Print #iAGV_LP, OutputString
End If
Next k
Next i
```

```
   For i = 1 To nbAGV
   For j = 1 To nbgrid
   OutputString = ""
   If AGVInitialArray(i, j) = 1 Then
   OutputString = OutputString & "location(" & j & "," & i & "," & 1 & ")= 1"
    Print #iAGV_LP, OutputString
    End If
    Next j
    Next i

    OutputString = "dummy = 0"
    Print #iAGV_LP, OutputString


End Sub

'LoadProductLORDistArray_Bkup():
'Loads the array BaseLorDistArray for use in the creation
'of the LP.  Selects records from LORDist_temp table for the relevant BusinessType
'ordered by DOW-Product-LOR.  These are assigned in order to the array LorDistArray.

Sub LoadBaseCombinationArray()


    For i = 1 To nbgrid
       For j = 1 To nbgrid
         BaseCombinationArray(i, j) = 0
       Next j
    Next i

    If nbgrid = 4 Then
    BaseCombinationArray(1, 1) = 1
    BaseCombinationArray(1, 2) = 1
    BaseCombinationArray(1, 3) = 1
    BaseCombinationArray(2, 1) = 1
    BaseCombinationArray(2, 2) = 1
    BaseCombinationArray(2, 4) = 1
    BaseCombinationArray(3, 3) = 1
    BaseCombinationArray(3, 4) = 1
    BaseCombinationArray(3, 1) = 1
    BaseCombinationArray(4, 3) = 1
    BaseCombinationArray(4, 2) = 1
    BaseCombinationArray(4, 4) = 1

    ElseIf nbgrid = 9 Then
```

```
BaseCombinationArray(1, 1) = 1
BaseCombinationArray(1, 2) = 1
BaseCombinationArray(1, 6) = 1
BaseCombinationArray(2, 1) = 1
BaseCombinationArray(2, 2) = 1
BaseCombinationArray(2, 3) = 1
BaseCombinationArray(2, 5) = 1
BaseCombinationArray(3, 3) = 1
BaseCombinationArray(3, 4) = 1
BaseCombinationArray(3, 2) = 1
BaseCombinationArray(4, 3) = 1
BaseCombinationArray(4, 9) = 1
BaseCombinationArray(4, 4) = 1
BaseCombinationArray(5, 5) = 1
BaseCombinationArray(5, 2) = 1
BaseCombinationArray(5, 8) = 1
BaseCombinationArray(6, 6) = 1
BaseCombinationArray(6, 1) = 1
BaseCombinationArray(6, 7) = 1
BaseCombinationArray(7, 7) = 1
BaseCombinationArray(7, 6) = 1
BaseCombinationArray(7, 8) = 1
BaseCombinationArray(8, 5) = 1
BaseCombinationArray(8, 7) = 1
BaseCombinationArray(8, 9) = 1
BaseCombinationArray(8, 8) = 1
BaseCombinationArray(9, 9) = 1
BaseCombinationArray(9, 8) = 1
BaseCombinationArray(9, 4) = 1

ElseIf nbgrid = 16 Then
BaseCombinationArray(1, 1) = 1
BaseCombinationArray(1, 2) = 1
BaseCombinationArray(1, 8) = 1
BaseCombinationArray(2, 1) = 1
BaseCombinationArray(2, 2) = 1
BaseCombinationArray(2, 3) = 1
BaseCombinationArray(3, 3) = 1
BaseCombinationArray(3, 4) = 1
BaseCombinationArray(3, 2) = 1
BaseCombinationArray(4, 3) = 1
BaseCombinationArray(4, 5) = 1
BaseCombinationArray(4, 4) = 1
BaseCombinationArray(5, 5) = 1
BaseCombinationArray(5, 4) = 1
BaseCombinationArray(5, 6) = 1
```

```
BaseCombinationArray(5, 12) = 1
BaseCombinationArray(6, 6) = 1
BaseCombinationArray(6, 5) = 1
BaseCombinationArray(6, 7) = 1
BaseCombinationArray(7, 7) = 1
BaseCombinationArray(7, 6) = 1
BaseCombinationArray(7, 8) = 1
BaseCombinationArray(8, 1) = 1
BaseCombinationArray(8, 7) = 1
BaseCombinationArray(8, 9) = 1
BaseCombinationArray(8, 8) = 1
BaseCombinationArray(9, 9) = 1
BaseCombinationArray(9, 8) = 1
BaseCombinationArray(9, 10) = 1
BaseCombinationArray(9, 16) = 1
BaseCombinationArray(10, 10) = 1
BaseCombinationArray(10, 9) = 1
BaseCombinationArray(10, 11) = 1
BaseCombinationArray(11, 11) = 1
BaseCombinationArray(11, 10) = 1
BaseCombinationArray(11, 12) = 1
BaseCombinationArray(12, 12) = 1
BaseCombinationArray(12, 11) = 1
BaseCombinationArray(12, 13) = 1
BaseCombinationArray(12, 5) = 1
BaseCombinationArray(13, 13) = 1
BaseCombinationArray(13, 12) = 1
BaseCombinationArray(13, 14) = 1
BaseCombinationArray(14, 14) = 1
BaseCombinationArray(14, 13) = 1
BaseCombinationArray(14, 15) = 1
BaseCombinationArray(15, 15) = 1
BaseCombinationArray(15, 14) = 1
BaseCombinationArray(15, 16) = 1
BaseCombinationArray(16, 16) = 1
BaseCombinationArray(16, 15) = 1
BaseCombinationArray(16, 9) = 1
Else
BaseCombinationArray(1, 1) = 1
BaseCombinationArray(1, 2) = 1
BaseCombinationArray(1, 12) = 1
BaseCombinationArray(2, 1) = 1
BaseCombinationArray(2, 2) = 1
BaseCombinationArray(2, 3) = 1
BaseCombinationArray(3, 3) = 1
BaseCombinationArray(3, 2) = 1
```

BaseCombinationArray(3, 4) = 1
BaseCombinationArray(4, 3) = 1
BaseCombinationArray(4, 5) = 1
BaseCombinationArray(4, 4) = 1
BaseCombinationArray(4, 9) = 1
BaseCombinationArray(5, 5) = 1
BaseCombinationArray(5, 4) = 1
BaseCombinationArray(5, 6) = 1
BaseCombinationArray(6, 6) = 1
BaseCombinationArray(6, 5) = 1
BaseCombinationArray(6, 7) = 1
BaseCombinationArray(7, 7) = 1
BaseCombinationArray(7, 6) = 1
BaseCombinationArray(7, 18) = 1
BaseCombinationArray(7, 8) = 1
BaseCombinationArray(8, 8) = 1
BaseCombinationArray(8, 7) = 1
BaseCombinationArray(8, 9) = 1
BaseCombinationArray(9, 9) = 1
BaseCombinationArray(9, 6) = 1
BaseCombinationArray(9, 4) = 1
BaseCombinationArray(9, 8) = 1
BaseCombinationArray(9, 10) = 1
BaseCombinationArray(10, 10) = 1
BaseCombinationArray(10, 9) = 1
BaseCombinationArray(10, 11) = 1
BaseCombinationArray(11, 11) = 1
BaseCombinationArray(11, 10) = 1
BaseCombinationArray(11, 12) = 1
BaseCombinationArray(12, 12) = 1
BaseCombinationArray(12, 1) = 1
BaseCombinationArray(12, 13) = 1
BaseCombinationArray(13, 13) = 1
BaseCombinationArray(13, 12) = 1
BaseCombinationArray(13, 24) = 1
BaseCombinationArray(13, 14) = 1
BaseCombinationArray(14, 14) = 1
BaseCombinationArray(14, 13) = 1
BaseCombinationArray(14, 15) = 1
BaseCombinationArray(15, 15) = 1
BaseCombinationArray(15, 14) = 1
BaseCombinationArray(15, 16) = 1
BaseCombinationArray(16, 16) = 1
BaseCombinationArray(16, 15) = 1
BaseCombinationArray(16, 9) = 1
BaseCombinationArray(16, 21) = 1

BaseCombinationArray(16, 17) = 1
BaseCombinationArray(17, 17) = 1
BaseCombinationArray(17, 16) = 1
BaseCombinationArray(17, 18) = 1
BaseCombinationArray(18, 18) = 1
BaseCombinationArray(18, 17) = 1
BaseCombinationArray(18, 7) = 1
BaseCombinationArray(18, 19) = 1
BaseCombinationArray(19, 19) = 1
BaseCombinationArray(19, 18) = 1
BaseCombinationArray(19, 20) = 1
BaseCombinationArray(19, 30) = 1
BaseCombinationArray(20, 20) = 1
BaseCombinationArray(20, 19) = 1
BaseCombinationArray(20, 21) = 1
BaseCombinationArray(21, 21) = 1
BaseCombinationArray(21, 22) = 1
BaseCombinationArray(21, 20) = 1
BaseCombinationArray(21, 16) = 1
BaseCombinationArray(21, 28) = 1
BaseCombinationArray(22, 9) = 1
BaseCombinationArray(22, 22) = 1
BaseCombinationArray(22, 23) = 1
BaseCombinationArray(22, 21) = 1
BaseCombinationArray(23, 23) = 1
BaseCombinationArray(23, 24) = 1
BaseCombinationArray(23, 22) = 1
BaseCombinationArray(24, 24) = 1
BaseCombinationArray(24, 13) = 1
BaseCombinationArray(24, 23) = 1
BaseCombinationArray(24, 25) = 1
BaseCombinationArray(25, 24) = 1
BaseCombinationArray(25, 25) = 1
BaseCombinationArray(25, 26) = 1
BaseCombinationArray(25, 36) = 1
BaseCombinationArray(26, 26) = 1
BaseCombinationArray(26, 25) = 1
BaseCombinationArray(26, 27) = 1
BaseCombinationArray(27, 27) = 1
BaseCombinationArray(27, 26) = 1
BaseCombinationArray(27, 28) = 1
BaseCombinationArray(28, 28) = 1
BaseCombinationArray(28, 27) = 1
BaseCombinationArray(28, 29) = 1
BaseCombinationArray(28, 21) = 1
BaseCombinationArray(29, 29) = 1

```
    BaseCombinationArray(29, 28) = 1
    BaseCombinationArray(29, 30) = 1
    BaseCombinationArray(30, 30) = 1
    BaseCombinationArray(30, 29) = 1
    BaseCombinationArray(30, 19) = 1
    BaseCombinationArray(30, 31) = 1
    BaseCombinationArray(31, 31) = 1
    BaseCombinationArray(31, 30) = 1
    BaseCombinationArray(31, 32) = 1
    BaseCombinationArray(32, 32) = 1
    BaseCombinationArray(32, 31) = 1
    BaseCombinationArray(32, 33) = 1
    BaseCombinationArray(33, 33) = 1
    BaseCombinationArray(33, 32) = 1
    BaseCombinationArray(33, 34) = 1
    BaseCombinationArray(34, 34) = 1
    BaseCombinationArray(34, 33) = 1
    BaseCombinationArray(34, 35) = 1
    BaseCombinationArray(35, 35) = 1
    BaseCombinationArray(35, 34) = 1
    BaseCombinationArray(35, 36) = 1
    BaseCombinationArray(36, 36) = 1
    BaseCombinationArray(36, 25) = 1
    BaseCombinationArray(36, 35) = 1
    End If

End Sub

Sub LoadTargetDeliveryCombinationArray()

  FileName = "c:\ASPAGV\TargetCombinations.txt"
    If Dir(FileName) <> "" Then Kill (FileName)

    iTargets = FreeFile
  Open FileName For Append As #iTargets
  Print #iTargets, "Pickup  Delivery"

    For i = 1 To nbgrid
       For j = 1 To nbgrid
         TargetDeliveryCombinationArray(i, j) = 0
       Next j
    Next i

    Dim pickup, delivery As Integer
    For k = 1 To NoOftargets
       pickup = Rand(1, nbgrid)
```

```
        delivery = Rand(1, nbgrid)
        Do While pickup = delivery
         delivery = Rand(1, nbgrid)
        Loop
        If k <= 4 Then
        Targets(k) = pickup & "," & delivery
        End If
        OutputString = pickup & "  " & delivery
        Print #iTargets, OutputString
       TargetDeliveryCombinationArray(pickup, delivery) = 1

    Next k

    Close #iTargets

End Sub

Sub LoadAGVInitialArray()

FileName = "c:\ASPAGV\AGVInitial.txt"
    If Dir(FileName) <> "" Then Kill (FileName)

    iAGVInitial = FreeFile
  Open FileName For Append As #iAGVInitial
  Print #iAGVInitial, "AGV#  InitialGrid#"

    For i = 1 To nbAGV
        For j = 1 To nbgrid
          AGVInitialArray(i, j) = 0
        Next j
    Next i

    For j = 1 To nbgrid
      StartGrid(j) = 0
    Next j

    For i = 1 To nbAGV
        count = Rand(1, nbgrid)
        Do While StartGrid(count) = 1
        count = Rand(1, nbgrid)
        Loop
        If i <= 4 Then
        AGVPositions(i) = count
        End If
        AGVInitialArray(i, count) = 1
        StartGrid(count) = 1
```

```
        OutputString = i & "  " & count
        Print #iAGVInitial, OutputString
    Next i

Close #iAGVInitial

End Sub

Public Function Rand(ByVal Low As Integer, ByVal High As Integer) As Integer
Randomize seed
Rand = Int((High - Low + 1) * Rnd) + Low
End Function
```

A.4.2 Sample OPL Code

```
// Binary Definition:

range Boolean [0..1];

// Non-decision Variable ranges:

int nbGrid = ...;

range Grid [1..nbGrid];

int nbAGV = ...;

range AGV [1..nbAGV];

int nbtime = ...;

range time [1..nbtime];

range time1 [1..nbtime-1];

// range time2 17..19;

// range time2 1..nbtime-2;

int Target[1..2] = [2,1];

// int Target[1..3] = [10,37,59];

//   int Target[1..2] = [10,59];

 int Capacity[1..2]=[1,1];

{int} moves[1..nbGrid] = ...;

// int w = 10;

// int d = 0;

// Decision Variable Definitions:

dvar int location[1..nbGrid][1..nbAGV][1..nbtime] in 0..1;
```

```
dvar int pickup[1..nbGrid][1..nbAGV][1..nbtime] in 0..1;
dvar int delivery[1..nbGrid][1..nbAGV][1..nbtime] in 0..1;
// var float+ w;
dvar float+ w1;
dvar float+ w2;
//dvar float+ w3;
//dvar float+ w4;

// Objective Function:
// minimize w
minimize w1+w2;
subject to {
// Constraint 1:
  forall(j in 1..nbAGV)
    forall(t in 1..nbtime)
      sum(i in 1..nbGrid)location[i][j][t] == 1;
// Constraint 2:
  forall(i in 1..nbGrid)
    forall(t in 1..nbtime)
      sum(j in 1..nbAGV)location[i,j,t]<=1;
// Constraint 3:
  forall(i in 1..nbGrid)
    forall(j in 1..nbAGV)
      forall(t in 1..nbtime-1)
        sum(k in moves[i])
         (location[k,j,t+1]) >=
          location[i,j,t];
// Constraint 4:
    forall(h in 1..2)
    sum(j in 1..nbAGV, t in 1..nbtime) pickup[Target[h],j,t] == 1;
// Constraint 5:
```

```
    forall(h in 1..2)
    forall(j in 1..nbAGV)
      forall(t in 1..nbtime)
        pickup[Target[h],j,t] <= location[Target[h],j,t];

// Constraint 6:
  forall(h in 1..2)
    sum(j in 1..nbAGV, t in 1..nbtime) delivery[Target[h],j,t] == 1;
// Constraint 7 - New Trial:
  forall(j in 1..nbAGV)
    forall(h in 1..2)
      forall(t in 1..nbtime-1)
        sum(c in 1..nbtime-1:c>=t+1)delivery[Target[h],j,c] >=pickup[Target[h],j,t];
  forall(j in 1..nbAGV)
    forall(h in 1..2)
      pickup[Target[h]][j][nbtime]==0;
// Constraint 8:
   forall(j in 1..nbAGV)
    forall(t in 1..nbtime)
      sum(h in 1..2,k in 1..nbtime:k<=t)(pickup[Target[h],j,k] - delivery[Target[h],j,k]) <=
Capacity[j];
// Constraint 9:
  forall(j in 1..nbAGV)
      forall(t in 1..nbtime)
        {
        delivery[2,j,t] <= location[3,j,t];
        delivery[1,j,t] <= location[4,j,t];
//          delivery[24,j,t] <= location[33,j,t];
//          delivery[31,j,t] <= location[44,j,t];
//          delivery[37,j,t] <= location[56,j,t];
//          delivery[46,j,t] <= location[27,j,t];
```

```
//          delivery[59,j,t] <= location[17,j,t];
          };


// Constraint 10:
   forall(i in 1..nbGrid, k in moves[i])
     forall(j in 1..nbAGV, l in 1..nbAGV:l!=j)
       forall(t in 1..nbtime-1)
         location[k,j,t] + location[i,l,t] + location[i,j,t+1] + location[k,l,t+1] <= 3;
// Constraint 14:
     w1 == sum(j in 1..nbAGV, t in 1..nbtime)(t*delivery[2][j][t]);
     w2 == sum(j in 1..nbAGV, t in 1..nbtime)(t*delivery[1][j][t]);
//     w3 == sum(j in 1..nbAGV, t in 1..nbtime)(t*delivery[31,j,t]);
//    w4 == sum(j in 1..nbAGV, t in 1..nbtime)(t*delivery[36,j,t]);
}
execute {
  for(i in 1..nbGrid)
    writeln("delivery[", delivery[i][1][1]);
}
```

A.5 Appendix 5: List of Files Contained in the CD

The CD included in the back of the thesis binder includes the software code for optimal problem formulation, the Visual Basic code used for the front-end development, and a .pdf version of the entire thesis text.