Task Allocation and Motion Coordination of Multiple Autonomous Vehicles

- With application in automated container terminals

by

Asela K. Kulatunga

A thesis submitted in fulfilment
of the requirements for the degree of
Doctor of Philosophy

University of Technology, Sydney
Faculty of Engineering
August, 2008

CERTIFICATE OF AUTHORSHIP/ORIGINALITY

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of requirement for a degree except as fully acknowledged within the text.

I certify that the thesis has been written by me. Any help that I have received in my research work and preparation of the thesis itself has been acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

Signature of Candidate

(Asela K. Kulatunga)

Sydney, August 2008

ABSTRACT

This thesis focuses on developing an approach to solve the complex problem of task allocation and motion coordination simultaneously for a large fleet of autonomous vehicles in highly constrained operational environments. The multi-vehicle task allocation and motion coordination problem consists of allocating different tasks to different autonomous vehicles and intelligently coordinating motions of the vehicles without human interaction. The motion coordination itself comprises two sub-problems: path planning and collision / deadlock avoidance. Although a number of research studies have attempted to solve one or two aspects of this problem, it is rare to note that many have attempted to solve the task allocation, path planning and collision avoidance simultaneously. Therefore, it cannot be conclusively said that, optimal or near-optimal solutions generated based on one aspect of the problem will be optimal or near optimal results for the whole problem. It is advisable to solve the problem as one complete problem rather than decomposing it. This thesis intends to solve the complex task allocation, path planning and collision avoidance problem simultaneously.

A Simultaneous Task Allocation and Motion Coordination (STAMC) approach is developed to solve the multi-vehicle task allocation and motion coordination problem in a concurrent manner. Further, a novel algorithm called Simultaneous Path and Motion Planning (SiPaMoP) is proposed for collision free motion coordination. The main objective of this algorithm is to generate collision free paths for autonomous vehicles, once they are assigned with tasks in a conventional path topology of a material handling environment. The Dijkstra and A * shortest path search algorithms are utilised in the proposed Simultaneous Path and Motion Planning algorithm.

The multi-vehicle task allocation and motion coordination problem is first studied in a static environment where all the tasks, vehicles and operating environment information are assumed to be known. The multi-vehicle task allocation and motion coordination problem in a dynamic environment, where tasks, vehicles and operating environment change with time is then investigated. Furthermore, issues like vehicle breakdowns, which are common in real world situations, are considered. The computational cost of solving the multi-vehicle STAMC problem is also addressed by proposing a distributed computational architecture and implementing that architecture in a cluster computing system. Finally, the proposed algorithms are tested in a case study in an automated container terminal environment with a large fleet of autonomous straddle carriers.

Since the multi-vehicle task allocation and motion coordination is an NP-hard problem, it is almost impossible to find out the optimal solutions within a reasonable time frame. Therefore, this research focuses on investigating the appropriateness of heuristic and evolutionary algorithms for solving the STAMC problem. The Simulated Annealing algorithm, Ant Colony and Auction algorithms have been investigated. Commonly used dispatching rules such as first come first served, and closest task first have also been applied for comparison. Simulation tests of the proposed approach is conducted based on information from the Fishermen Island's container terminal of Patrick Corporation (Pty.) Ltd in Queensland, Australia where a large fleet of autonomous straddle carriers operate. The results shows that the proposed meta-heuristic techniques based simultaneous task allocation and motion coordination approach can effectively solve the complex multi-vehicle task allocation and motion coordination problem and it is capable of generating near optimal results within an acceptable time frame.

ACKNOWLEDGEMENT

This PhD thesis could not have been successfully completed if not for the tireless assistance and guidance with understanding of a list of academics throughout my PhD candidature. It is with a high sense of gratitude that I wish to place on record, the valuable guidance of my principal supervisor that I received throughout. I would be failing in my duties if I do not purposely thank Professor Dikai Liu who has given me valuable advice and encouragement, with understanding throughout the course of study. A special word of thanks should necessarily go to Professor Gamini Dissanayake for his support and guidance given throughout my candidature which paved the way for me to realize my dream of successfully completing a PhD in one of the leading Autonomous research groups.

I greatly acknowledge the financial assistance provided by the ARC Centre of Excellence for Autonomous Systems at the University of Technology, Sydney, Australia; the Faculty of Engineering, University of Technology, Sydney; and the Presidential Fund of Sri Lanka for its financial support. Furthermore, I would like to express my gratitude to Professor Sarath Siyambalapitiya of the Faculty of Engineering, University of Peradeniya, Sri Lanka for all the support given as an alternative supervisor during my stay in Sri Lanka. I would like to convey my gratitude to Doctors Peter Wu, Brad Skinner, Haye Lau and Raymond Kwok for their kind assistance and support.

I wish to express my sincere gratitude also to Dr. Fook Choon Choi and Ms. Pui Yeng Wong for their whole hearted parental support rendered with understanding during the difficult times that I had to encounter. My sincere thanks also go to Ms. Iroshini Gunaratne, Ms. Erandi Wattegama and Mr. Mohan Samaranayake for their support given to me in formatting and writing the thesis. The author wishes to express his gratitude to the friends both at UTS and elsewhere for their support and encouragement. I would fail in my duty if I do not thank Mr. Janitha Wijesinghe for keeping company with me and often providing transport to travel in and around Sydney. Furthermore, I would like to thank Mr. Tonmoy Dutta-Roy, Mr. Bashar Ramadin, and Mr. Ashod Donikian who accompanied me in my stay in Australia. My acknowledgements also are extended to CAS research students, Mr.Mashall Yuan and Mr. Dalong Wang in particular for their kind-hearted support. I would also like to thank all the academics of the CAS centre, and my friends Mr. Manjula Gunaratne, Mr. Sujeewa Fernando and Mr.Arjuna Dissanayake from Sri Lanka.

My abundant love must go to my parents for raising me and making my education a priority in their lives, for my loving wife Nadeeka sacrificing many things in her life to support my higher studies and to my loving new-born daughter Amaya for giving the added strength to successfully complete thesis work.

Lastly, I would like to dedicate this thesis to my parents, grandparents, all the good teachers who taught me from kindergarten to University entrance at St' Anthony's College, Kandy, and to all the lecturers who taught me at the undergraduate level at the Faculty of Engineering, University of Peradeniya, Sri Lanka for their encouragement to pursue postgraduate studies and to choose an academic career.

With Metha !

Asela K. Kulatunga
University of Technology, Sydney
Australia
29/08/2008

LIST OF PUBLICATIONS

Book chapters

1. **A.K. Kulatunga**, B. T. Skinner, D. K. Liu & H. T. Nguyen (2007), 'Simultaneous task allocation and motion coordination of autonomous vehicles using a parallel computing cluster'**,** *Robotic Welding, Intelligence and Automation,* Volume 362/2007, 409-420, Springer, Berlin/Heidelberg.

2. D.K. Liu &**A.K. Kulatunga**, (2007) 'Simultaneous Planning and Scheduling for Multi-Autonomous Vehicles', in Dahal, K., Tan, K.C. and Burke, E. (eds) *Evolutionary Scheduling, Studies in Computational Intelligence,* Volume 49/2007, 437-464, Springer, Berlin/Heidelberg.

Refereed Conference papers

3. **A. K. Kulatunga**, D. K. Liu & G. Dissanayake (2004) 'Simulated annealing algorithm based multi-robot coordination'. *Proceedings of the 3rd IFAC Symposium on Mechatronic Systems,* September 2004, Sydney, Australia, (Paper No. 74), 411-416

4. **A.K. Kulatunga**, D. K. Liu & S. B. Siyambalapitiya (2006) 'Ant colony optimization technique for simultaneous task allocation and path planning of autonomous vehicles.' *Proceedings of the IEEE International Conference on Cybernetics and Intelligent Systems (CIS),* 7-9 June, 2006, Bangkok, Thailand, 823-828

5. D.K. Liu, X. Wu, **A. K. Kulatunga**, G. Dissanayake (2006), 'Motion coordination of multiple autonomous vehicles in dynamic and strictly constrained environments.' *Proceedings of the IEEE International Conference on Cybernetics and Intelligent Systems (CIS),* 7-9 June, 2006, Bangkok, Thailand**,** 204-209

CONTENTS

List of Figures

List of Tables

Abbreviations

ACO	Ant colony optimization
AGV	Automated guided vehicle
AV	Autonomous vehicle
BD	Breakdown
BS	Batch size
BSA	Beam Search Algorithm
CIM	Computer Integrated Manufacturing
COF	Close proximity task first
CR	Cooling rate
CT	Container Terminals
DR	Dispatching Rules
FCFS	First-Come-First-Served
FMS	Flexible Manufacturing Systems
GA	Genetic algorithm
LAN	Local area network
MC	Motion Coordination
MPI	Message-Passing Interface
MS	Makespan
PP	Path Planning
RSI	Rescheduling interval
SA	Simulated Annealing
SCs	straddle carriers
SiPaMoP	Simultaneous Path and Motion Planning
SPS	Shortest Path Search
STAMC	Simultaneous Task Allocation and Motion Coordination
stu	simulation time units
TA	Task Allocation
TEU	Twenty feet Equivalent Units
TS	Tabu Search

Chapter 1

Introduction

1.1. Background of Autonomous Vehicle Operations

Supply chain management, which includes all activities covering the flow and transformation of goods from raw materials to end-user, has started to play an important role in the growth of the global economy. With rapid developments in technology over the past several decades, many improvements have taken place in order to cope with "Just-In-Time" delivery and to reduce the lead-time delays. This has been achieved by planning and scheduling all transportation activities using information technology wisely, and automating some of the transportation activities and the material handling in warehouses, inter-mode transportation terminals and manufacturing systems. A material handling system consists of material handling equipment such as trucks, forklifts and straddle carriers, and operational and management staff, information, materials and related planning and control systems.

Transportation activities have steadily boomed in order to deal with the rapid expansion of global trade. For example, in the container transportation sector, the capacity of ships has recently increased from 400 TEU (Twenty feet Equivalent Units) to the level 10,000 -12,000 TEU (Stahlbock and Vob, 2008). According to the February, 2006 issue of *Cargo News* "the container terminal business has expanded by more than 10 percent, annually, over the past 15 years. Fuelled by the globalisation of the world economy, this rate of growth is likely to continue. Even cautious forecasts indicate that the present demand will at least be doubled by 2015, with around 650 million TEU handled in the world's ports at that time" (CargoNews, 2006). Therefore, in order to deal with these trends, the performance of material handling systems has to be improved. This can be achieved by introducing new technologies such as automatic systems to increase the efficiency of material handling. Better planning and coordination strategies can achieve efficient material handling. According to Matson and White (1982), Operations Research concepts have been utilised to an extent to overcome these issues. This is further reinforced by the review of (Stahlbock and Vob, 2008).

Material handling systems are widely used in warehouses, manufacturing centres and container terminals. Warehouses are used to store inventories of raw materials and finished or semi-finished products. In manufacturing centres, material or product components flow from one conveyor to another or from one work centre to another. In container terminals, the 20 or 40 TEU boxes need to be loaded onto or unloaded from ships, and sometimes to other modes of transport. In addition, the boxes need to be stored until they are transported or transhipped.

The most attractive strategy to increase the operational efficiency of material handling in warehouses, inter-mode transportation terminals and manufacturing plants is to automate material handling systems. This is essential, since manually operated systems cannot achieve a high rate of efficiency in modern day supply chains. Furthermore, with the expectation of high levels of accuracy, human operators will soon be subject to work-related-stress and fatigue. This will directly cause accidents, hazardous situations and accidents ultimately leading to heavy financial losses. One of the ways to automate material handling systems is replacing manually operated vehicles and equipment with autonomous vehicles (AVs) such as automated guided vehicles (AGVs) or automated straddle carriers (ASCs).

An autonomous vehicle is an operator-less vehicle, which traverses on its own intelligent capabilities and other available information. The vehicle can usually carry a unit load or, in some applications, multiple loads. In most cases, there are many AVs working together as a group because of the high workload generated within a stipulated time.

In addition, these vehicles are operated in strictly constrained environments because the space is very limited in these applications. Due to these limitations, guided paths are often used for AV's to traverse in both directions (bi-directional). Since many AVs operate at the same time in a constrained area, it is essential to plan and coordinate them. Otherwise, it is impossible to achieve the expected productivity levels. Therefore, it is essential to have a proper mechanism to address the planning and coordination issues of multiple autonomous vehicles in material handling environments.

1.2. Planning and Coordination

When multiple autonomous vehicles are operating simultaneously in the same environment, there are many combinations or choices for allocating a task to a vehicle and it has to be performed in a productive manner. As these AVs operate in a space-constrained area and AVs traverse through planned paths, each AV should be given a safe and shorter route to travel from its pick-up location to its drop-off location on the path network. When AVs travel on these paths, it is essential to avoid collisions between them and blockages. The selection of particular vehicle to undertake particular task is known as **Task Allocation** (TA).Choosing the best possible path to travel is known as **Path Planning** (PP). In addition to path planning, an efficient collision and deadlock avoidance method also plays an important role. It is possible to consider path planning, velocity control and collision avoidance together, which is named **Motion Coordination** (MC). Planning and coordination of multiple autonomous vehicles consist of the following key functions: task allocation, path planning, and collision avoidance. Therefore, it is reasonable to formulate these issues collectively as a **multiple-vehicle task allocation and motion coordination** problem.

Apart from the above operational aspects of material handling systems, good layout design needs to be determined as well as the number of vehicles required to transport all tasks between various areas. However, good layout design and the number of vehicles required are beyond the scope of this research, and the focus of this thesis is on the task allocation and coordination aspects of the vehicles. In material handling environments, most of the tasks and vehicles (AGVs, SC, etc.) have equal priority and features, such as capability and carrying capacity. Furthermore, there will be many tasks to be performed within the same pre-defined time interval, or there will be a group of tasks, which are available for allocation at the same time.

The vehicles should be provided with clearly defined paths or routes to travel from one place to another. The commonly used method is to define the path edges by separating the storage areas and travelling areas. In the case of fully automated operation, metal strips or modern autonomous vehicle navigation systems guide these paths. Depending on the layout of the material handling area, there will be many possible routes to travel from one place to another in many instances; consequently, each vehicle may get a chance to select the path in which it travels to perform a task. When AVs travel on these paths, they maintain different speeds and accelerate or decelerate at different path segments; this is especially needed in order to prevent collisions when multiple vehicles travel on the same routes.

When multiple vehicles operate on the same network simultaneously, there should be an efficient traffic management system to coordinate them. Otherwise, they tend to collide with each other or meet with deadlocks or live-locks. Collision occurs mainly in the following ways: when two or more vehicles travel on the same path towards each other in a different or the same speed (head-on collision), when two or more vehicles travel on the same path at the same direction at different speeds (catch-up collision), or in a situation where different vehicles travel from different directions towards the same intersection point. This problem is aggravated when route segments are used for uni-directional movements. Therefore, it is essential to have an efficient and effective method to avoid collisions and deadlocks or live-locks.

Some research studies (Corréa et al., 2007, Le-Anh and De Koster, 2006, Meersmans, 2001a) have attempted to solve one aspect of the three issues highlighted earlier. When the task allocation aspect is investigated, it is commonly assumed that vehicles use the shortest paths from Pick-up locations to Drop-off locations. In addition, it is also normally assumed that there will not be deadlocks or collisions in the shortest paths. Similarly, when path planning or collision avoidance is discussed, it is assumed that task allocation is being done in an efficient manner.

Currently, the sub-problems of task allocation, path planning and collision avoidance of the multiple-vehicle task allocation and motion coordination problem are considered in a sequential manner. The available tasks are allocated among the available vehicles. Paths are then planned based on the AVs allocated tasks. Finally, the vehicles traverse the paths while collisions are avoided using different mechanisms. This scenario is shown in the Figure 1-1(a). Since task allocation, path planning and collision avoidance are handled at different levels in a sequential manner, it is difficult to guarantee the optimality of the results for the whole task allocation and motion coordination problem. When attempting to optimise the solution for one aspect and then for the other aspect consecutively, the optimal results found for the first aspect may not match the final results. Therefore, it is important to consider them simultaneously. This approach is shown in Figure 1-1(b). However, this has not been widely attempted.

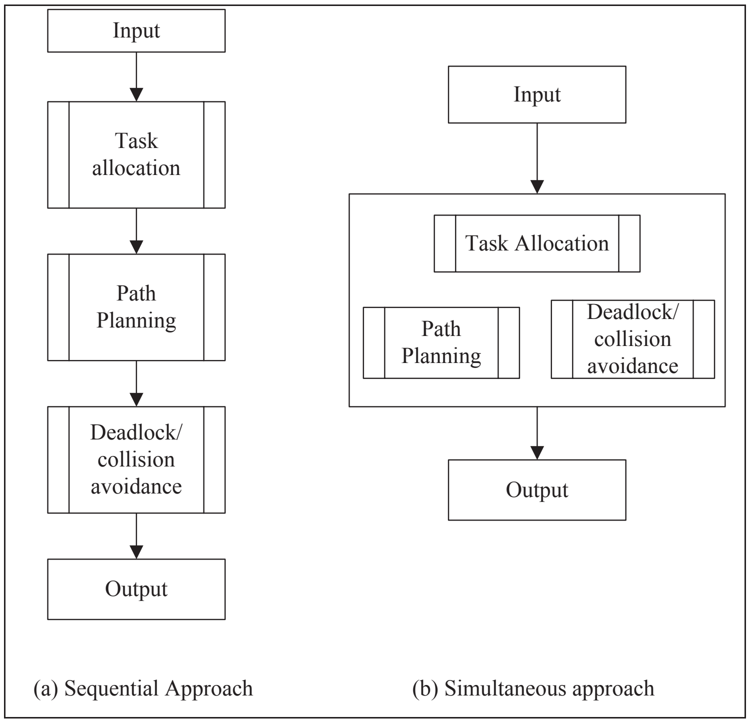


Figure 1-1: Two possible approaches to solve the multiple-vehicle task allocation and motion coordination problem

Integrating task allocation and motion coordination of autonomous vehicles is a challenging job. Task allocation itself becomes a tedious job due to nature of complexity, because there will be many combinations to select a vehicle for a particular task. Selection of the best or optimal combination requires the exploration of all possibilities. Therefore, the task allocation problem is categorised as NP-hard in mathematical terms. Path planning/routing part of the motion coordination problem is a necessity to search for the best possible route. As with task allocation, path planning is very time consuming. This problem is aggravated when it is necessary to avoid collision prone paths out of all possible paths.

Solving such a problem amounts to making discrete choices so that an optimal/near optimal solution is found among a finite or countable number of alternatives. It is impossible to find an optimal solution without the use of an essentially enumerative algorithm. This increases computational time exponentially with the problem size.

1.3. Scope of the Research and Contributions

Efficiency and effectiveness are very important for the overall productivity of the multiple autonomous vehicle system. Although there are different modes of transportation activities, (e.g. in the case of a container terminal, loading/ unloading of ships, moving containers inside the yard, and even responding to external loading/unloading activities), are required to perform in an integrated way. Collaboration among vehicles is very important for smooth functioning of the multiple autonomous vehicle system. In order to synchronise these activities, efficient task allocation or scheduling of autonomous vehicles and coordination of their motions are paramount necessity. Therefore, any approach should address the issues of task allocation and motion coordination in an integrated way.

There are many varieties of tasks, which require to be carried out at different time intervals in material handling systems. For example in Container Terminals (CT), the loading and unloading of containers from ships should be attended with urgency in order to reduce the turnaround time of ships and to reduce the congestion in crane areas. Loading/ unloading from external trucks and trains also needs to be given certain priority, with lower priority allocated to the movement of containers within yard.

Inefficient coordination can cause considerable traffic congestion near the crane areas and stacking areas, or in the terminal yard. Efficient task allocation would not solve all practical problems because there are a large number of vehicles working with limited resources or at high speeds. Therefore, routing of vehicles is as vital as good scheduling. Many congestion issues around quay crane areas as well as stacking/yard areas can be minimised by efficient routing.

The potential collision or deadlock issues of vehicles have to be addressed particularly when operating of large fleet of AVs. This is essential when vehicles travel bi-directionally in a road network in order to minimise the travelling time between two points and to reduce waiting times. Even uni-directional movement has to confront this situation in congested areas such as quayside and the yard/stack. Path planning should be done dynamically to facilitate smooth flow in the road segment inside the terminal.

In addition, new tasks normally arrive irregularly, and some new tasks may have higher priority. Therefore, it is necessary to accommodate new tasks within the allocation process and perform them accordingly. Sometimes, there will be uncompleted tasks, due to unavoidable circumstances such as vehicle breakdowns.

This research attempts to solve the multi-vehicle task allocation and motion coordination problem in material handling environments, where operating space is highly limited. This research work expects to consider task allocation, path planning and collision avoidance simultaneously in order to improve the collective solution quality of the multi-vehicle task allocation and motion coordination problem. Furthermore, there are many uncertainties involved in the multiple-vehicle task allocation and motion coordination problem, such as new tasks, vehicle breakdowns, or even unexpected operating environmental changes. The tasks also need to be allocated based on their priority or urgency. The proposed approach also tries to address these issues.

There are a number of approaches proposed in the literature of similar types of optimal assignment problems. Branch and bound or dynamic programming algorithms are often used to find exact solutions for such problems with the help of problem-specific information to reduce search space. However, this is valid only to very specific type of problems. Researchers have also come up with many greedy search algorithms, which are again dependent on specific conditions in their applications.

Many variations of local search algorithms for solving NP-hard problems have been proposed and investigated. Since the quality of the solutions obtained by local search algorithms strongly depends on the initial conditions, these local algorithms have the potential to perform poorly under some conditions.

Meta-heuristic and evolutionary techniques, which involve trial and error and some contemplated intuition, can produce an approximate solution, which will be near optimal. There is a clear trade-off between the solution quality and the computational time. In many practical situations, it is necessary to be content with near optimal solutions produced within reasonable time (Bagchi, 1999). This is acceptable and has been a practice in many fields such as engineering, economics and management. Therefore, this research will investigate the applicability of meta-heuristic and evolutionary techniques for task allocation in the multi-vehicle task allocation and motion coordination problem. The motion coordination (path planning and collision avoidance) element of the main problem will be handled along with task allocation problem simultaneously. The ways and means to reduce the computational time necessary to generate the near optimal results will be investigated in this research as well.

The principal objective of this research is to develop an integrated approach and algorithms, which would solve the complex multi-vehicle task allocation and motion coordination problem for a large fleet of autonomous vehicles operating in strictly constrained environments. The original contributions of this thesis include:

1. Simultaneous task allocation and motion coordination approach

   There are very few integrated approaches tackling task allocation and motion coordination in a simultaneous manner for large fleet autonomous vehicles. Furthermore, few have tried to address the solution quality of the available methods for a large fleet of autonomous vehicle systems. This thesis proposes a novel approach called the **S**imultaneous **T**ask **A**llocation and **M**otion **C**oordination **(STAMC)** to solve the complex multi-vehicle task allocation and motion coordination problem for a large fleet of autonomous vehicles.

2. Simultaneous path and motion planning algorithm for path and motion planning

   In order to solve the motion coordination sub-problem of the complex multi-vehicle task allocation and motion coordination problem, a novel algorithm, **Si**multaneous **Pa**th and **Mo**tion **P**lanning **(SiPaMoP)** is proposed and verified. The Dijkstra algorithm is used for the shortest path search while the path topology's node weights are changed dynamically to avoid potential collisions and deadlocks.
3. Dynamic STAMC approach

   In reality, there will be many unexpected events, which can create difficulties in setting up a schedule. Furthermore, there will be new tasks arriving to be allocated to vehicles from time to time. Therefore, a dynamic task allocation and motion coordination approach is studied in this research to handle these issues.

4. Decentralised architecture of STAMC approach

   In order to generate quality solutions to the complex multi-vehicle task allocation and motion coordination problem more efficiently, a distributed computational architecture is proposed in this research.

5. Case study – An application in a fully automated container terminal
   The proposed approach and algorithms have been tested in a large-scale map (with 14949 nodes) where up to 25 vehicles are operating in the environment simultaneously. The main purpose of this study was to investigate the effectiveness of the proposed algorithms in a large-scale operating environment with a large fleet of vehicles.

1.4. Organisation of the Thesis

The rest of the thesis is organized as follows. Chapter 2 presents a comprehensive review on: (1) integrated approaches to solving the multi-vehicle task allocation and motion coordination problem, including the exact method and heuristic/approximation methods; (2) task allocation methods for multiple vehicle problems; (3) routing methods for multiple vehicles; (4) deadlock and collision avoidance mechanisms available in multiple autonomous vehicles. Drawbacks of existing approaches are highlighted and possible avenues for research and development are presented.

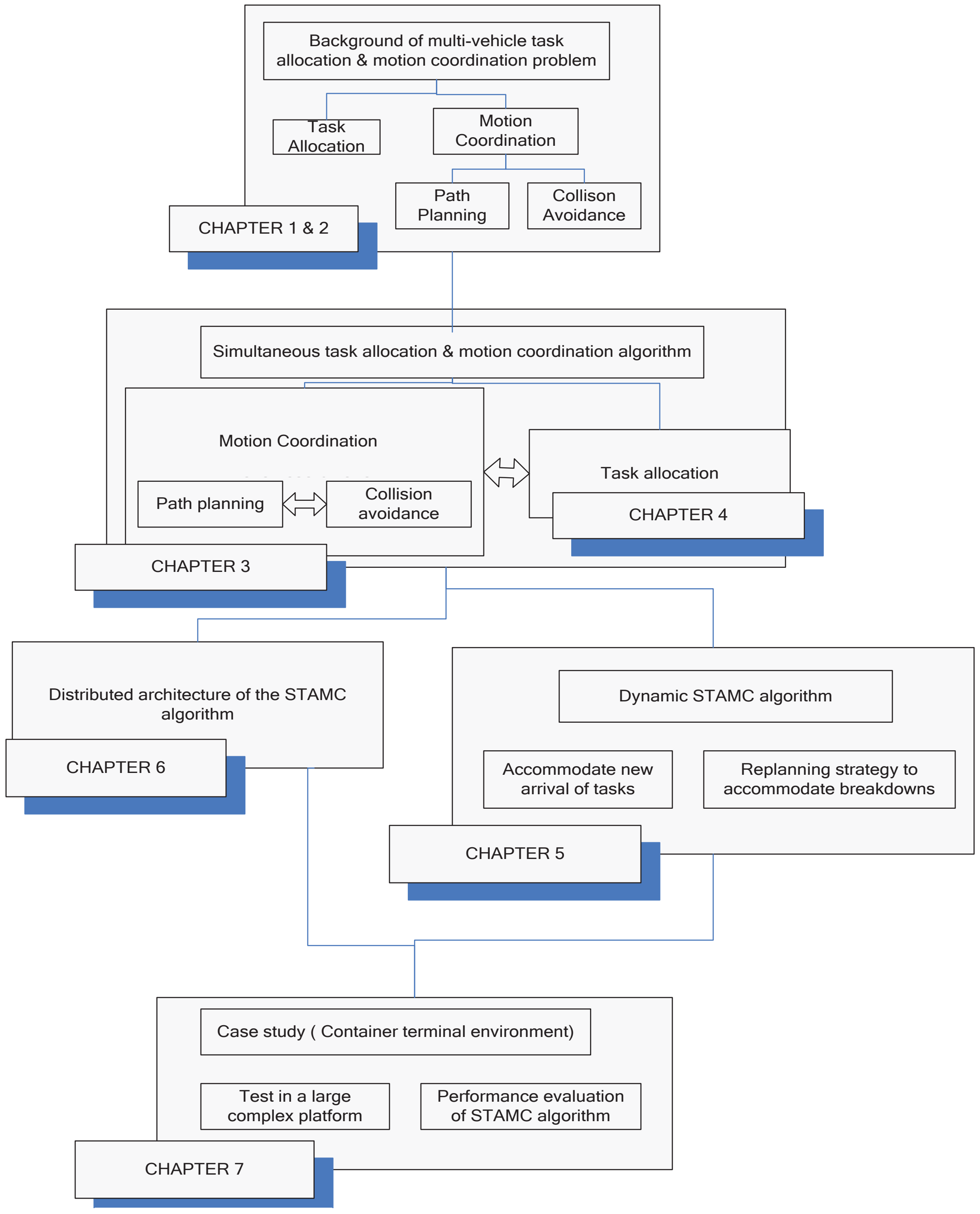
Chapter 3 presents the multiple autonomous vehicles task allocation and motion coordination problem. This Chapter is dedicated to the motion coordination aspect of the proposed approach and the SiPaMoP algorithm is presented including simulation studies.

In Chapter 4, the simultaneous task allocation and motion coordination approach is developed in addition to an extensive discussion of the task allocation component of the task allocation and motion coordination problem. Furthermore, simulation studies related to the STAMC approach are presented in this chapter.

The dynamic task allocation and motion coordination problem is presented in Chapter 5. This chapter aims to highlight the dynamic characteristics of the proposed simultaneous task allocation and motion coordination approach. Rescheduling, when new tasks arrive or due to vehicle breakdowns or external disturbances, is investigated in this chapter.

The implementation of the STAMC approach in distributed architecture is presented in Chapter 6 along with architecture development steps and the simulation results.

A case study of the proposed approach of an automated container terminal is presented in Chapter 7 along with simulation results. This is followed by the conclusions in Chapter 8. Further, outline of the thesis is given in Figure 1-2.



Background of multi-vehicle task allocation & motion coordination problem
Task Allocation
Motion Coordination
Path Planning
Collison Avoidance
CHAPTER 1 & 2
Simultaneous task allocation & motion coordination algorithm
Motion Coordination
Path planning
Collision avoidance
Task allocation
CHAPTER 4
CHAPTER 3
Distributed architecture of the STAMC algorithm
CHAPTER 6
Dynamic STAMC algorithm
Accommodate new arrival of tasks
Replanning strategy to accommodate breakdowns
CHAPTER 5
Case study (Container terminal environment)
Test in a large complex platform
Performance evaluation of STAMC algorithm
CHAPTER 7

Figure 1-2: Outline of the thesis

Chapter 2

Literature Survey

2.1. Introduction

This chapter introduces the background of multi-vehicle task allocation, motion coordination, path planning and deadlock/collision avoidance. It provides a detailed review on previous research in relation to the integrated approaches used to solve multiple autonomous vehicles task allocation and motion coordination problem. Exact approaches and approximation/heuristic approaches are investigated and reviewed. Detailed accounts on various methodologies found in the literature are given. The research papers on automated material handling found in literature mainly falls into two specific areas of manufacturing and transhipment terminals such as CTs. However, scale of the problem and complexity differ in the two application areas. For example, CT environments have large route networks compared to manufacturing environments.

The Chapter 2 is organised as follows: Section 2.2 review the integrated approaches related to multi-vehicle task allocation and motion coordination. Exact and approximation approaches are discussed. Sections 2.3, 2.4 and 2.5 present methods used for task allocation, routing and deadlock avoidance, respectively. In Section 2.6, possible research and development issues are highlighted with respect to overall efficiency and solution quality, optimisation techniques used to find solutions, and path and motion planning. This is followed by conclusions in Section 2.7. The structure of Chapter 2 is given in Figure 2-1.

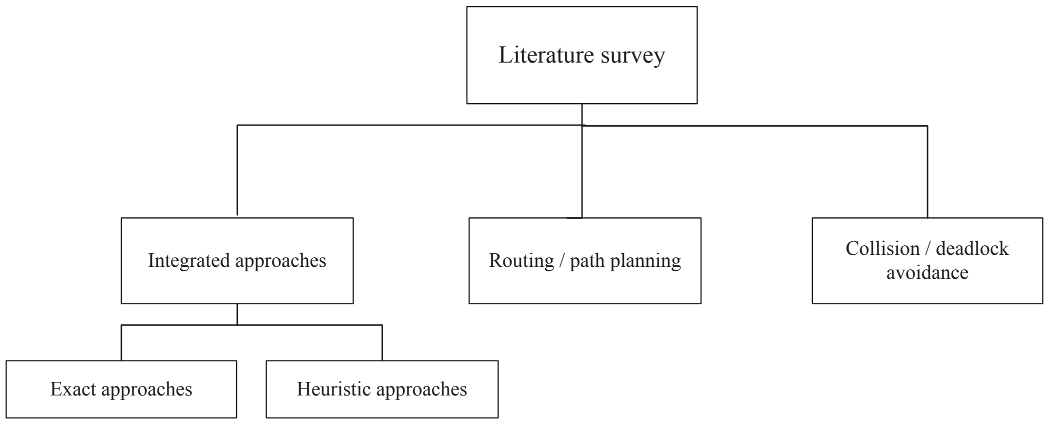


Figure 2-1: Organisation of literature survey

2.2. Integrated Approaches for Multi-Vehicle Task Allocation and Motion Coordination Problem

There are few attempts to solve the problem of task allocation, routing and collision avoidance in an integrated manner in the literature. Furthermore, these approaches can be subdivided based on the techniques used to solve the problem.

2.2.1. Exact Approaches

Exact approaches find the best or in mathematical terms, an optimal solution within a reasonable computational time. There are few approaches in the literature, which tried to find exact solutions for the multi-vehicle task allocation and motion coordination problem. A hybrid approach presented by Corréa, Langevin, and Rousseau (2007) for scheduling and routing of AGVs in a flexible manufacturing system is one of them. A decomposition method was adopted, where the master problem handles scheduling of AGVs and conflict free routing was considered as a sub problem. Constraint programming and mix-integer programming were used for the master and the sub problems, respectively. The optimal solution space of the scheduling problem was pruned by the logic cuts generated by the sub problem. This approach was capable of finding an exact solution for up to six AGVs and assumes the speeds of AGVs to be constant and limited only to a static environment.

Le-Anh and DeKoster (2005) investigated on-line dispatch rules and the behaviour of single and multi-attribute dispatch rules in internal transport systems. The impact of reassignment of vehicles was also investigated. Three performance criteria i.e. minimising average load waiting time, keeping the maximum load waiting time as short as possible and better utilisation of vehicles were used to investigate the efficiency of the approach. The results revealed that, the multi-attribute dispatch rules and reassignment of vehicles provide better results. In this research, routes were selected as shortest paths and collision issues were not considered. Furthermore, this approach could be used to solve small-scale problems and the speeds of the vehicles were assumed as constant.

A column generation method for scheduling of AGVs was proposed by Desaulniers, et.al, (2003) along with the dynamic programming technique for collision free route planning for a flexible manufacturing system. They used number of simplifications and assumptions in this research: (1.) Constant speeds for the AGVs,(2.) when one path is selected by an AGV, all nodes which were in the selected path were locked until particular AGV finishes its task, so that others cannot use them to plan their journeys, (3.)Simulations were limited to four AGVs.

2.2.2. Heuristic / Approximation Methods

Heuristic methods are the most common tool used for the problems when an optimal solution cannot be found within a reasonable computational time (NP-hard problems). There are a number of resource allocation related applications which were solved using heuristics techniques (Czarnas, 2002, Kim and Moon, 2003, Baker and Ayechew, 2003, Choi et al., 2003, Zhang et al., 2008, Seo et al., 2007, Chen et al., 2007). In the case of multiple vehicle task allocation and motion coordination related situations, there are few cases published in the literature. However, they were also focused on solving one aspect of the main problem with the assistance of heuristic techniques. The relevant research studies are presented below.

An integrated approach was proposed by Chen et al., (2007) to schedule the entire container handling equipment in a terminal. The problem was formulated as a Hybrid Flow Shop Scheduling problem with precedence and blocking constraints (HFSS-B).Minimising the makespan, or the time taken to serve a given set of ships was considered as the functional objective. A Tabu Search algorithm was used as the heuristic technique to solve the problem. Certain mechanisms were developed along with the Tabu Search algorithm to assure solution quality and efficiency. The performance of the Tabu search algorithm was analysed from the point of view of the computational cost. Meersmans and Wagelmans (2002) also proposed a similar kind of method to solve the scheduling and routing of all containers handling equipment in a terminal. The beam search algorithm was used as the heuristic technique. The path planning and collision avoidance of the vehicles were done with the assistance of traffic rules in a loop based path topology. The main schedule was prepared at the beginning of each day with incomplete information of future tasks. Subsequently, initial schedule was modified, when exact information of tasks arrives to the system. Longer planning horizons of the scheduling problem were achieved due to the strategy of scheduling with incomplete information. The main advantage of this approach was that this could be used for a large fleet of vehicles and its ability to solve the scheduling problem in an integrated manner. However, the vehicles' speeds and travelling times were considered as constant values. This study, further revealed that there is an advantage in planning on a longer horizon with inaccurate data, rather than doing it for shorter horizons and later to update the plan very frequently with accurate data as they arrive. An integrated approach proposed by Hartmann (2004)to schedule container terminal equipment, consisted of a general scheduling framework which includes SCs, AGVs, stacking cranes and the workers who handle reefer containers (containers which have a refrigeration facility). The priority rule based heuristic method and GA were used to solve the scheduling problem. To test the method, simulated data were used in this research. In the trial runs, the GA based method has given better results than priority rule based method. The collision avoidance and routing issues were not considered in this study and it was assumed that vehicles travelled at constant speed in a static environment.

Lau, Wong and Lee (2007)have presented an immunity-based control framework for a fleet of AGVs for material handling purposes of an automated warehouse. This method had the ability to detect changes in a dynamic environment and to coordinate AGV's activities. A robust and flexible automated warehouse system was developed through the self-organised and fully decentralised fleet of AGVs.

Further, there are number of researches (Chen et al., 2007, Lau and Zhao, 2008) which were developed to solve one of the common and important problems of integrated scheduling of all the equipment at container terminals. An intelligent decision making mechanism for loading operations in the CT was proposed by Zeng and Yang (2008). This approach generates an initial container loading sequence according to a certain dispatching rule, which was then improved by GA. A simulation study was carried out to evaluate the improved solution. Lau and Zhao (2008) have proposed a Mixed Integer Programming model for scheduling of CT equipment. A heuristic algorithm called multi-layer Genetic Algorithm (GA) was developed to reduce the computational difficulty in solving the mathematical model. The proposed method was tested in a fully automated CT environment with a cyclic path topology. Chen et al., (2007), have proposed the Tabu Search algorithm for the scheduling of CT equipment. The scheduling problem was modelled as a hybrid flow shop-scheduling problem with precedence and blocking constraints. The loading and unloading operations were considered separately. The AGVs were allocated for different ships while sharing yard cranes between different blocks for loading operations. The routing/path planning and collision avoidance related aspects were not considered in this study.

A Tabu Search algorithm based fleet sizing and vehicle routing method was proposed by Koo, Lee and Jang (2004) for container terminals with several yards. The main objective of this study was to find the smallest fleet size and routes for vehicles to fulfil all transportation requirements within a static planning horizon. The results revealed that the proposed method delivers quality solutions when compared with the existing method. Vehicle travelling time was considered as constant between two points. Kim and Kim (1999) attempted to minimise the total container handling time in a yard based on genetic and a beam search algorithms. Numerical experiments were carried out to compare the performance of the proposed algorithms against the optimal solution.

Langevin et al. (1996) presented a method for dispatching, conflict-free routing, and scheduling of AGVs in a flexible manufacturing system. The problem was solved optimally in an integrated manner, contrary to the traditional approach in which the problem was decomposed in three steps, sequentially. Rolling time horizon based solution was developed on dynamic programming technique. A heuristic version of the algorithm was also proposed as an extension to a large fleet of vehicles.

In addition, few research studies have attempted to solve the scheduling of machines and material handling AGVs of manufacturing systems simultaneously. One of them was proposed by Ulusoy et.al., (1997), to address the problem occurring in a flexible manufacturing system. The objective of the problem was to minimise the makespan of the schedule. GA was used as the optimizing technique and results revealed that for 60% of the problems, GA reached the lower bound indicating the optimality. The average deviation from the lower bound over all problems was found to be 2.53%. Additional comparison was made with a time window approach. In 59% of the test problems, GA outperformed the time window approach where the reverse was true in only up to 6% of the problems.

Further, one of the pioneering researches on multi-robot coordination was done at the LAAS centre (Aguilar et al., 1995) to address the movement coordination of a large fleet of ten or more robots in a network-like environment. Container transhipment centres such as harbours, airports and marshalling yards were considered as the application domains. Due to the complexity of the coordination problem, optimality was not considered when generating solutions. Task allocation was considered incrementally due to uncertainties in the dynamic environment.

2.3. Task Allocation for Multiple Autonomous Vehicles

Dispatching Rules (DR) has been used frequently for task allocation by many researchers for both static and dynamic environments. Bose et.al, (2000) proposed different dispatching strategies for straddle carriers which were dedicated to different gantry cranes. The main objective of this research was to reduce a vessel's turnaround time at a port by maximising the productivity of gantry cranes. This was achieved by an efficient schedule of given straddle carriers. Bish et.al, (2001) focused on the vehicle-scheduling-location problem of assigning yard locations to import containers and dispatching vehicles to the containers in order to minimize the total time for unloading a vessel. A heuristic algorithm was presented and the algorithm's performance was tested. This research was further extended by Bish et.al, (2005) and easily applicable heuristic algorithms were developed. According to their findings, in simple settings, most of those algorithms were able to find an optimal solution. In more generic settings, those algorithms were able to obtain near-optimal results for the dispatching problem.

Grunow, et.al, (2006)presented a simulation study of AGV dispatching strategies in a seaport container terminal, where AGVs could be used in the single or dual-carrier mode. The dual carrier mode allowed transporting of two small-sized or one large-sized container at a time, while in the single carrier mode, only one container was loaded onto an AGV irrespective of the size of the container. In their investigation, a typical on-line dispatching strategy adopted from flexible manufacturing systems was compared with a more sophisticated, pattern-based off-line heuristic approach. The performance of the dispatching strategy was evaluated using a scalable simulation model. The design of the experimental study reflects conditions, which were typical in a real automated terminal environment. Results of the simulation study revealed that the pattern-based off-line heuristic approach outperforms its on-line counterpart. For the most realistic scenario investigated, a deviation from a lower bound of less than 5% was achieved when the dual-load capability of the AGVs was utilised.

A solution topick up, dispatching and load-selection of multiple AGVs was proposed by Ho and Liu (2006).Nine pickup-dispatching rules were proposed. The impact of the proposed rules on each other's performance were also investigated. The experimental results revealed that the rule that dispatches vehicles to machines with the largest output queue length, was the best in all performance measures. Distance-based or due-time-based rules did not perform as well as queue-based rules. It further revealed that the performance of pickup-dispatching rules was affected by different load-selection rules.

An inventory based dispatching method was proposed by Briskorn et.al, (2006). The focus of this research was on the assignment of transportation jobs to AGVs within a terminal control system operating in real time. First, a common problem was formulated based on due times of the jobs. A greedy priority rule-based heuristic strategy and an exact algorithm were used to solve this problem. Subsequently, an alternative formulation of the assignment problem was proposed without due times. This formulation was based on a rough analogy to inventory management and which was solved using an exact algorithm. The idea behind this alternative formulation was to avoid estimation of driving times, completion times, due times, and tardiness because such estimation was often highly unreliable in practice and does not allow for accurate planning. By means of simulation, different approaches were analysed and it was shown that the inventory-based model leads to better productivity on the terminal than the due-time-based formulation.

Das and Spasovic (2004) developed a terminal scheduler to schedule straddle carriers in a container port. The objective was to minimise the empty travel of straddle carriers, while at the same time minimising any delays in serving customers. An assignment algorithm that dynamically matches straddle carriers and trucks conducted the scheduling procedure. Using a simulation model of a real system, the superiority of the proposed scheduler over two alternative scheduling strategies was justified.

Kim and Kim (2003) discussed approaches and decision rules for sequencing pickup and delivery operations for yard cranes and trucks, respectively. Their goal was to maximise the service level of trucks by minimising their turnaround time, both for automated and conventional terminals. A dynamic programming model for a static case (all arrivals of trucks are known in advance) was suggested. For a dynamic case (new trucks arrive continuously), a learning-based method for deriving decision rules was proposed alongside with several heuristic rules. Kim and Moon (2003) extended the problem presented by Kim and Kim (2003) to general yard-side equipment, such as gantry cranes or straddle carriers. Experiments revealed that the proposed Beam Search algorithm outperformed GA.

Grunow et al., (2004) presented a dispatching strategy for multi-load AGVs. A flexible priority rule was proposed and it compared with an alternative mix integer programming formulation for different scenarios. The main objectives of the research were to reduce lateness of AGVs in the multi-load mode and to improve the terminal's overall operational performance.

An auction algorithm based method was proposed by Lim et al., (2003) for dispatching AGVs in a general context with the objective of minimising the total empty travel time of AGVs. This method implemented a distributed decision making process with the facility to communicate among related vehicles and machines for matching multiple tasks with multiple vehicles. Future events were also taken into account. The results revealed that the distributed dispatching method outperformed shortest distance dispatching rule.

A vehicle management system was developed by van der Heijden et al. (2002), for a Dutch pilot project on an underground cargo transportation system using AGVs. Several rules and algorithms for empty vehicle management were developed, varying from First-Come-First-Served (FCFS) via look-ahead rules to integral planning. The various rules were embedded in a framework for logistics control of automated transportation networks. The planning options were evaluated for their performance in terms of customer service levels, AGV requirements and empty travel distances. Based on the experiments, it was concluded that look-ahead rules had significant advantages over FCFS and a more advanced serial scheduling method out-performed the look-ahead rules.

Steenken (1992) investigated methods to optimize the straddle carrier operation at the truck working area. The problem of assigning jobs to straddle carriers was solved with a linear assignment procedure combining movements for export and import containers. Different algorithmic approaches were used to solve the routing problem. Solutions were implemented in a real time environment and resulted in considerable gains of productivity.

2.4. Vehicle Routing and Path / Motion Planning

As discussed in Chapter 1, routing of vehicles in an effective and efficient manner contributes greatly to the overall efficiency of multi-vehicle coordination. There are many research studies conducted on vehicle routing problems (VRP), they include Lau et al., (2003), Baker and Ayechew, (2003), Tan et al., (2001), Liu and Shen, (1999), K.C. Tan, (1999), Barbarosoglu and Ozgur, (1999), Ochi et al., (1998), Achuthan et al., (1997) and Renaud et al., (1996).However, there are many differences of planning and coordinating requirements of vehicles in general VRPs and autonomous vehicles such as AGVs and SCs. All the VRP problems discussed in the literature are about situations where human operators (drivers) are involved. Conversely, autonomous vehicles are operator-free. The deadlock and collision are significant issues. Therefore, most of the findings related to driver driven vehicles cannot be directly applied to AGV based systems. Initial research on routing of AGVs came from manufacturing areas such as Flexible Manufacturing Systems (FMS) and Computer Integrated Manufacturing (CIM) systems and so forth. During the last decade, research works have been extended to areas like automated or semi-automated container terminals and warehouses. The pioneering research works on AGV routing can be found in Steenken et al. (1993), Dhouib and Kadi (1994), Taghaboni-Dutta and Tanchoco (1995), Evers and Koppers (1996) and Langevin et al. (1996).

Routing of AGVs largely depends on the path topology (route network layout) of the operating environment. The path topologies can be divided into three groups (Le-Anh and De Koster, 2006): single loop; tandem; and conventional. In some applications in a manufacturing sector, only one or two AGVs can fulfil the tasks. Frequently AGVs need to travel to different workstations in more or less the same sequence, for example, in FMS or CIM. For these types of operations, single loop path topology is suitable. In the case of automated container transhipment terminals, there will be a number of gantry cranes operating at different berths for loading and unloading of different ships. In order to cater for these cranes, a large fleet of AGVs is needed. AGVs are normally dedicated to different cranes. Hence they operate in different loops. In these instances, tandem path topology is used. In semi-automated material handling environments, fleets of AGVs operate in uni- and bi-directional paths in a complicated path topology with many crosses and path segments. The next section presents the previous works done on AGV routing in the above three path topologies.

Seifert (1998) presented a simulation model to analyse an AGV system operating under selected vehicle routing strategies. The proposed model could handle an arbitrary system layout as well as arbitrary numbers of AGVs and pedestrians causing congestion in the system. A dynamic vehicle routing approach was introduced and this was based on hierarchical simulation where, at the time of each AGV routing decision in the main simulation, subordinate simulations were performed to evaluate a limited set of alternative routes in succession until the current routing decision can be finalised and the main simulation resumed.

Routing of straddle carriers for loading operations of export containers was discussed by Kim and Kim (1999), with the objective of minimising the total travel distance of straddle carriers in the yard. A Beam Search Algorithm (BSA) was used to solve the routing problem and the proposed method was evaluated in numerical tests. In Kim and Kim (1999), the number of containers picked up by a straddle carrier at each bay and the sequence of bay visits were determined in order to minimise the total travel distance/time. The proposed integer programming model was solved by a two phase procedure.

Routing of AGVs in the presence of interruptions was studied by Narasimhan and Batta, (1999). Re-routing due to interruptions was achieved by accessing a route database to quickly obtain previously generated paths and using a flexible re-routing strategy. A simulation experiment was conducted based on data collected from a large manufacturing facility to justify the approach. Another dynamic conflict-free routing approach was proposed by Oboth and Batta (1999), which addressed the design and operational control issues of AGVs. In addition to design issues, operational control factors such as demand selection and assignment, route planning and traffic management, idle AGV positioning and AGV characteristics, were addressed.

The survey of scheduling and routing algorithms (Qiu et al., 2002) showed similarities and differences between scheduling and routing of AGVs and related problems such as VRP, the shortest path planning problem etc. The authors classified algorithms into groups of general path topology (static/time-window based/dynamic methods), of path optimisation (0-1-integer-programming model, intersection graph method, and integer linear programming model), specific path topologies (linear/loop/mesh topology) and dedicated scheduling algorithms.

The route planning method proposed by Sarker and Gurav (2005) suggested a bi-directional path layout and a routing algorithm, which generated conflict-free, shortest-time routes for AGVs in a manufacturing environment. Based on the path layout, a routing algorithm and mathematical relationships were developed among certain key parameters of vehicles and paths. A high degree of concurrency was achieved in vehicle movement. Routing efficiency was analysed in terms of distance and time required for AGVs to complete all pickup and drop-off jobs. The routing algorithm assumed that each AGV travels at constant speed.

Two classes of routing algorithms were proposed by Maza and Castagna (2005), namely optimised pre-planning algorithms and real-time routing algorithms. It was revealed that pre-planning algorithms have the advantage of producing optimal conflict-free routes, but cannot deal with changing situations such as vehicle delays and failures. Real-time algorithms have the advantage of being reactive, but cannot generate optimal path. In this paper, it was proposed to combine the advantages of both, as a two-stage approach. In the first stage, a pre-planning method was used to generate the fastest conflict-free routes for AGVs. In the second stage, conflicts were avoided in a real-time manner when needed. The objective of the second was to avoid deadlocks in the presence of interruptions while maintaining the established AGVs routes. The efficiency of this approach was analysed using developed simulations.

Nishi et al., (2007) proposed a distributed routing method based on motion delay to prevent disturbance for multiple AGVs. The proposed method has the characteristic to derive its optimal route of each AGV to minimise the sum of the transportation time and the penalties with respect to collision probability with other AGVs. The proposed method was applied to a routing problem for transportation in the semi-conductor fabrication bay with 143 nodes and 20 AGVs. The results showed that the total transportation time obtained by the proposed method was shorter than that of the conventional method. For dynamic transportation environments, an optimal timing for re-routing multiple AGVs under motion delay was determined by the trade-off between the total computation time and the uncertainties for re-routings. Markov chain was used to represent uncertainty distribution for re-routings. The proposed method was implemented in an experimental transportation system with 51 nodes and 5 AGVs.

A mathematical model for the unidirectional path design problem was developed by Seo, et.al., (2007) for an AGV system. To obtain a near optimal solution within a reasonable computation time, a TS algorithm was used to solve the routing problem.

2.5. Collision and Deadlock Avoidance

One of the most important issues of multi-vehicle coordination is to avoid deadlocks. Previous researchers answered this problem in different ways (Corréa et al., 2007, Le-Anh and De Koster, 2005).Deadlock prevention can be incorporated into either the integrated approach or the routing methods. These methods can be classified into three categories: traffic control rule-based methods, zone control policies, and collision-free best possible routes.

Traffic rule based collision avoidance can be applied in any path topology. These rules act similar to normal traffic light systems in the intersections of each path, but rules can be defined based on the priority and network congestion. In zone control policies, network was partitioned into zones. Each zone was exclusively assigned to one AGV to avoid collisions. In time-window based methods, the occupation times of all connections were maintained in a database. When a new route was required to be selected, free time slots of the selected path segments were checked. If there were no free timeslots for the selected path segments, other feasible paths that contain free time slots were selected.

Traffic rules were used for collision avoidance in Sarker and Gurav (2005), Grossman (1998). Liu et al. (2004) used traffic rules to eliminate traffic jams in a grid-based road network environment. In his research, restricted route selection was compared with autonomous route selection and revealed that the proposed approach produced near optimal results even for a large fleet size. Sarker and Gurav (2005), proposed traffic rules to be used in the intersections of the bi-directional path topology of a manufacturing facility. The routing efficiency was calculated in terms of the AGVs travel distance and time required for an AGV to complete its tasks. Liu et al. (2004) also used traffic rules in the intersections to avoid collisions in an automated container terminal environment. In addition, Narasimhan and Batta (1999) and Qiu, et al. (2001) also used traffic control rules in their routing approaches.

Kim et al. (2006) proposed a deadlock avoidance method for the AGVs. The objective of this study was to develop an efficient deadlock prediction and prevention algorithm for AGV systems in an automated container terminal. Since the size of an AGV was much larger than the size of a grid-block on a guide path, this study assumed that an AGV might occupy more than one grid-block at a time. This study proposed a method for reserving grid-blocks in advance to prevent deadlocks. A graphical representation method was suggested for a reservation schedule and a priority table was suggested to maintain priority consistency among grid-blocks. It was shown that the priority consistency guarantees deadlock-free reservation schedules for AGVs. The proposed method was tested in a simulation study.

Moorthy et al. (2003) proposed a zone control-based collision avoidance method. The main advantage of this method was that it could accommodate a fleet size up to 80 AGVs. Here, the routes were divided into working and service lines. The proposed method was implemented in AutoMod simulation software. A similar type of method was proposed by Oboth and Batta (1999), which selected the shortest route and blocked the selected path till the respective AGV finishes its job.

The agent-based technique developed by Singh and Tiwari (2002) and real time AGV routing approach proposed by Mohring et al. (1998) use a time-window approach to overcome collision issues.

The conflict-free routing scheme proposed by Zeng and Hsu (2008) varied AGVs' speeds in order to change the time for the AGV to reach a node where there were possibilities of collisions. This was tested in a mesh like path topology of a container yard layout.

2.6. Research and Development Challenges

2.6.1. Overall Efficiency and Solution Quality

The selection of best strategies for task allocation, path/route planning and handling the single and dual cycle modes especially in container terminals directly affects the overall quality of solution and the efficiency. For example, in some instances, for task allocation, tasks are selected in a first-come first-served basis or following a task sequence determined previously (Bish et al., 2005). However, there can be many task sequences, which may be able to increase the overall efficiency. Therefore, it is worthwhile to investigate the possible alternative task sequences. Conversely, routing and deadlock/collision prevention strategies used in the literature more often reserve either adjacent zones (Moorthy et al., 2003) or path segments for a vehicle till it finishes its task (Desaulniers et al., 2003, Oboth and Batta, 1999). Therefore, other vehicles cannot use those zones or path segments to deliver their tasks and this reduce the solution quality.

In some of the fully automated material handling environments, AGVs are guided through loops in order to prevent collisions and deadlocks. This especially occurs in path layouts such as loop based systems and zone based systems. An example for this instance is automated container terminals where vehicles are used for either loading or unloading. This is achieved by traversing in loops. However, this leads to more empty travels and thereby reducing the overall efficiency. In contrast, though the dual cycle mode is complex to coordinate, higher overall efficiency can be achieved by reducing empty travel times. Although dynamic schedule proposed by Meersmans (2002) for the automated container terminal, they considered only loading activity for the scheduling. However, in Das and Spasovic (2004), they did not differentiate the allocation process based on cycle modes as they considered loading of landside transportation. Whenever SC finishes its task, it would be allocated with a new task available in the pool. Since this approach was not limited to one cycle, there was not many empty travels of SCs. However, if they accommodate seaside transportation to the terminal scheduler, there was a possibility to achieve overall high transportation efficiency of the terminal. The general framework presented by Hartmann (2004) for scheduling equipment and manpower only considered the transportation of both sides of the terminal. Further, it was not narrowed down to cycle modes of loading/unloading, though this system was not run on real data sets.

The approach used by Bose et al.,(2000), single and double cycle modes were discussed. In a semi-dynamic assignment, a fixed number of SCs was allocated for one/ gantry crane of one ship while in dynamic assignment. A fixed number of SCs was assigned to all gantry cranes. Nevertheless, in order to simplify the problem, this approach ignored the stacking and destination of discharge and assumes a constant number of SCs and a constant velocity.

Based on the above research studies, Hartmann and Bose (2006) and (Bose et al., 2000) consider a dual cycle mode without limit into loading or unloading. However, Hartmann's method did not use real data to evaluate his approach. Further, this work did not take into account the velocity changes, the breakdowns of the vehicles etc. The same comment applies for Bose et al. (2000), due to its assumptions.

2.6.2. Optimisation Methodologies

Three types of methodologies are commonly used for the task allocation in the literature, namely: the heuristic and evolutionary approaches, mixed integer linear programming, and the dispatching rules. Evolutionary algorithms such as GA have been used in many studies (Bose et al., 2000, Lau and Zhao, 2008, Qingcheng and Zhongzhen, 2008 and some cases as a comparison (Briskorn et al., 2006). Heuristic approaches such as Beam Search Algorithm (BSA) (Meersmans, 2002, Kim et al., 2004, Kim and Kim, 2003), Tabu Search (TS) (Koo et al., 2004, Seo et al., 2007, Chen et al., 2007), Simulated Annealing (SA) (Kim and Moon, 2003) are also studied by logistic/transportation researchers. However, according to our understanding, a proper comparison has not been done about these techniques and that is a necessity.

GA was used as the general scheduling approach in (Briskorn et al., 2006), which schedules SC, AGVs and workforce of a container terminal. The GA based results have outperformed priority rule based heuristics with less computation time. GA was used to improve the solutions of dispatching strategies in Bose et al. (2000). The results revealed that GA has the potential to improve the solution quality. Ulusoy et al., (1997) used GA to schedule machines and AGVs simultaneously in manufacturing environment.

BSA was used by Meersmans (Meersmans, 2002) as a prime tool for integrated scheduling in static and dynamic environments in an automated container terminal. Further, BSA was used to minimise the total handling time of cranes and trucks of a container terminal by Kim et al., (Kim et al., 2004a). Their results revealed that the BSA outperforms the Ant algorithm and neighbourhood search. (Kim and Kim, 1999) used BSA for loading operations of the SCs. The objective was to minimise the total travelling time of the SCs in the yard. TS algorithm was used for fleet sizing and vehicle routing by Koo et al. (2004) in a container terminal with several yards. Their objective was to find the minimum fleet size and routes of vehicles. This work was done for a static environment and assumed that the vehicles travel times are fixed.

Scheduling/task allocation problems are also solved by linear assignment procedures (Steenken, 1992), as an assignment problem by (Das and Spasovic, 2004b), by dispatch rules (Leong, 2001; Grunow et al., 2004b; Le-Anh and De Koster, 2005; Bish et al., 2005; Corréa et al., 2007) and by auction algorithms (Lim et al., 2003, Henesey et al., 2003).

2.6.3. Path and Motion Planning Issues

Some issues in this section have already been presented in previous sections. This section focuses on issues that can arise when planning the paths for the transportation tasks of vehicles. Most of the complexities arise due to path layouts or path topologies of the environment. For example, it is easier to schedule and route the AGVs in loop-based path topology than a conventional type path topology, which contains a grid network environment where each path segment facilitates the bi-directional movement. There is always a trade-off between transportation efficiency and path topology of the system. Where the layout of the path is based on a uni-directional or a loop system, then empty travel distances will be higher and transportation efficiency will be less. Conversely, when routes are bi-directional, empty movements are reduced and transportation efficiency goes up. Path planning issues are more complex in bi-directional transportation systems. This is mostly due to the collisions among the vehicles operating in such environments.

A traffic control system based on semaphore techniques was proposed by Evers and Koppers (1996). In the semaphore based system, only one vehicle can be used in certain segments of the path until it moves. In other words, the particular segment is locked by the occupied vehicle. Path planning for a large fleet is not practical with this method as transportation efficiency will be reduced when fleet size increases.

In order to avoid deadlocks and collisions, many path and motion planning approaches to locking the selected path segments or nodes until the respective vehicle finishes its task have been proposed by many researchers (Corréa et al., 2007, Le-Anh and De Koster, 2005, Narasimhan and Palekar, 2002 and Qiu et al., 2001). During the locked period, none of the other vehicles can use those path segments or the nodes, affecting the overall efficiency. The exception of Wallace's (2001), all other zone-based systems use exclusive zones in their routing or path planning. Therefore, it is necessary to introduce path and motion planning methods, which have less usage restrictions.

Most of the researches discussed above have not considered motion aspects of the transport vehicles in their approaches. They have assumed the speed of the vehicles to be constant. However, this is not realistic. It would be preferable for speed variations to be accommodated in the path planning process. Thus, they can be referred to as either path or motion planning or motion coordination problems.

In considering all the aspects of the available methods, there remains a need for an efficient path and motion planning technique, which could especially be useful in conventional (bi-directional) types of path topology for a large fleet of vehicles.

2.7. Summary

Based on the findings in the literature, several important areas, which require more investigation that is extensive, have been identified. These can be classified into the following areas;

i. Efficiency of overall container terminal operation

   A great deal of research has been done on one operational aspect of the container handling problem, such as quay crane scheduling, prime mover scheduling, deadlock and collision avoidance etc. Only a few attempts have been made to integrate these sub-problems into a single problem. Therefore, it is difficult to enhance the overall terminal efficiency.

ii. Solution quality

   A number of past research papers have used dispatching rules for task allocation. Only a few studies have focused on TA related solution quality improvement. However, there are a number of ways to get at least near optimal solutions within a reasonable time frame.

iii. Scalability and applicability

Most research studies have been conducted based on scaled down environments with a number of assumptions, which limits their adaption in large-scale real world material handling environments such as container terminals.

iv. Uncertainty

The real world material handling systems have to face many unexpected events such as sudden vehicle breakdowns, blockages in the routes, etc. However, these issues have been extensively addressed in research studies. Addressing these contingencies is essential to manage real world large-scale material handling systems.

v. Computational efficiency

There is always a trade-off between the quality of solution and computational efficiency needed to generate the optimal or near-optimal solution. Even if there is a method that can find the optimal solution it will have to be concerned with computational efficiency. Otherwise, it will not be possible to be used in real world situations.

Several reviews (Vis and Koster, 2003, Qiu et al., 2002, Le-Anh, De Koster, 2006, Vis, I. F. A. 2006) have highlighted that more effective and efficient methods are needed to address the breakdown, scheduling and routing issues of a large fleet of autonomous vehicles. Therefore, this study aimed at developing a methodology, which will address the overall efficiency of task allocation and motion coordination while enhancing scalability.

The proposed way to address the efficiency and solution quality issue is to develop an integrated approach to task allocation, path planning and collision avoidance. Task allocation and motion coordination will be done simultaneously. In addition, to reduce empty travel time and distances, the proposed method will use conventional type path topology, which facilitates bi-directional movement.

Furthermore, in the motion coordination stage, where collision-free paths are planned, the selected path segments will not be blocked for other vehicles. Thus, other vehicles can use them with some safety constraints. This proposed approach will also have provision to react to unexpected events such as vehicle breakdowns or sudden environment changes like road blockages. A comprehensive comparison among different methods in real applications is necessary. For example, when evolutionary algorithms or generally applicable heuristics are used, it is always better to compare them with other types of algorithms such as heuristic. Evolutionary methods are dependent on the problem set up. It is rare to see many comparisons of different approaches to this problem. Furthermore, studies have not covered the computational efficiency-related issues under real world scenarios. The proposed simultaneous task allocation and motion coordination approach will address this research gap.

Chapter 3

Problem Formulation and Simultaneous Path and Motion Planning Algorithm

3.1. Introduction

This chapter presents the multi-vehicle task allocation and motion coordination problem and the method proposed for motion coordination. The problem is defined as a generic multi-vehicle task allocation and motion coordination problem, which is applicable to automated CTs, warehouses and manufacturing systems. As discussed previously, when multiple autonomous vehicles operate in a strictly constrained environment, the space allocated for paths is very limited. Paths are, therefore, designed to facilitate bi-directional movement. This type of path topology is called a conventional path layout (Le-Anh and De Koster, 2006). However, this path topology leads to more collisions and deadlock situations than other path topologies. Efficient coordination techniques are required to coordinate motions of multiple AGVs.

Most of the earlier research studies have focused on some aspects in task allocation and motion coordination for specific applications such as container terminal operation or manufacturing systems. It is possible to consider all transporting operations in a container terminal as a set of tasks to be performed by a fleet of vehicles in a prioritised or sequential manner.

The importance of tasks varies with time. For example, in a container terminal, the tasks of loading and unloading of ships are of paramount importance. In contrast, when there are no ships in the terminal, transporting containers within the yard or loading to outside trailers might be more important. In general, it is justifiable to assume different priority levels for transporting tasks at different time periods.

The rest of the chapter is organised as follows. The task allocation and motion coordination problem is presented in Section 3.2. The mathematical representations and modelling of the path and motion planning and collision avoidance sub-problems and the proposed SiPaMoP algorithm is given in Section 3.3. The simulation environment is presented in Section 3.4. Simulation and case study results that show the performance of the approach are presented and discussed in Section 3.5. Finally, Section 3.6 gives a discussion and concluding remarks.

3.2. Task Allocation and Motion Coordination Problem

In automated container terminals or manufacturing environments, a fleet of AGVs or automated SCs are normally in operation concurrently. These vehicles can carry unit loads (e.g. a box in a container terminal or a palette in a manufacturing plant) at any instance. Therefore, the transportation operation can be modelled as a constraint transportation problem. This problem consists of ***m*** number of vehicles and ***n*** number of tasks. Task allocation (TA) decides which task is allocated to which vehicle. If tasks are allocated to vehicles by using commonly used dispatching rules, it is difficult to guarantee the optimality of the allocation. Selecting the most suitable vehicle for a given task is a crucial decision, which significantly affects the overall efficiency of the transportation system. This is categorised as an optimal assignment problem, which is known to be NP-hard (Bagchi, 1999).

Simultaneously planning the appropriate path for a vehicle to undertake the allocated task and avoiding possible collisions with other vehicles while travelling make the complex allocation problem even more complicated. Multi-vehicle task allocation and motion coordination problem consists of three main components, namely, task allocation, path planning and collision/deadlock avoidance, as explained in Chapter 1. Collisions occur mainly in the following ways:

i. When two or more vehicles travel on the same path towards each other at a different or the same speed (head-on collision)
ii. When two or more vehicles travel on the same path in the same direction at different speeds (catch-up collision)
iii. In a situation where different vehicles travel from different directions towards the same intersection point at the same time.

The multi-robot path planning and collision/dead lock avoidance, in general terms, can be referred to as motion coordination. These two aspects need to be integrated in order to come up with a deadlock free optimal path, once a vehicle has been chosen for a task. The multi-vehicle task allocation and motion coordination problem is schematically represented in Figure 3-1. In the task allocation and motion coordination problem, autonomous vehicles are handling transport operations where task pick-up and destination positions are scattered around different locations. These vehicles travel in a bi-directional path network to perform tasks. When one vehicle finishes its current task, that vehicle will be given another task to perform unless the entire available tasks are allocated. The main challenges of solving a multi-vehicle task allocation and motion coordination problem are;

i. to allocate the most appropriate vehicle for a given task
ii. to select the best path (route) for a vehicle based on the current situation
iii. to avoid possible collisions among vehicles while travelling on a bi-directional path network.

Usually these three challenges are addressed separately.

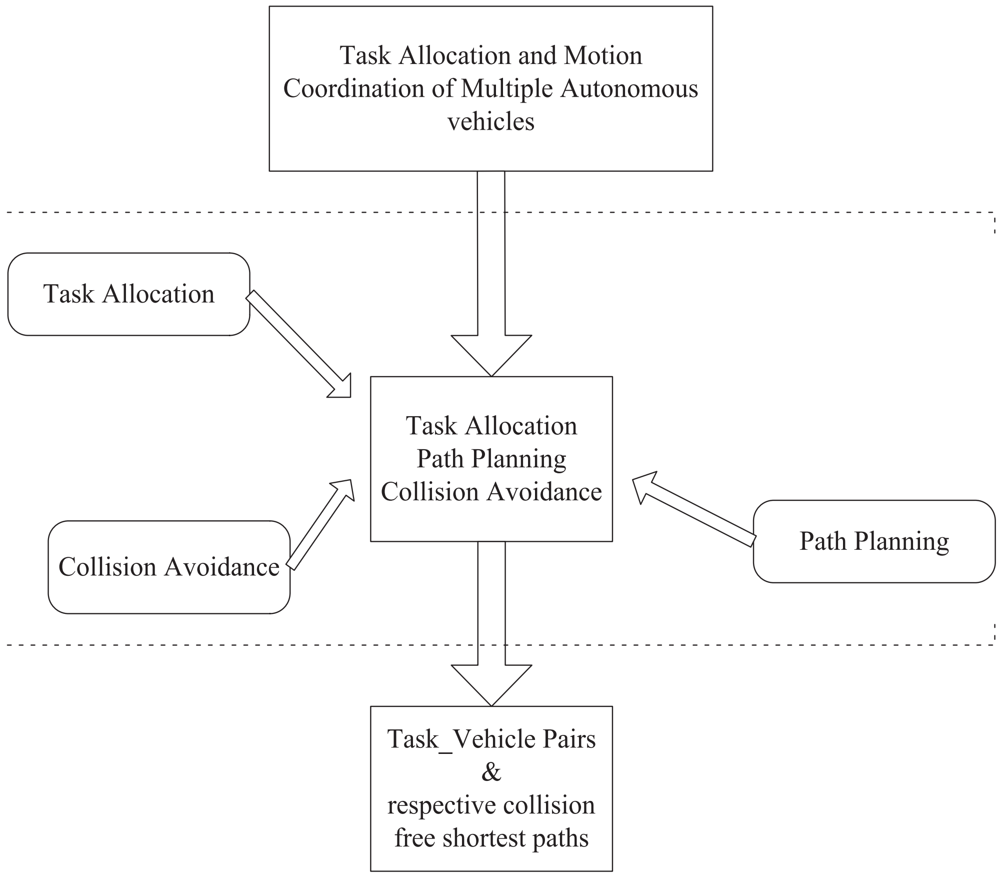


Figure 3-1: Schematic representation of the multi-vehicle task allocation and motion coordination problem with three key sub-problems

The overall efficiency of the multi-vehicle system is expected to be increased very considerably, if task allocation, path planning and collision avoidance are solved simultaneously. In this chapter, path planning and collision avoidance issues are addressed together as an integrated problem. A novel algorithm called **S**imultaneous **P**ath and **Mo**tion **P**lanning (SiPaMoP) is proposed for motion coordination of multiple AGVs in a strictly constraint environment.

The task allocation aspect integrated with motion coordination problem is presented separately in Chapter 4, along with the proposed simultaneous task allocation and motion coordination approach.

3.3. Motion Coordination and SiPaMoP Algorithm

Motion coordination plays an important role in the multi-vehicle task allocation and motion coordination problem. Motion coordination consists of both collision-free path and motion planning for multi-autonomous vehicles in various environments and operating conditions. Unlike single autonomous vehicle motion planning, motion planning for multiple autonomous vehicles not only requires getting vehicles from pick-up to drop-off without colliding with obstacles either stationary or moving, but also requires the achievement of a minimum level of congestion while ensuring maximum productivity of the whole system.

Path planning algorithms have been studied for autonomous vehicles operating in various (known, unknown or partially known) environments, for example, D* (Stentz, 1994), Delayed D* (Ferguson and Stentz, 2005) and E* (Philippsen et.al, 2005) algorithms. Meta-heuristic and evolutionary algorithms have also been studied for applications in path planning, examples include particle swarm optimisation (Qin et.al, 2004), genetic algorithm (Chiba et.al, 2004) etc. For applications in known environments such as road networks, path planning is usually solved in two steps: (1) build a graph to represent the geometric structure of the environment and (2) perform a graph search to find a connected path between the pick-up and destination points.

Finding shortest paths in known and dynamic environments appears to have divergent approaches. Chabini (1997 and 1998) has classified dynamic shortest path problems. Fu and Rilett (1996) have investigated the dynamic and stochastic shortest path problem by modelling connection travel time as a continuous-time stochastic process. The motion coordination of a fleet of autonomous vehicles in a strictly constrained environment is a very hard problem to solve. The majority of published research studies on shortest path planning algorithms, which are often used to find the ‘best’ paths for road vehicles in known environments, have dealt with static road networks that have fixed topology and few constraints. The roadmap approach, which is based on the concepts of configuration space and continuous path, is one of the most common approaches to vehicle path planning in road networks. Zhan and Noon (1996) presented a comprehensive study on shortest path planning algorithms on 21 real road networks in the U.S., with networks ranging from 1600/500 to 93000/264000 nodes/arcs. In this study, Dijkstra-based algorithms outperformed other algorithms in one-to-one or one-to-all fastest path problems. Husdal (2000) reported that the A* algorithm and Dijkstra-based algorithms have been preferred in most of the literature.

Autonomous vehicle path planning in a known environment such as container terminals has been attempted to provide more realistic and versatile solutions over the last decade. However, the following issues still remain unresolved:

i. Most of the shortest path algorithms generate the shortest path without taking into account the number of vehicles, and the vehicle speed/turning variations. Path planning and scheduling are separated, and no generic search approach has been reported to conduct planning, scheduling and collision avoidance simultaneously.
ii. Current approaches cannot efficiently manage congestion and bottleneck areas, which cause the biggest difficulties in planning.
iii. How path and speed/turning planning algorithms accommodate dynamic traffic conditions.
iv. Inefficiency and lower productivity caused by simple rules-based collision avoidance methods.

Focusing on solving the above problems, this section presents a Simultaneous Path and Motion Planning approach (Liu, Wu and Kulatunga, 2006) for multi-autonomous vehicle motion coordination in strictly constrained environments. This approach plans collision-free paths and speeds for all vehicles in an environment by dynamically changing the vehicles’ path and/or travel time/speed between nodes to minimise the completion time of a set of tasks, and as a result, maximise the productivity.

Path and motion planning of autonomous vehicles in container terminals and material handling environments strive to maximise the productivity of the whole system in a given period of time, for example, one day or one week. Due to the dynamics and uncertainties of operation, productivity is significantly affected by a number of factors such as the environment, vehicle path and collision avoidance strategy or bottleneck area. Figure 3-2 shows a schematic simultaneous approach for path and speed planning and collision avoidance with the objective of minimising travel time and avoiding collisions with any stationary or moving obstacles. Allocated tasks, dynamic traffic information and static and dynamic obstacles (e.g. containers in the terminal) previously located and new arrivals are the main inputs to the simultaneous path and motion planning algorithm.

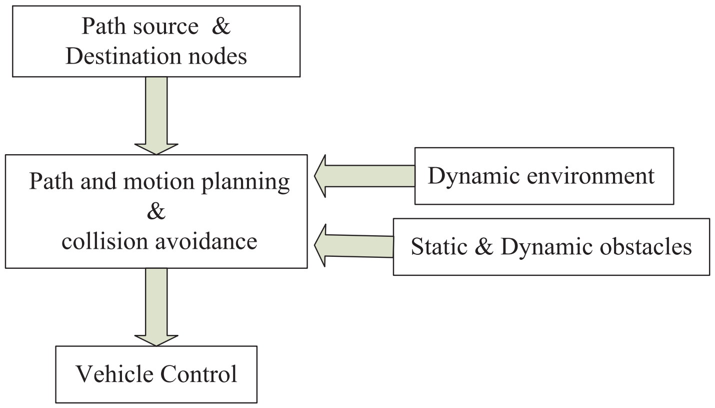


Figure 3-2: Schematic representation of simultaneous path and motion planning approach

The pick-up and drop-off nodes of a task define a task where an automated material handling equipment (AGV or SC) picks-up a unit load (it can be a container or pallet) from the source node and transports to its drop-off node via linked connections (connected with adjacent nodes). The following assumptions are considered in formulating the path and motion-planning problem:

i. Vehicles are running on time
ii. All transport orders are known in advance
iii. No expected delivery time applies to the transportation orders
iv. There are n tasks to be allocated to m vehicles

The main objective of the simultaneous path and motion-planning problem is to minimize the total completion time of all the vehicles, which involve transportation activities. The total completion time of vehicles can be given as:

$$t_{total} = f\left(P_{s,i}, V_{s,i}, r_{s,i}, t_w, t_{re-p}, t_0\right) = \sum_{s=1}^{m} \sum_{i}^{n_s} t_{s,i} \qquad (3.1)$$

The total completion time is the function of path $\mathbf{P_{s,i}}$, speed $\mathbf{V_{s,i}}$, waiting time $\mathbf{t_w}$ and path replanning time $t_{re\text{-}p}$ needed to complete the tasks. And **m** is the number of vehicles, $\mathbf{n_s}$ is the number of tasks allocated to vehicle **s,** $\mathbf{t_{s,i}}$ represents the travel time for the vehicle **s** to complete its task **i.**

The $t_{\mathbf{s,i}}$, (time needed for a vehicle to complete a task), can be calculated by:

$$t_{s,i} = \sum_{j=1}^{k_{s,i}} \frac{S_{s,j}}{v_{s,j}} \Big|_{v_{s,j\neq 0}} + \sum t_w \Big|_{v_{s,t=0}} + t_0 \qquad (3.2)$$

$t_{s,i}$ has five components; time for vehicle s to perform task i is determined by the number of connections $k_{s,i}$ among nodes in the path $P_{s,i}$ the distance $S_{s,j}$ between nodes, average speed $v_{s,j}$ of the vehicle travelling $S_{s,j}$, sum of waiting time t_w (when the speed is zero) and loading or unloading time t_o. The parameters of $k_{s,i}$, $P_{s,i}$, $S_{s,j}$, $v_{s,j}$, and t_w, are undetermined at the planning stage and they vary with the change of environment, traffic conditions, obstacles, planning strategies, etc.

A vehicle's motion is expressed as: $\boldsymbol{P}_{s,i} : [\boldsymbol{p}_{s,1}, \boldsymbol{p}_{s,2}, \ldots, \boldsymbol{p}_{s,k_{s,i}}]$ path segments of vehicle s for performing task i. The length of a path segment, $p_{s,j}$ is the $S_{s,j}$ shown in Equation (3.2), and $\boldsymbol{V}_{s,i} : [\boldsymbol{v}_{s,1}, \boldsymbol{v}_{s,2}, \ldots, \boldsymbol{v}_{s,k_{s,i}}]$ is the average speed of vehicle s from the first path segment to the last connection.

Based on the inputs, namely allocated tasks, dynamic environment changes and the location of static and moving obstacles, the simultaneous path and motion planning method finds the path $P_{s,i}$ and speed $V_{s,i}$ for all vehicles by minimizing the total travel time for a set of tasks. This approach is further explained using an example, which is given below.

Assume nodes ***i, j*** and ***k*** are nodes connected serially from node ***i, j*** and then to ***k***, and the weight for each connection, for example, connection between nodes ***i*** and ***j***, is the travel time for a vehicle at the given speed. While planning the motion (e.g. a path and speed) of a vehicle to undertake a task, the SiPaMoP algorithm will change the connection weight (travel time) between nodes ***i*** and ***j*** in the way of:

$$w^{d}_{i,j} = w_{i,j} + y \tag{3.3}$$

where $\boldsymbol{w_{i,j}}$ is the current connection weight which is equivalent to the travel time between the two nodes; $\boldsymbol{w^{d}_{i,j}}$ is the new connection weight between nodes ***i*** and ***j*****,y** is parameter which is determined as;

$$y = \begin{cases} 0 & \textit{connection between nodes } i \textit{ and } j \textit{ is free} \\ T + \Delta T & \textit{otherwise} \end{cases} \tag{3.4}$$

Where, ***T*** is the time difference between two relevant vehicles when they approach the same node ***j***. $\boldsymbol{\Delta T}$ is the minimal time difference between two vehicles according to safety requirements.

Vehicle $\mathbf{V_1}$ is travelling from node ***d*** to ***a*** via intermediate nodes ***c, j*** and ***b***, for example, as shown in Figure 3-3. The SiPaMoP algorithm ~~i~~s planning vehicle $\mathbf{V_2}$'s path from node ***h*** to node ***l***, and finds that $\mathbf{V_1}$ and $\mathbf{V_2}$ will pass the same node **j** within $\pm\boldsymbol{\Delta T}$. So the algorithm will automatically change the connection weight between node ***i*** and node ***j*** by letting $\boldsymbol{T = t_{cj}}$ and $\boldsymbol{w^{d}_{i,j} = w_{i,j} + t_{cj} + \Delta T}$.Because of this weight increase between nodes ***i*** and ***j***, $\mathbf{V_2}$ will either change its path, for example, travel from ***h*** to ***l*** via ***i, b*** **(or *c*)** and **k**, or reduces its travel speed from ***i*** to ***j*** to allow $\mathbf{V_1}$ to pass node ***j*** safely. This change in connection weight will automatically be removed after $\mathbf{V_2}$ leaves the node ***j*** towards node ***k***, which allows other vehicles pass the connection without change in speed if there is no collision in this connection.

This dynamic change of connection weight between nodes ***i*** and ***j*** can also solve the collision problem in the case that $\mathbf{V_1}$'s path is from node ***d*** to ***l*** via ***c, j*** and ***k***, and $\mathbf{V_2}$ travels from node ***h*** to ***l*.** In this case, $\mathbf{V_2}$ will pass node ***j*** after $\mathbf{V_1}$ with a minimum time difference of $\boldsymbol{\Delta T}$.

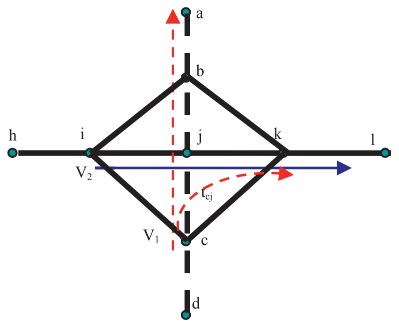


Figure 3-3: The connections of nodes for the given example

The SiPaMoP algorithm is further illustrated as a flow chart in Figure 3-4. It works as follows. Initially, a map is labelled with different node numbers and a database is maintained which includes nodes and available connections. Then a task pick-up node is set as labelled. Next step is to select the adjacent node and calculate the arrival time. This is repeated for each adjacent node. If a labelled or adjacent node is occupied at a point of time, then the weight of the labelled and adjacent nodes are updated. This is repeated till all the adjacent nodes are scanned. Once this is finished, shortest distance node is labelled. This is repeated until all nodes are labelled. Once all the nodes are labelled, shortest path is registered with respect to travelling time.In SiPaMoP algorithm, Dijkstra algorithm (Dijkstra 1959) is used to search the shortest path by considering the updated node weights. Other algorithms such A* can be used in the SiPaMoP algorithm as well. The path and motion coordination for the autonomous vehicles are performed in a sequential manner in this approach. Therefore, once a path is decided for a vehicle, the planned paths and connection weights will be considered in the planning of the next path.

By changing the connection weight, the SiPaMoP algorithm is able to avoid collisions by coordinating the motions of a coming vehicle with the previously planned vehicles in following ways,

Method 1: Waiting until other vehicles pass the connection
In this method, a vehicle will be asked to wait until the other vehicles pass the connection where collision may occur. (Example: between nodes ***j*** and ***k***of Figure 3-3). This happens normally in bottleneck areas.

Method 2: Changing the path of the incoming vehicle
This method involves replacing a path planned without weight change by a better path that takes less travel time, compared to the path with changed weight on current environment and traffic conditions.

Method 3: Changing the vehicle speed

In this method, for example, from node ***i*** to node ***j***, the same path is kept as planned but travelling speed is changed. As we assume the paths planned previously have higher priority than the current path being planned, and vehicles are running with the set speed when there is no collision on their path, the change in weight, i.e. increase in weight, will slow down the coming vehicle(s).

Method 4: Combinations of the above three methods

This is useful when more than two vehicles are involved in traffic congestion. In order to avoid possible collisions, a combination of the above methods is adopted here. Clearly, this weight change does not affect previously planned paths and vehicle speeds. Nevertheless, it will affect the paths and speeds yet to be planned. The results to be obtained from the SiPaMoP algorithm is the paths and speeds of all vehicles $\mathbf{V_{s,i}}$ for their assigned tasks $\mathbf{P_{s,i}}$, where $\mathbf{s=1,2,\ldots, m}$ and $\mathbf{i=1,2,\ldots, n_s}$.

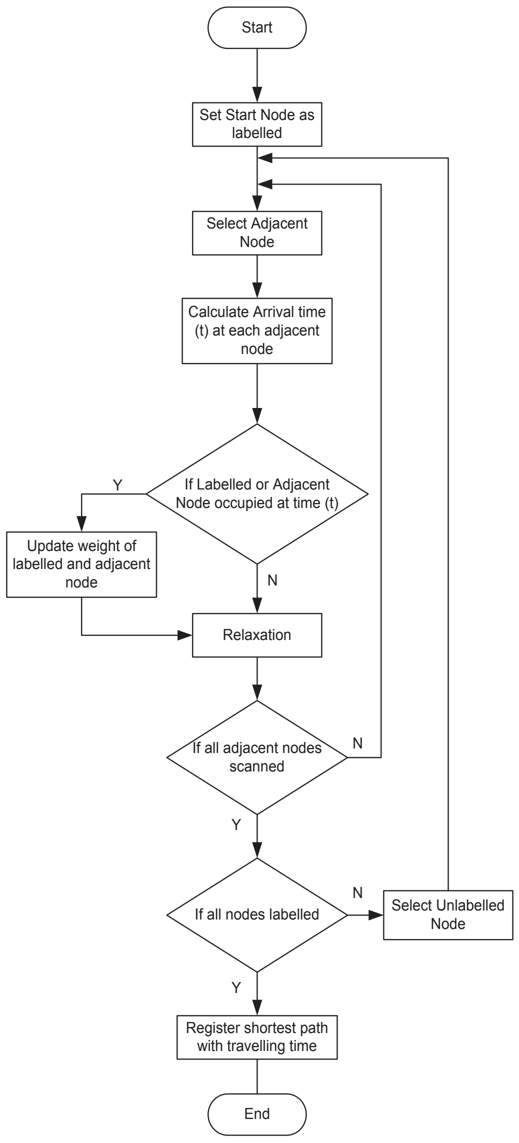

Start
Set Start Node as labelled
Select Adjacent Node
Calculate Arrival time (t) at each adjacent node
If Labelled or Adjacent Node occupied at time (t)
Y
Update weight of labelled and adjacent node
N
Relaxation
If all adjacent nodes scanned
N
Y
If all nodes labelled
N
Select Unlabelled Node
Y
Register shortest path with travelling time
End

Figure 3-4: The flowchart of the SiPaMoP algorithm

3.4. Simulation Environment

An indoor environment, which consists of a number of restricted areas for movement of vehicles, is used for verification of the SiPaMoP algorithm. Figure 3-5 shows the environment in which a fleet of autonomous vehicles are designed to deliver and collect materials from and to any place in the environment. This environment is represented by a network map (Figure 3-6) which represents all geometric relationships. The crosses (x) in the map represent the nodes that the vehicles can reach, and the lines are the connections among nodes and the paths the vehicles have to follow. Each autonomous vehicle is supposed to move from node to node following the connections between nodes. Each connection facilitates bi-directional movement for the vehicles.

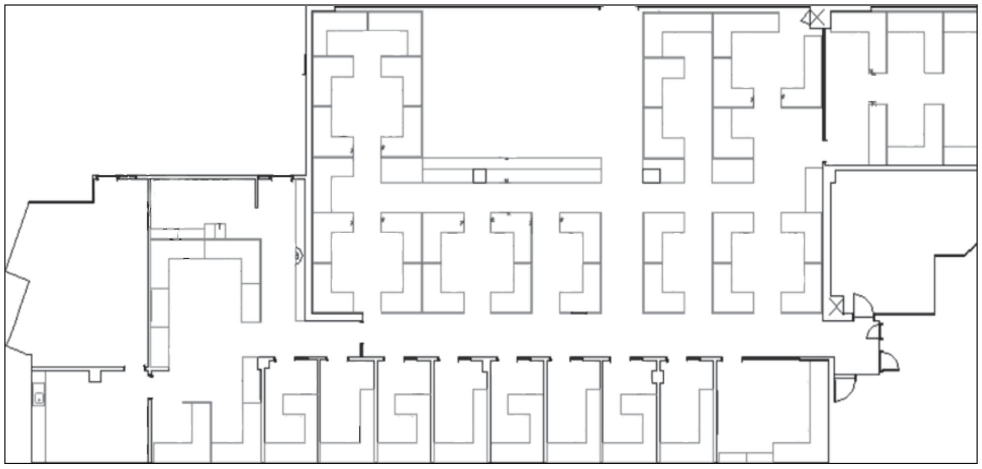


Figure 3-5: The plan view of the simulation environment

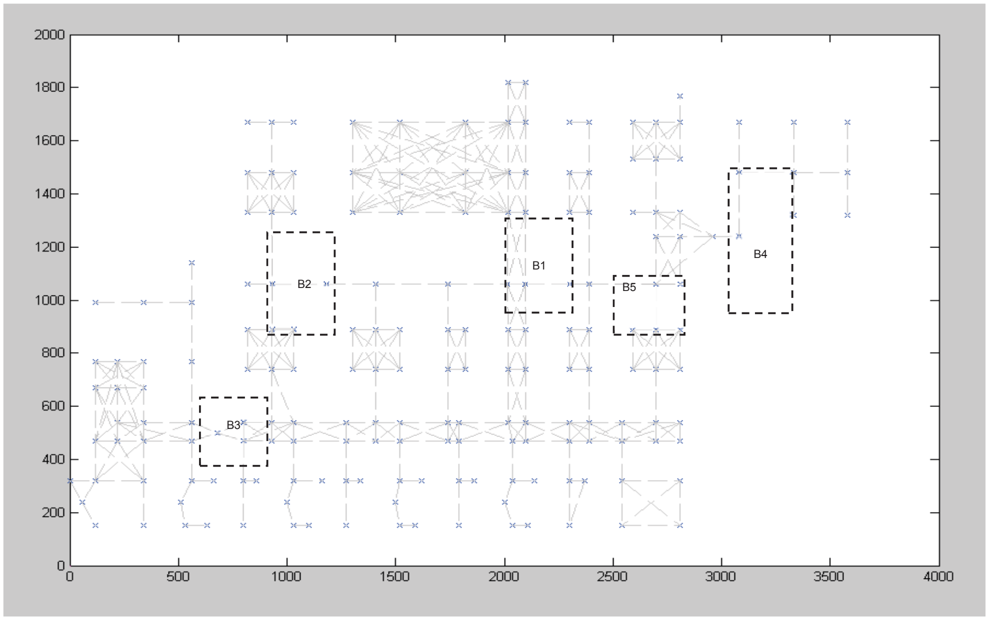


Figure 3-6: The network map of the environment shown in Figure 3-5

It can be seen from the map that there are many bottleneck areas. B1, B2, B3, B4 and B5 are five examples of bottleneck areas (Figure 3-6). Traffic congestion is unavoidable in these areas. This situation becomes worse when more autonomous vehicles are employed. How to manage these areas efficiently becomes a crucial issue in path and motion planning for a fleet of vehicles. Commonly adopted method to solve this problem is waiting and passing based on the allocated priority. Thus, vehicles with lower priority have to give way to vehicles with higher priority. For vehicles with the same priority level, the vehicle with the longest travel path will be given higher priority.

3.5. Simulation Studies

The performance of the SiPaMoP algorithm was tested in two stages. At the first stage, the main focus was to show the collision avoidance capabilities of the SiPaMoP algorithm. This has been presented with the assistance of examples, which show possible collisions in the simulation environment (Liu, Wu and Kulatunga 2006). In the second stage of the simulation study, the efficient route/path selection capabilities were presented with comparisons to the loop-based path topology and conventional bi-directional path topology with the SiPaMoP algorithm.

3.5.1. Collision Avoidance Capability

As stated before, the SiPaMoP algorithm is able to coordinate the motions of a coming vehicle with the previously planned vehicles by either changing the coming vehicle's path, speed of the vehicle, or by waiting, until other vehicle passes the respective connection where a collision may occur. These instances are elaborated in the following examples for explanation purposes.

Example 1: **One vehicle waits until the other one passes safel**y (Method-1)

In this example, vehicle V1 travels from node 50 to node 18 (Red), and vehicle V2 from node 35 to node 48 (Black). The paths for the two vehicles are first obtained by applying the Dijkstra algorithm: the path of V1 starts from node 50 to node 18 via node 34 and V2's path starts from node 35 to 48 via nodes 34, 33, 54 and 49. Those two vehicles collide at node 34 (Figure 3-7 and Table 3-1). By applying the SiPaMoP algorithm, vehicle V1 does not change its path and travel speed (Figure 3-8), but V2 delays its travel by 4.2 simulation time units (stu) in order to avoid collision with V1. With this extra waiting time, V2's travel time increases from 20.7 stu to 24.9 stu(Table 3-1). In this case, waiting at V2's pick-up node is better than decreasing V2's speed because of the requirement of safety distance between the two vehicles.

Table 3-1: Path selection of example 1(One vehicle waits till the other one passes away the connection)

Vehicle 1								Vehicle 2							
without collision avoidance				From SiPaMoP				without collision avoidance				From SiPaMoP			
Pick-up node	End node	Node time	Total time	Pick-up	End node	Node time	Total time	Pick-up node	End node	Node time	Total time	Pick-up	End node	Node time	Total time
50	34	3.2	3.2	50	34	3.2	3.2	35	34	5.0	5.0	**35**	**34**	**8.8**	**9.2**
34	18	3.8	**7.0**	34	18	3.8	**7.0**	34	33	5.0	10.0	34	33	5.0	14.2
								33	54	4.8	14.8	33	54	4.8	19.0
								54	49	2.4	17.2	54	49	2.4	21.4
								49	48	3.5	**20.7**	49	48	3.5	**24.9**

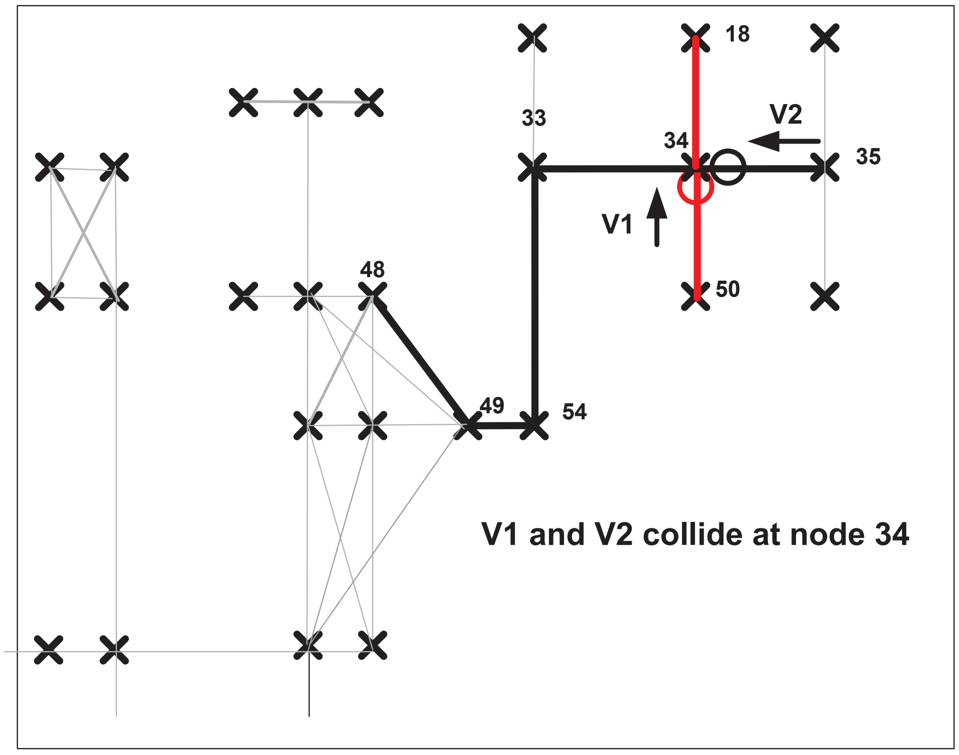


Figure 3-7: V1's and V2's Paths obtained without considering collisions by Dijkstra algorithm (Example 1)

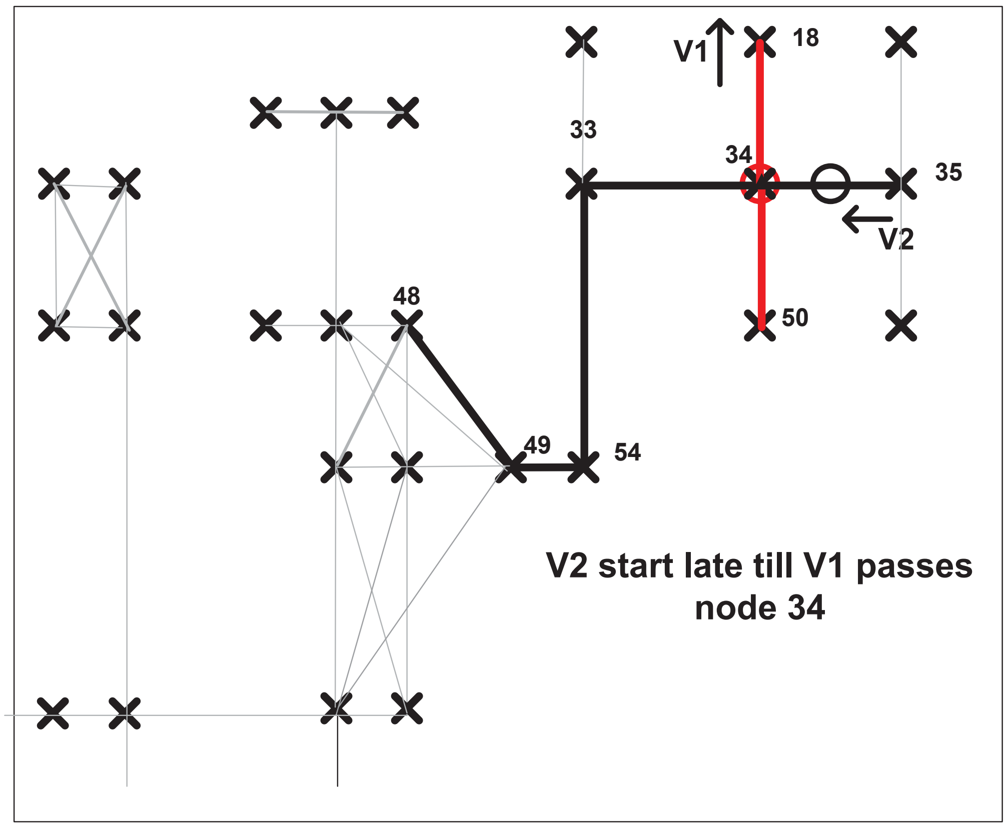


Figure 3-8: V1's and V2's Paths obtained by the SiPaMoP algorithm: V2 wait till V1 passes node 34 to avoid collisions (Example 1)

Example 2: **Two vehicles collide: one changes its path.**

In this case, vehicle V1 travels from node 3 to node 89 (Red), and vehicle V2 from node 112 to node 38 (Black). Similar to example 1, the paths for the two vehicles are first planned by applying shortest path search mechanism: the Dijkstra algorithm. At this stage, potential collisions are not considered when paths are planned. As shown in Figure 3-9 and Figure 3-10, the path of V1 starts from node 3 to node 89 via nodes 4, 21, 37, 57 and 71, and the path of V2 starts from node 112 to 38 via nodes 90, 71, 57 and 37. Those two vehicles collide between nodes 37 and 57. When the proposed SiPaMoP algorithm is used, vehicle V1 does not change its original path and travel speed when its path is planned using SiPaMoP algorithm. There is no other vehicle occupying the path segments V1 intends to use (Figure 3-10:).However, when V2 plans its path (that happens after V1 finalized its path), it has to avoid the collision between nodes 112 and 57 (Figure 3-10). Therefore, V2 changes its path without changing its speed. Due to the change of path, V2 travels a longer path than the shortest path obtained from the Dijkstra algorithm in order to avoid a collision with V1. As a result, V2's travel time increases from 18.3 (stu) to 35 (stu) (Table 3-2). This change in path in this case is better than a speed change because the two vehicles collide in a bottleneck area and V2 needs to wait at least 18 (stu) (the travel time of V1 from node 37 to node 71 plus the safety distance between the two vehicles).

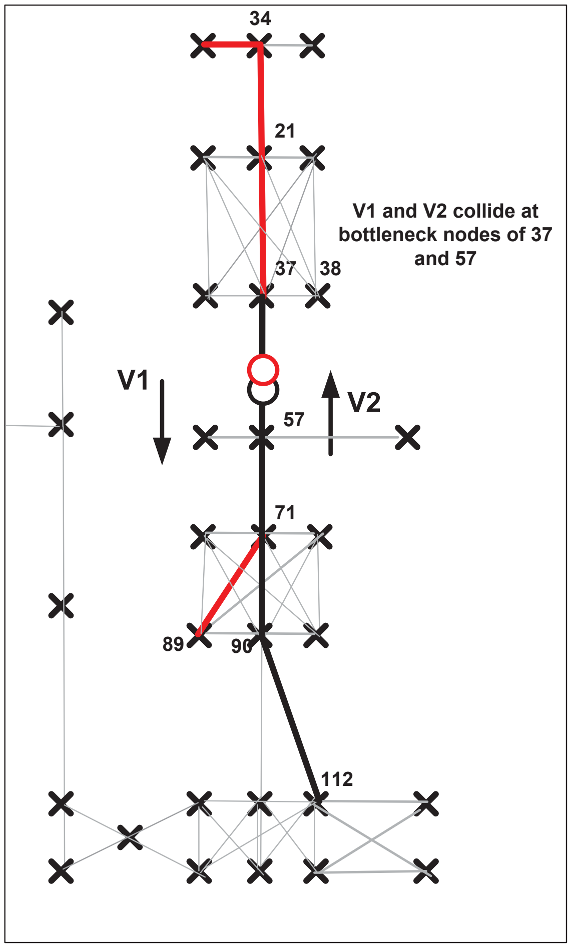


Figure 3-9: V1's and V2's Paths obtained without considering collisions by Dijkstra algorithm (Example 2)

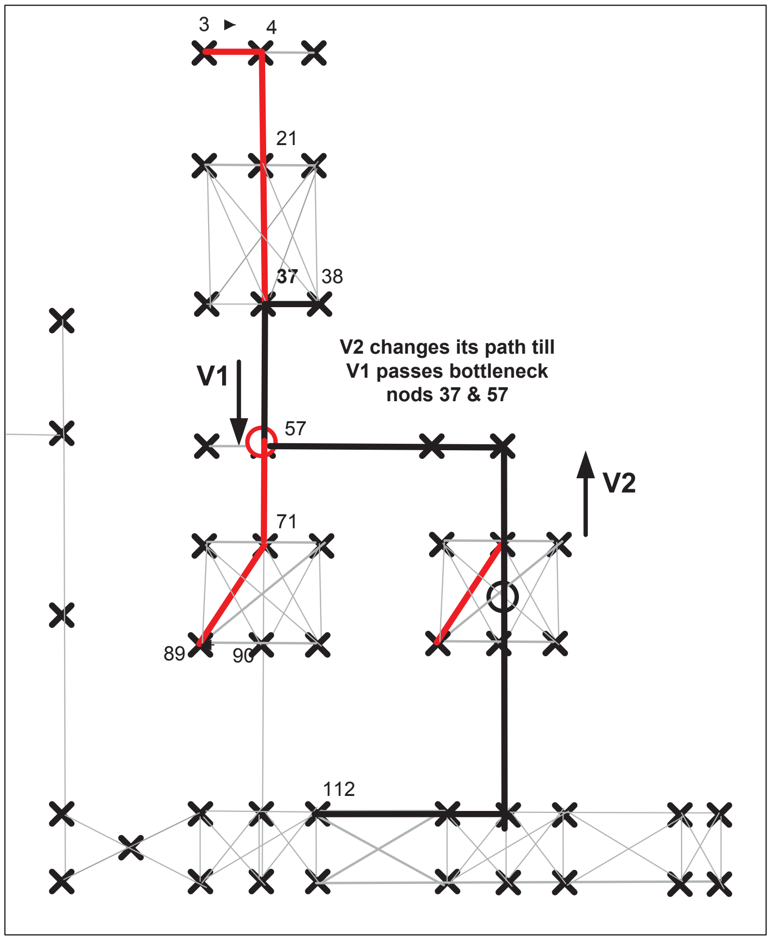


Figure 3-10: V1's and V2's Paths obtained by the SiPaMoP algorithm: By changing V2's path to avoid collisions between nodes 37 and 57 (Example 2)

Table 3-2: Path selection of the vehicles in example 2 (V2 changes its path to avoid collision)

Vehicle 1								Vehicle 2							
Without collision avoidance				From SiPaMoP				Without collision avoidance				From SiPaMoP			
Pick-up node	End node	Node time	Total time	Pick-up node	End node	Node time	Total time	Pick-up node	End node	Node time	Total time	Pick-up	End node	Node time	Total time
3	4	2.2	2.2	3	4	2.2	2.2	112	90	4.5	4.5	112	**113**	4.8	4.8
4	21	3.8	6.0	4	21	3.8	6.0	90	71	3.0	7.5	113	**114**	2.8	7.6
21	37	3.0	9.0	21	37	3.0	9.0	71	57	3.4	11.0	114	**93**	4.0	11.6
37	57	5.4	14.4	37	57	5.4	14.4	57	37	5.4	16.3	93	**74**	3.0	14.6
57	71	3.4	17.8	57	71	3.4	17.8	37	38	2.0	**18.3**	74	**59**	3.4	18.0
71	89	3.7	**21.6**	71	89	3.7	**21.6**					59	**58**	4.6	22.6
												58	57	5.0	27.6
												57	37	5.4	33.0
												37	38	2.0	**35.0**

Example 3 :(Method 3)-This example shows how multiple vehicles select their own paths without interfering with others and work simultaneously. In this example, there are 13 tasks allocated to four vehicles, with vehicle V1 allocated four tasks and each of the other three vehicles allocated three tasks (Table 3-3). The vehicles' current positions and the tasks' pick-up and drop-off positions are also listed in Table 3-3. The first four tasks start from time zero, all other tasks start immediately after their preceding tasks.

The vehicles' paths obtained by the Dijkstra algorithm without collision avoidance and by the SiPaMoP algorithm are presented in Figure 3-11 and Figure 3-12, respectively. The SiPaMoP algorithm avoided collisions in three ways (i.e. changes in path, travel speed and waiting) and makes all the collisions automatically in the planning stage depending on which one can reduce the completion time of all tasks.

Those paths obtained by Dijkstra algorithm are not collision free, but the makespan obtained for completing all the tasks is 151.8 (stu) which should be the ideal target for the SiPaMoP algorithm. The closer the makespan obtained by the SiPaMoP algorithm to the target time, the better the SiPaMoP performance. The makespan obtained by the SiPaMoP algorithm is 158.6 (stu) which is very close to the target value of 151.8 (stu).

Table 3-3: Task allocation information to four vehicles of example 3

Task No	Vehicle allocated	Vehicle current position	Task pick-up node	Task drop-off node	Pick-up time (stu)
1	V1	176	176	56	0
2	V2	3	3	172	0
3	V3	135	135	25	0
4	V4	143	143	74	0
5	V1	56	56	174	19.18
6	V2	172	172	20	60.20
7	V3	25	25	53	21.61
8	V4	74	74	18	36.59
9	V1	174	174	142	45.91
10	V2	20	20	68	94.72
11	V3	53	53	170	48.46
12	V4	18	18	20	84.71
13	V1	142	142	2	92.39

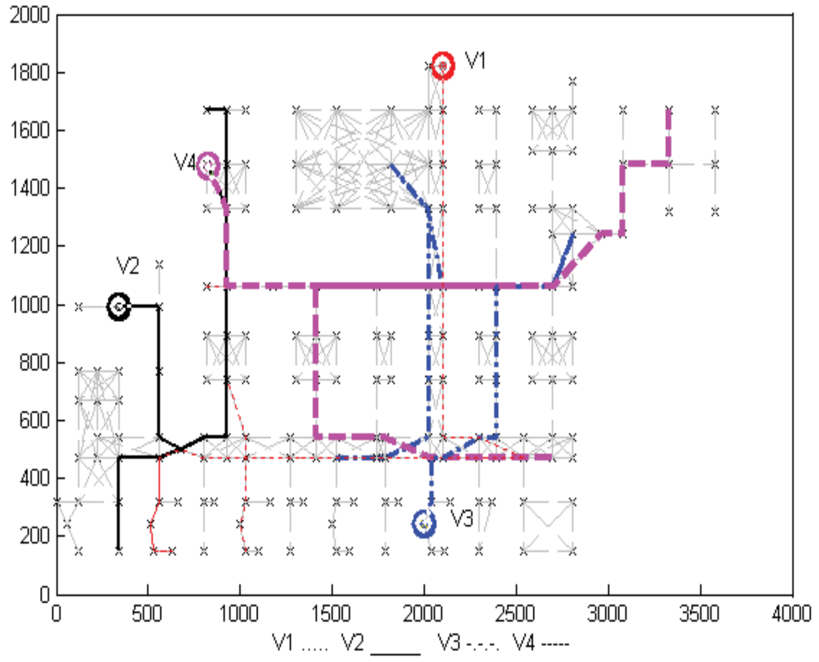


Figure 3-11: Paths of all vehicles obtained by Dijkstra algorithm without considering collisions (Example 3)

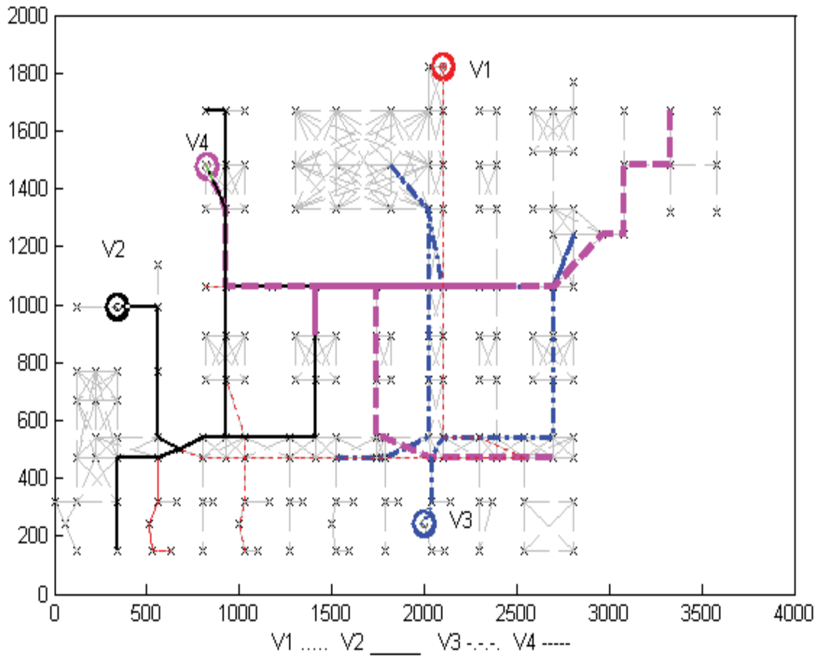


Figure 3-12: Paths of all vehicles obtained by SiPaMoP algorithm by considering collisions (Example 3)

3.5.2. Efficient Motion Coordination Capability

A number of different traffic handing and collision avoidance strategies have been used by researchers and practitioners in different path topologies in material handling systems in order to avoid collisions. Zone-based systems and looping or cyclic systems have been used due to their simplicity. However, they are not always applicable to some path topologies. Therefore, the second stage of simulation is designed to investigate the overall efficiency of the SiPaMoP algorithm for bi-directional path topology against the loop-based path topology. This is further explained using an example. Four tasks and two vehicles are used in this simulation. The task and vehicle information of the simulation example is given in Table 3-4 and Table 3-5 along with task completion times and planned paths to accomplish those tasks in the loop-based approach and the SiPaMoP algorithm, respectively.

Table 3-4: Task allocation information and path details in loop based path topology

Task	Allocated Vehicle	Pick-up node	End node	Task finish Time (stu)	Selected Path in loop based topology
1	1	110	96	13.20	110-116-96
2	2	130	60	16.96	130-137-60
3	1	111	77	47.27	96-76-60-57-111-77
4	2	131	96	46.90	60-57-90-111-131-96

Furthermore, Figure 3-13 to Figure 3-16 illustrates the different travelling segments of both vehicles to complete the allocated tasks in a loop-based path topological scenario. In the first segment of the loop, vehicle V1 starts from the task 1's pickup location of node 110 to its drop off location situated at node 96. Conversely, vehicle V2 starts its journey from its first allocated task's pickup location of node 130 to drop off location of node 60 in the first segment of the loop. Figure 3-13 shows these movements: from 110 – 117 of V 1 and 130 – 137 of V 2 on their way to complete their first tasks assigned to them. Figure 3-14 shows the loop segments from node 117 -96 and 137-60 of V1 and V2 respectively.

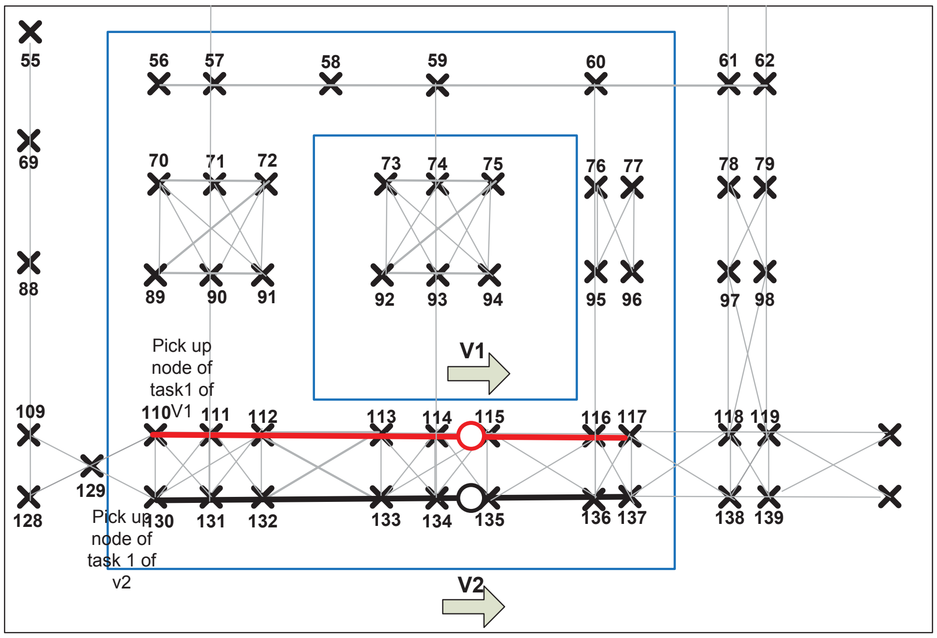


Figure 3-13: Vehicles V1 and V2 performing their first tasks (Path segments between node 110 - 117 of V1 and from nodes 130 –137 of V2 are shown here)

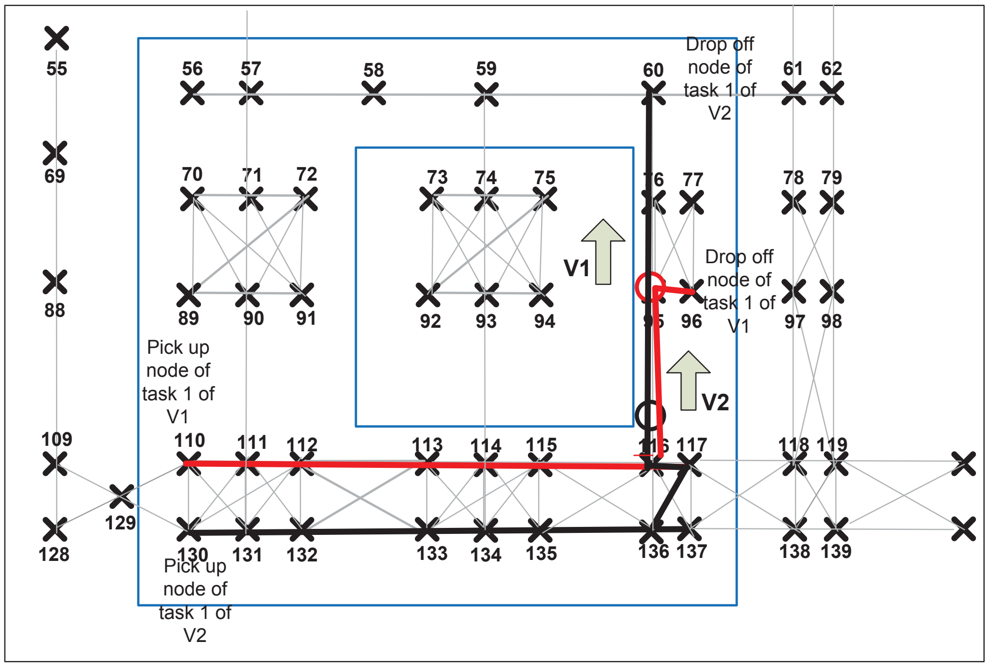


Figure 3-14: Vehicles V1 and V2 towards the completion of their task 1 (both use the same path segments from nodes 116 to node 95 and vehicle 2 travels behind the vehicle 1)

Figure 3-15 shows the V1 and V2's movements along the loop from their first tasks drop off nodes (nodes 96 and 60 respectively) to second tasks pick up nodes (node 111 and 131 respectively).

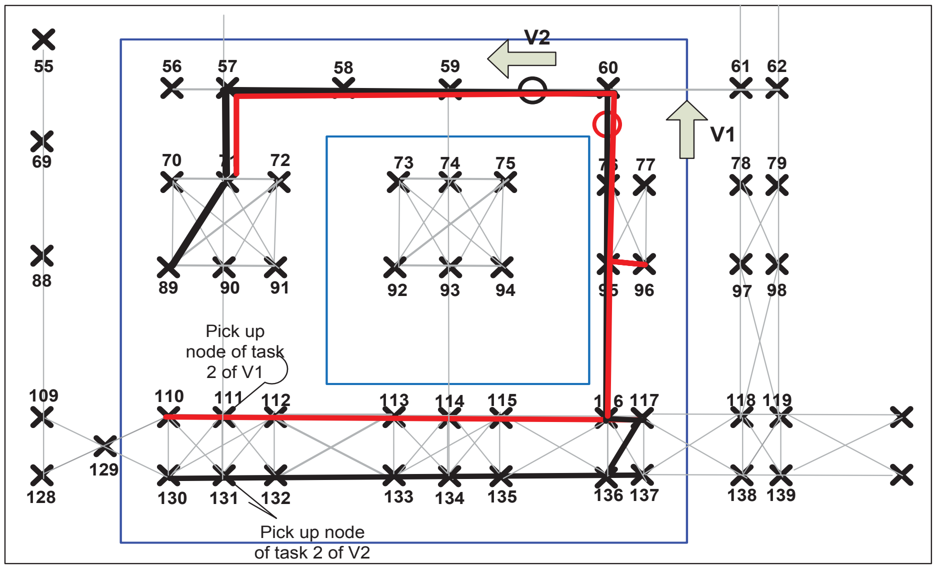


Figure 3-15: Vehicles V1 and V2 travelling to pick-up their 2nd tasks (at node 111 and 130 respectively) in a loop path topology

Figure 3-16 shows the last loop segment of vehicles V1 and V2 travelling from 71 to 111 of V1 and 89 to 131 of V2 to pick up their second tasks from nodes 111 and 131. Since the vehicles are travelling in a loop, they cannot design the shortest path out of the all connections. This leads to extra travel time to undertake tasks. However, the potential collision probability reduces in the loop based systems since it facilitates uni-directional movement only. Between nodes, 71 and 90 there is a potential collision. This is avoided by selecting an alternative route to reach node 90 by the vehicle 2 (node 71-89 90). The potential collision region is highlighted in the Figure 3-16.

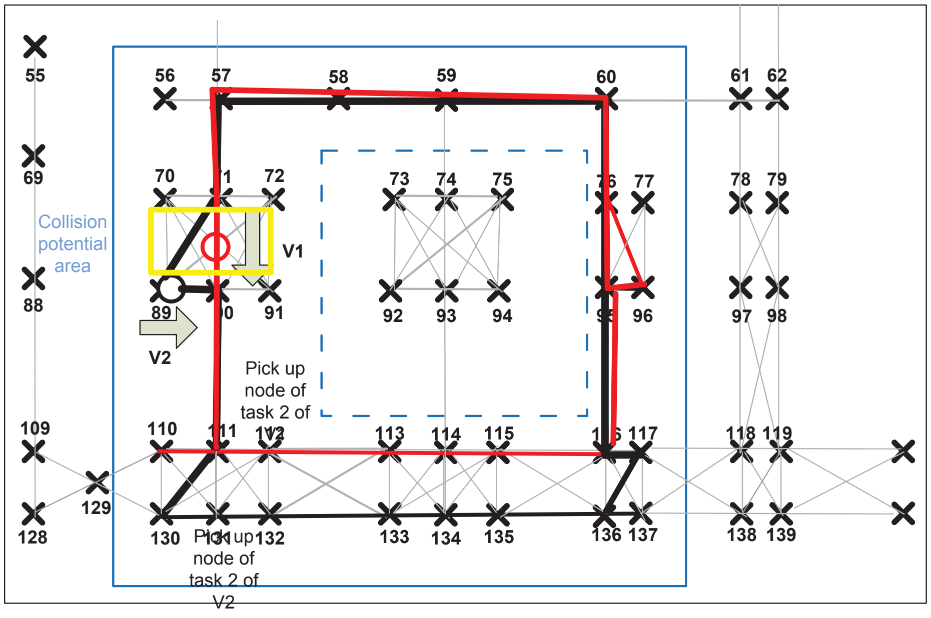


Figure 3-16: Vehicles V1 and V2 travelling to pick-up their 2nd tasks (Between loop segment 71 - 111 and 89 -131 respectively)

Table 3-5 presents the details of the simulation carried out to show the path selection done by the SiPaMoP algorithm in conventional path topology. It shows the task allocation information, pick up and drop off nodes of each task, task completion time and the planned path to complete each tasks.

Table 3-5: Vehicles task allocation information and path details in conventional path topology

Task	Allocated Vehicle	Pick-up node	Drop off node	Task completed time (stu)	The plannedpathin conventional topology
1	1	110	96	12.20	110-116-95-96
2	2	130	60	15.30	130-135-115-95-60
3	1	111	77	35.10	96-116-111-77
4	2	131	96	38.90	60-59-74-93-113-131-113-96

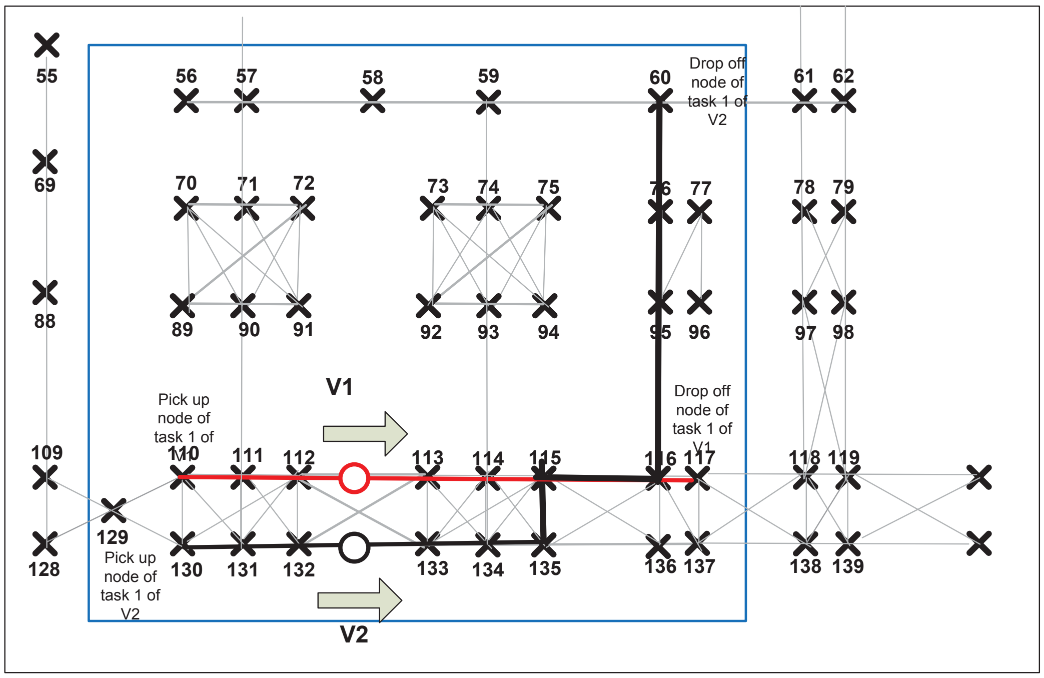


Figure 3-17: Vehicle 1 (from node 110 to 117)and Vehicle 2 (from node 130 to 60)perform their first tasks by following the paths planned by the SiPaMoP algorithm

Figure 3-17 to Figure 3-21 shows how two vehicles selected their paths to carry out four tasks in conventional path topology where the SiPaMoP algorithm is used for the path planning and collision avoidance. Figure 3-18 shows the segment of two vehicles' movements to fulfil their first tasks. Vehicle V1 picks up its first task from 111 and carried it up to the drop off node at 96. Conversely, vehicle 2 picks-up its first task from node 130 and carries that to the drop off node of 60. Vehicle 1 and vehicle 2 move in a parallel path up to nodes 114 and 135 and then vehicle 2 follows the same path as vehicle 1 does up to node 95. This can be seen in Figure 3-18, and the path segments are also shown in red and black separately for vehicle 1 and vehicle 2.

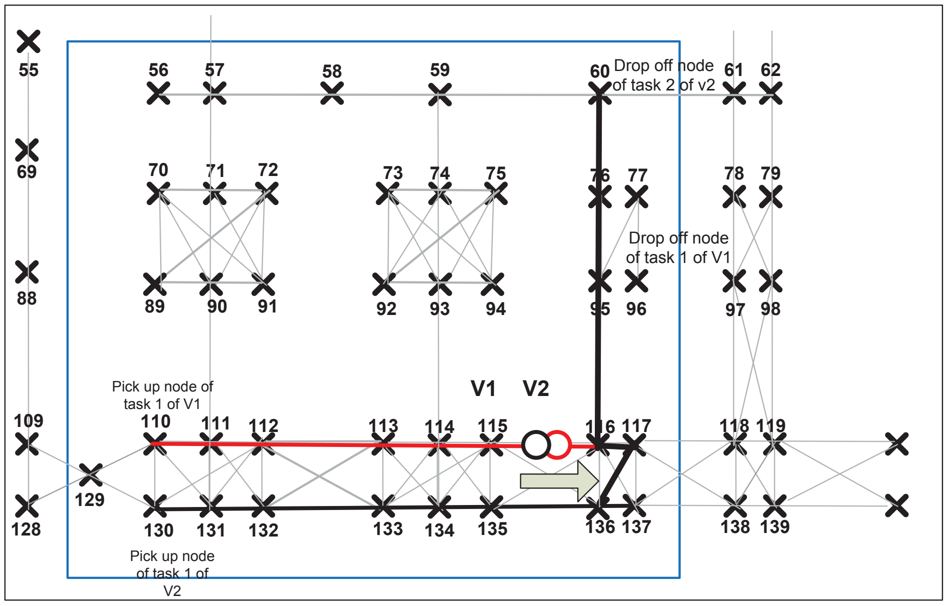


Figure 3-18: Vehicle 1 (from node 115 to node 96) and vehicle 2 (from node 135 to node 60) perform their initial tasks planned by the SiPaMoP algorithm

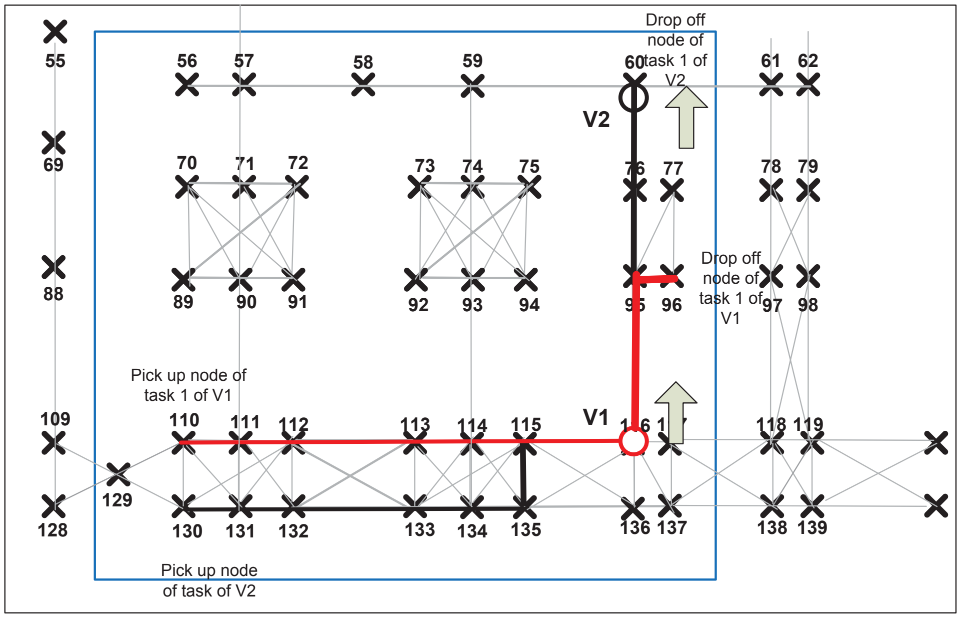


Figure 3-19: Vehicle 1 returns to its 2nd task's origin while vehicle 2 is reaching its initial task's drop-off node (from the SiPaMoP algorithm)

In Figure 3-20, vehicle 1 moves back from the same route it used to travel to the first task's drop-off node 96 to collect its second task from node 111. While vehicle 2 moves from node 95 to node 60, vehicle 1 has travelled from node 96 to node 116.

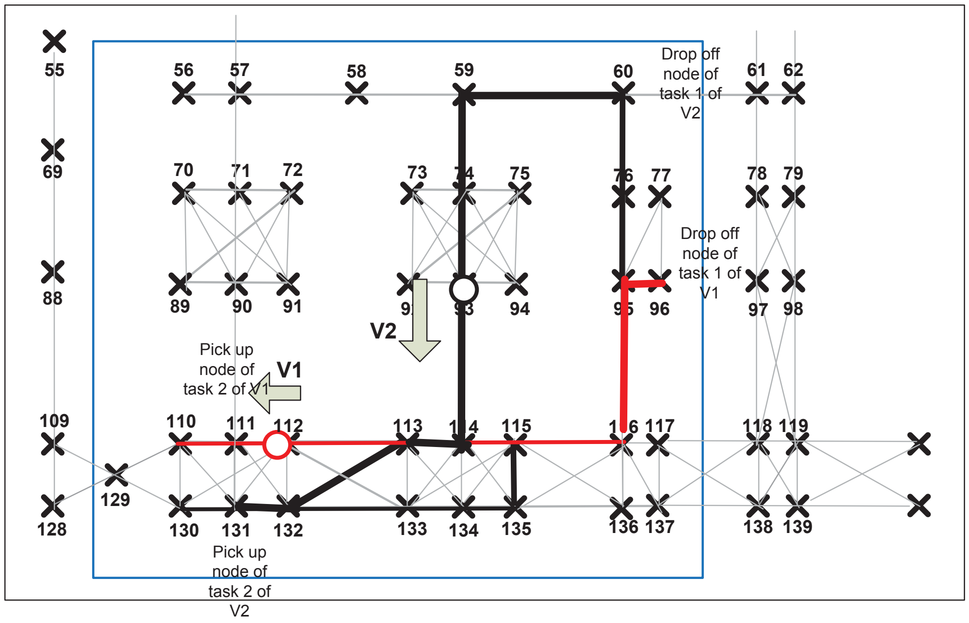


Figure 3-20: Vehicle 1 travels towards its 2nd task'spick-upnode of 111 while vehicle 2 travels towards its 2nd task's pick-up node of 131

Figure 3-21 shows that vehicle 1 undertakes its 2nd task from node 111 towards node 77 while vehicle 2 is still returning to pick-up node 131 of its 2nd task. Vehicle 1 travels from node 111 to node 77, this time with a new set of nodes in order to avoid a possible head-on collision with vehicle 2 between nodes 113 to 114. Therefore, vehicle 1 takes node 133, 134, 135, then to 116, and from there to node 95 and 77.

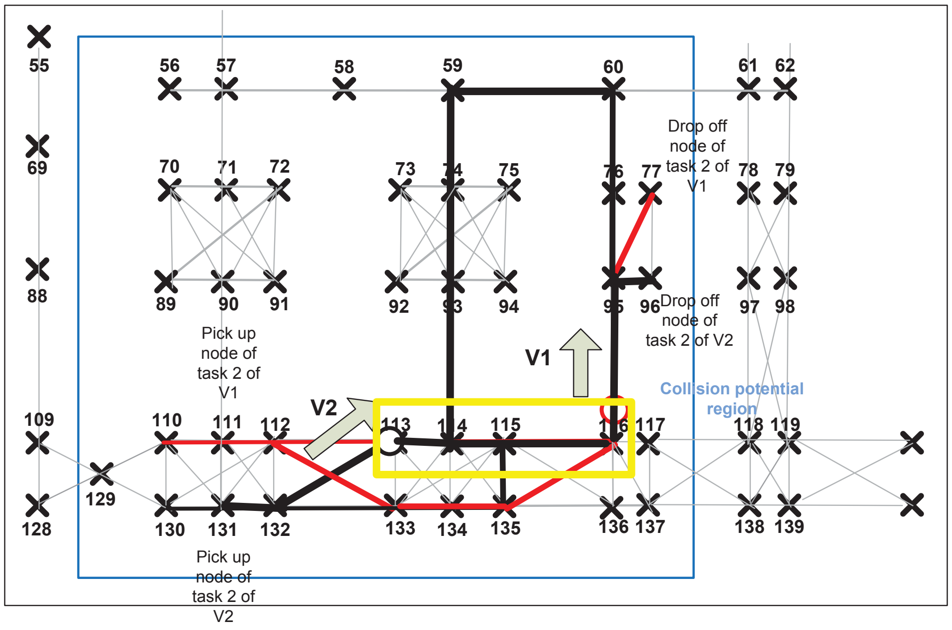

55
56
57
58
59
60
61
62
69
70
71
72
73
74
75
76
77
78
79
Drop off node of task 2 of V1
88
89
90
91
92
93
94
95
96
97
98
Drop off node of task 2 of V2
Pick up node of task 2 of V1
V1
V2
Collision potential region
109
110
111
112
113
114
115
116
117
118
119
129
128
130
131
132
133
134
135
136
137
138
139
Pick up node of task 2 of V2

Figure 3-21: Vehicles 1 and 2 perform their 2nd tasks by following the paths from the SiPaMoP algorithm

Table 3-6 presents the completion time and empty travel time of two approaches of each task separately. From the first task to the fourth task, the empty travel time was increased along with the task completion time. However, there is a significant difference with respect to the two measured parameters between the loop-based approach against the SiPaMoP algorithm. In all the four tasks the SiPaMoP algorithm has outperformed the Loop based approach in both aspects measured in the simulations.

Table 3-6: Completion and empty travel times obtained from the two approaches

Task #	Loop-based approach		SiPaMoP algorithm	
	Completion time (stu)	Empty travel time (stu)	Completion time (stu)	Empty travel time (stu)
1	13.20	4.51	12.20	4.78
2	16.96	7.31	15.30	6.22
3	47.27	12.09	35.10	10.67
4	46.90	14.35	38.90	12.99

As previously mentioned, many path-planning strategies have been used in all sorts of material handling systems. However, these have been divided into three categories: conventional, loop-based and zone based depending on the path topology.

Therefore, when validating the SiPaMoP algorithm, it was a necessity to investigate the capabilities of the proposed algorithm against other path planning strategies. However, since many material handling systems adopt loop-based systems due to its capability to avoid collisions among the vehicles easily, it was decided to investigate the proposed SiPaMoP algorithm against loop-based strategy for the same problem set-up. A simulation study was done to fulfil this requirement.

Of these two approaches, it can be seen that the loop-based approach requires more time to complete the tasks and it has taken longer empty travel time to reach the pick-up node of next task. However, in the SiPaMoP algorithm, task completion times are shorter and the empty travel times are less. This simple simulation demonstrates that the SiPaMoP algorithm is capable of selecting efficient paths while avoiding collisions among the vehicles.

3.6. Conclusion and Remarks

This chapter presented a novel path planning and motion coordination algorithm named SiPaMoP. The main features of the SiPaMoP algorithm are its capability to plan the shortest collision-free paths while coordinating the motion of a fleet of vehicles simultaneously. This approach can avoid potential collisions effectively. In order to select a shortest path, the Dijkstra algorithm is used. However, other shortest path search algorithms such as A* or D* can be easily adapted to the SiPaMoP algorithm. By changing the connection weight, the SiPaMoP algorithm avoids collisions by coordinating the motions of the vehicles with previously planned vehicles.

Simulation studies were carried out in this chapter, using different sets of tasks whose pick-up and drop-off nodes are generated randomly, and in different environments. The obtained simulation results demonstrate that the SiPaMoP algorithm can plan multi-autonomous vehicles' paths and speeds simultaneously, avoid potential collisions with static obstacles, other moving vehicles and stopped vehicles, and maximise the usage of bottleneck areas. It has been further revealed that in conventional bi-directional path topology the SiPaMoP algorithm performs very well. With the collision/dead lock avoidance capability embedded in the SiPaMoP algorithm, potential collisions can be avoided effectively, without locking all the planned paths.

There are limitations in this algorithm. The paths of different vehicles are planned in a sequential manner. Due to this reason, most critical connections might be occupied by previously planned vehicles, and then considerable waiting for new vehicles is to be expected. Furthermore, connections from pick-up to drop-off nodes of the paths are planned once, but if there is an unexpected event such as vehicle breakdown or path blockage, replanning is required. In order to overcome this issue, paths can be planned incrementally with frequent inter-communication between the vehicles.

Chapter 4

Simultaneous Task allocation and Motion Coordination - Static environment

4.1. Introduction

Vehicle/robot task allocation or dispatching refers to a strategy used to select a vehicle to perform a task. Usually, it is a continuous and dynamic process as the numbers of both vehicles and tasks could change continuously. Two ways can be used for task allocation: work-centre-initiated task allocation and vehicle-initiated task allocation (Egbelu and Tanchoco, 1984). In work-centre-initiated task allocation, an AGV is selected from a set of competing idle AGVs. Various rules, for example, the random vehicle rule, nearest vehicle rule and least utilised vehicle rule, can be employed for assigning tasks to AGVs. Some of the task allocation policies in the vehicle-initiated approach include the shortest travel time/distance rule, the maximum outgoing queue size rule, and the modified first-come-first served rule (Srinivasan et al., 1994, Yamashita, 2001). Evaluation on the performance of those rules and policies has been conducted. For example, De Koster et al. (2004) evaluated the performance of several real-time vehicle dispatching rules in three different environments, namely a European distribution centre, a container terminal and a production site. Bish et al. (2001) investigated the problem of dispatching vehicles to containers in combination with the assignment of containers to locations in the storage area.

Various algorithms have been proposed and studied for task allocation. They vary from simple dispatching rules, local search algorithms, heuristics and exact methods. Branch and bound or dynamic programming algorithms are often used to find solutions to them with the help of problem specific information to reduce search space. However, this is valid only for specific types of problems and cannot always be adopted. When the problem size increases, it is difficult to get an optimal solution since computational time increases exponentially (Bagchi, 1999). In contrast, if simple dispatch rules are used, it is difficult to find good quality solutions (optimal or near optimal solutions). This difficulty has led to the development of greedy search algorithms.

Many variations of local search algorithms for solving NP-hard problems have been proposed and investigated. Since the quality of solutions obtained by local search algorithms strongly depends on initial conditions, these local algorithms have the potential to perform poorly for some sets of initial conditions. Heuristic approaches, which involve trial and error and some contemplated intuition, can produce an approximate solution to a task allocation problem. Although heuristic approaches normally cannot find optimal solutions, they are capable of finding near-optimal solutions within a reasonable time. Therefore, it is necessary to make a trade-off decision on expected solution quality and computational time.

As discussed in Chapter 2, there are many examples in the literature on task allocation of multi-vehicles. Examples include a dynamic deployment algorithm for dispatching AGVs to a container in order to minimise the loading and unloading time for a vessel (Leong, 2001), an Auction algorithm for dispatching AGVs (Lim et al., 2003a); a dynamic model for real-time optimization of the flow of flatcars (Powell et.al., 1998) and a mixed integer linear programming model to dispatch multi-load AGVs (Grunow et al., 2004). Other heuristic and computational algorithms studied for application in vehicle task allocation include Markov decision processes, fuzzy logic and neural network approaches. For real life applications with a large number of vehicles, more research into advanced heuristics and optimisation approaches is required (Vis, 2006).

A few research studies have been conducted in respect of integrated task allocation, path/motion planning and collision avoidance. The container-handling method investigated by Meersmans and Wagelmans (2001) combines the task allocation process with path planning. A heuristic search based Beam Search Algorithm is applied for TA. Collisions among vehicles are avoided by a loop (uni-directional) based path topology. This method results in more empty travels and waiting time, and low efficiency. Bish et al. (2005) modelled a transportation system for container terminals, which makes the assumptions of constant vehicle velocity and uni-directional vehicle movement. Congestion is not taken into account in this research. An algorithm proposed by Koo et al. (2004) does fleet sizing and vehicle routing for container transportation based on the Tabu Search algorithm in a static environment. This algorithm initially starts with a lower bound of fleet size and increases it until the makespan criteria are satisfied. The vehicle travel time between each segment is fixed and vehicle speed is assumed to be constant. The methodologies presented by Qiu et al. (2001) gives collision- free routing for AGVs in a bi-directional path layout. The path topology is created to suit a particular application. The vehicle speed is also assumed to be constant in this work. Simultaneous machine and AGV scheduling in a flexible manufacturing system have been introduced by Ulusoy et. al.,(1997) in order to minimize the makespan. However, this research has not taken routing into consideration.

Autonomous vehicle path planning in known environments such as container terminals has attempted to provide more realistic solutions over the last decade. However, all the research works done so far deal with the three stages of task allocation, path planning and scheduling separately which results in low efficiency. It is expected that integration of these three stages would be able to increase the efficiency of planning and scheduling, and productivity.

4.2. Simultaneous Task Allocation and Motion Coordination

In order to overcome the issues identified in Chapter 2 and in the Section 4.1, a STAMC approach is presented. This approach performs task allocation and motion coordination simultaneously, to generate efficient schedule and collision free paths for a large fleet of autonomous vehicles in strictly constrained environments such as fully automated container terminals. It can efficiently manage congestion and bottleneck areas, avoid collisions among and between vehicles and handle dynamic changes in high traffic movements.

Figure 4-1 shows the schematic representation of the STAMC approach. It shows that vehicle path planning and collision avoidance are taken into consideration in task allocation. This simultaneous approach is totally different from the traditional sequential approach shown in Figure 4-2, which deals with the task allocation, vehicle path planning and collision avoidance separately at different stages.

The traditional sequential approach is capable of finding solutions for the three separated sub-problems (task allocation, path planning and collision avoidance). However, the overall solution quality suffers due to separation of the three sub-problems. In contrast, the simultaneous approach takes into account all the three sub-problem simultaneously when finding solutions to the task allocation and motion coordination problem. Hence, the solution quality is significantly improved. As shown in Figure 4-1, collision avoidance and path planning methodologies are embedded in the task allocation process. That means, when STAMC approach allocates tasks, it considers path planning and collision avoidance.

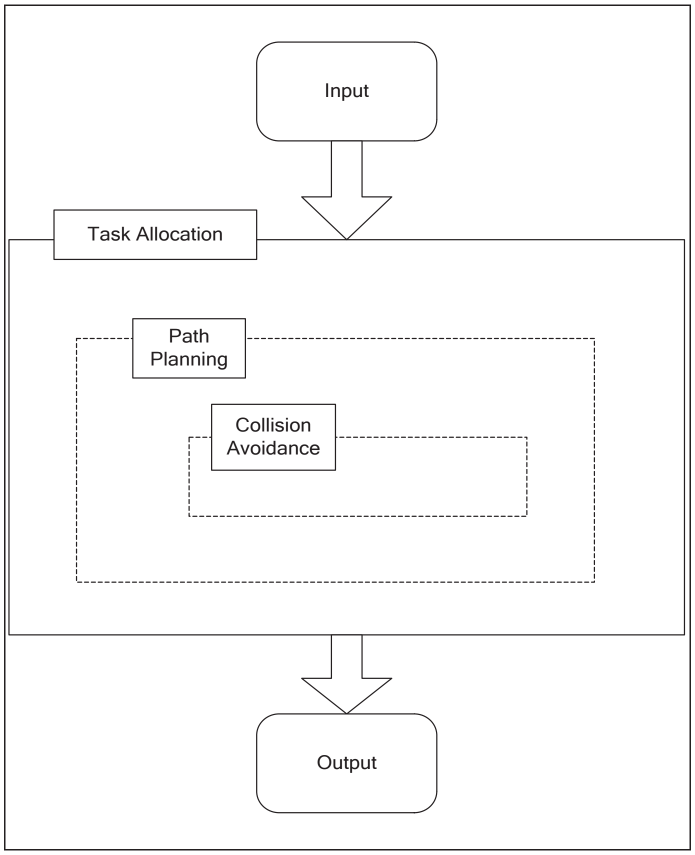
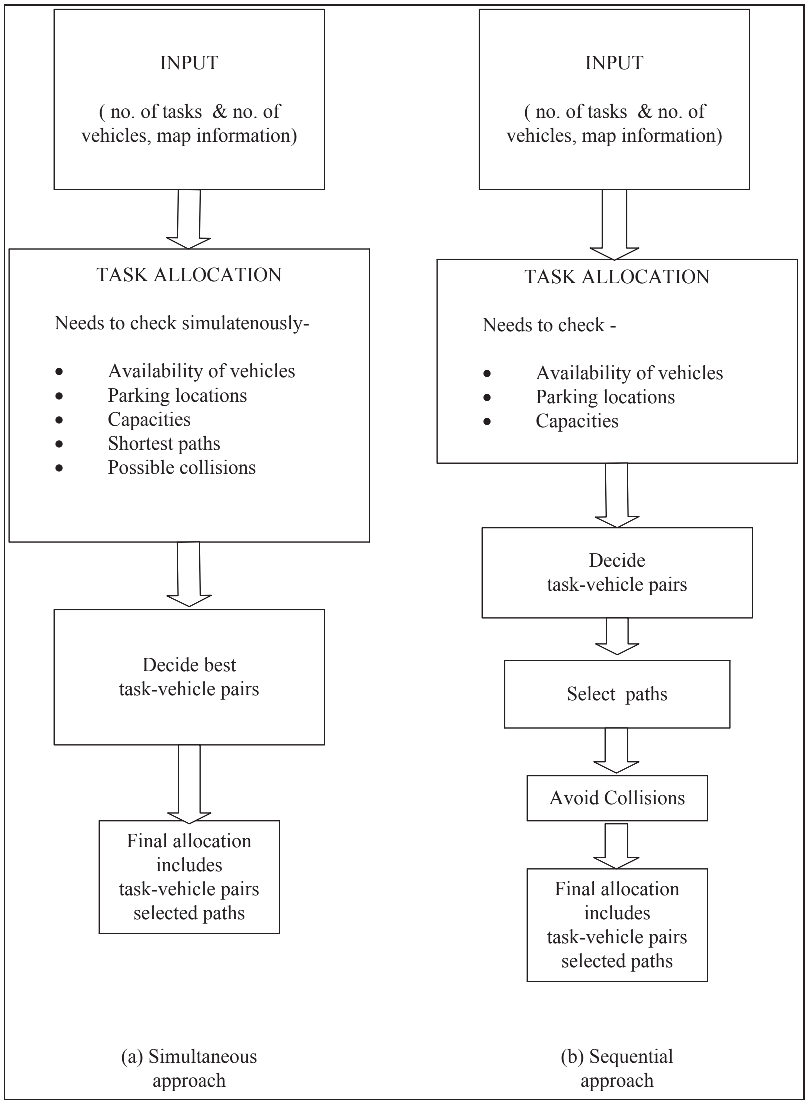


Figure 4-1: Schematic representation of the simultaneous approach

The traditional sequential approach and the simultaneous approach work as follows (Figure 4-2 (a) and Figure 4-2 (b)).

Initially both approaches are fed with the information on tasks, vehicles and map. Then in the sequential approach, tasks are allocated to appropriate vehicles and then paths are planned to perform those tasks accordingly. While the paths are being decided, potential collisions are handled. Tasks-vehicle pairs are selected finally. In the simultaneous approach, when suitable vehicles for the given tasks are selected, collision-free path planning is conducted. Therefore, in the simultaneous approach the best possible task- vehicle pairs are selected at the same time by considering path planning and avoiding collisions.



INPUT
(no. of tasks & no. of vehicles, map information)
TASK ALLOCATION
Needs to check simulatenously-
• Availability of vehicles
• Parking locations
• Capacities
• Shortest paths
• Possible collisions
Decide best task-vehicle pairs
Final allocation includes task-vehicle pairs selected paths
(a) Simultaneous approach
INPUT
(no. of tasks & no. of vehicles, map information)
TASK ALLOCATION
Needs to check -
• Availability of vehicles
• Parking locations
• Capacities
Decide task-vehicle pairs
Select paths
Avoid Collisions
Final allocation includes task-vehicle pairs selected paths
(b) Sequential approach

Figure 4-2: The simultaneous approach and the sequential approach

4.3. Mathematical Modelling

Bi-directional connections network topology is used for the material handling model proposed in this chapter, which is a common feature in many material handling environments such as fully automated container terminals. A task is defined as a transport activity performed by a vehicle, which initially travels to the pickup location from the vehicle's current location and picks up a unit load (e.g. a container/

pallet)from the pick-up node of the task (original location)and transports the unit load to the drop-off node of the task (destination location),through a set of connections known as a path. Once a vehicle drops off its load at the destination, it travels to the next allocated task pick up position. If there is no task allocated, it remains at the last completed task's drop-off location and this location is considered as the respective vehicle's current parking location. At the inception of the task allocation process, it is assumed that all vehicles are parked at different locations of the map.

The time taken to complete the task is considered as a measurement for the decision-making purposes. Time taken to complete a task consists of the following components:

i. **Task pre-processing time** – the time taken for a vehicle to travel from its currently parking location to a task's pick up location(example: t_{AB} or t_{CD} in Figure 4-3)**.**
ii. **Task processing time -** the travelling time from the task pick up location to its drop-off location (e.g. time taken for the vehicle to travel from B to C in Figure 4-3)
iii. **Loading time** – the time taken to load the vehicle at the pickup location (Point B in Figure 4-3)
iv. **Unloading time** – Time taken to unload the vehicle at drop-off location(Point C in Figure 4.3)
v. **Task time**– the total time taken for a vehicle to reach task's pickup location (task pre-processing time), loading time, task processing time and unloading time.

The task time is a function of velocity of the vehicle, travelling distances and loading and unloading times. All vehicles available for material handling will be working on their allocated tasks simultaneously in a material handling environment. Soon after one vehicle completes a task, it travels to the next allocated task's pick-up location and so on. The task procession will continue until all the available tasks are completed by the vehicles.

Figure 4-3 shows an example of what task processing means. In this example, there are two tasks to be processed by a vehicle. The two task pick-up and drop-off locations B, D and C, E, respectively.

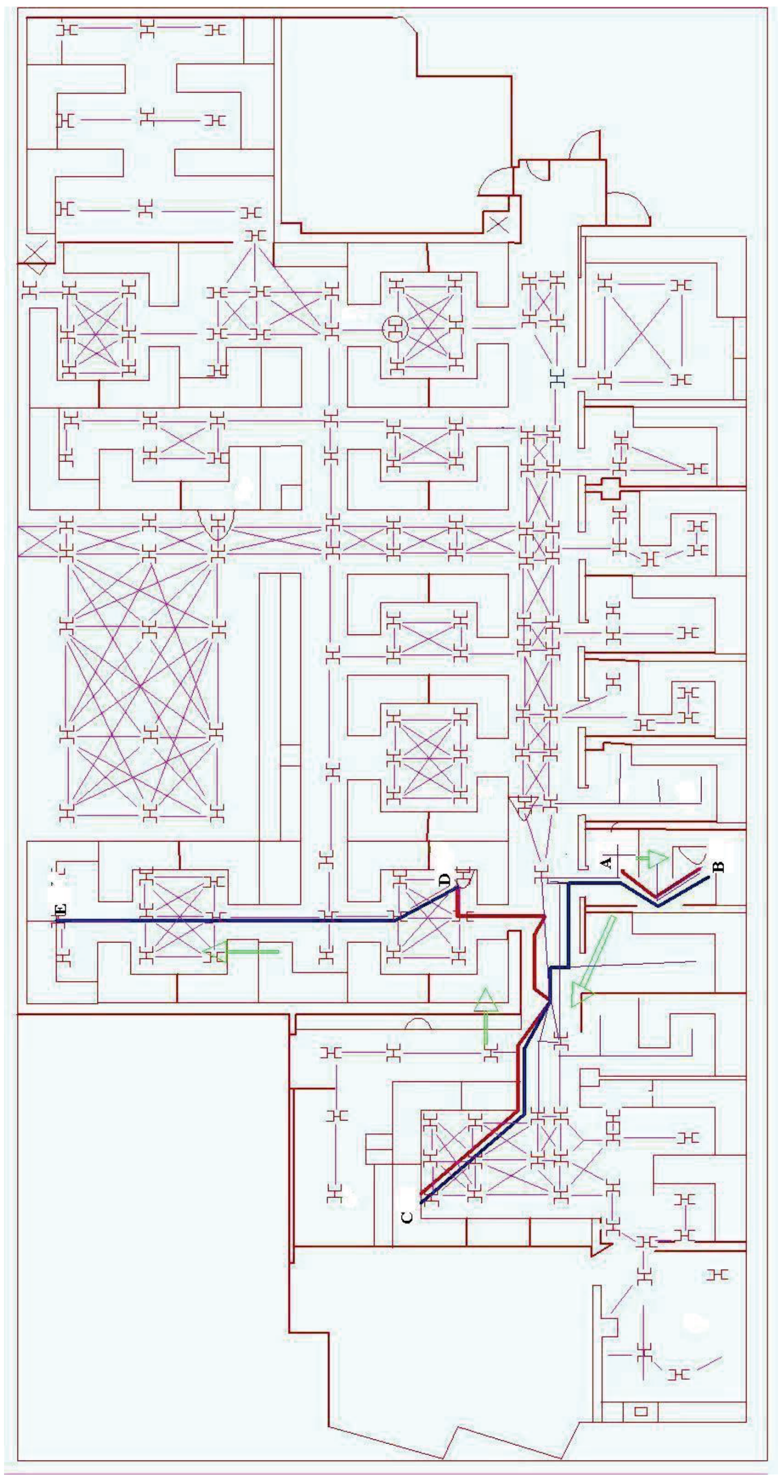


Figure 4-3: An example of tasks, vehicles' start and drop-off nodes

At the inception of the task allocation process, the vehicle is parked at location A. The task allocation sequence is task (1) and task (2). The vehicle starts from location ***A*** and travels towards location B via connections and picks up task (1) and carries that to the drop-off location C. Then, it travels towards location D, which is the pickup location of the task (2) and carries that to the drop-off location E.

4.3.1. Mathematical Model

The mathematical model of multi-vehicle STAMC problem consists of three main elements: map information; task information; and vehicle information. A map consists of nodes and connections. Paths are generated by inter-linked connections. The pickup and drop-off locations are considered as nodes of the map. Task information consists of priority category, pickup and drop-off locations. Vehicle information consists of vehicle class, loaded and unloaded speeds and vehicle capacity. Following assumptions are made when developing the mathematical model.

i. At the inception of task processing, vehicles are parked at different locations of the material handling environment
ii. Each vehicle can process one task (unit load) at a time
iii. The starting time of the first task of each vehicle is the same at the inception
iv. All tasks have equal priority
v. All vehicles have the same capacity

The nomenclature of the problem and the notations of the variables are as follows.

Number of tasks to be allocated	: n
Number of available vehicles	: m
Available tasks for allocation	: T = [T_1, T_2, T_3, . . .T_n]
Avalable véhicules	: V = [V1, V_2, V_3 ...V_m]
Time taken to process all available the tasks	: Makespan (MS)

Vehicle specific information

Initially parked location	: PL_{Vi}
Mean travel speed of an empty vehicle	: V_E

Mean travelspeed of a loaded speed vehicle	: V_L
Loading time of a task	: t_L
Unloading time ofa task	: t_U
Total travelling and waiting time of vehicle V_i	: TV_i
Number of tasks allocated to vehicle V_j	: N_j
Empty travel time of vehicle V_j	**:** $t_{i,(ett)}$

Task specific information:

Pick-uplocation	: L_s
Drop-offlocation	: L_d
Priority group	: P_g
Expected starting time of task T_i	: t_{esti}
Actual starting time of task T_i	: ts_i
Starting time of task T_i of j^{th} vehicle	: ts_{ij}
Expected finishing time of task T_i	: t_{eft}
The actual time of task T_i by vehicle V_j	: TT_{ij}

The selection of the most appropriate vehicle for a given task is called task allocation. The main objective of the task allocation is to decide the best task–vehicle in such a way that the handling cost is minimized. However, this allocation process cannot be considered individually. It has to be performed in such a way that the overall cost function of the total allocation should be satisfied. The time taken to complete a set of tasks is a crucial factor to define the overall efficiency of the material handling terminals (ex. ship turnaround time is used to define container terminal performance). Therefore, time taken to complete a set of given tasks was taken as the cost function. In scheduling terminology, from the time when the vehicles start to do their first task to the time when the last task is completed is called **makespan** of the schedule or the allocation. Therefore, here onwards, in this research, the time taken to complete a set of tasks is called the makespan of a given schedule.

The allocation decision has to be made by considering possible choices as shown in Figure 4-4. For each task, there exits ***m*** number of alternatives / combinations and each combination associates with a cost factor (for example, for task T_i, vehicle V_j the cost factor is $C_{i,j}$) which depends on Task time for vehicle V_j task T_i combination. Each of these cost factors are embedded in the objective function of the task allocation. For example, Task T_1 can be performed by vehicle V_2. Hence the cost factor associated with this combination is $C_{1,2}$. Similarly, other m-1 vehicles can perform task T1. The cost factor for Task $\mathbf{T_i}$ and vehicle $\mathbf{V_j}$ combination is represented by $\mathbf{C_{i,j}}$. The respective cost components of vehicle–task combinations are shown in Figure 4-5 as a matrix.

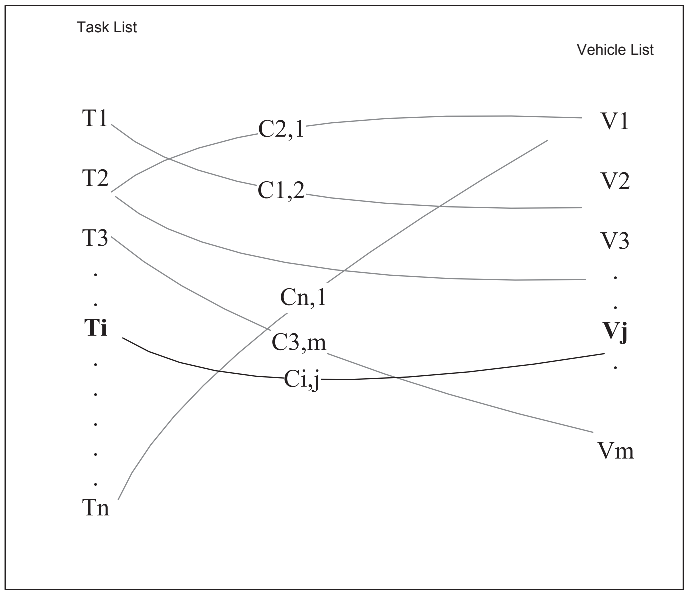


Figure 4-4: Task allocation process: selecting appropriate task-vehicle pairs

Vehicle / Task	V_1	V_2	V_3		V_j		V_m
T_1	$C_{1,1}$	$C_{1,2}$	$C_{1,3}$	---	---	---	$C_{1,m}$
T_2	$C_{2,1}$	$C_{2,2}$	$C_{3,3}$	---	---	---	$C_{1,m}$
T_3	$C_{3,1}$	$C_{3,2}$	$C_{3,3}$	---	---	---	$C_{3,m}$
---	---	---	---	---	---	---	---
T_i	$C_{i,1}$	$C_{i,2}$	$C_{i,3}$	---	$C_{i,j}$	---	$C_{4,m}$
---	---	---	---	---	---	---	---
T_n	$C_{n,1}$	$C_{n,2}$	$C_{n,3}$	---	---	---	$C_{n,m}$

Figure 4-5: Vehicle-task pair's cost matrix

The cost component is directly proportionate to the task time (TT $_{ij}$-task time of task i – vehicle j)that can be calculated as;

$$TT_{ij} = t_{reach(ij)} + t_L + t_{pro(ij)} + t_U \tag{4.1}$$

where $t_{reach(ij)}$ is the travelling time to reach task's pickup location and $t_{pro(ij)}$ is the time taken to travel from a pickup location to a drop off location. Each task time consists of these two travelling components, and the loading and unloading time as well.These two path segments consist of a number of small path segments. Therefore, $TT_{(ij)}$can be given as;

$$TT(ij) = \sum_{k=1}^{QR} \frac{REACH_{(ij)k}}{V_E} + t_L + \sum_{k=1}^{QP} \frac{PRO_{(ij)k}}{V_L} + t_U \tag{4.2}$$

where $REACH_{(ij)k}$ is the path connections vehicle V_j needs to travel through from its parked location to reach the task T_i pickup location(e.g., A to B or C to D as shown in Figure 4-3).$PRO_{(ij)k}$ is the path segments vehicle V_j needs to travel through from the task T_i pickup location to its drop-off location(e.g., B to C or D to E as shown in Figure 4-3). The QR and QP are the number of connection segments contained in $REACH_{(ij)k}$ and $PRO_{(ij)k}$respectively.

If vehicle V_j completes N_j number of tasks in a given set of allocation, the total travelling time of the vehicle V_j will *be;*

$$Total\ traveling\ time\ of\ V_j = \sum_{i=1}^{Nj} TT(ij) \tag{4.3}$$

4.3.2. Optimisation Criterion

Many criteria have been used in scheduling or resource allocation in the literature. For job shop scheduling problems, minimising the makespan, job tardiness and average flow time are commonly used. In vehicle routing problems and traveller salesman problems, the overall tour cost has been used. In automated material handling systems, the overall system's efficiency and resource utilisation are sensible criteria to be used. The performance measurements of the problem presented here are based on several parameters, which are defined below.

4.3.2.1. Overall Efficiency

In this research, the overall efficiency of the system is calculated based on the overall time required to complete the whole batch of tasks (makespan of the schedule) and the lateness of individual tasks.

The total time taken by vehicle V_j (travel time) to complete the allocated tasks is given by equation 4.3. Therefore, the makespan (MS) will be the largest one among the total travelling time of all vehicles for the set of tasks. This can be calculated using equation 4.4.

$$\boldsymbol{Makespan} = \max_{\boldsymbol{for\ j=1\ to\ V_m}}(\boldsymbol{Total\ traveling\ time\ of\ V_j}) \tag{4.4}$$

The starting time of the first task of each vehicle is the same and is equal to the start time of the scheduling. This is set as zero in the mathematical from when representing the task allocation.

$$\sum_{j=1}^{m}\left|ts_{1j}\right| = 0 \tag{4.5}$$

Furthermore, starting times of the remaining tasks are subjected to the following constraints:

The start time of task **T** $_{\mathbf{i}}$of the vehicle **j ($\mathbf{ts_{ij}}$)** is less than the start time of task **T** $_{\mathbf{(i+1)}}$ of vehicle **j.**

$$ts_{ij} < ts_{(i+1)j} \tag{4.6}$$

The expected start time of task $\mathbf{T_i}$ should be equal to or less than the actual starting time.

$$test_i \leq ts_{ij} \tag{4.7}$$

The expected starting time of task $\mathbf{T_i}$ should be equal to or less than expected finish time.

$$test_i \leq teft_i \tag{4.8}$$

The expected starting time of task $\mathbf{T_i}$ should be equal to or less than the actual finish time.

$$test_i \leq taft_i \tag{4.9}$$

Tardiness is defined as the lateness of a task T_i, that is the time difference between the expected finish time ($t_{i,(eft)}$) and the actual finish time ($t_{i,(aft)}$). This can be calculated from equation 4.10

$$Tardiness\_T_i = [\ t_{i,(aft)} - t_{i,(eft)}] \tag{4.10}$$

Therefore, the overall tardiness of the whole batch of tasks can be defined as;

$$Overall\_tardiness = \sum_{i=1}^{n} [T_{i,(aft)} - T_{i,(eft)}] \tag{4.11}$$

4.3.2.2. Vehicle Utilization

In order to utilise vehicles optimally, it is essential to minimise the time where vehicles are not doing any useful work (idle times). To investigate the idle time, two optimisation criteria, namely the percentage of the total empty travel time and the overall idle time of vehicles, are used to measure the resource utilisation in the multi-vehicle STAMC problem.

The empty travel time $\mathbf{(t_{i,(ett)})}$of vehicle $\mathbf{V_j}$ can be calculated as;

$$t_{j,(ett)} = \sum_{i=1}^{N_J} \left[t_{(ett),i} \right] \tag{4.12}$$

Therefore, percentage total empty travel time can be found using the following equation;

$$\textit{Percentage travel time} = \frac{\sum_{j=1}^{Nj} t_{j,(ett)}}{\sum_{j=1}^{Nj} TT_j} \tag{4.13}$$

A vehicle will be in an idle state once the following condition has been met;

$$t_{i,ft} < t_{(i+1),st} \tag{4.14}$$

where $t_{i,ft}$ and $t_{(i+1),st}$are the finish time of task ***i*** and the start time of next task ***(i+1)*** to be undertaken by the same vehicle.

Vehicle idle time is a very good measurement to understand the efficiency of the task allocation process. If any vehicle needs to travel long distances empty in order to undertake a task, during that time respective vehicle will not perform any useful work. Conversely, if vehicles spend most of their operating times on transportation operations on loaded mode, it is obvious that they are performing useful work. The time difference between previously loaded tasks finish time and new tasks loaded task is considered as idle time of the vehicle. Therefore, it can be calculated as;

$$Idle\_time\_v_j = \sum_{i=1}^{N_J} (t_{(i+1),st} - t_{i,ft}) \tag{4.15}$$

Furthermore, overall vehicle idle times also an important parameter to understand the overall efficiency of the task allocation process. The total idle time will be the summation of the idle times of all vehicles and it can be calculated as

$$Total\_Idle\_time = \sum_{i=1}^{m} Idle\_time\_v_j \tag{4.16}$$

4.4. Meta-heuristic algorithms for Simultaneous Task Allocation and Motion Coordination

The multi-vehicle task allocation and motion coordination problem, as explained before, can be categorised as an NP-hard type problem. Hence, it is impossible to find the optimal solution within a reasonable time-frame for a task allocation and motion coordination problem in complex environments and with a large fleet of autonomous vehicles. Therefore, some simple dispatching rules such as first-come-first-served, or shortest processing time first or due date/time are commonly used.These dispatching rules are unable to generate good solutions of near optimal quality. In the last two decades, a new area called meta-heuristic and evolutionary techniques has been studied in the operational research field, which were mainly derived by studying nature patterns.

Generally, applicable heuristics and evolutionary algorithms are appropriate for a wide range of combinatorial problems. Besides SA (Kirkpatrick et al., 1983), other heuristic algorithms such as TS (Glover, 1989), Auction Algorithms (AA) (Bertsekas, 1978) and evolutionary algorithms such as GA (Holland, 1975), Particle Swarm Optimisation (Kennedy and Eberhart, 1995), Ant Colony Optimization (ACO) algorithm (Dorigo, 1992, Dorigo et al., 1996) have been proposed and further investigated in recent years.

These techniques have proven their capacity to generate feasible solutions to many NP-hard problems such as TSP, VRP, etc.Therefore, it was decided to meta-heuristic and evolutionary algorithms to generate near-optimal solutions to the multi-vehicle task allocation and motion coordination problem,due to its NP–hard nature.Of many heuristic and evolutionary algorithms in the literature, simulated annealing and ant colony optimization algorithms, which are widely used in solving similar kinds of problems, are extensively studied in this research. Furthermore, these two algorithms are compared with a commonly used approach called Auction algorithm in the multi-robot task allocation problem.

4.4.1. Simulated Annealing Algorithm

Once SA was introduced by Metropolis et. al.,(1953), it was first used for solving optimisation problems by Kirkpatrick et. al., (1983). This algorithm has proven its ability to find near optimal solutions to many NP-hard combinatorial optimisation problems such as the travelling salesman problem, graph partitioning, quadratic assignment and scheduling. SA algorithm based applications span over diverse areas. The most recent works include: (1) Investigation of vehicle routing problem with time windows using a parallel-simulated algorithm (Czarnas, 2002). Excellent solutions were found for Solomon's benchmark problems (Solomon, 1987). (2) Kim and Moon (2003) used SA for berth scheduling in container port. (3) Melouk et al. (2003) used SA for scheduling in a batch processing plant to improve solution quality and reduce computation time. (4) Sadeh (1996) used a modified SA to solve a job shop scheduling problem which was subjected to tardiness and inventory costs. Two cost functions were used in different temperature ranges of the annealing process. (5) Chiang (1996) worked on an SA-based vehicle routing problem with two different neighbourhood structures. This work is a good example of an application of SA for solving large-scale problems. Nevertheless, to the best of our knowledge, fewhave tried to accommodate the dynamic behaviour of the scheduling problem.

The main feature of the simulated annealing algorithm is that it accepts not only the solutions with improved cost, but also the limited extent of the solutions with deteriorated cost. This feature gives the algorithm hill climbing capability. Initially, the probability of accepting inferior solutions is large. However, this probability is reduced with the search process proceeding. SA is effective, robust and relatively easy to implement and to modify. Regardless of the initial configuration, it can produce high quality solutions. There are several factors to be considered with the application of SA. They are(1)concise description of a system configuration, (2) randomly generating steps or rearrangement of elements, (3) a quantitative objective function; and an annealing schedule of temperatures.

The flowchart of a standard simulated annealing algorithm is presented in Figure 4-6. Application ofthe SA algorithm to solve the task allocation problem consists of following steps:

Step 1: Set initial conditions: the number of tasks (***n***), the number of vehicles (***m***), initial temperature $\boldsymbol{\theta_0}$; and the stopping criteria, task list, vehicle states, etc.

Step 2: Generate an initial task sequence at random. The schedule generated (which task to which vehicle) depends on this task sequence. The simulated annealing algorithm effectively searches the space of schedule combination to find one that minimises the objective function.

Step 3: Generate task-vehicle pairs using a first-come-first-served heuristics. Calculate the initial scheduling time for the given sequence.

Step 4: Set a counter.

Step 5: Modify the task sequence (swapping) as dictated by the Metropolis algorithm and generate ***S*** new schedule.

Step 6: Calculate the difference between the initial and the new solutions of makespan (***Δ***).

Step 7:If the difference is negative, allocate a new value to the best solution and reduce temperature (θ).

Step 8: If the difference is positive, then generate a random number (0<r<1) and compare with P, ($\boldsymbol{P= exp^{(-\Delta/\theta).}}$)

Step 9: If P > r, accept the current solution as the best schedule time, update counters, go to step 5, where P is calculated by the $\mathbf{P= exp^{(-\Delta/\theta)}}$

Step 10: Reduce temperature and check stop condition, if not repeat the process.

The pseudo code of the SA algorithm is given in Appendix 2.

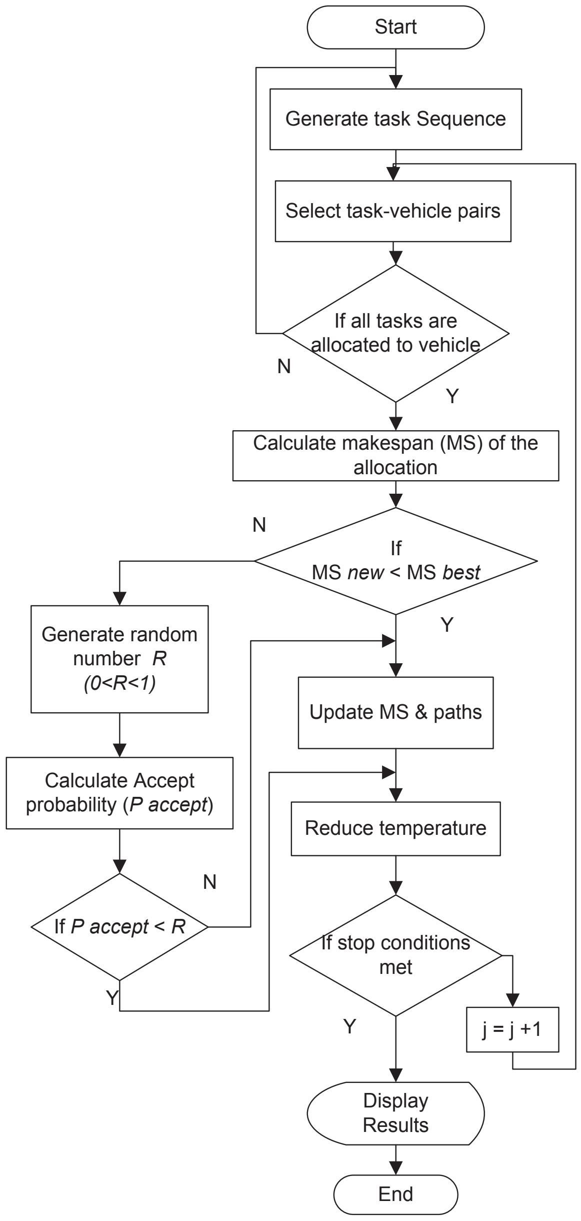

Start
Generate task Sequence
Select task-vehicle pairs
If all tasks are allocated to vehicle
N
Y
Calculate makespan (MS) of the allocation
N
If MS *new* < MS *best*
Y
Generate random number *R* *(0<R<1)*
Update MS & paths
Calculate Accept probability (*P accept*)
Reduce temperature
N
If *P accept* < *R*
Y
If stop conditions met
Y
j = j +1
Display Results
End

Figure 4-6: Flow chart of the Simulated Annealing Algorithm

4.4.2. Ant Colony Optimisation

Ant colony algorithms are inspired by the observation of ant colonies. Ants are social insects that live in colonies and whose behaviour is directed more to the survival of the colony. As a whole, the behaviour of social insects has captured the attention of many scientists because of the structural level their colonies can achieve, especially when compared to the relative simplicity of the individuals. An important and interesting behaviour of ant colonies is their foraging behaviour, and, in particular, how ants can find the shortest paths between a food source and their nest. It was found that ants are able to communicate information concerning food sources via an aromatic essence called pheromone. While they are moving, ants lay down pheromone in a quantity that depends on the quality of the food source discovered. Other ants, observing the pheromone trail, are attracted to follow it. Therefore, the path will be reinforced and will attract more ants. This behavioural mechanism can be used to solve combinatorial optimisation problems by simulating with artificial ants searching the solution space instead of ants searching their environment. Additionally, the objective values corresponding to the quality of the searched food as an adaptive memory is equivalent to the pheromone trails. Furthermore, to guide their search through the set of feasible solutions, the artificial ants are equipped with a local heuristic function.

Ant colony algorithms were first proposed by Dorigo and colleagues (Colorni et al., 1991, Dorigo et. al., 1991) as a multi-agent approach to difficult combinatorial optimisation problems such as the travelling salesman problem and the quadratic assignment problem. There is currently much ongoing activity in the scientific community to extend and apply ant-based algorithms to many different discrete optimisation problems (Dorigo, 1992). Recent applications cover problems such as vehicle routing (Bullnheimer et al., 1998), job shop scheduling (Maniezzo et al., 1994), quadratic assignment problem (Maniezzo et al., 1994), and so on.

When an ant colony optimization algorithm is applied to task allocation for multi-autonomous vehicles, an ant represents a vehicle and starts from its respective vehicle pick-up node (depot). The first task of each ant is allocated randomly. Then each ant selects the next task from the available task list until all tasks are selected. For the selection of tasks, two aspects are taken into account: how good was the choice of that task in previous runs and how promising is the choice of that task in general. The first information is stored in the pheromone trails $\boldsymbol{\tau_{ij}}$ associated with each ***task-vehicle*** pair, whereas the second is the local heuristic function. This measure of desirability, called visibility, is denoted by $\boldsymbol{\eta_{ij}}$.

Each ant's total travel time is calculated based on its selected tasks, planned routes and travel speeds. The maximum travel time of all the ants (vehicles) is considered as the makespan (MS).

In ACO, each ant selects its next task to be performed based on the probability of the respective choice .If task list (**T$_L$**)contains the feasible tasks to be allocated and task **T$_j$** is to be allocated after task **T$_i$**, the probability function can be stated as;

$$P_{ij} = \begin{cases} \dfrac{\left[\tau_{ij}\right]^{\alpha}\left[\eta_{ij}\right]^{\beta}}{\sum\limits_{u \in T_L}\left[\tau_{iu}\right]^{\alpha}\left[\eta_{iu}\right]^{\beta}} \end{cases} \tag{4.17}$$

This is valid when **T$_i$** is an element of task list (**T$_L$**) otherwise **P$_{ij}$** will be zero. The probability distribution is biased by the parameters α and β that determine the relative influence of the trails and the visibility, respectively.

In this equation $\boldsymbol{\tau_{ij}}$ is equal to the amount of pheromone of selecting task ***'j'*** after task ***'i'***. The value of $\boldsymbol{\eta_{ij}}$ is defined as the inverse of the complete travel time which is the sum of the transient time (for the vehicle from its current position to the pick-up node of the task), and the task processing time (the travelling time of the vehicle from the task pick-up node to end node). Allocated tasks are removed from the remaining task list.

In order to improve the quality of the solutions, the pheromone trails of ants must be updated to reflect the ants' performance. This update is a key element to the adaptive learning technique of ACO and helps to ensure improvement of subsequent solutions. The update is conducted by reducing the amount of pheromone elements in the pheromone table of each task-ant combination of the respective schedule in order to simulate the natural evaporation of pheromone and to ensure that no one task-vehicle combination becomes dominant. This is achieved by the following equation:

$$\tau_{ij} = (1-\lambda)\tau_{ij} + \lambda\tau_0 \tag{4.18}$$

where: $\boldsymbol{\lambda}$ is a parameter that controls the speed of evaporation and $\boldsymbol{\tau_0}$ is the initial pheromone value assigned. In our algorithm $\boldsymbol{\tau_0}$ is the inverse of the complete tour cost of each ant (total travel time of respective ant for batch of tasks allocation (one-schedule)). These travel times come from the collision-free path planning by the SiPaMoP algorithm. Each ant calculates the probabilities to select the next task based on Equation 4.1. The task, which gives highest probability, is selected as the next task to perform. Then task selection opportunity moves to the next ant. Each ant selects one task at a time until all the tasks in the feasible list are selected. After all the tasks are allocated to vehicles (ants), makespan can be calculated and accordingly the best makespan so far can be updated. This process repeats until the given number of cycles is performed.

The flow chart of the ACO algorithm used in simultaneous TA and motion coordination is given in Figure 4-7. The number of ants and the number of tasks available in the task allocation problem represent the pheromone table. Initially, all the elements of the pheromone table are set to one. The ACO parameters such as***α, β, λ*** are set to their respective values. At the start of each cycle, the first task for each ant is assigned randomly. From that step onwards, each ant selects the next task based on the probabilities calculated in Equation 4.17.

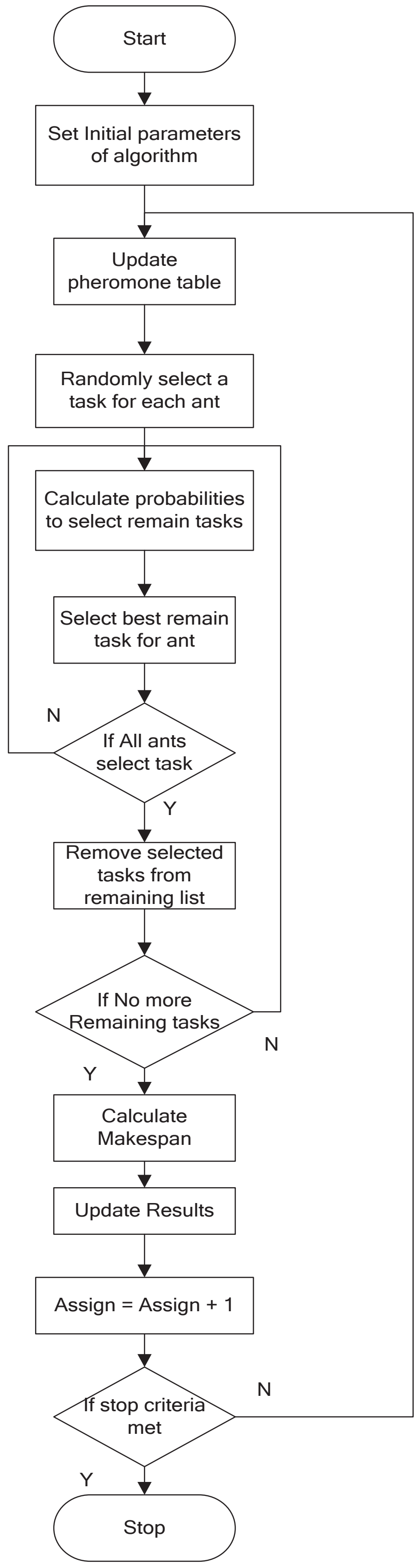

Start
Set Initial parameters of algorithm
Update pheromone table
Randomly select a task for each ant
Calculate probabilities to select remain tasks
Select best remain task for ant
N
If All ants select task
Y
Remove selected tasks from remaining list
If No more Remaining tasks
N
Y
Calculate Makespan
Update Results
Assign = Assign + 1
N
If stop criteria met
Y
Stop

Figure 4-7: Flow chart of the ACO algorithm

The algorithmic representation of the ant colony algorithm based task allocation approach is given below:

Step 1: Set initial conditions: the number of tasks (n); the number of vehicles/ants (m),random state of the problem; other algorithmic constants and stopping criteria
Step 2: Do while remaining tasks not equal to empty matrix
Step 3: Calculate probabilities for each task available to respective ants
Step 4: Sort out the probability table
Step 5: Select most suitable tasks for ants
Step 6: Remove selected tasks from the remaining task list
Step 7: Update ants' new locations
Step 8: Go to step 3
Step 9: When allocation is finished
Step 10: Do a local search based on the best allocation for the cycle
Step 11: After completing the local search, update the pheromone table
Step 12: Repeat the above steps until stopping criteria (when number of cycles equal to pre-defined number)is met

The pseudo code of the ACO algorithm is given in Appendix 3.

4.4.3. Auction Algorithm

The AA was introduced by Bertsekas (1979) for the classical assignment problem. The motivation was to solve the problem by using parallelism in a natural way. It turned out that the resulting method was very fast in a serial environment as well. Subsequent work extended the auction algorithm to other linear network flow problems. In particular, an extension to the minimum cost problem, the ε-relaxation method, was proposed (Bertsekas, 1986). An auction algorithm for transportation problems was studied by Bertsekas and Castanon (1989),and used for shortest path planning by Bertsekas (1991).It was recently used for scheduling activities of JIT production (Nishi et al., 2000) and in a decentralised environment (Takeda et al., 2000) to increase the rate of production.

This algorithm is an intuitive method for solving the classical assignment problem. It outperforms other algorithms in some problems and is also naturally well suited for parallel computation. Even though there are several modified versions of auction algorithms for the simultaneous task allocation and motion coordination problem, the simplest version was used. Following are the steps used in the AA.

Step 1: Set initial parameters such as number of tasks, the number of vehicles, stopping criteria, etc.
Step 2: For each vehicle, compute its utility (shortest reaching time for a selected task)
Step 3: Vehicles bid for tasks based on their utility values
Step 4: Task allocation is done based on the first auction cycle for the descending order of the respective bids
Step 5: Remove allocated tasks from the remaining task list
Step 6: Go to step 2
Step 7: Repeat the steps until all the tasks are allocated.

The auctioning process and the flow charts are given in Figure 4-8 and Figure 4-9. The pseudo code of the AA is given in Appendix 4.

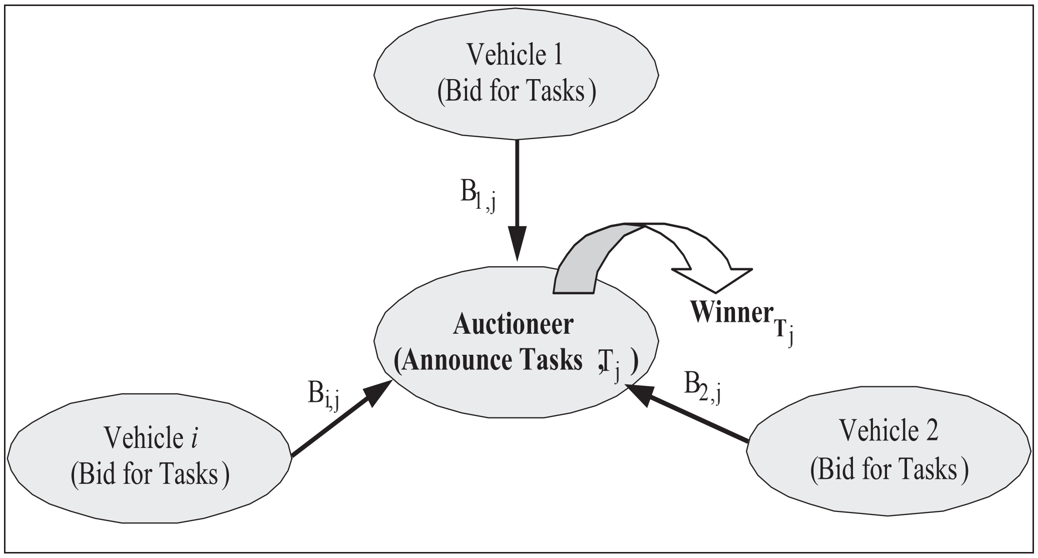


Figure 4-8: Simple Auction Process

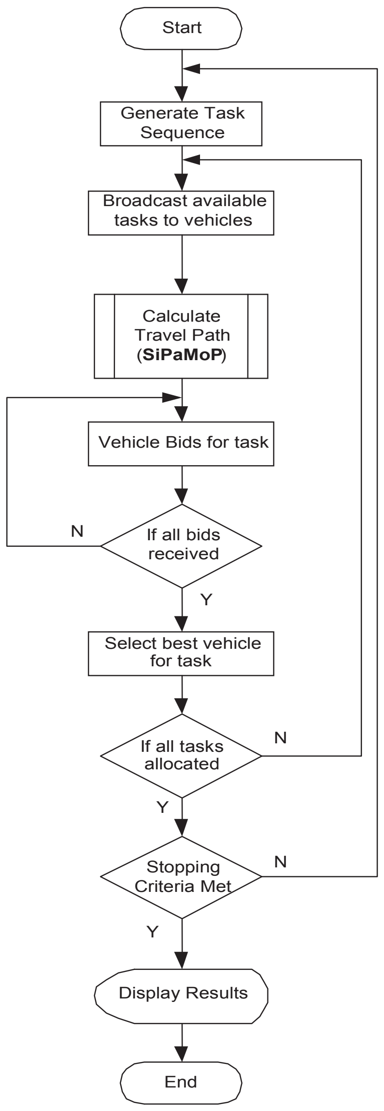


Figure 4-9: Flow chart of Auction Algorithm

4.5. Simulation Studies

In order to investigate the appropriateness of SA, ACO and Auction algorithms with the proposed STAMC approach, a number of simulations were performed. The first simulation study was done with the objective to demonstrate the performance of the STAMC approach against the sequential approach and this is presented in Section 4.5.1. Later in Section 4.5.2, a comparison study was performed to decide the appropriateness of meta-heuristic algorithms for task allocation in the STAMC approach. The rest of this section presents the simulation studies followed by discussion and conclusions.

The following assumptions are made in the simulations:

i. Static task allocation problem is considered here: all information is known before the allocation, no replanning, no vehicle breakdown and no new coming tasks occur

ii. All tasks have equal priority

iii. All the tasks are available for allocation at the start of the task allocation process

iv. Loading and unloading times are considered to be constant for all tasks

v. Speeds of the vehicles when loaded and empty are considered the same.All vehicles have the capacity to undertake one task at a time.

4.5.1. Simultaneous Approach versus Sequential Approach

This simulation was designed to compare the difference between the simultaneous approach (STAMC) and the sequential approach. Here, the multi-vehicle task allocation and motion coordination problem was solved in simultaneous and sequential manner separately. The SA algorithm was used for task allocation purposes in both situations; however, there is no particular reason to select SA algorithm over the other for this simulation.

In both approaches, the parameters of the numbers of tasks, the number of available vehicles and vehicle motion parameters such as velocities, SA algorithm parameters (start temperature $\boldsymbol{\theta_0}$, cooling rate - ***CR***, and stop temperature $\boldsymbol{\theta_{End}}$) and the map information of the material handling environment were given initially. The task allocation process started with a randomly selected task order of allocation.

The objective of the simultaneous approach was to find the best ***task-vehicle*** pairs in order to minimise the overall completion time of all tasks, which is called the makespan (MS) of the schedule generated by task allocation. The makespan varied with different task selection sequences. The role of the SA algorithm was to find the best ***task-vehicle*** pairs with best task allocation order while considering the collision-free shortest paths given by the SiPaMoP module. In the ***task-vehicle*** pair selection stage, the SiPaMoP module was called for each pair in the main algorithm, in order to calculate collision-free shortest path from the vehicle's present node to the pick-up node and then to the drop-off node of the tasks. After all the task-vehicle pairs had been planned in the initial selection stage, the SA algorithm decided whether to accept the allocation or not, which depended on the best possible allocation made up to then. If the SA algorithm based module accepted the results, this allocation was considered as the best allocation. Then, the temperature in the SA algorithm was reduced, based on the cooling ratio of the annealing process. If the solution generated was unacceptable, then it tried to accept the generated solution with a probability. This allocation process was continued until the stop criterion of the SA algorithm was met. The flow chart of with the SA algorithm based STAMC approach is given in Figure 4-10.

The SA algorithm with the SiPaMoP algorithm was used as well for the sequential approach. Here, task allocation was performed initially, based on the information of the shortest path without considering collision avoidance. The Dijkstra algorithm was used to find the shortest paths in this instance. Once all the ***task-vehicle*** pairs were selected, the SiPaMoP module was called in order to find collision-free paths for the selected **task-*vehicle*** pairs. The flow chart of this approach is given in Figure 4-11.

Initially, problem parameters such as the number of vehicles and tasks and the map information were confirmed. Then the task allocation process was started by generating a task sequence randomly. Based on the task sequence, the most suitable ***task-vehicle*** pairs were generated based on the shortest paths. This was done by using the Dijkstra algorithm without considering dynamic change of the connection weights as in the SiPaMoP algorithm. This approach was called the Shortest Path Search (SPS) mechanism. Since this approach will not consider the dynamic changes of the connection weights, the travelling times are shorter than any other approach. Hence this will deliver the lowest makespan values and becomes the lower bound of the makespan. When all the tasks had been allocated, the MS of this schedule was calculated and it was compared with the current best makespan (MS_{best}). If the new MS was better than the current MS_{best} then it was updated with the new value and the respective allocation (task-vehicle pairs) and paths. Then the annealing temperature was reduced based on the cooling ratio selected in the SA algorithm. If the stop criterion was not met, then it continued searching for better task-vehicle pairs slightly changing the initially generated task sequence. If the new MS was inferior to the best MS (MS_{best}), then the SA algorithm had a tendency to accept solutions with some probability as explained Section 4.4.1. This allocation process was continued until the stop criterion of the SA algorithm is met.

Different problem scenarios were created for the same number of tasks and the same number of vehicles by changing the tasks pickup and drop-off locations only(Figure 4-3). Furthermore, each problem scenario was differentiated from others in the following format: number of tasks (n) – number of vehicles (m) – case instance. For example, 8n-4m-case 3 means that eight tasks-four vehicles are available in task allocation and the problem instance was case 3. That means that, for the same number of tasks and vehicles combination, there are different situations (cases) where tasks' pick-up and drop-off nodes and initially parked nodes of vehicles are different from one case to another. These nodes were selected randomly, in order to avoid any bias towards a particular area of the map. Due to this reason, different problem scenarios with same task-vehicle combinations have different makespan values. Each of these cases was run for a fixed number of iterations and the best solution reached up to that time was selected. The results of this simulation are given in Table 4-1 and Figure 4-12.

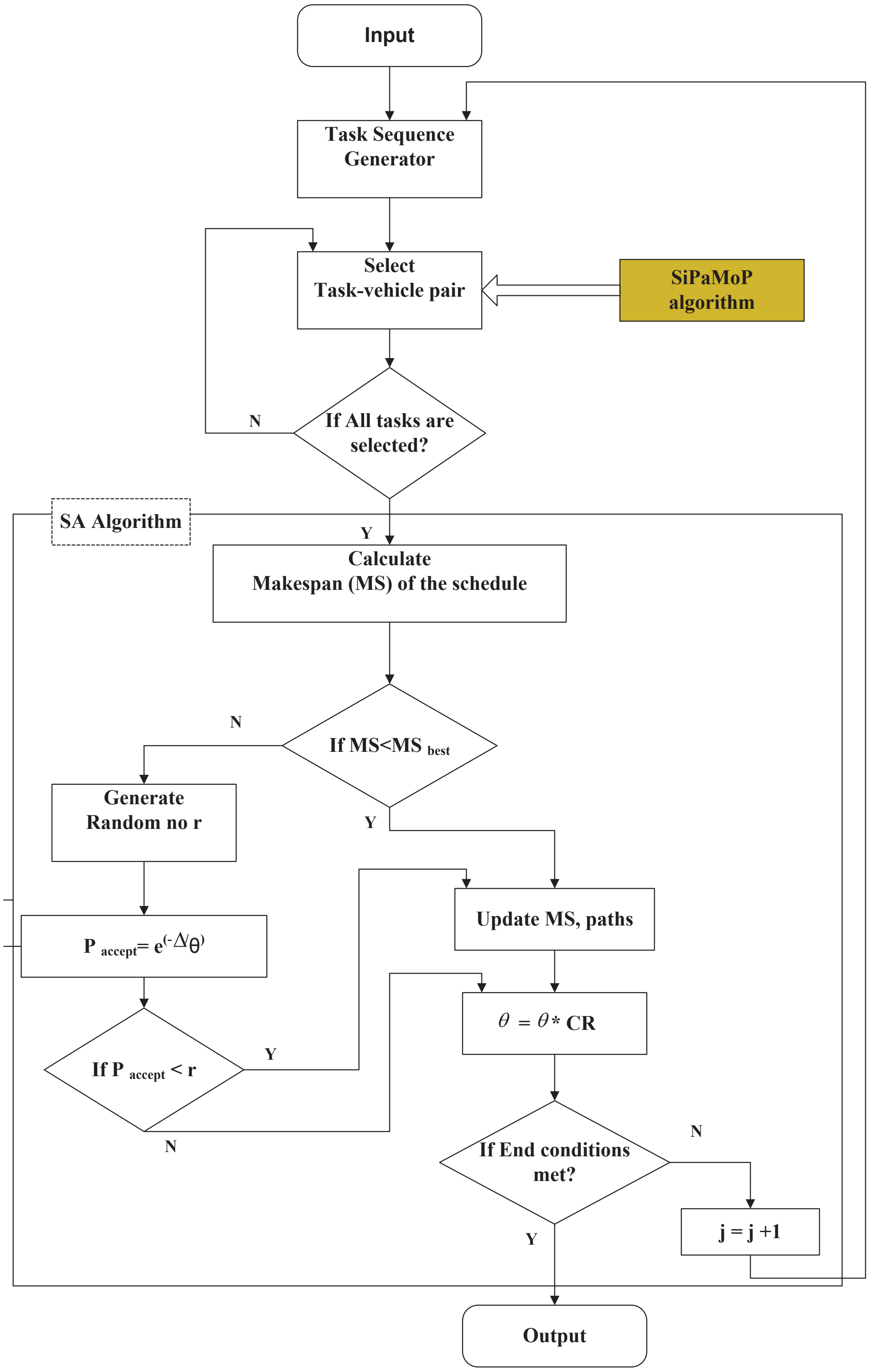


Figure 4-10: Flow chart of the simultaneous approach with the SA algorithm

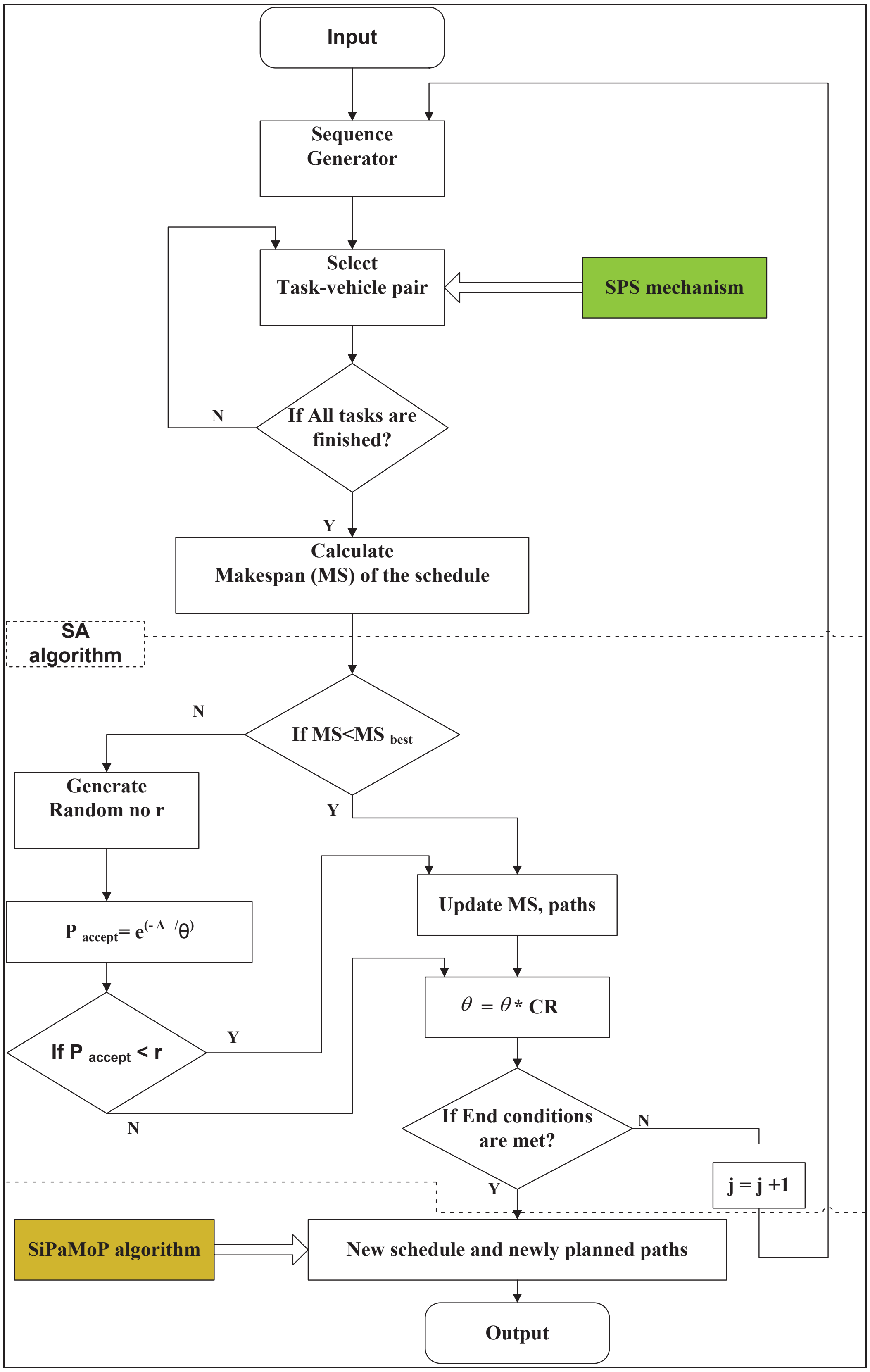


Figure 4-11: Flow chart of the sequential approach with the SA algorithm

In Table 4-1, the makespan values from three approaches, SPS, simultaneous and sequential are presented along with the percentage of improvement of the solutions generated by the simultaneous approach against the sequential approach. Further, the deviation of the makespan of the simultaneous approach against the lower bound of the makespans (SPS approach) is given in the fifth and sixth columns of the Table 4-1. Ideally, the SPS approach based makespan should have the smallest value followed by the simultaneous approach and the sequential approach should give the largest makespan for the same case. This variation is seen in eight out of the nine cases, except in the 8n-4m-case 1. Five out of the nine cases show that the percentage improvement was higher than 3 %of makespans of the simultaneous approach against the sequential approach. The maximum variation of 3.7 (stu) can be seen in 8n-4m-case 2. Seven out of nine cases show a deviation of less than 3.0 (stu) from the lower bound of the makespans of each cases.

The variation of the results in Table 4-1 and Figure 4-12 is due to the following reasons. In the SPS method, only the shortest paths were selected without considering potential collisions among the vehicles during the path planning and motion coordination. These paths were not executable. However, in the simultaneous approach, path planning was integrated with the task allocation phase for each task-vehicle combination by considering the possible collisions among the vehicles. Hence, the STAMC approach was able to generate better solutions for makespans since it could detect and avoid collisions in the early stages of the allocation process. In the sequential approach, task allocation was done based on the shortest path information without considering potential collisions. Later these paths were modified by applying the SiPaMoP algorithm for each shortest path selected before. Due to this, previously selected paths were slightly changed and respective task times were increased considerably. Therefore, the simultaneous approach achieved a better makespan than in the sequential approach.

Table 4-1: Different simulation problem sizes and makespan values

Problem	Makespan (stu)				
	Based on SPS Mechanism	Simultaneous approach	Sequential approach	Percentage Improvement (%)	Deviation from lower bound
8n-4m-case 1	46.0	46.8	46.8	0.0	0.8
8n-4m-case 2	49.0	52.7	53.2	0.9	3.7
8n-4m-case 3	64.2	64.9	72.3	10.2	0.7
8n-4m-case 4	53.4	56.5	57.1	1.1	3.1
8n-4m-case 5	60.9	61.7	63.6	3.0	0.8
12n-4m-case 6	71.8	74.8	82.9	9.8	3.0
12n-4m-case 7	92.9	94.8	95.6	0.8	1.9
12n-4m-case 8	92.2	93.5	101.9	8.2	1.3
16n-4m-case 9	107.7	108.9	114.0	4.5	1.2

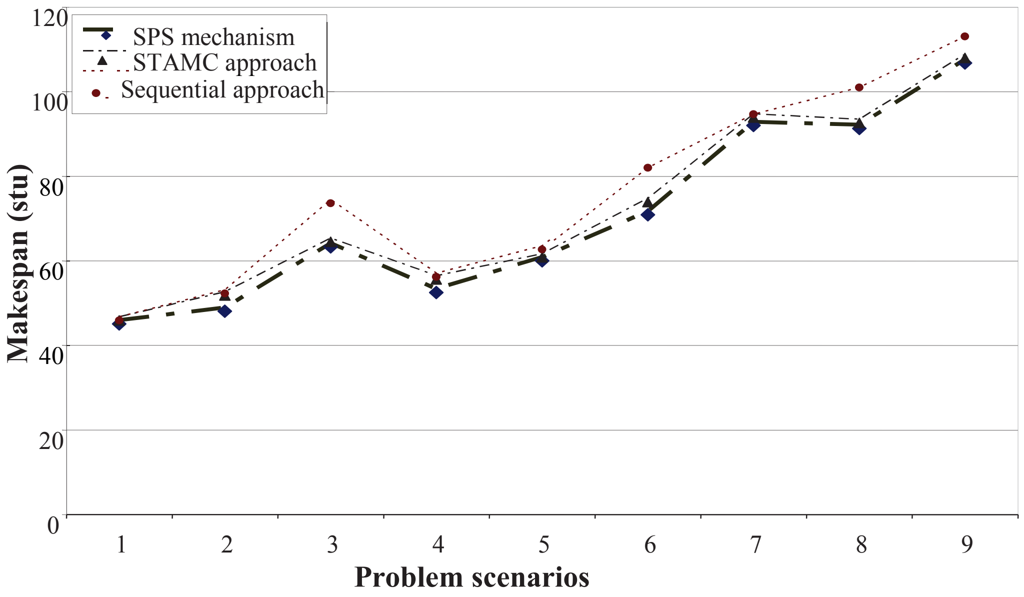


Figure 4-12: Variation of makespans obtained by simultaneous, sequential and SPS (without collision avoidance) respectively

Figure 4-13 and Figure 4-14 show the Gantt charts of task allocation using simultaneous and sequential approaches for the 8n-4m-case 3. Tasks allocated for each vehicle, the implementation order of tasks and the total completion time of each vehicle are visible in Figure 4-13 and Figure 4-14. It can be seen that the simultaneous approach has allocated tasks 6 and 4 to vehicle 1, tasks 7 and 3 to vehicle 2, tasks 1, 2 and 8 to vehicle 3 and task 5 to vehicle 4. The sequential approach has allocated similar task orders as the simultaneous approach to vehicles 1 and 4, but different tasks to vehicles 2 and 3. The makespan obtained in the simultaneous approach (64.78 stu), was less than that of the sequential approach (72.29 stu).The workloads among the vehicles were fairly balanced in the simultaneous approach.

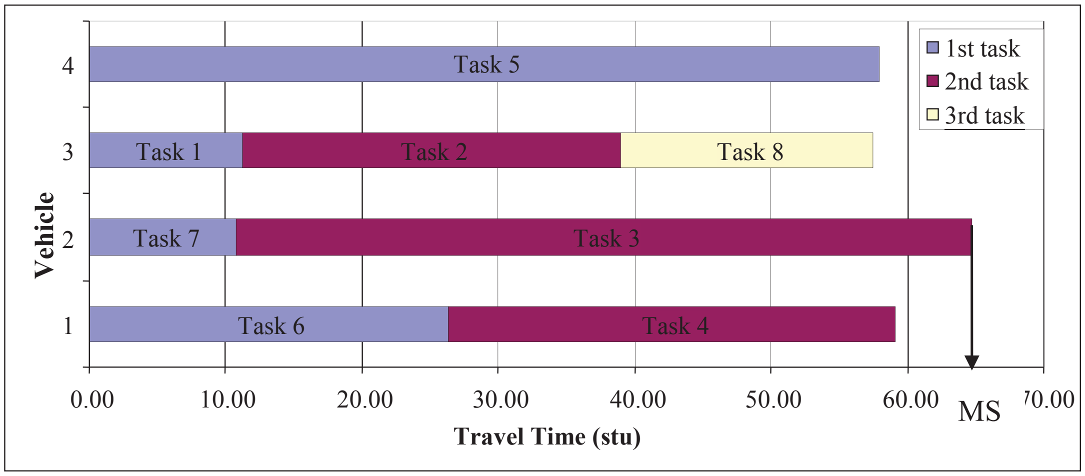


Figure 4-13: Tasks allocation among vehicles, order of implementation and completion time obtained by the simultaneous approach for 8n-4m-case 2: makespan is 64.78 (stu)

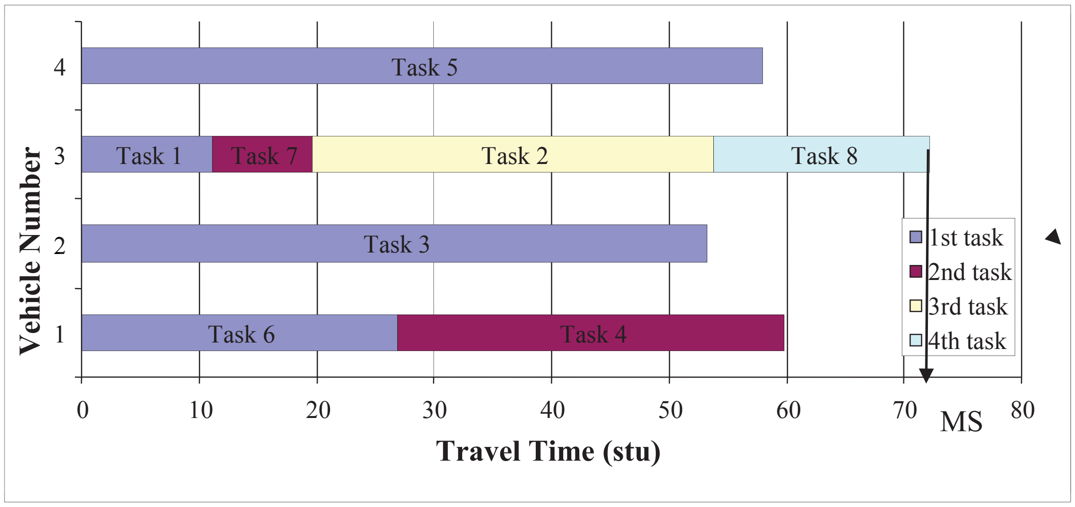


Figure 4-14: Task allocation among vehicles, order of implementation and completion time obtained by the sequential approach for 8n-4-case 2: makespan is 72.29 (stu)

4.5.2. Comparison of the SA, ACO and Auction algorithms

This simulation was designed to investigate the solution quality of the STAMC obtained by the meta-heuristic approaches. Initially, the solutions from heuristic approaches were compared with the optimal results found from exhaustive search for small-scale problems. Subsequently, comparison between the meta-heuristics was investigated. In addition to quality of the solution, the computational times of all the algorithms were investigated.

Of the three meta-heuristic approaches, ACO was selected or comparison with the optimal results generated by exhaustive search since ACO algorithm based STAMC approach managed to deliver reasonably better quality results for small-scale problems than SA and AA. Since exhaustive search evaluates all the possible combinations of the task order considered for the allocation, only the problem size up to four vehicles and eight tasks were compared with different initial settings as explained in Section 4.5.1, which vary the task pickup and drop off locations and the vehicles initial parked locations. The ACO algorithm based STAMC approach was run for a pre-defined number of iterations based on the minimum number of iterations required to saturate with best possible solution for a given problem. For example, 50 iterations used for 6n-2m, 60 iterations for 6n-3m and 75 iterations for 8n-4m respectively. The best makespan generated out of the respective number of iterations was considered as the best solution. The results of this comparison are presented in Table 4-2. Furthermore, Figure 4-15 and Figure 4-16 show the Gantt charts of the schedules generated by ACO algorithm and ES approaches respectively for 8n-4m case.

Table 4-2: Makespan comparison of ACO algorithm with optimal Value

Problem	**Makespan (stu)**		**Deviation from the optimal solution**
Tasks-vehicles	**Optimal solution from exhaustive search**	**ACO (best solution)**	
6n-2m-case 1	75.6	80.1	4.5
6n-2m-case 2	55.3	62.8	7.5
6n-2m-case 3	90.9	102.5	11.6
6n-2m-case 4	99.1	109.3	10.2
6n-3m-case 1	54.5	66.6	12.1
6n-3m-case 2	40.7	42.4	1.7
6n-3m-case 3	72.9	72.9	0.0
6n-3m-case 4	68.1	70.8	2.7
8n-4m-case 1	45.3	59.9	14.6
8n-4m-case 2	43.8	49.8	5.9
8n-4m-case 3	60.3	73.0	12.7
8n-4m-case 4	53.4	60.2	6.9
8n-4m-case 5	56.9	64.2	7.2

The difference in makespan in *8n-4m-case1* was 14.6 (stu).This implied that the schedule generated by the ACO algorithm-based STAMC approach takes 14.6 (stu) longer than the schedule generated by the ES method. Based on the comparison of this approach with the optimal values presented in Table 4-2, the average percentage difference of the ACO based solutions was 12.71 % while out of 13 cases, 4 showed less than 5% difference while 10 cases showed less than 10 % difference and 12 out of 13 cases showed less than 25% difference.

Furthermore, each autonomous vehicle's empty travel times (time taken to reach the tasks' pick-up nodes from the vehicles' current positions) are presented in Table 4-3. It can be seen that out of four vehicles only two showed more empty travel time than the optimal schedules and in fact vehicle 3 and 4 showed less empty travel time than the ES method. When considering all four vehicles, the ACO based STAMC approach contains more empty travel time than the ES.

Table 4-3: Empty travel times (stu) of 8T-4m-case 1

Method	Vehicle 1	Vehicle 2	Vehicle 3	Vehicle 4	Total
ES	19.47	5.48	26.9	22.52	69.17
ACO	30.77	20.9	23.53	14.28	89.48
Difference	11.3	15.42	-3.37	-8.24	20.31

However, based on the results shown in Table 4-2 and Table 4-3, it is evident that the ACO algorithm-based STAMC approach has shown that itcan generate near-optimal results for 31% within the accuracy of 5% difference and 77% fall within the 15% accuracy and 92% of the cases showed the accuracy of 25%. Furthermore, it is evident that the ACO based STAMC approach has not shown large empty travel time too.

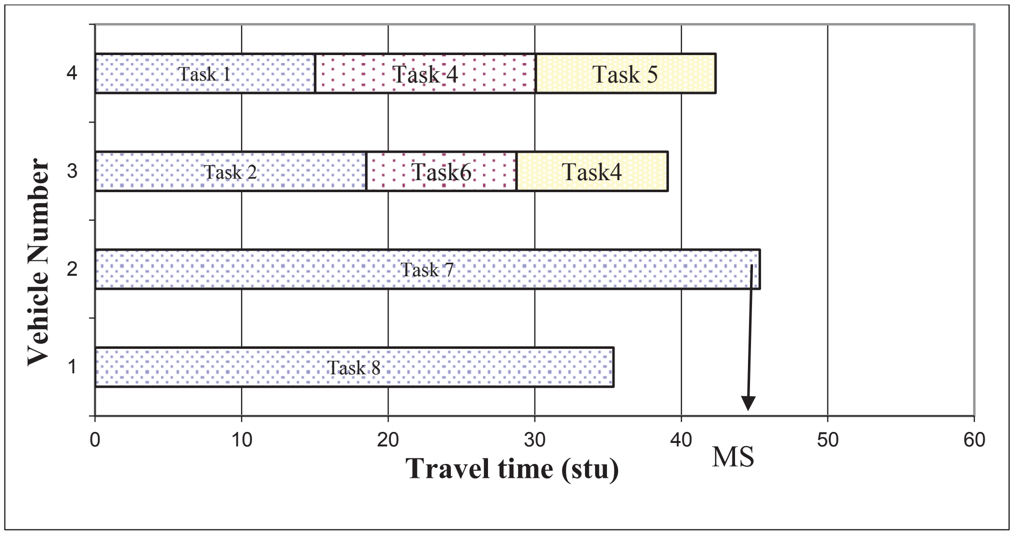


Figure 4-15: Gantt chart of the ES based the task allocation

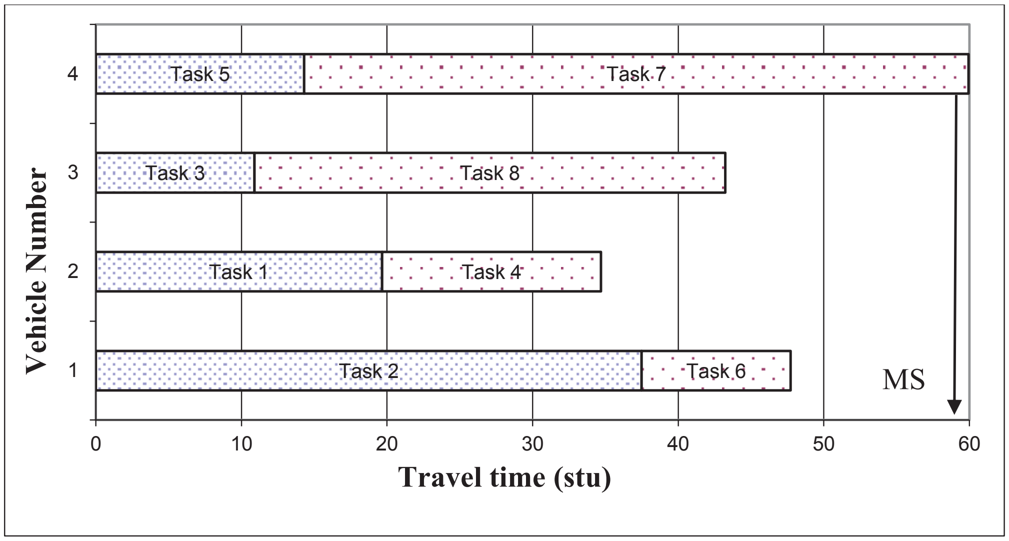


Figure 4-16: Gantt chart of the ACO algorithm based task allocation

Next, simulation was conducted to investigate the expandability of the STAMC approach for larger task allocation and motion coordination problems**.** However, due to the expensive computational cost, ES could not be used for comparison purposes. Instead, the SA and ACO algorithms were compared in this simulation. The problem size was expanded up to 40n-10m. Makespan, computational time and total empty travel times were evaluated. The results are presented in Table 4-4. For smaller task-vehicle combinations, the ACO algorithm gave better makespans compared to the SA algorithm with less computation time. However, with respect to empty travel time, there was no significant variation. The SA algorithm showed better makespan than the ACO algorithm when the task-vehicle combinations increase (more than 20n-5m). Concerning computational time the SA approach shows better results when the problem size is larger than 20n-5m and for empty travel time it was better when the problem size is more than 10n-5m.

Table 4-4: Comparison of makespan, CPU time and empty travel times of ACO and SA algorithms

Problem	Makespan (stu)		CPU time (seconds)		Empty travel time (stu)	
Tasks-vehicle	**ACO**	**SA**	**ACO**	**SA**	**ACO**	**SA**
8n-4m-case 1	58.25	55.07	48.20	103.59	87.06	92.84
10n-5m-case 2	66.93	73.69	109.04	255.28	116.02	113.28
20n-5m-case 3	109.39	112.57	1031.97	773.22	341.54	311.38
30n-10m-case 4	103.20	102.00	3037.45	3430.50	353.05	442.15
40n-10m-case 5	124.90	115.50	6662.63	4908.88	650.07	557.36

Based on the simulations investigated, it was found that the ACO based STAMC approach gives quality results for the simultaneous task allocation and motion coordination of a multi-vehicle problem. The makespans of this approach for small size task allocation and motion coordination problems showed 25% difference from the optimal makespan for 95% of different cases with considerably less computational time. In the case of the large-scale task, allocation and motion coordination problems where the number of vehicles and tasks are higher (more than 10n-5m), the SA algorithm performs well against ACO algorithm. Hence, it has been shown that the ACO is capable of providing near-optimal results for small size task allocation and motion coordination.

In these simulation studies, different problem scenarios/cases were generated randomly for the same number of tasks and the same number of vehicles. Here the main reason to generate different problem scenarios / cases was to start the problem based on different pickup and drop off locations for the tasks and vehicles. This is further explained in detail with the example below. The same problem setting of 8 tasks and 4 vehicles (8m-4m) are solved by SA, AA and ACO algorithms for task allocation. Here, he pick-up and drop-off nodes of the 8 tasks and the 4 vehicles' pick-up positions are listed in Table 4-5 and Table 4-6.

Table 4-5: Eight tasks' pick-up and drop-off nodes

Task No	Pick-up Node	Drop-off node
1	177	153
2	43	83
3	113	115
4	91	148
5	166	172
6	142	138
7	85	33
8	4	76

Table 4-6: Four vehicles' initial positions

Vehicle No	Pick-up node
1	174
2	171
3	77
4	167

The simulation results of task allocation by the proposed SA algorithm-based method are listed in Table 4-7 and Figure 4-17. For the 8n-4m problem, vehicle V1 is assigned tasks 4 and 5, V2 task 7, V3 tasks 3, 6 and 2, and V4 tasks 1 and 8. The Gantt chart of the allocated task-vehicle pairs are shown in Figure 4-17. The planning order or the priority order of the 8 tasks is finally determined as 1, 7, 3, 6, 4, 5, 2 and 8. The task completion time of each task is also listed in the last column in Table 4-7. The makespan obtained from this SA algorithm is 45.96 (stu).

Table 4-7: Simulation results obtained by SA algorithm based STAMC approach

Task No	Pick-up node	Drop-off node	Allocated Vehicle	Planning order	Task completion time (stu)
1	177	153	V4	1	15.02
7	85	33	V2	2	45.34
3	113	115	V3	3	10.90
6	142	138	V3	4	15.30
4	91	148	V1	5	19.62
5	166	172	V1	6	12.26
2	43	83	V3	7	19.49
8	4	76	V4	8	30.94

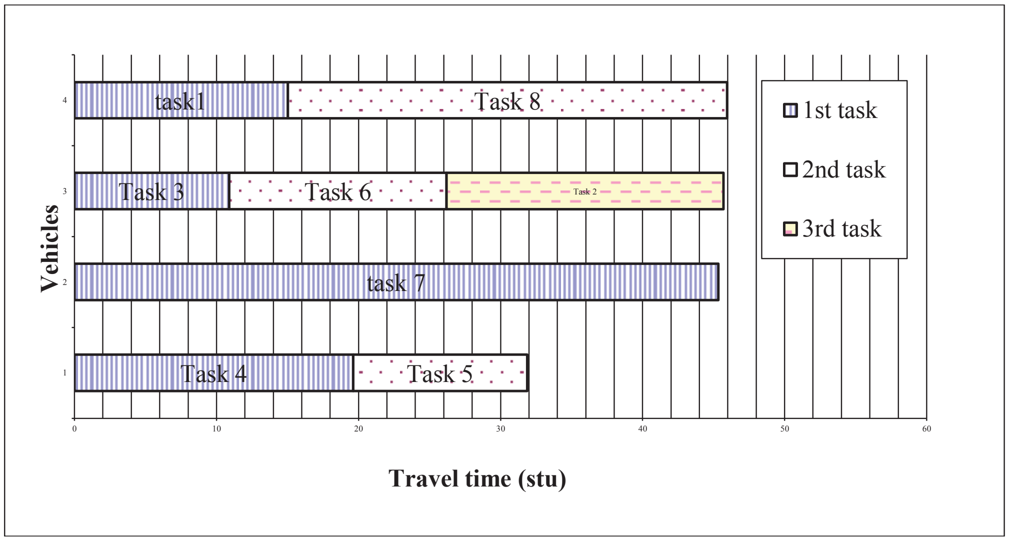


Figure 4-17: Task allocation results obtained from the SA algorithm

The simulation results obtained by the proposed ACO-based STAMC approach are listed in Table 4-8 and Figure 4-18. For the 8n-4m problem, vehicle V1 is assigned tasks 2 and 6, V2 tasks 1 and 8, V3 tasks 3 and 4, and V4 tasks 5 and 7. The Gantt chart of the allocated task and vehicle pairs are shown in Figure 4-18. The planning order or the priority order of the 8 tasks is finally determined as 2, 1, 3, 5, 6, 8, 4 and 7. The processing time of each task is also listed in the last column in Table 4-8. The makespan obtained from the ACO-based STAMC approach was 59.94 (stu) from V4.

Table 4-8: Simulation results obtained by ACO based STAMC approach

Task no	Pick-up node	Drop-off node	Allocated vehicle	Planning order	Processing time (stu)
2	43	83	V1	1	37.48
1	177	153	V2	2	19.66
3	113	115	V3	3	10.90
5	166	172	V4	4	14.32
6	142	138	V1	5	10.25
8	4	76	V2	6	30.94
4	91	148	V3	7	16.44
7	85	33	V4	8	45.62

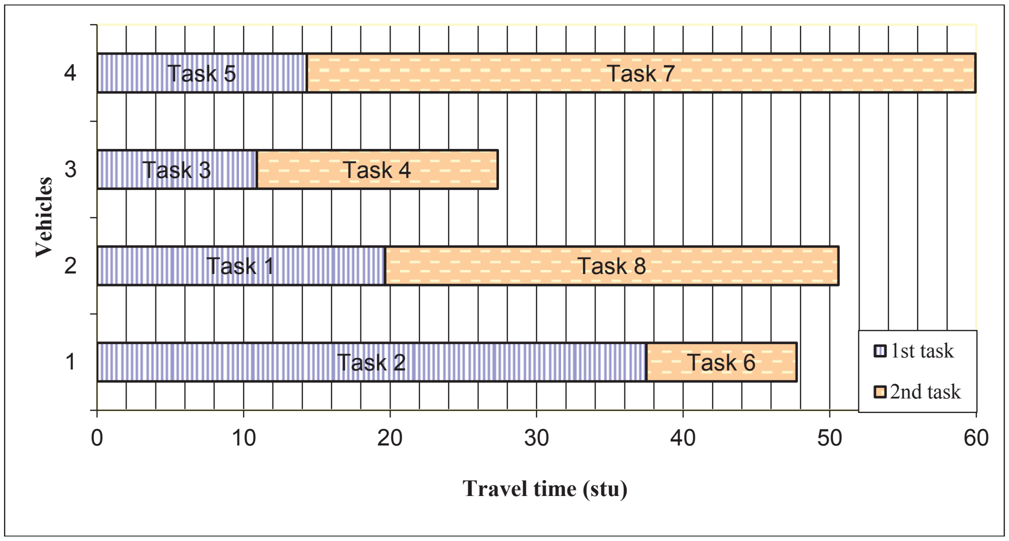


Figure 4-18: Task allocation results obtained from the ACO

The simulation results of task allocation by the proposed AA based method are listed in Figure 4-19 and Table 4-9. In the 8n-4m problem, vehicle V1 is assigned to task 8, V2 to task 7, V3 to tasks 3, 2 and 6, and V4 tasks 1, 4 and 5. The Gantt chart of the allocated task and vehicle pairs are shown in Figure 4-19. The planning order or the priority order of the 8 tasks is finally determined as 8, 7, 3, 1, 2, 4, 5 and 6. The processing time of each task is also listed in the last column in Table 4-9. The makespan obtained from the ACO is 45.34 (stu) from V2.

Table 4-9: Simulation results obtained by AA based STAMC approach

Task no	Pick-up node	Drop-off node	Allocated vehicle	Planning order	Processing time (stu)
8	4	76	V1	1	35.37
7	85	33	V2	2	45.34
3	113	115	V3	3	10.90
1	177	153	V4	4	15.02
2	43	83	V3	5	23.42
4	91	148	V4	6	15.04
5	166	172	V5	7	12.26
6	142	138	V3	8	10.25

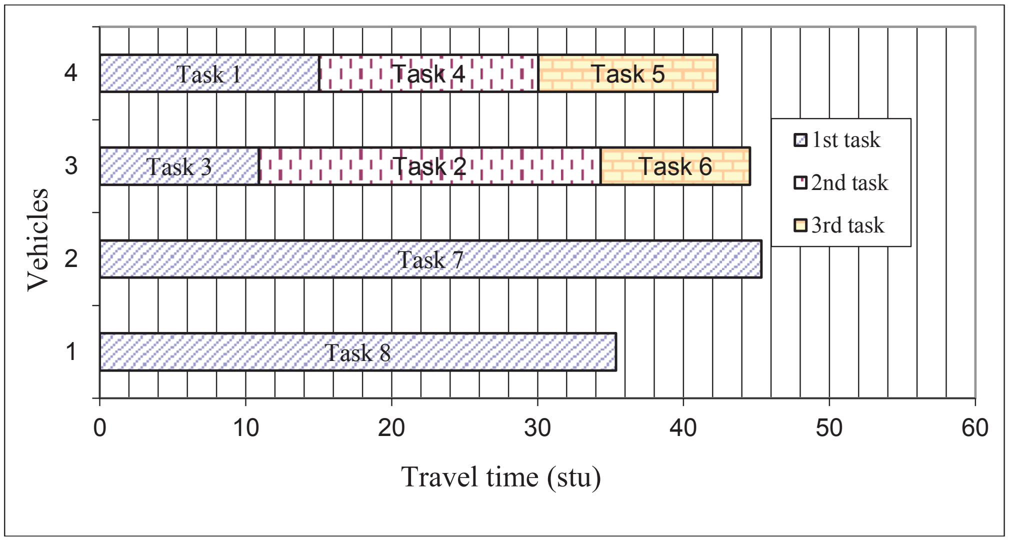


Figure 4-19: Task allocation results obtained from AA

It can be seen from the results and discussions above, that the task allocation results are different. The vehicle utilisations (i.e., the total travel time of all 4 vehicles for performing the 8 tasks) are also given in Table 4-10. Their makespans are within the range of 45 stu to 60 stu as shown in Table 4-11. Vehicle utilization in the SA algorithm-based STAMC approach (168.87 stu) was very close to that of the AA (167.6 stu), while the ACO had more vehicle utilisation (185.6 stu).

Since the task processing time includes two components, namely the travel time from a vehicle's initial node to a task's pick-up node and the time taken from the task pick-up node to its drop-off node, task processing times vary even though the same task is allocated to a different vehicle in the three approaches. In addition, the processing time of the same task can also change due to traffic congestion. This is clearly visible in task 8, where the SA and ACO algorithms take approximately 30 (stu) to process. The AA takes 35 (stu) to process the same task because different vehicles are allocated to do the task number 8.

Table 4-10: Comparison of Makespan and vehicle utilisation of three methods

Parameter	Method		
	SA	ACO	AA
Makespan	45.96	59.94	45.34
Vehicle utilisation	168.87	185.60	167.60

Table 4-11: Makespan comparisons of ACO, SA and Auction algorithms

Problem Tasks-Vehicles	No of iterations	Makespan (stu)		
		ACO	SA	AA
8n-4m-case 1	75	59.9	46.0	45.3
8n-4m-case 2	75	49.8	54.7	46.5
8n-4m-case 3	75	73.0	66.5	66.3
8n-4m-case 4	75	60.2	55.1	55.1
8n-4m-case 5	75	64.2	66.6	58.2
8n-4m-case 6	75	52.8	52.7	52.7
12n-4m-case 1	200	84.6	85.6	86.4
12n-4m-case 2	200	82.4	84.0	80.0
12n-4m-case 3	200	95.1	104.5	96.5
12n-4m-case4	200	98.0	94.3	86.3
12n-4m-case 5	200	81.8	81.6	71.4
12n-4m-case 6	200	76.9	75.8	68.8
20n-5m-case 1	400	130.1	124.0	125.7
20n-5m-case 2	400	107.3	112.6	108.7
20n-5m-case 3	400	137.8	141.1	141.1
20n-5m-case 4	400	118.8	112.6	106.1
20n-5m-case 5	400	94.4	95.2	89.5
20n-5m-case 6	400	110.4	105.2	105.6
40n-10m-case 1	200	108.3	105.4	100.0
40n-10m-case 2	200	119.0	109.4	107.0
40n-10m-case 3	200	121.1	127.4	112.0
40n-10m-case 4	200	133.3	115.5	116.0
40n-10m-case 5	200	124.1	119.7	115.0
40n-10m-case 6	200	125.3	107.6	110.0

Furthermore, the three methods (SA, ACO and AA) were extensively tested with different task-vehicle combinations while allowing each method to run a fixed number of iterations before generating the final results (makespan). The simulation results are shown in Table 4-11. Here, four task-vehicle pairs, i.e., 8 tasks and 4 vehicles (8n-4m), 12 tasks and 4 vehicles (12n-4m), 20 tasks and 5 vehicles (20n-5m), and 40 tasks and 10 vehicles (40n-10m), were used. For each task-vehicle combination, six different cases were generated by randomly generating the pickup and drop down locations of each task and the initially parked locations of the vehicles as explained previously. These results show that AA also generates competitive results compared to meta-heuristic approaches, irrespective of the problem size.

In Table 4-12 and Table 4-13, respectively, the hardware and software configurations used for the simulations and the parameter values of the heuristic algorithm: ACO and SA are given. The parameter values were selected based on separate test runs and the most suitable values in terms of solution quality and computational time. Some of these findings were published in Kulatunga et. al., (2006).

Table 4-12: Hardware and Software Specification of Simulation Studies

Computing Environment Component	Description
Processor Type and core Speed	Pentium 4 Hyper threading @ 3.0Ghz (Prescott)
Front-side Bus Bandwidth	800MHz
DRAM capacity and bandwidth	2GB DDR @ 400MHz
Network Type and Bandwidth	1000Mbps Ethernet
Network Switching Type	Gigabit Switching Fabric
OS Kernel Type and Version	Linux (2.4.21-20.EL)
MATLAB	7.0.4.352 (R14) Service Pack 2

Table 4-13: Algorithms and parameter values

ACO		SA	
Parameter	**Value**	**Parameter**	**Value**
α - comparison integer	3	Annealing start Temperature	100
β - comparison integer	5	Cooling rate	0.9
λ -speed of evaporation	0.7	Annealing stopping temperature	1

4.6. Discussion and Conclusions

This chapter presented the complex multi-vehicle task allocation and motion coordination problem and the STAMC approach with extensive simulations. The mathematical model of the multi-vehicle task allocation and motion coordination problem was first developed, and followed by optimisation criteria. Two types of optimisation criteria namely, the resource utilisation and overall efficiency were considered in the mathematical formulation. Subsequently, the proposed STAMC approach to solving the multi-vehicle task allocation and motion coordination problem was presented while emphasising the difference between the simultaneous approach and the sequential approach.

Various simulations were done to emphasise different features of the proposed approach. The summary of simulation studies is given in Figure 4-20.

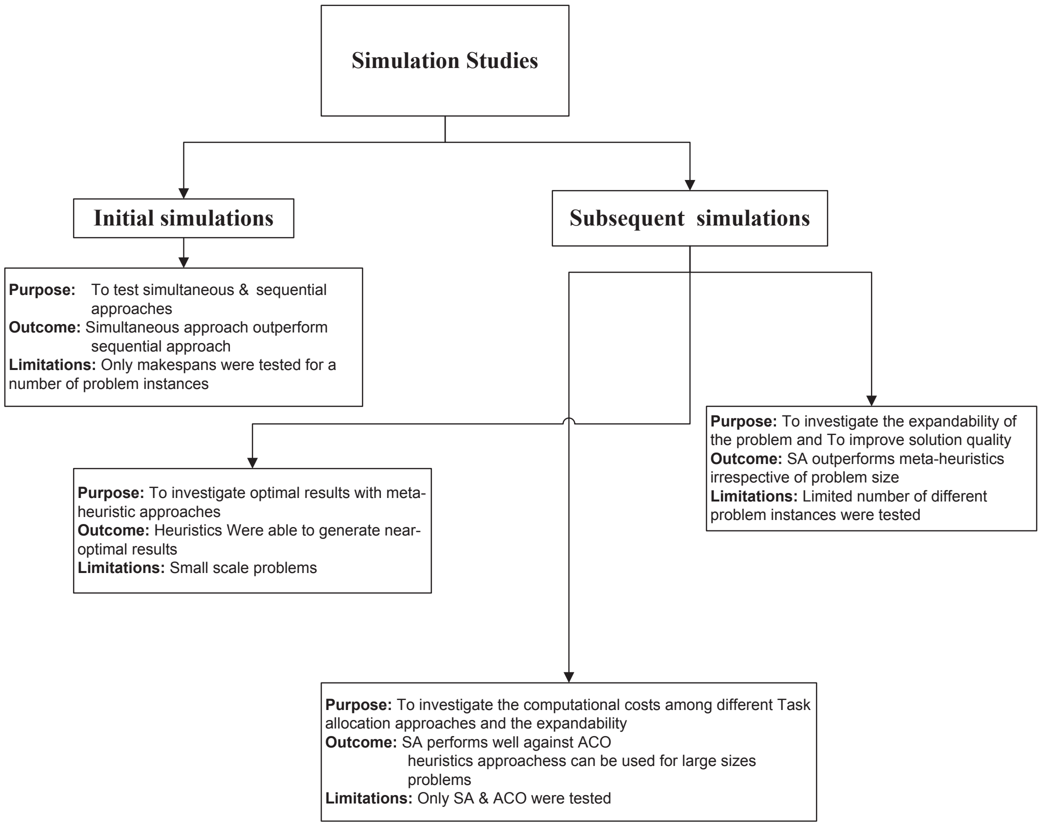

Simulation Studies
Initial simulations
Subsequent simulations
Purpose: To test simultaneous & sequential approaches
Outcome: Simultaneous approach outperform sequential approach
Limitations: Only makespans were tested for a number of problem instances
Purpose: To investigate optimal results with meta-heuristic approaches
Outcome: Heuristics Were able to generate near-optimal results
Limitations: Small scale problems
Purpose: To investigate the expandability of the problem and To improve solution quality
Outcome: SA outperforms meta-heuristics irrespective of problem size
Limitations: Limited number of different problem instances were tested
Purpose: To investigate the computational costs among different Task allocation approaches and the expandability
Outcome: SA performs well against ACO heuristics approachess can be used for large sizes problems
Limitations: Only SA & ACO were tested

Figure 4-20: Summary of the simulation studies

First, simulation was used to emphasise the advantages of the simultaneous approach against the sequential approach. The results revealed that the simultaneous approach can get results closer to the lower bound generated by the shortest path search approach without considering collision avoidance. In the next stage of the simulations, different meta-heuristic approaches were extensively investigated in the STAMC approach. Two commonly used algorithms namely SA and ACO were used with AA. A meta-heuristic, ACO was compared with the optimal results generated from exhaustive search for small-scale problems. The results showed that ACO based STAMC approach has shown that it can generate near-optimal results for 31% within the accuracy of 5% difference and 77% fall within the 15% accuracy and 92% of the cases showed the accuracy of 25%.

Next the proposed meta-heuristic techniques based STAMC approaches were tested for large problem sizes. The SA algorithm based solutions outperformed the ACO when the problem size increased to 40n-20m. Furthermore, empty travel times were also compared with SA, ACO and Auction algorithms, in these simulations.

Based on all the simulations, it can be concluded that the proposed meta-heuristics based STAMC approach is able to generate near optimal solutions for complex multi-vehicle task allocation and motion coordination problem within a reasonable time, even for large problem sizes. However, there are a number of issues that were not considered in this chapter.

It was assumed that all problem-specific information (available tasks and available vehicles and vehicles' operating environment) is fully known before starting the task allocation process. However, in real world situations this information is not fully available before task allocation starts. Most of the time, new tasks arrive for allocation while the allocation process is running and unexpected events such as vehicle breakdowns may occur. Therefore, in Chapter 5, these issues will be taken into account and the proposed approach for multi-vehicle task allocation and motion coordination problem will be tested in dynamic conditions where task allocation and motion coordination environment vary with time.

Furthermore, Chapter 4 has not paid much attention to the ways of improving the computational efficiency. Only computational costs were compared at one stage of the simulation between different task allocation methods. However, there is always a trade-off between the quality of solutions and the computational time spent. Therefore, if computations can be efficiently performed, then it is possible to find better solutions within a shorter time. This issue will be taken into consideration in Chapter 6.

In the real world outdoor material handling environments, the scale of the problem and operational environment are normally large, compared to the simulation environment used in Chapter 4. This was closer to an indoor material handling environment, where the number of path segments and nodes in the map was in the scale of hundreds. However, in outdoor material handling environments, these parameters are comparatively large and the number of vehicles operating in such environments is higher. These issues will be tested in Chapter 7, where a case study will be conducted based on the information obtained from a fully automated container terminal.

Chapter 5

STAMC Approach for a Dynamic Environment

5.1. Introduction

It is quite often that many resource allocation problems need to be handled without entire information of the problems. Some information may only be available during the resource allocation process. In order to overcome this issue, the resources have to be allocated or rescheduled with updated information. These types of resource allocation or scheduling problems are known as dynamic scheduling problems. The multi-vehicle task allocation and motion coordination problem is also similar to this. There may be new tasks arriving for allocation once the allocation process is proceeding or the road network where vehicles are operating can be changed due to a blockage such as a vehicle breakdown.

As Kim and Gunther (2007) stated, it is very difficult to perform pre-planning or scheduling in container terminals for longer time horizons. This is mainly due to the fact that the problem's specific information such as availability of vehicles, tasks to be allocated, constraints in the operating environment are not available all at once. The relevant information arrives at different time slots. In these situations, the solutions generated with the available information will not be accurate. Therefore, it would be difficult to find solutions in advance.

According to Bianchi (2000), there are two main characteristics in dynamic scheduling: (1) the information required to solve a problem is time dependent, and when new information is collected scheduling needs to be updated.(2) Solutions have to be found concurrently with incoming information. This means that it is impossible to find a-priori solutions. However, if problem-specific information varies over the time but variation pattern is known in advance or problem specific information of the schedule is non-deterministic, then it can be solved a-priori, with probabilistic information. However, these types of problems may not be categorised into a dynamic scheduling problem (Bianchi 2000).

In container terminals, containers arrive at different time intervals from different sources such as cargo vessels, freight trains and container carrier trucks. Planning and scheduling of a terminal is not limited to loading/unloading of containers from/ to cargo vessels. It also includes handling of containers, which arrive from other sources at the same time. These arrivals are always not known completely a-priori. However, it is essential to handle these new containers, whenever they arrive. Otherwise, the terminal management needs to bear the costs of keeping those new containers waiting. Conversely, if these can be handled promptly, the overall efficiency can be enhanced.

Arrival patterns of new tasks (containers to be handled in a CT) cannot be easily predicted. Unexpected events such as vehicle breakdowns, road congestions, interruptions can occur as well. Due to unexpected events and the unpredictable nature of material handling, task allocation, path planning and collision avoidance in attic environment are complicated. Because of the change of parameters and environmental condition, it is necessary to modify or partially change the solution already determined. This can be achieved by way of a dynamic task allocation / scheduling and motion coordination.

There are mainly two ways of handling previously highlighted changes over the scheduling time horizon. One method is to reschedule the whole problem at fixed time intervals with updates of the variations taken into account from time to time. The other method is to reschedule the whole problem or part of the problem whenever variations occur in the environment or in the scheduling problem parameters. These two methods are shown in Figure 5-1. If new tasks arrive or variations occur regularly (as Task $_i$ to Task $_{n+1}$ show in Figure 5-1(a)) the first method is suitable. However, the task arriving between two adjacent rescheduling intervals has to wait until the next rescheduling cycle. If task arrival occurs on a non-regular basis, the second method can be used (Figure 5-1(b)).

.

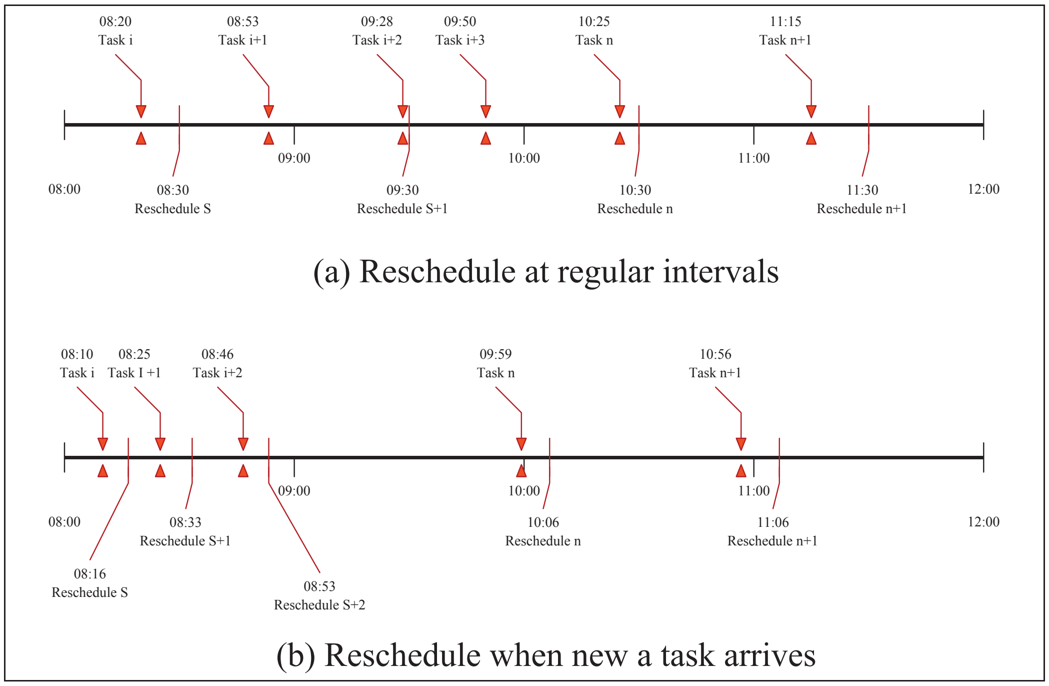


Figure 5-1: Typical rescheduling methods

The time between two rescheduling intervals significantly affects the overall performance of the material handling system. When the rescheduling interval is stretched, the newly arrived tasks need to wait for a longer time. Further, frequent re-scheduling is not advisable if computation cost for the rescheduling process is high. Rescheduling has to be done when unexpected events or disruptions occur in the system, such as vehicle breakdowns, or some disruptions occur in the available paths. Although such unexpected events occur rarely, it is important to study and rectify them. Otherwise, not all plans and schedules would be effective.

Therefore, it is necessary to address the multi vehicle task allocation and motion coordination problem by considering dynamic characteristics. In order to accommodate changes, it is rational to consider dynamic multi-vehicle task allocation and motion coordination as a series of static scheduling sub-problems.

Dynamic Vehicle Routing Problem (DVRP) is the closest problem to the dynamic version of multi-vehicle STAMC problem studied in the literature. This is similar to VRP. However, rescheduling is done at different time intervals to accommodate changes to the main problem over time interval. A comprehensive analysis of DVRP can be found in Bianchi (2000). Most recent works of DVRP include Yuan and Li (2007), Taniguchi and Shuimamoto (2004), Rizzoli et.al., (2007) and Kytojoki et.al., (2007). A few of them have used heuristic or evolutionary techniques as an optimization approach (Rizzoli et.al. 2007).

There are several studies related to material handling in dynamic environments. Meersman (2002) has studied an integrated scheduling problem for various types of handling equipment at an automated container terminal. The handling times were not known exactly in advance. Instead, schedules were done based on the partially available information and schedules were updated when new information on realizations of handling times became available. Among the BSA and several dispatching rules (DR), the BSA performed best on average, and in some situations, simple dispatching rules had performed almost as well as the BSA. Furthermore, this study has shown that it is important to have a plan on a longer horizon with inaccurate data, rather than to update the planning anticipating emergence of new data for updating.

Le-Anh and De Koster (2005) proposed an on-line vehicle-dispatching rule for warehouses and manufacturing facilities. Multi-attribute dispatching rules and the single attribute rules were tested. The results revealed that the multi-attribute DR performs well against single attribute DRs. Furthermore, the impact of reassigning moving vehicles based on both single and multi-attribute, were investigated and the results suggested that reassigning of moving-to-park vehicles has a significant positive effect on reducing the average load waiting time.

In addition to dynamic task allocation / scheduling problems due to new arrivals in material handling, uncertainties such as vehicle breakdown and path blockages in the already planned/scheduled tasks also frequently happen in practical situations. Due to the complexity of the problem, limited research is found in the literature dealing with vehicle breakdowns (Paul and Liu, 2006, Li et al., 2008). Different replanning approaches were presented in Paul and Liu (2006) for unexpected situations such as vehicle breakdown and path blockages. Mirchandani and Borenstein (2008) proposed a real-time VRP with time windows which is applicable to delivery and/or pick-up services that undergo service disruptions due to vehicle breakdowns. A Lagrangian relaxation based-heuristic is developed, which includes an insertion based algorithm to obtain a feasible solution.

The facts discussed so far reveal that the nature of multiple autonomous vehicles STAMC problem is dynamic. Therefore, any solution to this problem should address the dynamic nature of the problem. The rest of the chapter is structured as follows: Section 5.2 presents the formulation of dynamic multi-vehicle task allocation and motion coordination problem. Section 5.3 presents the STAMC approach for the dynamic multi-vehicle task allocation and motion coordination problem. Section 5.4 presents the simulation studies and the results. This is followed by conclusions in Section 5.5.

5.2. Formulation of Dynamic Multi-Vehicle Task Allocation and Motion Coordination Problem

There are two significant differences between the dynamic and the static versions of the multiple vehicle task allocation and motion coordination problem. In the dynamic version, tasks' arrival time and priority are considered as two important factors. However, in the static version, it is assumed that all the tasks are available at the time of scheduling. Different types of tasks can be categorized into groups based on their priorities. Tasks within a priority group have equal priority. The group of tasks with highest priority should be allocated first. The group of tasks with least priority is allocated last.

Furthermore, in the dynamic scheduling problem, rescheduling is to be performed at regular time intervals over the operational time horizon to accommodate a number of new tasks. However, it is not mandatory to reschedule at fixed intervals but this is decided by the arrival pattern of the new tasks for the schedule.

5.2.1. Mathematical Model

It is assumed that tasks have different priorities based on their importance. They can be categorised into priority groups in descending order. Further, the number of tasks and vehicles available at each scheduling or rescheduling interval varies since the task arrival pattern and vehicle breakdowns cannot be predicted.

Problem specific information

Rescheduling interval	: RSI
Time between two adjacent rescheduling intervals	: t_{int}

Rescheduling intervals	: [0, 1, 2, 3 …e ...f]

Where, **e** is the intermediate arbitrary rescheduling interval and **f** is the final interval

Number of Priority groups	: [1, 2, 3,...,p,...l]

Where **p** and **l** are arbitrary and least priority groups

Number of tasks of each priority group at $\boldsymbol{e^{th}}$ interval: [$N_{1,e}$, $N_{2,e}$, $N_{3,e}$…$N_{L,e}$]

In addition to the information on tasks and vehicles presented in Chapter 4, the following information is needed in the dynamic multi-vehicle task allocation and motion coordination problem.

Additional task and vehicle specific information

Task arrival time	: t_{ari}
Task priority	: p
Batch size of the tasks	: BS
Remaining travel time of the previously allocated tasks	: tt_{pre}

The same simulation environment used in Chapters 3 and Chapter 4 is considered for this chapter too. Hence, definitions and equations used in Chapters 3 and 4 are valid. However, slight modifications have been made to equation (4.1) in order to match it with the dynamic scheduling problem. Since rescheduling is done at different time intervals, the available tasks and vehicles are checked before each rescheduling instances.

The number of available tasks for rescheduling at rescheduling interval ***e*** is taken as $\boldsymbol{T_{en}}$. However, only a limited number of tasks (Batch of tasks) is considered for allocation in a given rescheduling interval with a batch size of ***BS.*** The number of available vehicles at $\boldsymbol{e^{th}}$ interval is considered as $\boldsymbol{V_{em}}$

Available task list and the vehicle list at interval ***e*** can be given as;

$T_e = [T_1, T_2, T_3 ...T_{en}]$

$V_e = [V_1, V_2, V_3 ...V_{em}]$

The loaded and empty speeds of the vehicles and loading/unloading times to/from vehicles are considered the same as in Section 4.5 of Chapter 4. Therefore, the total completion time of task $\boldsymbol{T_i}$by vehicle $\boldsymbol{V_j}$ can be calculated using Equation (4.3) in Chapter 4. Starting time of the first allocated tasks for all vehicles is assumed to be the same, at the beginning of the scheduling process (at the initial schedule when $\boldsymbol{t = 0}$).Furthermore, it is assumed that the absolute time at $\boldsymbol{e}$th rescheduling instance is $\boldsymbol{t_e}$ and the absolute time at the last rescheduling instance is $\boldsymbol{t_f}$. Therefore, the total completion time of the task $\boldsymbol{T_i}$ by vehicle $\boldsymbol{V_j}$at time interval $\boldsymbol{e}$ can be calculated based on Equation (4.1) as follows;

$$TT_{ij} = t_{avi} + t_{reach(ij)} + t_L + t_{pro(ij)} + t_U \tag{5.1}$$

where $\boldsymbol{t_{avi}}$ is the available time of vehicle $\boldsymbol{V_j}$ to undertake new tasks assigned to it at $\boldsymbol{e}$ rescheduling interval. If vehicle $\boldsymbol{V_j}$ completes $\boldsymbol{n}$ number of tasks at the rescheduling interval of $\boldsymbol{e}$, the total travelling time of the vehicle $\boldsymbol{V_j}$ will be:

$$Total\ Traveling\ time\ of\ V_j = \sum_{i=1}^{en} TT_{ij} \tag{5.2}$$

Therefore, makespan of the reschedule at time interval $\boldsymbol{e}$ is:

$$Makespan_e = max_{j=1}^{s}\ Total\ Traveling\ time\ of\ V_j \tag{5.3}$$

Where $\boldsymbol{s}$ is the number of available vehicles.

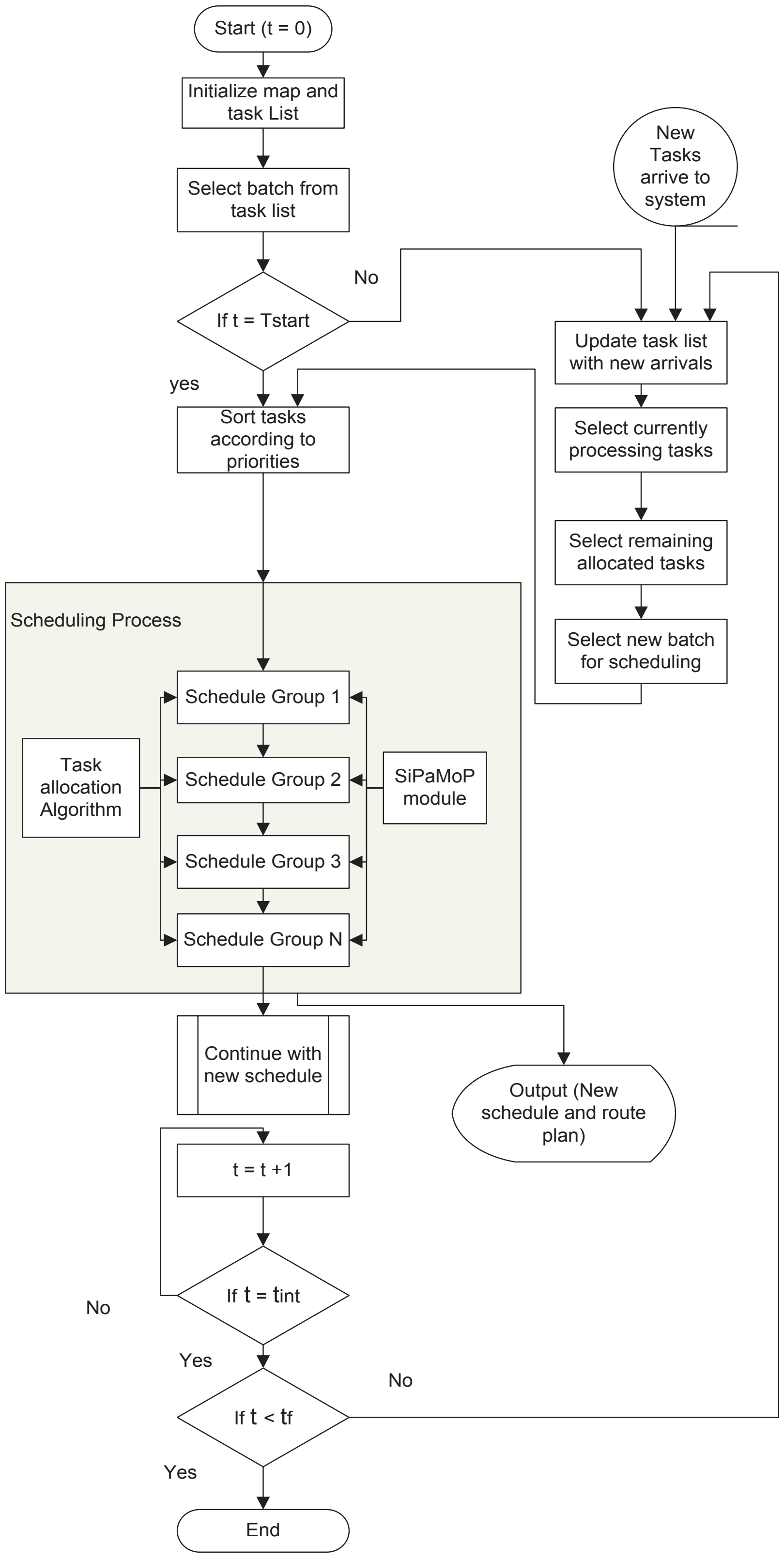
For each rescheduling interval, tardiness can be calculated based on Equation (4.10)and total tardiness as in Equation (4.11) since at each rescheduling interval one schedule (Gantt chart) will be generated and conditions governing tardiness are not differ from the conditions used for Equation (4.10) and Equation (4.11).

5.3. The Dynamic STAMC Approach

As previously stated, newly arrived tasks should be allocated for appropriate vehicles as quickly as possible. Moreover, usually at the beginning of the scheduling process, there will be a pool of tasks available for scheduling. Generally, new tasks arrive to the pool in an arbitrary manner. As stated previously, these tasks have different priorities, which have to be considered when scheduling them. In the proposed system, rescheduling is done at fixed time intervals in order to make the rescheduling problem simple. The new arrivals during a rescheduling interval are grouped together with the tasks that have not been started in the current rescheduling interval. It is assumed that the number of tasks arriving within a rescheduling has an upper bound and lower bound.

The scheduling at each rescheduling interval is done in stepwise manner based on the priority levels. In each rescheduling interval, the highest priority task group is considered first, followed by the intermediate priority task group and last the lowest priority group.

The flow chart of the proposed algorithm is given in Figure 5-2. The absolute time at the beginning of the scheduling process is taken as zero ($t = 0$).At the beginning of rescheduling, map information, available tasks and vehicle information are collected. Then, tasks are sorted, based on their priorities and expected start times. Later batches of tasks are selected from the sorted task list for scheduling. Once the allocation of all the tasks within a batch is completed, the schedule generated is released for the vehicles. This process is continued until the next rescheduling interval ($t = t_{int}$). These steps are continued until the final reschedule interval (t_f) is reached.



Start (t = 0)
Initialize map and task List
Select batch from task list
If t = Tstart
No
yes
New Tasks arrive to system
Update task list with new arrivals
Select currently processing tasks
Select remaining allocated tasks
Select new batch for scheduling
Sort tasks according to priorities
Scheduling Process
Schedule Group 1
Schedule Group 2
Schedule Group 3
Schedule Group N
Task allocation Algorithm
SiPaMoP module
Continue with new schedule
Output (New schedule and route plan)
t = t +1
If t = tint
No
Yes
If t < tf
No
Yes
End

Figure 5-2: Flow chart of the priority based dynamic STAMC approach

5.4. Simulation Studies and Results

A number of simulation studies were carried out in order to investigate the performance of the STAMC approach in a dynamic environment. In the first simulation study, the appropriate batch size (BS) and rescheduling interval (RSI) for dynamic scheduling has been investigated by varying the batch and rescheduling intervals with the solution quality by trial and error method. In the second simulation study, different task allocation algorithms have been tested with the identical task allocation problem scenarios. The main purpose of this study was to investigate the different approaches adopted in Chapter 4. The third simulation study was used to investigate the replanning capabilities due to vehicle breakdowns and path blockages.

5.4.1. Simulation study 1

Tasks with equal priorities were considered in this simulation study. The number of new tasks arriving for scheduling was set to a fixed number in order to determine suitable values for BS and RSI. The batch size varied from 12 to 24 and RSI varied from 10 (stu) to 50 (stu) at 10 (stu) steps. The number of vehicles in the simulations was set to be four and the Auction algorithm was used in task allocation. The number of tasks scheduled in each trial was varied with different RSIs and the new task arrival frequency was set to be fixed. The total tardiness was calculated based on Equation 4-10. Altogether, 12 trials were done by varying the batch size and rescheduling intervals. The makespan values, the total tardiness and the number of late tasks of each schedule were presented in Table 5-1 while the variation of these parameters was shown in Figure 5-3 to Figure 5-6.

Based on the results presented in Table 5-1, Figure 5-3 and Figure 5-4, it can be seen that the total tardiness is comparatively higher from trails 5 to 15, than those in other trials. However, it is evident that trials 11 and 12 have considerably lower makespan values, total tardiness and even a lower number of late tasks. The average and the standard deviation of the makespans are 128.80 (stu) and 16.58 (stu), respectively.

Table 5-1: Variation of rescheduling intervals with tardiness, late tasks and tasks scheduled

Trail number	Schedule interval	Batch size	Makespan	Tardiness	Late tasks
1	10	12	101.55	36.45	5
2	20	12	104.06	66.62	11
3	30	12	166.72	174.45	12
4	40	12	130.99	165.51	10
5	50	12	141.49	245.51	10
6	10	16	123.44	80.02	8
7	20	16	132.76	155.75	11
8	30	16	139.88	231.79	11
9	40	16	144.63	163.68	11
10	50	16	146.09	321.34	14
11	10	20	116.10	77.78	6
12	20	20	119.25	105.56	8
13	30	20	118.08	251.33	11
14	40	20	126.64	200.53	11
15	50	20	141.74	267.47	12
16	10	24	97.66	137.89	10
17	20	24	122.91	99.51	10
18	30	24	126.09	184.54	10
19	40	24	144.26	174.02	10
20	50	24	131.58	155.52	9

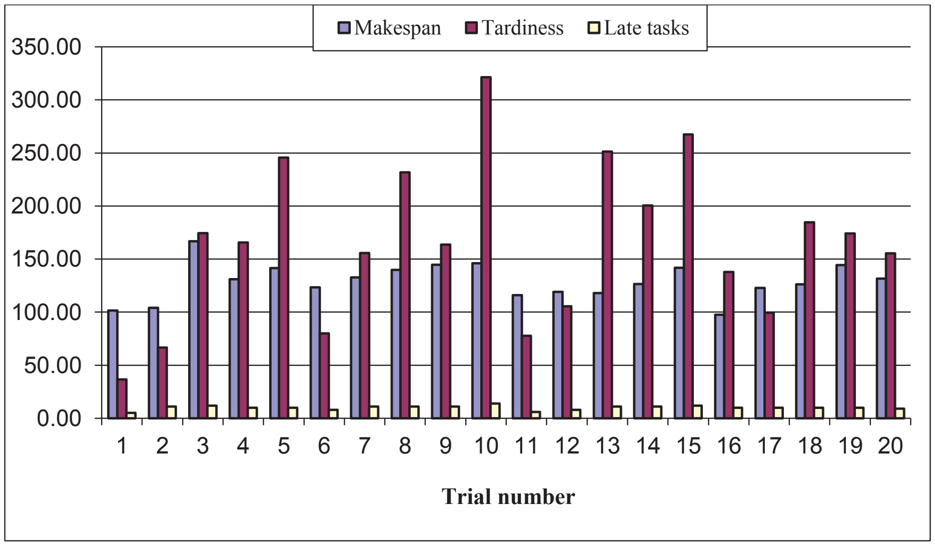


Figure 5-3: Variation of makespan, tardiness and late tasks in simulation 1

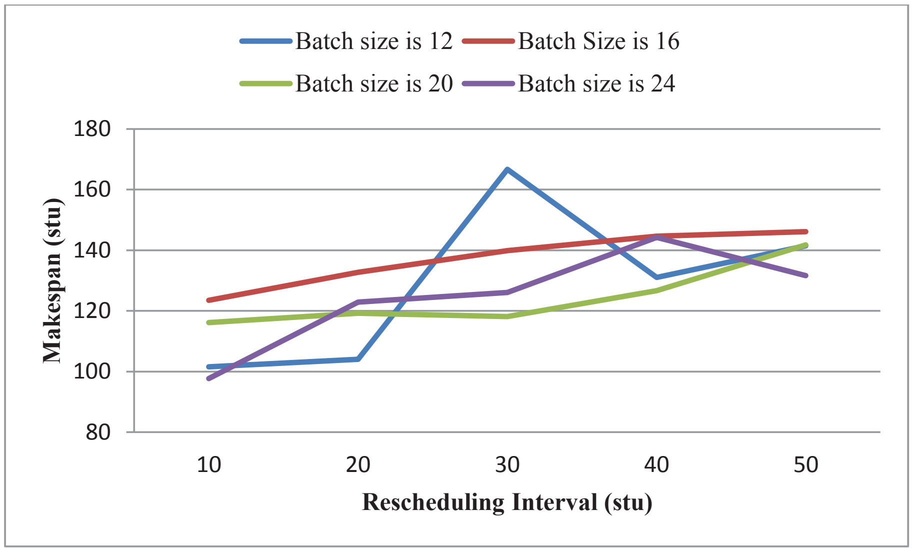


Figure 5-4: Variation of makespan with re-scheduling intervals for different batch sizes

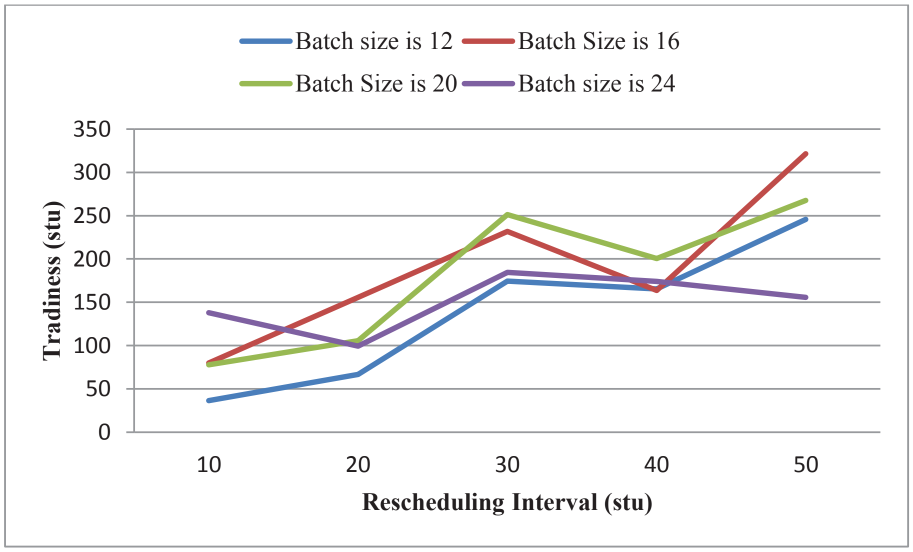


Figure 5-5: Variation of Tardiness with re-scheduling intervals for different batch sizes

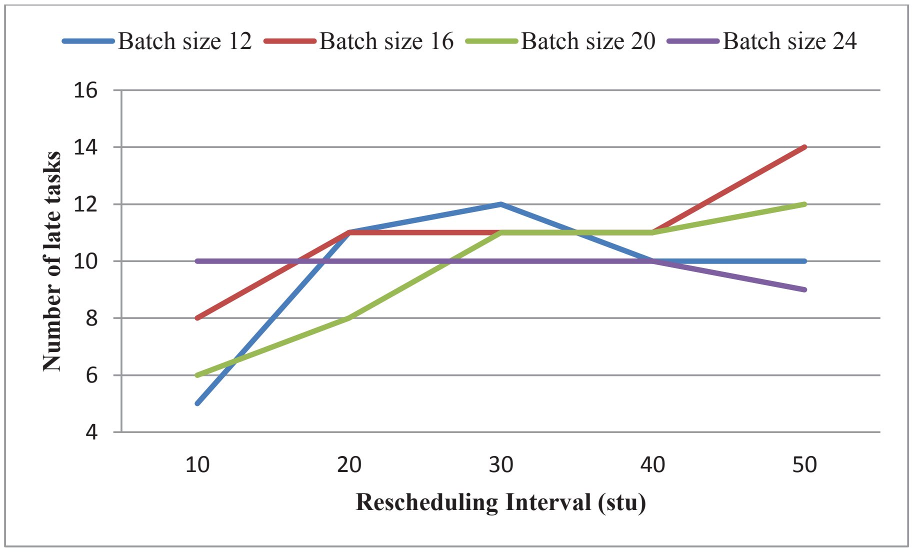


Figure 5-6: Variation of number of late tasks with rescheduling intervals for different batch sizes

The BS and RSI values were selected as 24 (tasks) and 20 (stu), respectively based on Figure 5-3 to Figure 5-6. The main reasons behind the section is that the first simulation study gave results statistically closer to the average values (Average Makespan 128.80 stu, standard deviation 16.58 stu, tardiness, and number of tasks delayed). However, these values will depend on the task allocation problem and environmental constraints such as vehicle speeds, distances of the paths and traffic congestions etc. Therefore, it would be useful to carry out different simulation studies by considering the constraints highlighted above as a future work.

5.4.2. Simulation study 2

The second simulation study was to investigate the behaviour of the STAMC approach in dynamic environment with different task allocation techniques presented in Chapter 4. Here tasks with different priorities were investigated. The SA and AA techniques were compared with general dispatching rules (DR) which performs close proximity task first (COF), and then goes to the next closest task from its current position. For example: each vehicle looks for the closest tasks out of the available tasks based on task priorities. After one vehicle completes its initially selected task, it looks for the next available task, which is closest to the vehicle’s current location. The dynamic task allocation problem was modelled in such a way that new tasks are generated in an arbitrary manner (with the assistance of random number generation in Matlab environment). These tasks were introduced to the scheduling at the same time as prior scheduled tasks were in progress. In order to schedule the new tasks, rescheduling was performed at regular time intervals. Therefore, the schedule horizon for each schedule or reschedule was the time between two RSIs. In the rescheduling a task's priority was taken into consideration. Tasks were allocated to three priority groups as highest, medium and lowest, based on the urgency to be performed. A fixed scheduling horizon was considered in the simulations. New tasks are considered at the next earliest rescheduling interval. At each rescheduling interval in the simulation, Gantt charts were generated in order to visualize the schedules generated from the three techniques, namely SA, AA and DR.

For simplicity in the simulation, if a task belongs to the highest priority group, its expected completion time was taken as 20 (stu). This means that, once a high priority task was considered for scheduling it has to be completed within 20 stu's from its available time. Similarly, for medium and lowest priority tasks, expected completion times were taken as 40 (stu) and 60 (stu) from respective tasks available time .The tardiness of each task was calculated based on the Equation 4.10 and cumulative tardiness (Equation 4.11) was used to decide the overall efficiency of the dynamic STAMC approach.

The dynamic task allocation problem was run for five re-scheduling instances starting from 0 (stu) to 100 (stu) at a scheduling interval of 20 (stu). Gantt charts of these three techniques up to 3rd RSI are shown from Figure 5-5 to Figure 5-16. Furthermore, the task time, priorities, task available time, expected completion times and tardiness obtained from each algorithm are given in Figure 5-4. In addition, each task's schedule related information are given in Appendix 5.

Figure 5-4 shows that the completion times obtained from the SA algorithm are always below the expected completion time. Hence, there is no tardiness. However, tasks 37, 38 and 39 were not considered for scheduling in the 3rd reschedule since this algorithm based task allocations could not complete many tasks as in the AA . Similar performance was shown in the AA-based solutions but tasks 37, 38 and 39 were considered in the 3rd reschedule. Meanwhile DR-based schedules, three tasks finished with some delay out of the 15 tasks arrived between the start of the scheduling and the 3rd rescheduling interval.

.

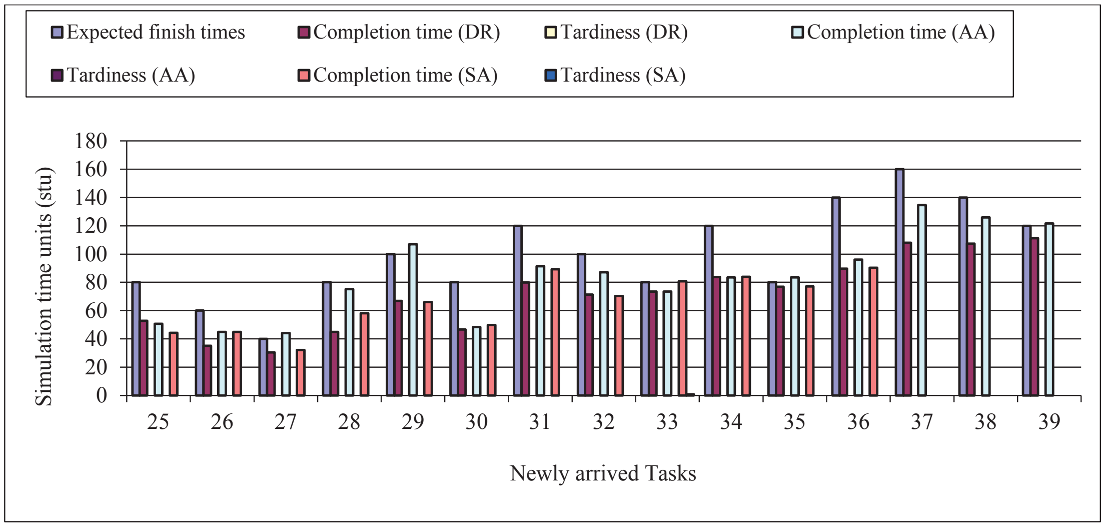


Figure 5-7: The completion time and tardiness of newly arrived tasks

The Gantt charts of schedules generated by the three task allocation techniques at the start of (0 stu) the scheduling process are shown from Figure 5-8 to Figure 5-10. In each Gantt chart, task processing times for each vehicle are shown as horizontal bars. The numerical numbers shown on each of the bars represent the task number and priority group**(task no/priority group)** it falls into: the highest priority, the intermediate priority and the lowest priority groups, which are represented by 1, 2 and 3 respectively. The priorities were considered as a dominant factor when tasks were allocated. This means that initially, highest priority tasks are allocated followed by the medium priority tasks and lastly by lowest priority tasks. There are certain tasks with lower priority which had started before higher priority tasks, for example, task 12 in Figure 5-8 or task 14 in Figure 5-9. However, if another vehicle completes these tasks, then, the task completion times or the makespan of the schedule would be higher due to the changes in task pre-processing times since pre-processing times are dependent on the initial locations of the vehicles before they undertake new tasks. The changes of task time can even grow due to the potential collisions between the initial locations of the vehicles and the pick-up locations. The task time bars shown in Gantt charts includes all four segments (time taken to reach tasks initial node, loading time, task processing time and unloading time) of task related times as a single entity. Therefore, different vehicles initiate to reach a task from different locations. The length of the bar of a selected task will vary depending on the vehicle that undertakes the task.

It can be seen from Figure 5-8, which most of the tasks were allocated to Vehicle 3 in the DR based method and only one task was allocated to Vehicle 1. The makespan obtained from the DR is almost double those obtained from other algorithms. Furthermore, the SA and Auction algorithms achieved a better tasks distribution among the vehicles compared to the DR.

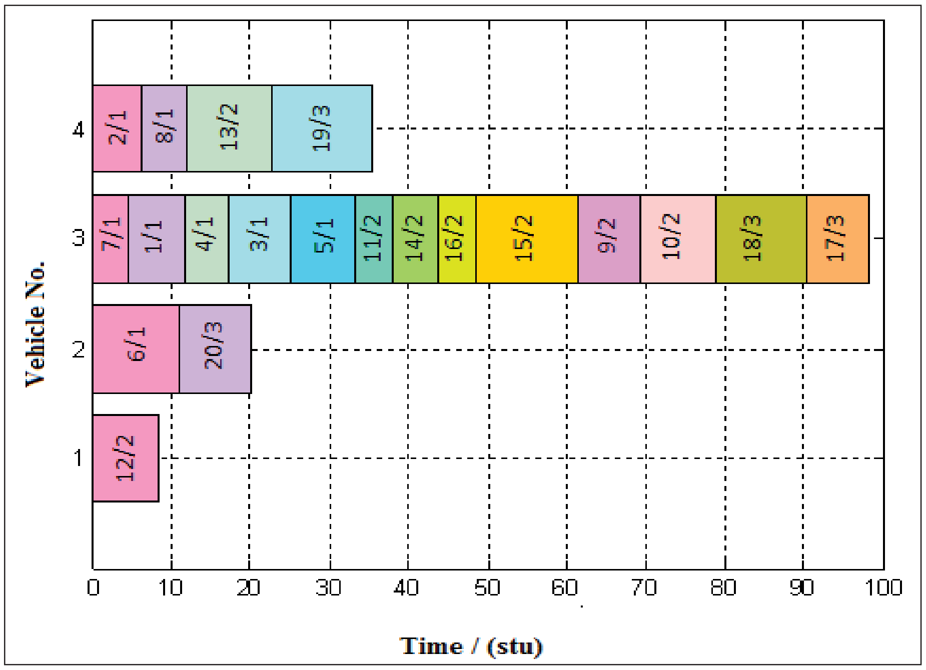


Figure 5-8: Gantt chart of the initial schedule at Time =0 (stu) based on the DR

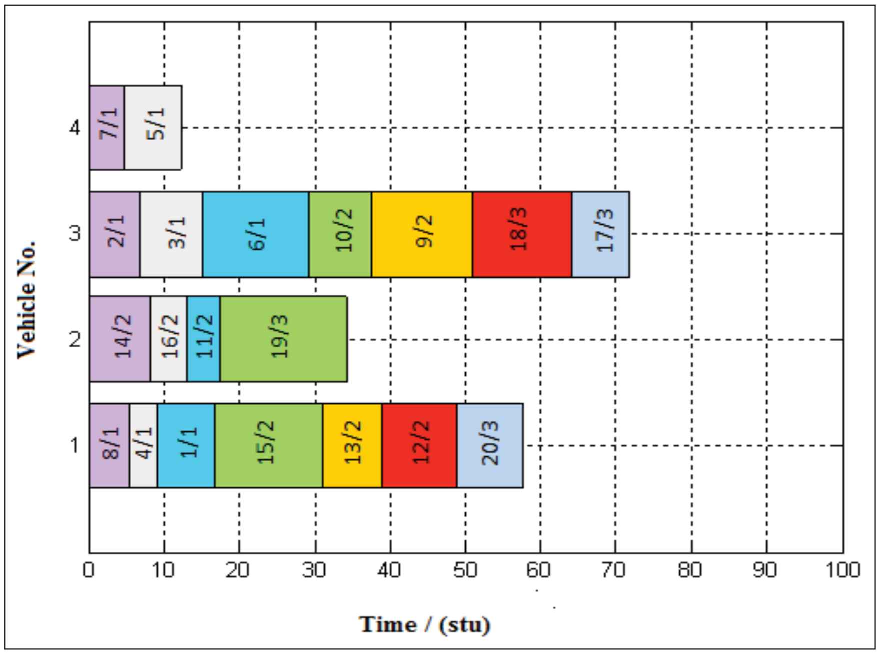


Figure 5-9: Gantt chart of initial schedule at Time = 0 (stu) based on AA

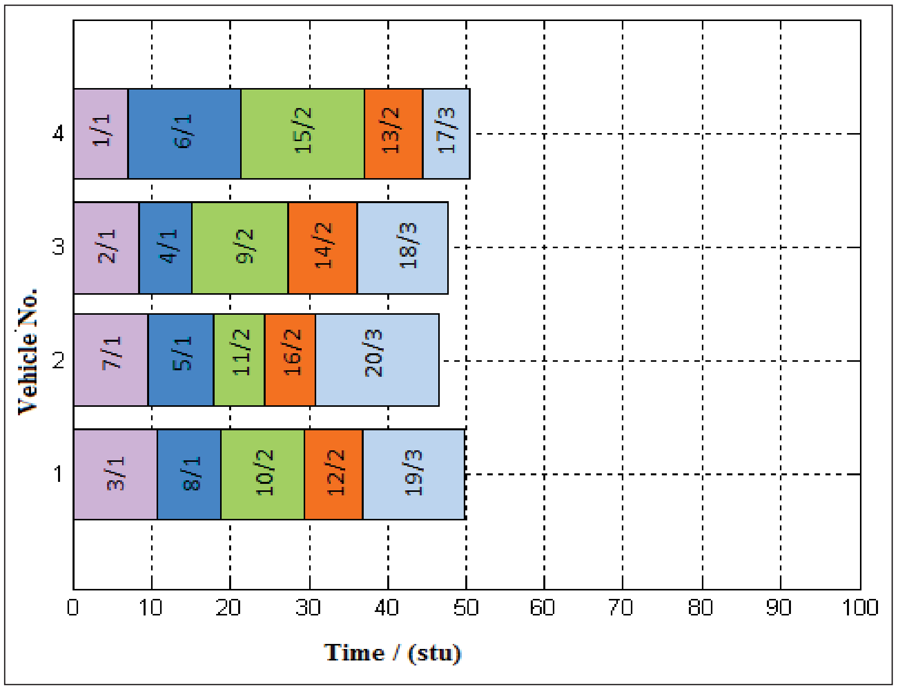


Figure 5-10: Gantt chart of initial schedule at Time = 0 (stu) based on SA algorithm

The same results can be seen in the first rescheduling interval (at 20 (stu) after the initial schedule) as in the initial schedule with regard to vehicle utilisation and task distribution among the vehicles. However, there is a slight improvement with respect to the makespan from the DR. In the case of the SA and Auction algorithms, the AA showed a good distribution of tasks among the vehicles and a shorter makespan than in the SA. The Gantt charts generated by the three task allocation algorithms are shown from Figure 5-11 to Figure 5-13.

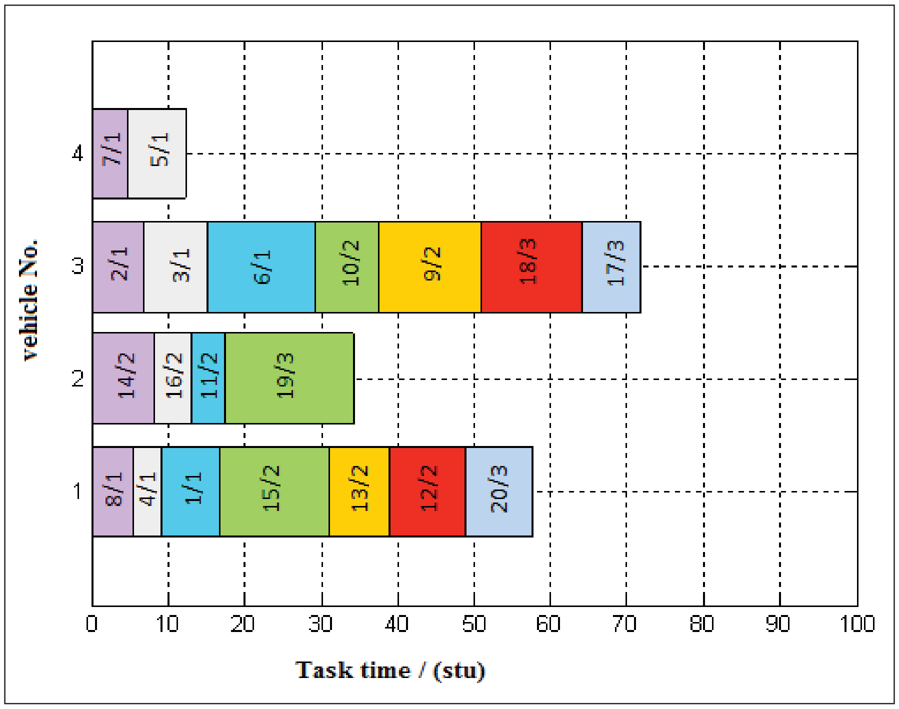


Figure 5-11: Gantt chart of 1st reschedule at Time =20 (stu) based on DR

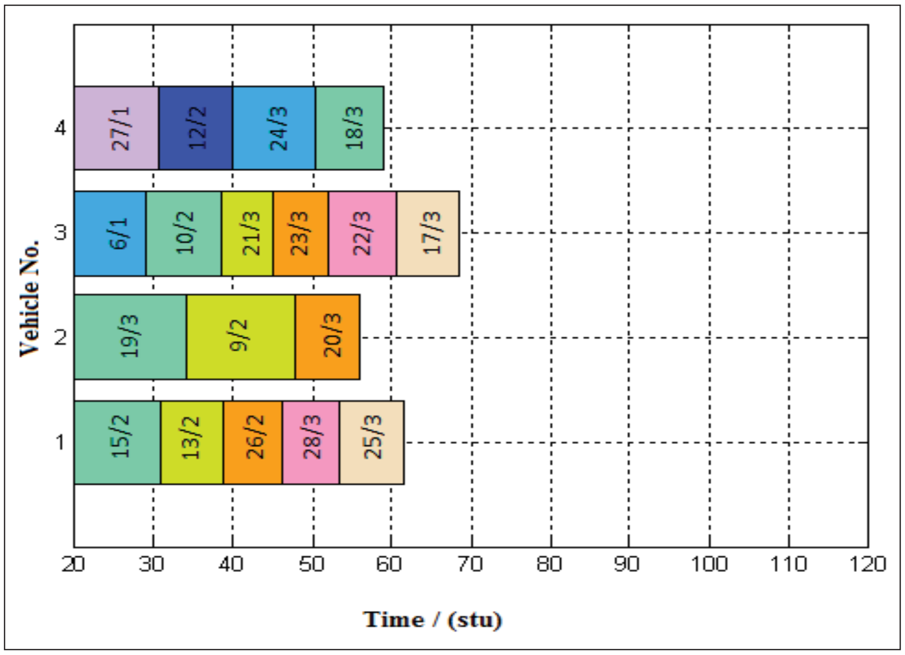


Figure 5-12: Gantt chart of 1st reschedule at Time = 20 (stu) based on AA

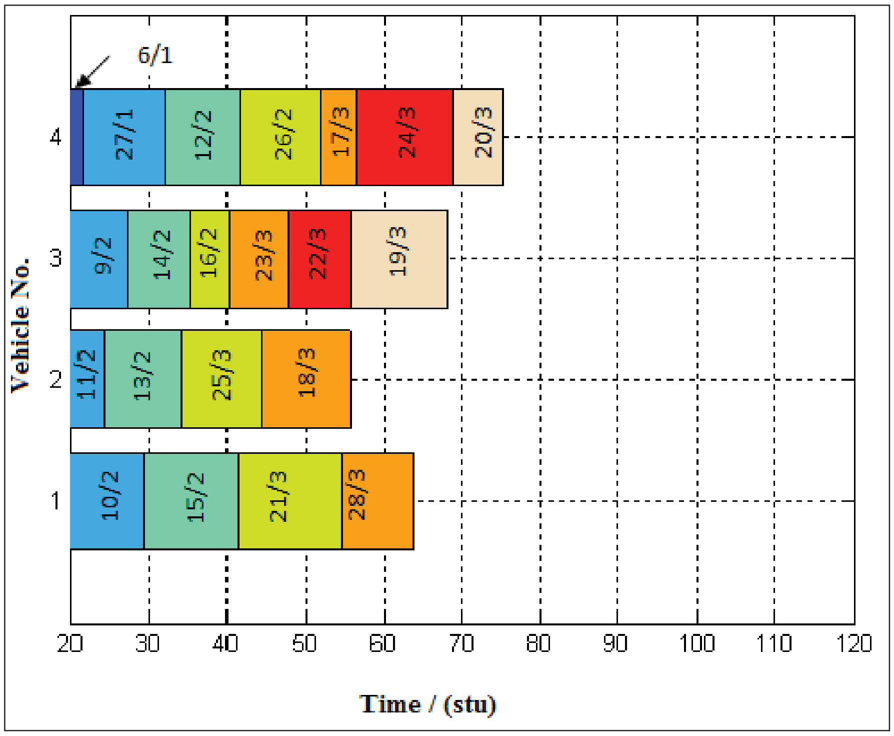


Figure 5-13: Gantt chart of 1st reschedule at Time = 20 (stu) based on SA algorithm

It can be seen that there is a slight improvement in the DR with respect to vehicle utilisation in the second reschedule interval (i.e. when t = 40 (stu)), however, even in this rescheduling the AA and SA algorithms outperformed with the minimum makespans then in the DR. Of the SA and AA, the SA has delivered shorter makespan, better vehicle utilisation and task distribution than in the AA. The Gantt charts generated at the second rescheduling interval are given in Figure 5-14 to Figure 5-16. It can be noticed from the Gantt charts shown in Figure 5-14 to Figure 5-16, the number of tasks considered for rescheduling differ from one another. This is mainly due to the fact that different task allocation algorithms generate different task-vehicle pairs at different time intervals (starting and finishing time interval of a task) and this leads to quicker task completion than in the other algorithms.

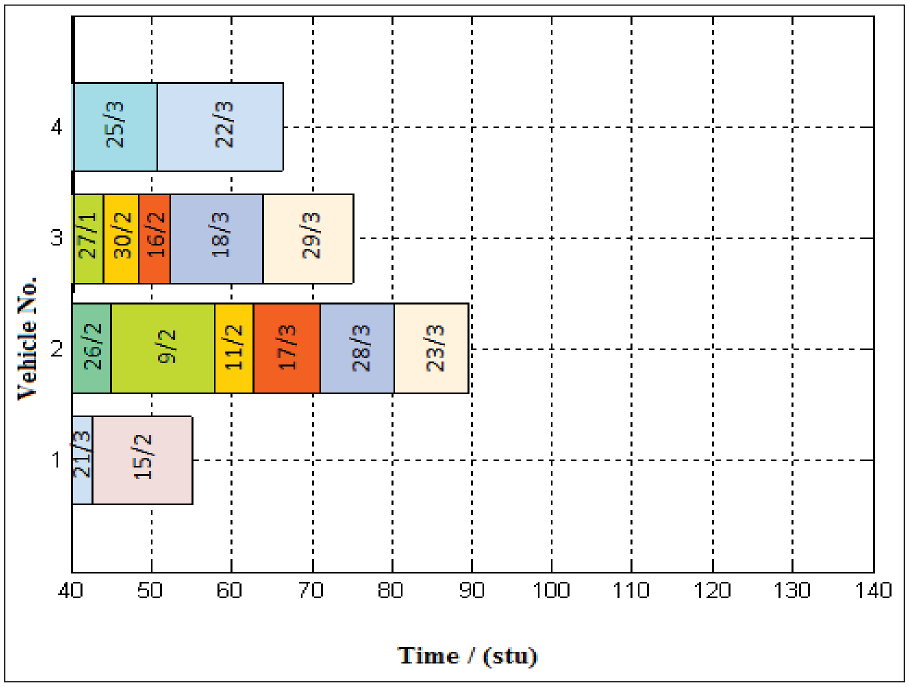


Figure 5-14: Gantt chart of 2nd reschedule at Time = 40 (stu) based on DR

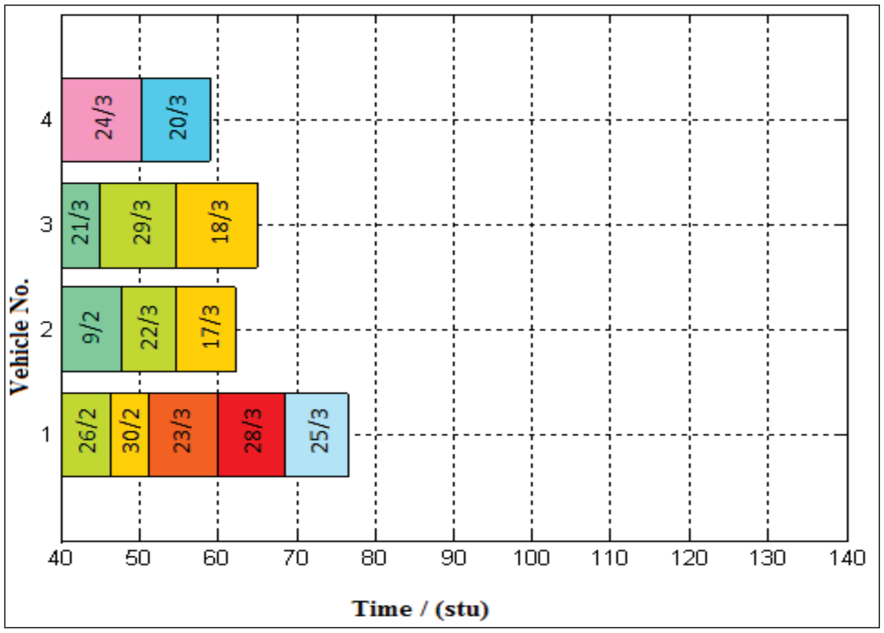


Figure 5-15: Gantt chart of 2nd reschedule at Time = 40 (stu) based on AA

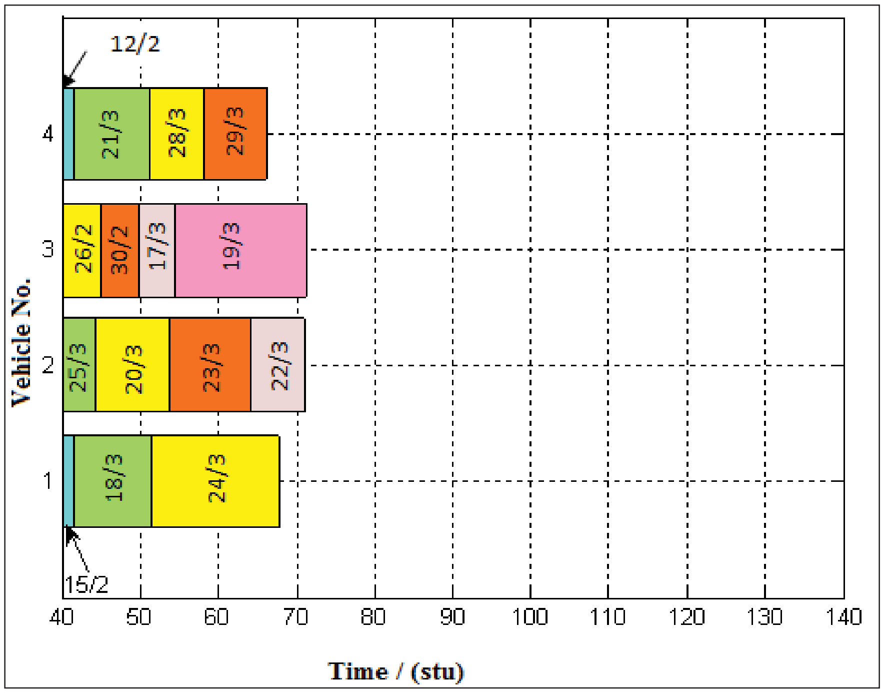


Figure 5-16: Gantt chart of 2nd reschedule at Time = 40 (stu) based on SA algorithm

From the three schedules / reschedules presented in this section in the form of Gantt charts from Figure 5-8 to Figure 5-16, it is evident that the SA and AA can generate better schedules than those of the DR. The schedule quality was determined based on vehicle utilisation and the task distribution among the vehicles and the makespan value. For vehicle utilization, the SA outperforms the AA and the DR. In the case of task distribution, the SA gave the best results when compared to the AA and DR. The DR gave the worst results with respect to the makespan value, the SA and AA generated similar results. Therefore, the SA is the best method among the three methods and DR is the worst method.

5.4.3. Re-planning Due to Unexpected Events (Simulation study 3)

In practical operation, there may be some unexpected and unavoidable instances such as vehicle breakdowns and path / route blockages, which directly affects the already generated schedules and plans. Predicting the events of this nature is difficult. There are some studies which attempted to address similar types of problems in Material handling (Narasimhan and Batta 1999, Paul and Liu 2006, Li et al., 2008) as extensively discussed in Chapter 2 and in the Introduction section of this chapter. Any approach for multiple vehicle planning and coordination should have the capability to accommodate unexpected events. The dynamic STAMC approach overcomes this issue by adopting a re-planning strategy when these unexpected events occur. This scenario is explained with a simulation study. The re-planning strategy is illustrated in Figure 5-17.

The re-planning strategy works as follows. When an unexpected event occurs, the location of the event is detected and the information is updated in the database. Accordingly, the STAMC approach will update its database and re-plan the previously allocated but not completed tasks. In this case, priority is given to the tasks which are being processed at the time of the stoppage. These tasks are assumed to be performed by the vehicles allocated before, but new routes will be planned. The rest of the tasks will be rescheduled with the remaining vehicles, based on the updated map information and operation condition.

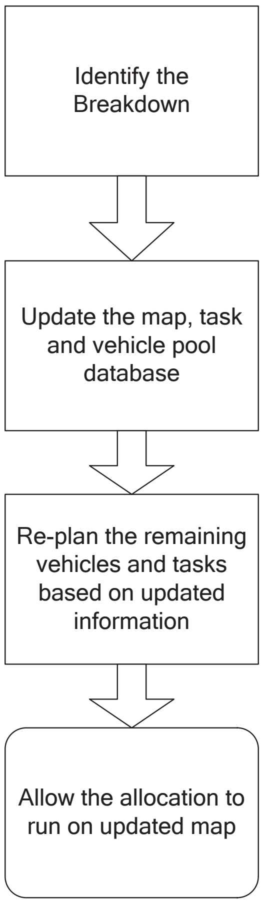

Identify the Breakdown
Update the map, task and vehicle pool database
Re-plan the remaining vehicles and tasks based on updated information
Allow the allocation to run on updated map

Figure 5-17: Schematic representation of the re-planning Strategy of the STAMC approach

In the simulation, a scenario of vehicle breakdown is randomly generated and the response to this unexpected event is explained below. Here, the vehicle number 4 broke down in the connection between nodes 98 and 79 (Figure 5-18 and Figure 5-20) at 30 (stu), after it started finishing tasks. Due to this event, the rest of the tasks allocated to other vehicles need to be re- planned. This can be seen in Figure 5-19. Vehicle 1 has two tasks that are (11 and 12) to be completed (Figure 5-20). Similarly, for vehicle 2 ,tasks no 6 and 9 are not completed, and for vehicle 3, task no 8 and task no 10 need to be completed. The Gantt charts of the schedules before and after the breakdown are given in Figure 5-18 and Figure 5-19, respectively. Furthermore, the task allocation summary before and after the breakdown is shown in Table 5-2.

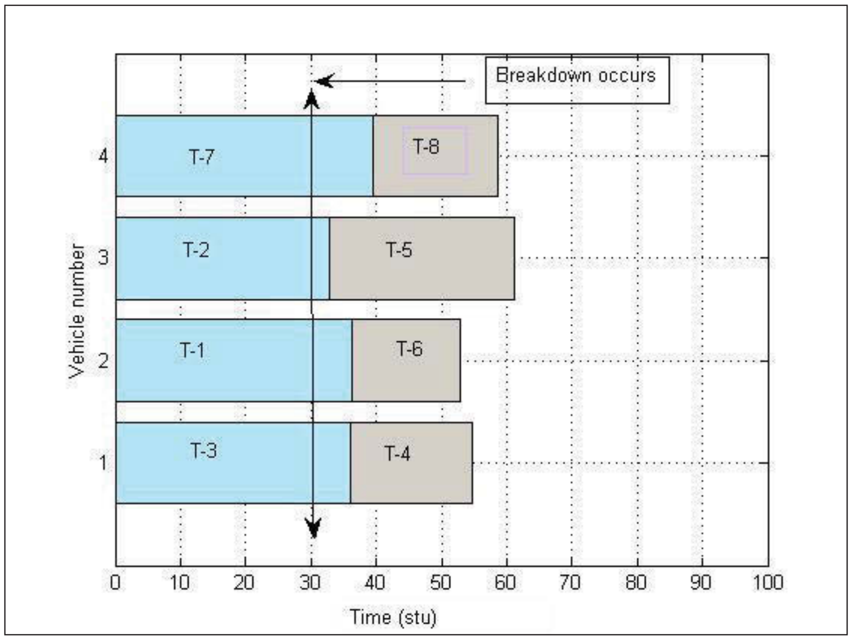


Figure 5-18: Gantt chart of the schedule before the vehicle breakdown

The information after re-planning due to the breakdown is given in Appendix 5. This information includes complete route and vehicle details before and after the breakdown. It can be seen that from 30 (stu) onwards, Vehicle 4 is not operating. However, there were none of the other vehicles already planned to use the blocked path caused by the breakdown based on the travel information shown in the tables of Appendix 5 .In addition, the travelling information of all four vehicles before the breakdown and after the breakdown have been shown Appendix 5.

It can be noted that the 4th vehicle slot in the Gantt chart in Figure 5-19 is empty in the re-planned schedule. Furthermore, in the replanning stage, new tasks waiting in the list also were considered for the allocation. Therefore, it can be seen in Figure 5-19 that Vehicle 1 has two more tasks that are new: T-11 and T-12. Similarly for Vehicle 2: T-6, T-9 and Vehicle 3: T-8, T-10.

.

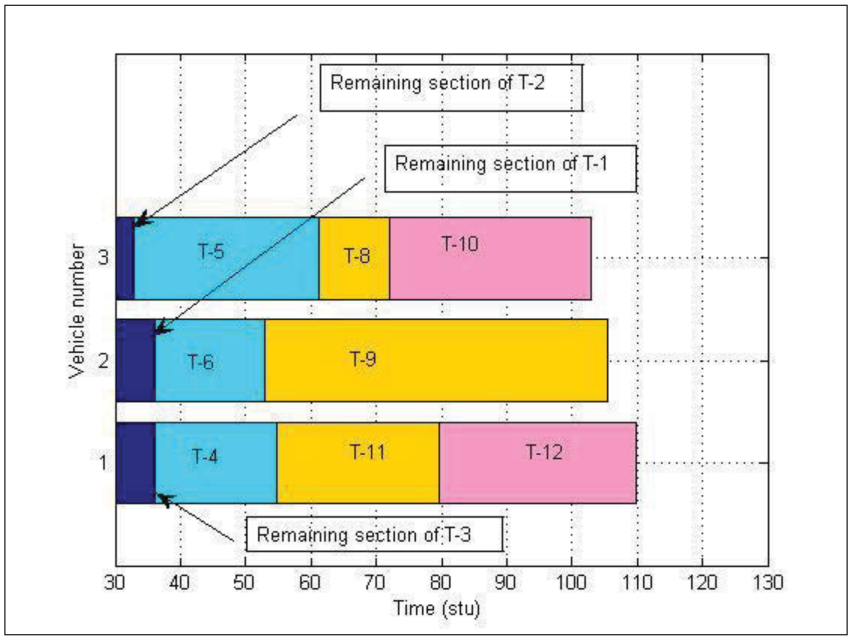


Figure 5-19: Gantt chart after re-planning of the same example

Table 5-2: Task allocation among the four vehicles before and after vehicle 4 breaks down

Vehicle	Task allocation before breakdown			Task allocation after breakdown			
	Finished	Processing	To be finished	Finished	Processing	To be finished	New Tasks
1	-	3	4	-	3	4	11, 12
2	-	1	6	-	1	6	9
3	-	2	5	-	2	5, 8	10
4	-	7	8	Broken down			-

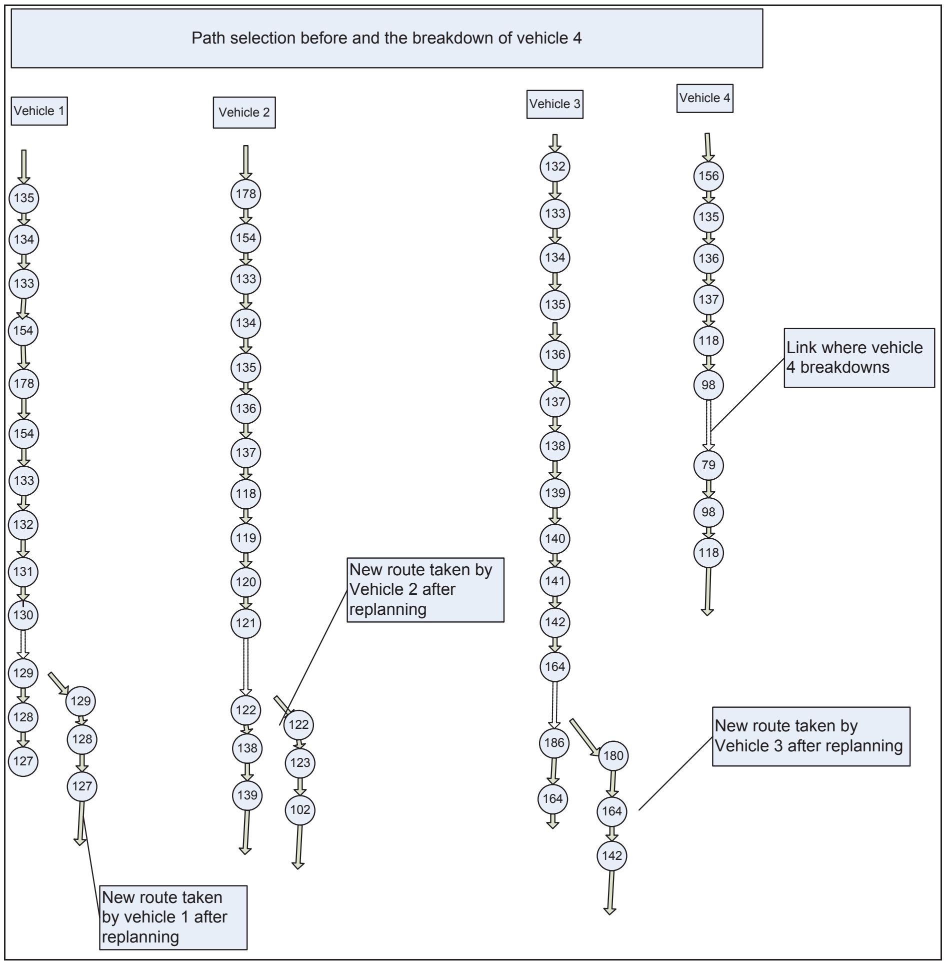

Path selection before and the breakdown of vehicle 4
Vehicle 1
Vehicle 2
Vehicle 3
Vehicle 4
135
134
133
154
178
154
133
132
131
130
129
128
127
129
128
127
178
154
133
134
135
136
137
118
119
120
121
122
138
139
122
123
102
132
133
134
135
136
137
138
139
140
141
142
164
186
164
180
164
142
156
135
136
137
118
98
79
98
118
Link where vehicle 4 breakdowns
New route taken by Vehicle 2 after replanning
New route taken by Vehicle 3 after replanning
New route taken by vehicle 1 after replanning

Figure 5-20: Path representations before and after the breakdown

Figure 5-20 shows the paths of the four vehicles before the breakdown (BD) and the remaining vehicles' paths after the breakdown of Vehicle 4. Figure 5-20 show st hat Vehicle 1 and Vehicle 3 have not changed paths even after the breakdown of Vehicle 4. Nevertheless, the path of Vehicle 2 has changedslightly after the breakdown. The dotted arrows of all four paths indicate the respective connections of the four vehicles when the breakdown occurred. It can be seen, from Table 5-2 that, task 8, allocated to Vehicle 4 before the breakdown is now allocated to Vehicle 3 since Vehicle 4 is not available due to the breakdown. Furthermore, none of the vehicles had completed any tasks as they were transporting at the time of the breakdown. All four vehicles were carrying out their initially allocated tasks before the breakdown.

5.5. Conclusions and Discussion

The focus of this chapter is to investigate the performance of the STAMC approach in dynamic environments. Two aspects are emphasised in the dynamic STAMC approach. The first aspect is to demonstrate its adaptability for rescheduling to accommodate new tasks that arrive while vehicles are operating. The second aspect is to demonstrate its re-planning capabilities in situations where vehicles breakdown, and path blockages or other unexpected events occur.

The performance of the dynamic STAMC approach was evaluated with three task allocation methods: DR, AA and SA algorithms. Of them the AA based method gave minimum tardiness and makespan for a majority of Rescheduling intervals. The SA and AA achieved similar makespans at all RSIs. In addition, the SA and AA based methods gave better performance with respect to vehicle utilisation and task distribution among the vehicles. Hence, it can be concluded from the simulation results that SA and AA can be used to solve complex scheduling and routing problems in dynamic environments.

The computational time taken to generate results in a dynamic environment plays an important role. In the dynamic task allocation and motion coordination problem, solutions were generated within a short time interval by the STAMC approach. This was achieved by fine-tuning the task allocation algorithm's parameters to generate solutions more efficiently. However, during this process, solution quality suffered to a certain extent. This was done by setting cooling rate of the annealing process to 0.6 instead of 0.9 used in Chapter 4. The upper bound of the computational time was set to 60 seconds to generate results with the STAMC approach with the hardware specified in Table 4.12 in Chapter 4.

The computational cost to generate solutions plays a crucial role in dynamic scheduling since it affects the rescheduling intervals. This will be discussed in Chapter 6. Furthermore, it revealed that the STAMC approach is capable of re-planning in unexpected situations such as vehicle breakdowns or route blockages. The proposed re-planning strategy of the STAMC approach can reschedule the remaining tasks to be performed at the time of the breakdown in addition to the re-routing of uninterrupted vehicles.

However, this approach has its limitations. It cannot reschedule the task, which was affected due to the vehicle breakdown. Practically, human intervention is needed to sort out this kind of work even in fully automated material handling systems. Furthermore, in our replanning strategy, the exact locations of the other vehicles, which are not directly affected by the breakdown, are approximated to the ending points of the respective connections they were travelling at the time of the vehicle breakdown.

Chapter 6

Distributed implementation of the STAMC approach

6.1. Introduction

As shown in the previous chapters, the simultaneous task allocation and motion coordination approach for solving the multi-vehicle task allocation and motion coordination problem is implemented in a single serial computing node with the MATLAB software computing platform. Due to the nature of the problem and its complexity, it takes considerable computation time to generate results. In order to achieve a reduction in the computation time of the STAMC approach, this chapter investigates a distributed control topology for implementing the proposed STAMC approach.

The integration and encapsulation of the Message-Passing Interface (MPI) specification into the existing MATLAB environment makes it possible to implement the STAMC approach in a distributed architecture by using the MPI Toolbox (MPITB) (Baldomero,2001).This enables coarse-grain and out-of-loop parallelisation of the expensive SiPaMoP algorithm on distributed nodes of a Linux computing cluster. The rest of this chapter is organised as follows. Section 6.2 describes the parallel/distributed architecture of the STAMC approach, Section 6.3 describes the details of parallelisation under MATLA Busing MPITB and Section 6.4 presents the description of the experiments conducted. The results followed by conclusions are given in Section 6.5.

6.2. STAMC Approach in Distributed Environment

In the distributed architecture, the motion coordination component of the STAMC approach is allowed to run on different workstations while the task allocation component is running on one workstation. That means the STAMC approach as a whole is a parallel algorithm enabling distributed computation of the expensive motion coordination sections. Furthermore, the STAMC approach is completely amenable to serial computation as described in Section 6.5 for the purpose of empirical comparison.

The auction algorithm is selected for the task allocation purposes in this chapter to explain the distributed architecture. However, other meta-heuristic algorithms can also be adopted for the task allocation purpose with the distributed architecture of the STAMC approach. The auctioning procedure used for the task allocation is presented in Section 4.4.3 and illustrated in Figure 4-9 of chapter 4. The remainder of the section presents how it is being modified in the distributed manner.

Initially, the task sequence is generated randomly by the task generator in the master workstation, which is responsible mainly for the task allocation phase of the STAMC approach. The first task is broadcast to all autonomous vehicles allowing them to place a future bid for the task by the master workstation. Each vehicle calculates their respective bids based on the collision-free paths generated by the motion coordination component of the STAMC approach in their own workstations and announces it back to the master workstation. After all the vehicles have returned their bids, a winner is determined and is allocated the first task by the master workstation. The second task is then broadcast, followed by bids from each vehicle, and a winner is selected. This auction process of broadcast task, followed by bidding and the selection of a winner continues until all tasks have been allocated.

For each vehicle, the calculation of bids is based on the travelling time to complete the current broadcasted task and any previously allocated tasks. Once the travel time has been calculated, it is used to post bids ($B_{1,j}$ to $B_{i,j}$). The travelling time of collision-free paths is calculated using the SiPaMoP algorithm. A winner is then determined, based on the lowest travel time to finish the respective task (completion time). When calculating the completion time of a task for each autonomous vehicle, the previous task commitments of each vehicle are also considered. This will help to reduce the chance of allocating too many tasks to one vehicle and therefore balance the usage of vehicles. For example, if a previous task is allocated to a particular vehicle, then there will be fewer tendencies for the same vehicle to win the next task. In addition, load balancing of the vehicles can be achieved partially. After all tasks of the current task sequence are allocated, the makespan is calculated. This process continues for a fixed number of cycles with a different task sequence generated randomly in each cycle. Eventually, the best task sequence is selected, which provides the minimum makespan. The flow diagram of the simultaneous task allocation and path-planning algorithm is illustrated in Figure 6-1. The parallel computation of the SiPaMoP algorithm (motion coordination aspects) occurs towards the middle of the flow diagram. It is during this stage that the motion and path is calculated for each vehicle independently using different computing nodes.

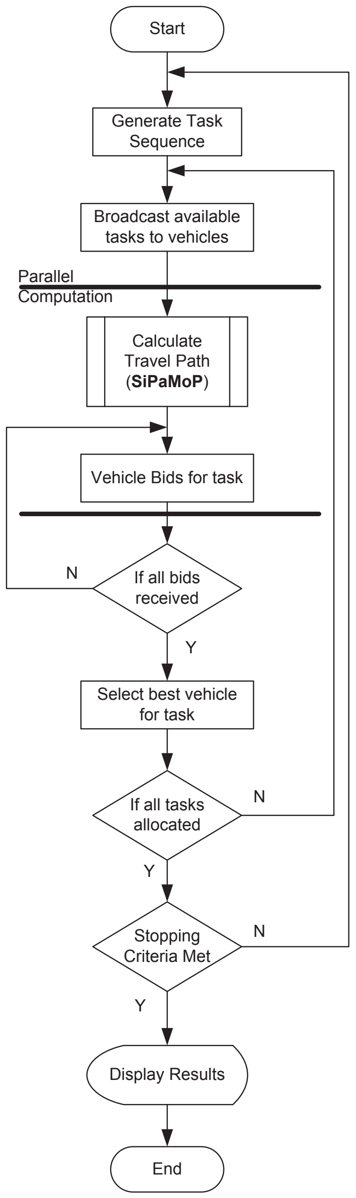


Figure 6-1: Flow diagram describing the simultaneous task allocation and motion coordination (STAMC) approach in parallel mode

The travel path and resulting travel time to complete the path is calculated using the SiPaMoP algorithm. This portion of the STAMC approach is distributed and computed in parallel for each autonomous vehicle. If there exists *n* autonomous vehicles requiring computation of a travel path, which takes t_{path} time to compute, the serial STAMC approach requires nt_{path} time to compute all paths for all vehicles. The parallel STAMC approach requires only t_{path} time to compute the travel path for all autonomous vehicles.

A Master-Worker topology is used to distribute the STAMC approach. The Master node takes on the role of 'Auctioneer', by issuing tasks, receiving bids for tasks and determining the winning bid from each worker node. The worker nodes perform parallel computation of the path and motion planning using the SiPaMoP algorithm. The distributed computing environment and master-worker topology introduce a necessary communication time (t_{comm}), but $t_{comm} << t_{path}$.

Partitioning of the STAMC approach onto a parallel computing architecture is illustrated in Figure 6-3 and the serial architecture is illustrated in Figure 6-2.

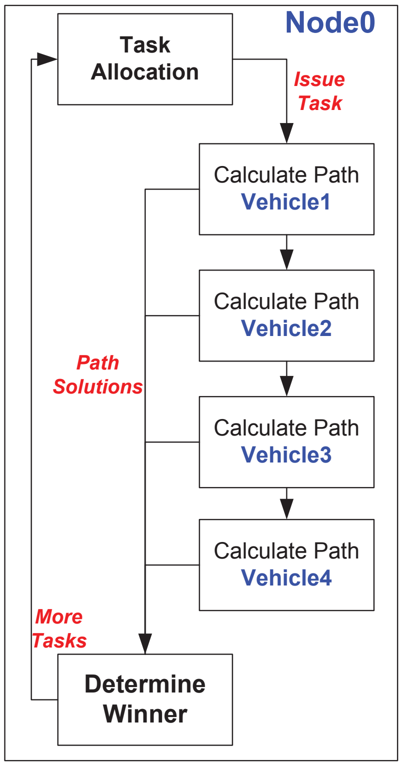


Figure 6-2 : The data path of serial computation for the task allocation and motion coordination algorithm for autonomous vehicles

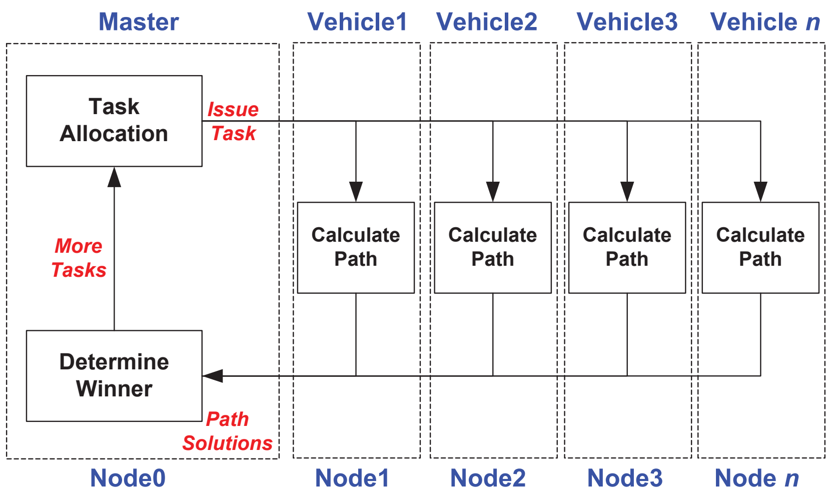


Figure 6-3 : The data path of parallel computation for the task allocation and motion coordination algorithm for autonomous vehicles.

In this chapter, the STAMC approach is implemented on a distributed memory computing system known as a compute cluster, which is discussed further in Section 6.4.2. The combined arrangement of hardware and software of the compute cluster provides a platform for coarse-grained parallelisation of the complete SiPaMoP algorithm on each of its computing nodes as illustrated in Section 6.2.

The cluster computing architecture and master-slave topology mutually provide for two possible mappings between the number of computing nodes and number of vehicles requiring motion and path planning. The first mapping is 1:1 and is used exclusively in this paper. Here, a single node computes the motion and path for a single vehicle. The second mapping, 1:*n* allows a single node to compute the motion and path for multiple vehicles. For example, if there exist three computing nodes and nine vehicles, only three vehicles can be computed in parallel at one instance. As a result, three groups would be computed *sequentially*. In this case a single node would perform the motion and path computation for three vehicles, a 1:3 mapping. A third mapping of *n*:1 allows the motion and path computation to be further distributed across *spare* nodes. For example, if there exist nine computing nodes and three vehicles it would be ideal to further distribute the path and motion computation for each vehicle across the spare six nodes, thus allocating all computational resources of the compute cluster to the calculations. The STAMC approach encapsulates the complete SiPaMoP algorithm into a coarse-grained implementation, preventing any further decomposition into finer-grained portions for execution on separate processors.

6.3. Integration of MPITB in the MATLAB Environment

The STAMC approach is implemented in the interpretive MATLAB environment, which has no native support for distributed computing. In order to arrange the STAMC approach into a Master-Worker topology on a distributed computing architecture, a Message-Passing Interface (MPI) was required.

The integration of the Message-Passing Interface specification (MPI.Forum, 2005) and the interpretive MATLAB environment allows researchers to achieve coarse-grained and out-of-loop parallelisation of scientific and engineering applications developed in MATLAB. Developed at the University of Granada in Spain, MPITB for MATLAB allows researchers to include MPI function calls in a MATLAB application, in a way similar to the bindings offered for C, C++ and Fortran. Decomposition and coding of a serial problem into a parallel problem is still the responsibility of the researcher, as there is no *automatic* parallelisation method in MPITB. This method of explicit parallelisation coupled with user knowledge of the application provides a good chance for sub-linear or linear computational speedup. Furthermore, the onus is placed upon the researcher to develop *safe* distributed code free of livelocks and deadlocks, which occur due to the loss of message synchronisation between distributed computing nodes.

Baldomero, (2001) provides a summary of several other parallel libraries that achieve coarse-grain parallelisation for MATLAB applications. The toolboxes differ in the number of commands implemented from the MPI specification and the level of integration with existing MATLAB data types.

The level of computational performance provided to a parallel MATLAB application is dependent upon the underlying communication method implemented in the toolbox. Toolboxes using the file system to exchange messages between computers tend to be slow due to the explicit read/write latency of rotating hard disks. Conversely, toolboxes using a message-passing daemon to provide communication between computers provide much better performance due to the small latencies and large bandwidth capabilities of local area networks (LANs). In the latter case, messages (data) are transferred between the primary memories of parallel computers, without buffering them using the file system prior to transmission over the LAN.

The design of MPITB includes nearly all functions defined in the MPI-2 specification. The MPI functions are written in C source code and dynamically compiled into MATLAB MEX-files. The MEX-files encapsulate the functionality of the message-passing routines, allowing them to be directly called within the MATLAB environment, thus making the parallelisation of the application possible. With both MATLAB and a message-passing library installed, such as LAM-MPI, the precompiled MEX-files can perform both MATLAB API calls and message-passing calls from within the MATLAB environment as illustrated in Figure 6-4.

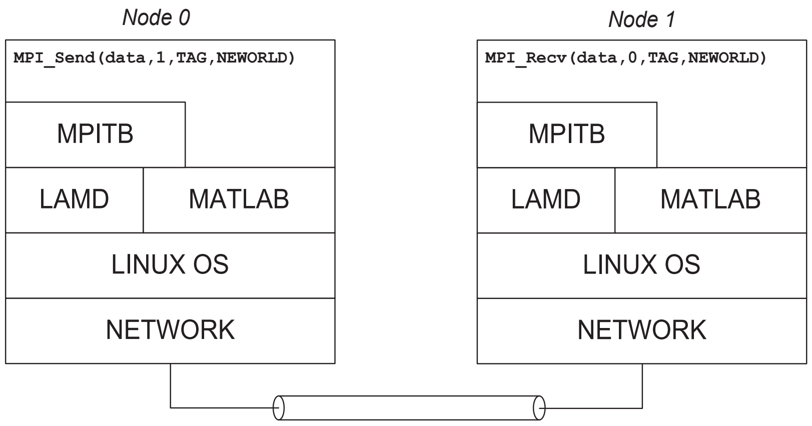


Figure 6-4 : Software architecture showing the role of MPITB and other software components.

The MPITB makes MPI calls to the LAM-MPI daemon and Matlab API. This method enables message-passing between Matlab processes executed in distributed computing nodes (Baldomero, 2001). Transmission of data between the master-worker MATLAB processes and execution of the STAMC approach can occur after booting and initialisation of the LAM-MPI library from the master process using:

LAM_Init(nworkers,rpi,hosts)

Where ***nworkers***is the number of cluster nodes designated as worker nodes, ***rpi*** is the LAM MPI SSI setting which is set to either ***tcp*** or ***lamd*** in our experiments, *hosts* is the list of host names on the Linux cluster. Once the underlying MPI library has been initialised, MATLAB instances must be spawned on worker nodes. This is achieved using the following command on the master process:

```
MPI_Comm_spawn('matnojvm',args,nworkers,NULL,0,MPI_COMM_SELF)
```

where *matnojvm* is a script containing the *matlab* UNIX command with the *–nosplash* and *–nojvm* inline switches. The switches disable the splash screen and JAVA virtual machine for headless operation of distributed MATLAB processes on the master node. The *nworkers* parameter is the number of cluster nodes designated as worker nodes, and the remaining are typical MPI parameters.

Finally, establishing an MPI communication domain, called a *communicator*,defines a set of processes that can be contacted. This is done using the following commands on the master process:

```
MPI_Comm_remote_size(processrank)
MPI_Intercomm_merge(processrank,0)
global NEWORLD
```

where *processrank* is an integer greater than zero assigned to each worker node in the MPI communicator; the master node has a processrank equal to zero. The global variable *NEWORLD* is the name of the MPI communicator. Transmission of messages between MATLAB processes can now be accomplished, permitting the arrangement of cluster nodes into any useful topology. The master-worker topology used in the experiments employs the fundamental point-to-point communication mechanism between master and worker nodes, with one side *sending*, and the other, *receiving*. The following commands perform the standard-mode blocking send and receive operations:

```
MPI_Send(buf,processrank,TAG,NEWORLD)
MPI_Recv(buf,processrank,TAG,NEWORLD)
```

where, *processrank* is the MPI rank assigned to each MATLAB process. When messages are sent *processrank* is the MPI rank value of the receiving process and when messages are received *processrank* is the value of the sending process. The parameter *buf* represents the MATLAB data to be sent or received, *TAG* is an integer associated with the message providing selectivity at the receiving node, and *NEWORLD* is the MPITB communicator.

Any valid MATLAB data type can be transmitted directly without being pre-packed into a temporary buffer, unless the message contains different data types. If the data to be sent is larger than a single variable, such as a matrix, then its size must be determined and sent prior to sending the matrix. The approach taken in this paper, is to calculate the size of the matrix in the sender using the MATLAB*size()* command, then send the size value to the receiver prior to sending the actual matrix, as illustrated by the following pseudo code:

Master pseudo code:

```
 numrows=sizeof(uwvpmat)
 for all vehicles(n)
   MPI_Send(numrows,vehicle(n),TAG,NEWORLD)
   MPI_Send(uwvpmat,vehicle(n),TAG,NEWORLD)
```

Vehicle (worker) pseudo code:

```
 MPI_Recv(numrows,master,TAG,NEWORLD)
 uwvpmatrix=zeros(numrows,numcols)
 MPI_Recv(uwvpmatrix,master,TAG,NEWORLD)
```

Another method uses the *MPI_Probe()* command to probe for incoming messages on the receiving side, assert the size of the incoming message, create a buffer of the corresponding size to receive the message and then receive the actual message in the buffer. A complete tutorial written by Sebastien Goasguen is available at http://falcon.ecn.purdue.edu:8080/cluster/.

The distribution and parallel computation of the STAMC approach requires the transmission of data between master and worker nodes. The size of the PATH_REGISTER matrix is dynamic between each task allocation cycle of the SiPaMoP algorithm. Pseudo code describing the parallel and distributed approach used for parallel task allocation and path planning is given in Appendix 6.

6.4. Experiment Description

The experiments involve execution of the task allocation and path-planning algorithm on a single serial computing node and in parallel on the distributed computing cluster. The serial computation uses a single node of the Linux cluster, whereas the parallel computation uses multiple cluster nodes.

The computation time for algorithms is often a measure of the number of objective function evaluations, because different algorithms may be compared regardless of their particular implementation. However, this often disregards communication times for parallel implementations of the same algorithm. In this study, the performance was measured by recording the wall-clock time, so all components of the execution time, including communications, are included. The wall-clock time is a fair measure of performance that is frequently used. Furthermore, the establishment of the MPI topology under MATLAB is not a part of the task allocation and path-planning algorithm and is not included in the computation times presented in this chapter.

6.4.1. Simulation Parameters

The set of algorithm and simulation parameters remained constant to encourage a meaningful empirical comparison between the parallel/distributed approach and the serial/centralised method. Table 6-1 presents the settings of the SiPaMoP algorithm and simulation parameters

Table 6-1 : Algorithm and simulation parameters

Parameter Name	Parameter Value
Num Master	1
Num Vehicles	4, 6 ,8
Num Tasks	24, 48, 72, 96, 120, 144, 168, 192, 216, 240
Num Map Nodes	187
Weights Update	Dynamic
Vehicle speed	100 cm/stu
scheduling time	1 batch
vturningspeed	1c m/stu
Safety time	0 sec
Turning rate	1.0
Number of cycles	25

In order to investigate the performance of the distributed implementation of the STAMC approach, four different sets of simulations were performed, The number of allocated tasks varied from 24 to 240 in steps of 24, while the number of vehicles remained fixed at either 4, 6, or 8 for each simulation respectively. The STAMC approach was executed for 25 cycles with the average computation time taken as the final result.

6.4.2. Cluster Computing Environment

All experiments were performed on a Linux computing cluster with specifications provided in Table 6-2. The serial and parallel versions of the task allocation and path planning algorithm were coded in MATLAB. Communications between cluster nodes employing the Message Passing Interface ver2.0 were implemented using the MPITB (Baldomero, 2001).

Table 6-2 : Cluster computing hardware and software environment

Computing Environment Component	Description
Number of Nodes Utilised	1,4,8
Processor Type and core Speed	Pentium 4 Hyper threading @ 3.0Ghz (Prescott)
Front-side Bus Bandwidth	800MHz
DRAM capacity and bandwidth	2GB DDR @ 400MHz
Network Type and Bandwidth	1000Mbps Ethernet
Network Switching Type	Gigabit Switching Fabric
Network Protocol	TCP/IP V4
OS Kernel Type and Version	Linux (2.4.21-20.EL)
MPI Type and Version	LAM 7.1.1 / MPI 2
MATLAB	7.0.4.352 (R14) Service Pack 2
MPITB	mpitb-FC3-R14SP1-LAM711.tgz

6.5. Results and Discussion

The average computation times for the parallel/distributed STAMC approach and the serial/centralised STAMC approach are illustrated from Figure 6-5 to Figure 6-8.

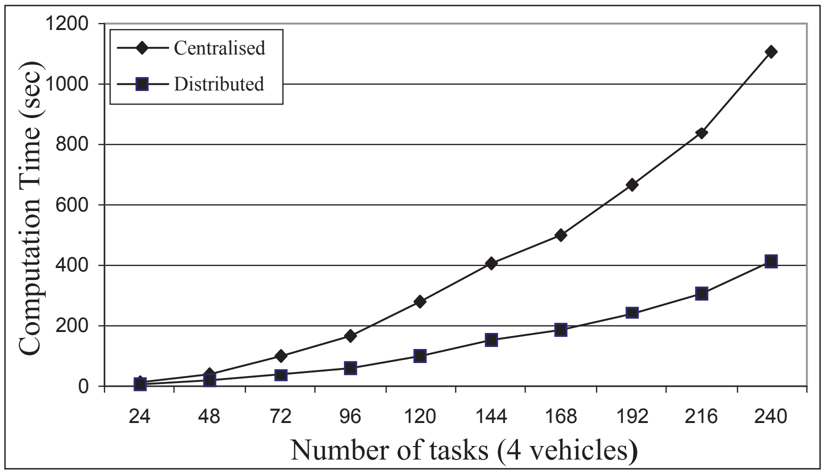


Figure 6-5 : Computation time (seconds) for the parallel/distributed and serial/centralised STAMC approach using 4 vehicles

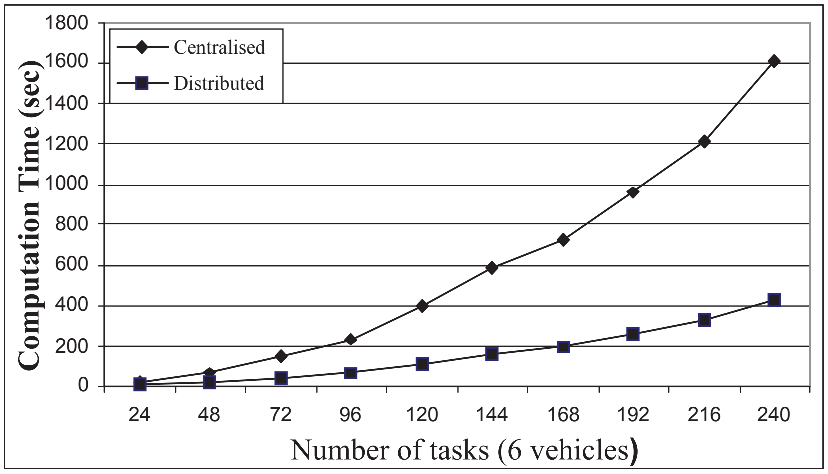


Figure 6-6 : Computation time (seconds) for the parallel/distributed and serial/centralised STAMC approach using 6 vehicles

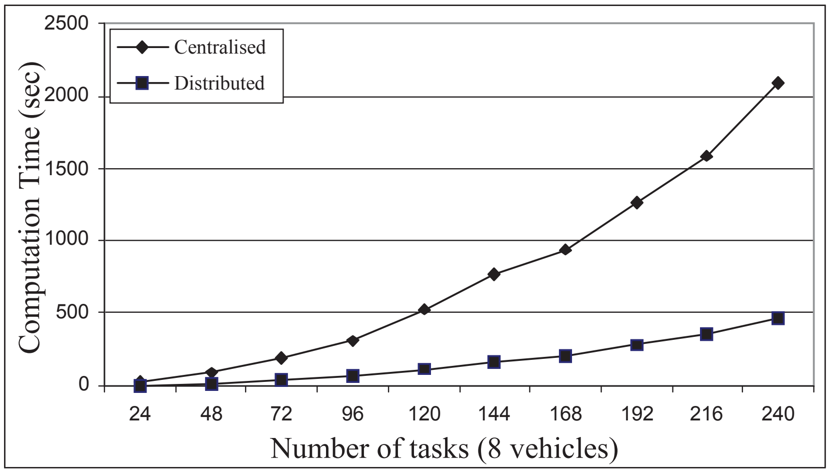


Figure 6-7 : Computation time (seconds) for the parallel/distributed and serial/centralised STAMC approach using 8 vehicles

From the three simulations using 4, 6, and 8 vehicles, there is a clear performance increase (less computation time) with the parallel/distributed STAMC approach compared to the serial/centralised STAMC approach. As the numbers of tasks are increased from 24 to 240, the difference in performance becomes even more significant between the two versions of the STAMC approach. In general, when the number of tasks of the simulation study increases, the computation time increases exponentially, but the rate of change of the gradient in the serial/centralised algorithm is much larger than the rate of change of the gradient in the parallel/distributed algorithm, because the STAMC approach is *evenly* distributed among cluster processors (1:1 mapping) and computed in *parallel* in the latter case.

Figure 6-8 illustrates the results of the parallel and distributed STAMC approach using 4, 6 and 8 vehicles. Results for the single vehicle situation are also given to provide a baseline for comparison against multi-vehicle simulations. For the multi-vehicle simulations, the computation time is similar from 24 to 240 tasks, with a maximum variation of approximately 14.3% between 8 and 4 vehicles at 216 tasks. This suggests a useful scalability property of the parallel and distributed STAMC approach arranged in a master-worker topology for an increasing number of vehicles.

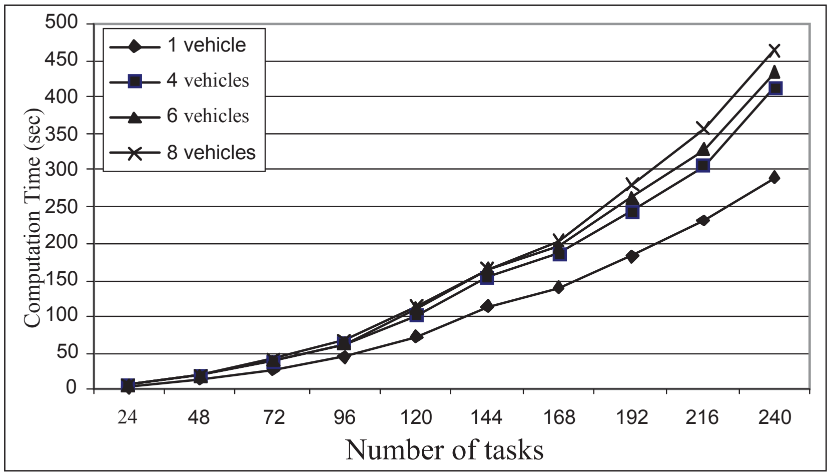


Figure 6-8 : Computation time (seconds) for the parallel/distributed STAMC approach using 4/6/8 vehicles

The STAMC approach attempts to find an optimal (minimum) schedule for the allocation of *all* tasks to available vehicles, whilst guaranteeing collision-free paths. This is a typical NP-hard problem, requiring time, which is exponential in log *n*, the number of tasks to be scheduled. As a consequence, the results are exponential in the number of tasks to be scheduled and not the number of vehicles. Because of the coarse-grained parallelisation of the SiPaMoP algorithm, variations between 4, 6, and 8 vehicles is small, even with 240 tasks as illustrated in Figure 6-8.

6.6. Conclusion and Further Investigations

The use of MPITB for MATLAB provides effective integration and encapsulation of a message-passing system like MPI into the interpretive environment of MATLAB. This enables the existing MATLAB code to be distributed and computed in parallel using clustered computing power. Scientific and engineering applications continue to maintain the interactive, debugging and graphics capabilities offered by the MATLAB environment, and can now reduce the computation time by taking advantage of clustered computing.

Due to the high granularity of the STAMC approach, the distributed and parallel versions achieved near-linear computational speedup over the serial STAMC approach. This result was achieved using 4, 6 and 8 cluster nodes for a number of tasks ranging from 24 to 240. The results also suggest good scalability of the parallel

STAMC approach, which becomes more important as the number of vehicles and number of tasks increase.

The simulation results show that by adapting the distributed architecture of the computation, the computational time taken by the STAMC approach can be reduced considerably. This can be even further improved with the introduction of more computers, or use of high performance computers. Furthermore, overall system reliability can be increased by the retirement of the master node to remove the single point of failure from the system and coupled with a fully-connected topology where redundancy is increased by allowing any cluster-computing node to adopt the master role of task allocation. In the practical scenarios with automated material handling systems, this can be achieved by allocating individual workstations' computations among the autonomous vehicles while maintaining one centralized workstation to coordinate all the activities. The outcomes of this chapter have already been published Kulatunga et.al.,(2007).

Chapter 7

A Case Study -Application of the STAMC Approach in an Automated Container Terminal

7.1. Introduction

So far, the proposed STAMC approach was tested in a scaled down, indoor material-handling environment where the number of nodes was 189 and the total area covered was 15m X 30m. However, in many real world material handling systems, especially outdoor environments such as container terminals, the number of tasks to be handled at a given time interval, and the route network, the staking / storage areas are comparatively large. Further, the operational area of container terminals is spread in a large area and the overall footprint of the material handling systems will be higher than indoor environments. Therefore, it is essential to validate the applicability of the proposed STAMC approach in a scaled up operational environment.

The Patrick Autostrad Container terminal was selected for the validation purpose. This terminal is located at Fisherman Island, Brisbane, Australia that covers the land area of about 40 hectares and owned by Patrick Corporation, a leader in freight transportation in Australia. This is known as the first fully automated container terminal on Australian soil and currently possesses a fleet of 25 automated straddle carriers and 12 Quay cranes and 3 forklifts with terminal yard capacity of 1.2M TEUs (http://www.patrick.com.au) . A snapshot of the terminal is shown in Figure 7-1.

Figure 7-1: Container terminal at FishermanIsland (http://www.patrick.com.au)

7.2. Representation of the Automated Container Terminal

The container yard is divided into different sections: stacking areas, exchange areas and travelling zones. The stacking areas are categorised into two groups where normal containers stack in ***A, B, C, D, X, Y*** and ***Z*** zones while reefer containers stack at zone ***re***. The travelling areas next to the crane operating area is represented as ***r1*** and other common road areas as ***r2*** and ***r3***. These sections can be compared with the aerial view of the terminal is shown in Figure 7-2 against the different regions demarcated in Figure 7-3.

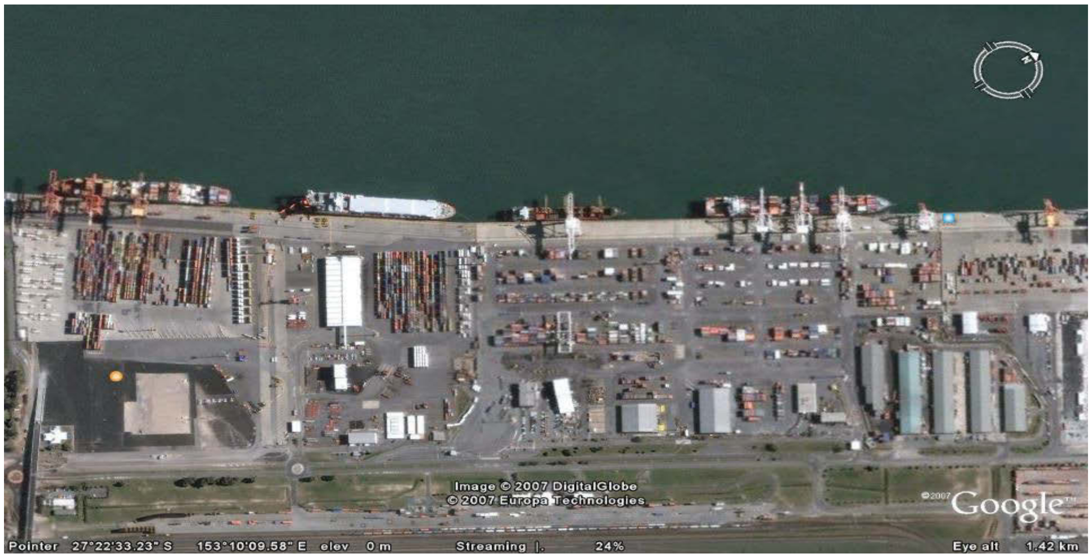


Image © 2007 DigitalGlobe
© 2007 Europa Technologies
©2007 Google
Pointer 27°22'33.23" S 153°10'09.58" E elev 0 m
Streaming |. 24%
Eye alt 1.42 km

Figure 7-2: Arial view of the FishermanIsland Container terminal (www.googlemaps.com)

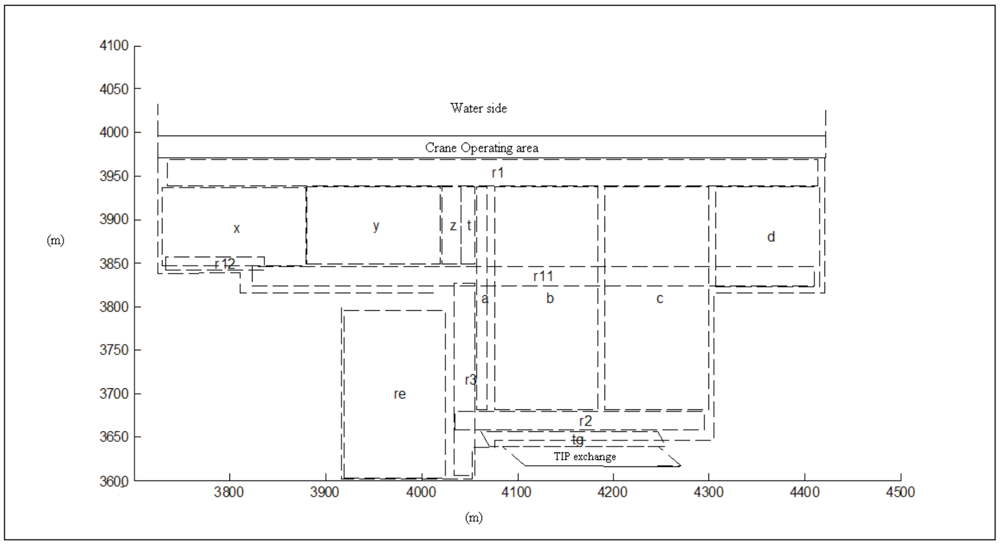


Figure 7-3: Different regions of the container yard at Fisherman's Island

Nodes and links connecting nodes are used to represent the terminal. The complete map of the terminal is consisted of 14949 nodes. The terminal yard is divided into grids of 8m x 4.4m in the stacking area and 4.4m x 7.0m in roadways. For each node, connections were established between the node and its adjacent nodes, thus creating a graph to represent the terminal. These connections signified the valid paths that could be executed by a straddle carrier, and pick up or drop-down tasks were defined based on these nodes and the connections between the nodes. Example vehicle paths in the terminal are represented in the MATLAB platform as shown in Figure 7-4.

The berthing facilities for container vessels were located on the water side of terminal next to the Quay Cane operating area. There were two forms of container arrival to the terminal: containers arrived from vessels(from the water side) or from the exchange area (from the land side).The space between water and land sides was used as a yard to stack containers until they were transferred or transhipped again to their respective destinations. The dedicated routes were kept in ***x*** and ***y*** directions of the map shown in Figure 7-4 in order to facilitate straddle carriers' movement between each of these zones.

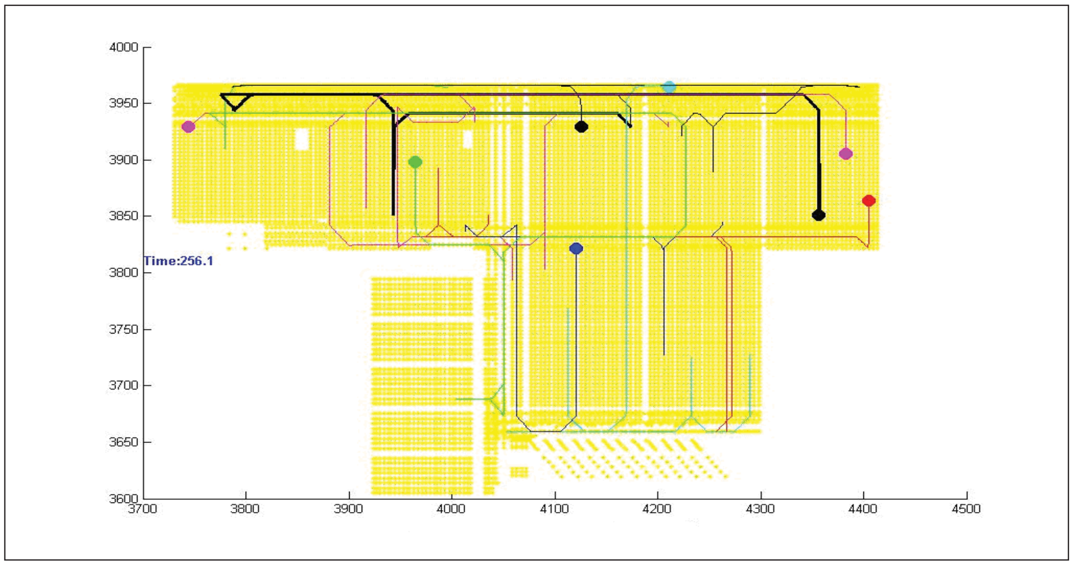


Figure 7-4: Vehicle movements screen short of the MATLAB simulation platform

7.3. Current Task Allocation and Motion Coordination Process

The current terminal planning was performed in a hierarchical manner with three different layers. Tasks to be completed were decided by the top layer and those instructions were transferred to an intermediate layer. At this layer, equipment (cranes and straddle carriers) allocation to different tasks has taken place. Once tasks were allocated to straddle carriers in the next layer, path planning (routing) and collision avoidance were performed. These planning activities were done with general purpose scheduling software.

Once all the decisions related to planning have been made in the three different layers, related information on the plan was transferred to SCs. SCs followed the instructions in order to complete the allocated tasks. An example task allocation is shown in Table 7-1. It shows the containers handled by respective straddle carrier, pick-up location and zone, drop-off off location and types of tasks (example: landside – LS outward or waterside –WS-outward). Based on the origin and destination, tasks were categorised into five types. Furthermore, the rehandling tasks such as shuffling from one location to another in the same or different zones were categorised into rehandling tasks.

Table 7-1: Task allocation information of the existing method of one hour duration

Vehicle ID	Task sequence ID	Container ID	Pick up zone	Pick up node	Drop-off zone	Drop-off node	Task category
16	X234	AB1	y	2041	TIP	14671	LS_outward
5	X238	AB2	c	7060	TIP	14239	LS_outward
2	X233	AB3	c	12531	TIP	14670	LS_outward
7	X231	AB4	d	5445	TIP	14228	LS_outward
2	X244	AB5	TIP	14670	TIP	14670	LS_outward
23	X239	AB6	TIP	14141	b	5093	LS_inward
5	X252	AB7	y	9188	TIP	14627	LS_outward
21	X248	AB8	c	8118	r1	1535	WS_outward
7	X247	AB9	c	5863	r2	13256	Rehandling
11	X258	AB10	x	6534	x	7134	Rehandling
13	X265	AB11	c	5128	c	14061	Rehandling
18	X282	AB12	b	9193	r1	203	WS_outward
16	X269	AB13	c	2597	TIP	14765	LS_outward
20	X278	AB14	b	12242	r1	4178	WS_outward
6	X287	AB15	TIP	14763	c	5428	LS_inward
1	X286	AB16	b	5107	TIP	14621	LS_outward
18	X297	AB17	c	2151	r1	1538	WS_outward
20	X299	AB18	c	2587	r1	203	WS_outward
19	X306	AB19	c	6468	r1	1538	WS_outward
2	X312	AB20	TIP	14766	c	5421	LS_inward
15	X302	AB21	b	12555	b	10658	Rehandling
11	X295	AB22	b	11504	b	10658	Rehandling
16	X300	AB23	c	12521	c	2151	Rehandling
1	X313	AB24	c	2161	c	2161	Rehandling
18	X310	AB25	b	7652	r1	502	WS_outward
20	X314	AB26	c	1085	r1	1085	WS_outward
21	X311	AB27	b	5554	r1	502	WS_outward
15	X317	AB28	b	7656	TIP	14756	LS_outward
11	X318	AB29	b	10658	b	10543	Rehandling
23	X315	AB30	b	11104	r1	282	WS_outward
15	X329	AB31	b	7201	TIP	14763	LS_outward
7	X323	AB32	b	12555	TIP	14761	LS_outward
6	X333	AB33	c	12521	TIP	14759	LS_outward
10	X328	AB34	TIP	14441	r1	1920	WS_outward

7.4. Experiments with the proposed STAMC approach

Simulations were done in two stages in order to show two key aspects of the STAMC approach in a large-scale outdoor material handling problem. The proposed STAMC approach was tested with a number of cases with different vehicle–task combinations in these stages. Meta-heuristic and evolutionary algorithms of SA, AA, ACO and commonly used two dispatching rules namely, First-Come-First-Serve (FCFS rule) and Closest-One-First (COF rule) were used in the simulations. The given task order was considered as task allocation sequence for the straddle carriers in the two dispatching rules. This is given in the second column of the Table 7-1. The best possible straddle carrier was selected based on the availably for the given task of the sequence. In the case of the FCFS rule, a straddle carrier which could reach the pick-up location of the task first (earliest reach time) was awarded the task. In the COF rule, the straddle carrier which is closest to the next task pick-up position is given the job. Here each straddle carrier looked for the closest available next task once it finished a task. All simulations were done in MATLAB environment with the software and hardware configurations being same as in the specifications given in Table 4-12 and Table 4-13.

7.4.1 The First Simulation Study

In the first simulation, the proposed approach was used alongside the previously discussed task allocation approaches, AA, ACO, SA, FCFS rule and COF rule. Tasks in the case study were selected from the schedules prepared with the existing system of the terminal for an eight-hour shift. A section of these schedules is shown in Table 7-1. The original schedule for eight-hour shift was divided into different segments as sub problems on an hourly basis. Later, these sub problems were solved separately on an hourly basis. Then, these problems were scheduled by the STAMC approach with different task allocation methods. During these experiments, it was possible to identify variations of the results from different task allocation approaches (makespans in these experiments)and straddle carrier utilisations, task distribution among the straddle carriers and variation of computational times for each method.

The scale of the scheduling problem (Number of vehicles – number of tasks combination) was different from hour to hour due to the availability of tasks in the respective time intervals. The AA, ACO, SA based STAMC approaches were run on a fixed number of iterations (each iteration generates a complete schedule for a scheduling problem) and of them the best schedules were selected as solutions. However, when dispatching rules were used schedules were selected from the first iteration itself since solution quality does not depend on the number of iterations. The results of these experiments are given in Table 7-2 and Figure 7-5.

Table 7-2: The makespan and computational cost of the first scenario

Time interval	Problem	Makespan (stu)					Computation time (seconds)				
		AA	ACO	SA	FCFS rule	COF rule	AA	ACO	SA	FCFS rule	COF rule
1st hour	19V-34T	93	127	118	122	127	8616	3697	11926	326	3224
2nd hour	**19V-33T**	93	115	116	121	117	8194	3163	13901	312	6423
3rd hour	19V-68T	133	220	154	164	216	16984	28765	31949	644	18339
4th hour	**19V-68T**	130	217	142	162	165	16152	28812	33058	663	30392
5th hour	19V-63T	134	163	149	159	172	13880	16696	30882	523	38263
6th hour	**19V-57T**	146	166	161	171	163	14036	16581	34542	529	46409
7th hour	19V-34T	91	126	117	121	140	9488	6219	25503	375	50745
8th hour	19V-34T	93	130	117	124	125	9463	6052	26915	373	54962

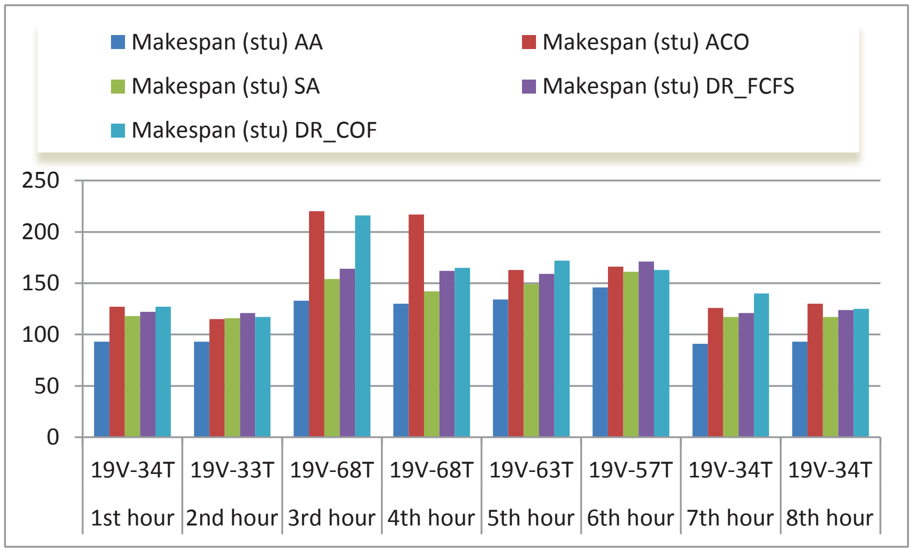


Figure 7-5: Results of the 1st simulation

The results of the first set of experiments show that AA out-performed all the other task allocation algorithms and dispatch rules as shown in Table 7-2 and Figure 7-5. In four scheduling sub-problems (1st, 2nd 7th and 8th hour schedules),the SA algorithm shows second best results. The third best solutions were generated by ACO algorithm except in two cases (3rd and 4th).Between the two dispatching rules, FCFS shows better results than the COF rule in all the cases. With respect to computational time, the FCFS approach shows the least, while the SA algorithm shows significantly more computation time than the others do. However, AA shows considerably less computational time than the meta-heuristics approaches.

Gantt chart representations of the best schedules generated by AA, SA, ACO, FCFS rule and COF rule for the 2nd(**19V-33T)**, 4th(**19V-68T**) and 6th(**19V-57T**) hour scheduling problems are given from Figure 7-6 to Figure 7-18.

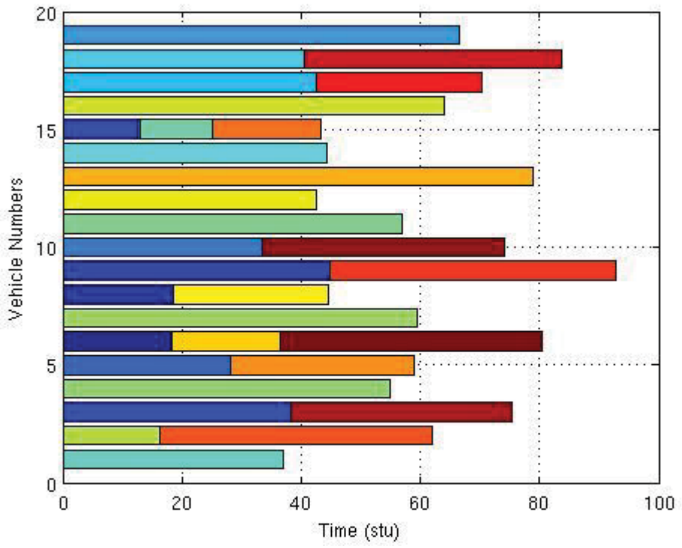


Figure 7-6: Gantt chart of the 2nd hour schedule based on the AA

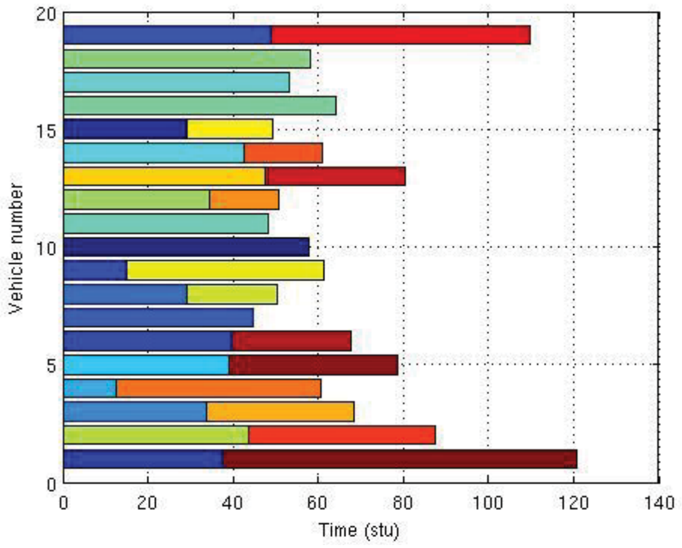


Figure 7-7: Gantt chart of the 2nd hour schedule based on the SA algorithm

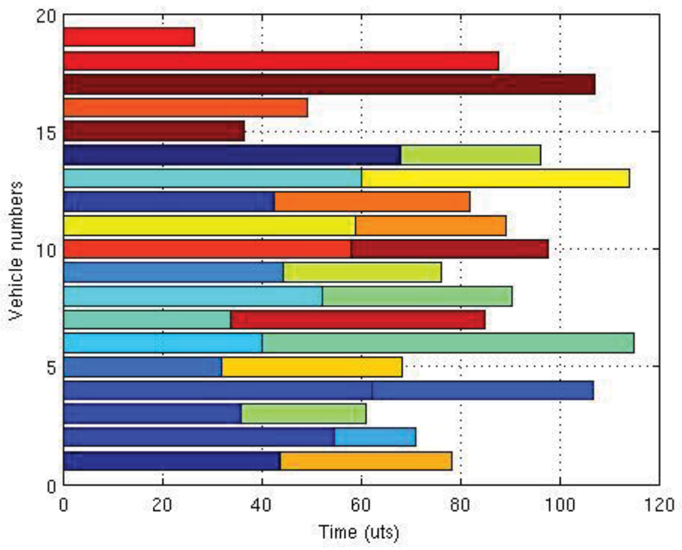


Figure 7-8: Gantt chart of the 2nd hour schedule based on the ACOalgorithm

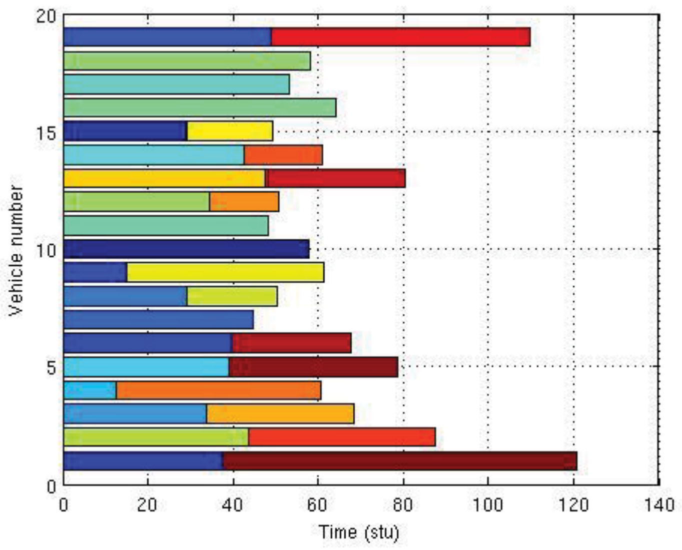


Figure 7-9: Gantt chart of the 2nd hour schedule based on FCFS rule

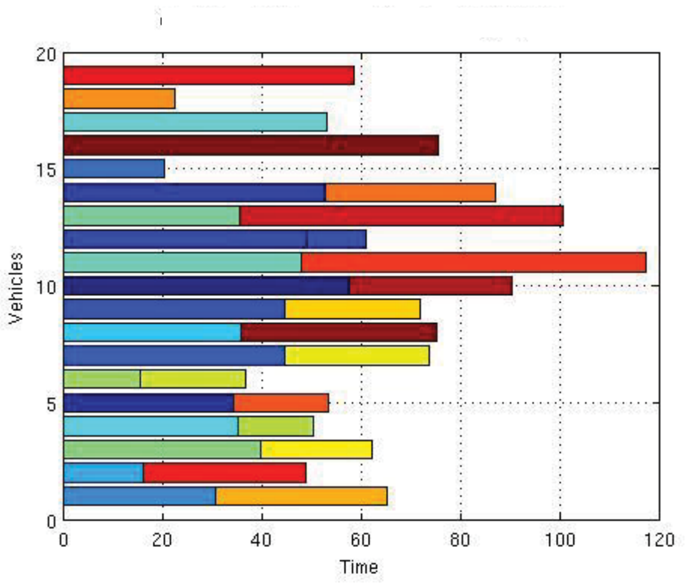


Figure 7-10: Gantt chart of the 2nd hour schedule based on COF rule

Among the Gantt charts, the AA showed the best makespan value of 93 (stu). Even the task distribution among the vehicles from AA was better than that of the other algorithms and dispatch rules.

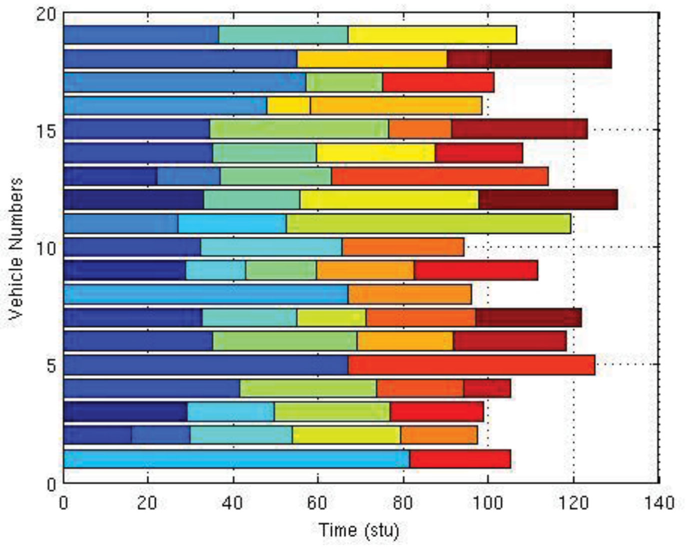


Figure 7-11: Gantt chart of the 4th hour schedule based on the AA

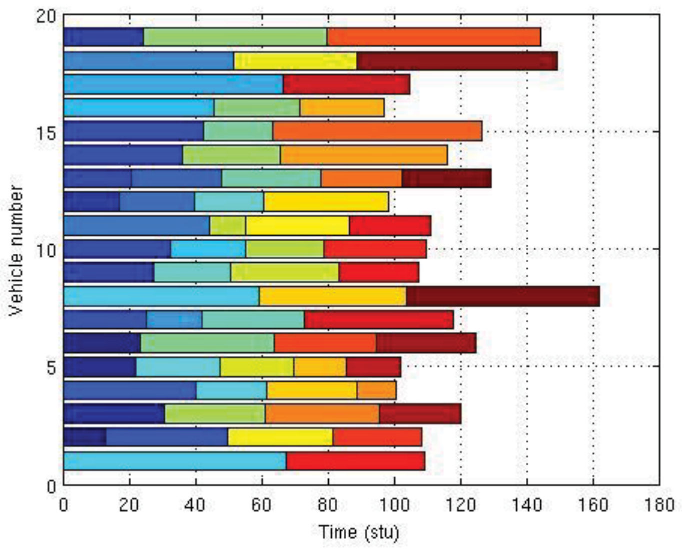


Figure 7-12: Gantt chart of the 4th hour schedule based on the SA algorithm

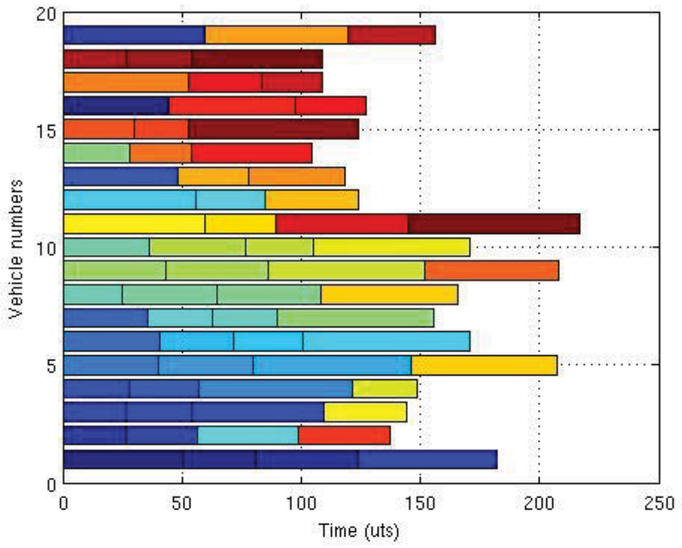


Figure 7-13: Gantt chart of the 4th hour schedule based on the ACO algorithm

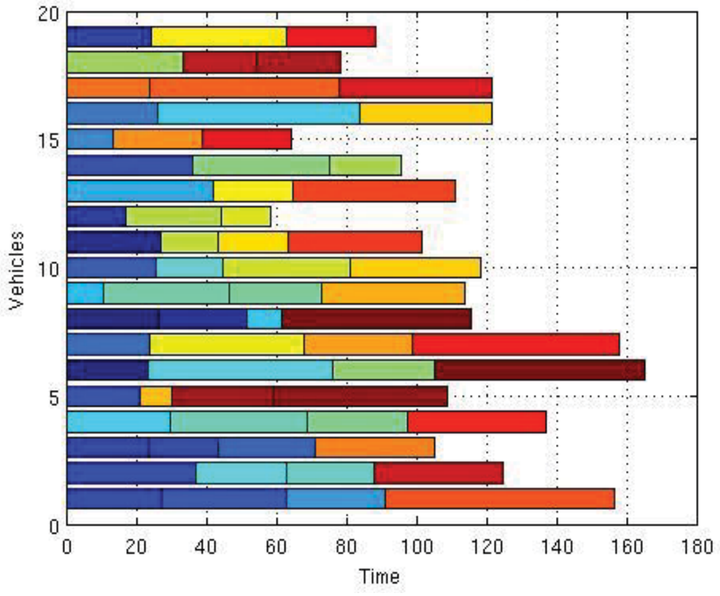


Figure 7-14: Gantt chart of the 4th hour schedule based on COF rule

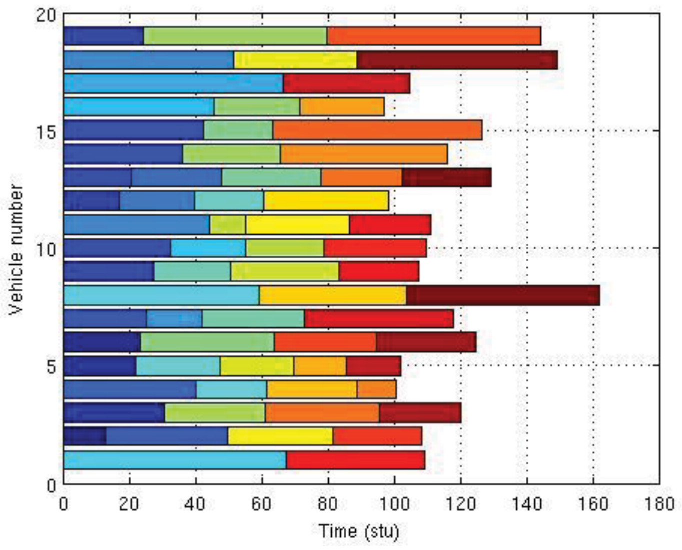


Figure 7-15: Gantt chart of the 4th hour schedule based on FCFS rule

In the 4th hour schedule, the AA gave the shortest makespan out of the five approaches and the task distribution among the vehicles was better than that of the others.

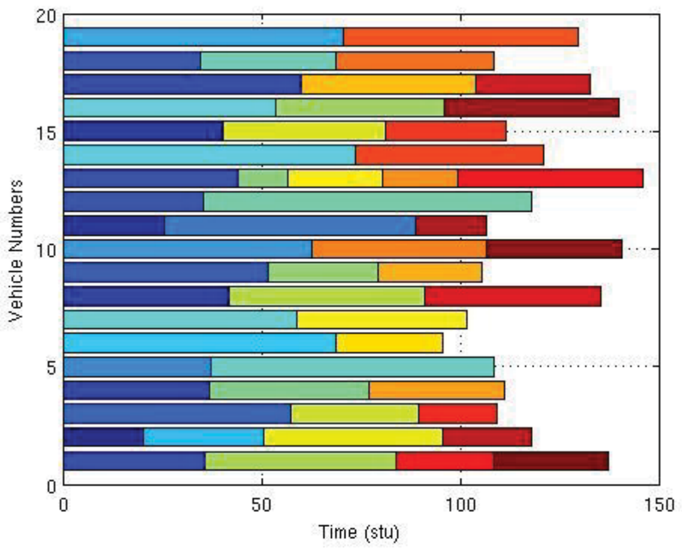


Figure 7-16: Gantt chart of the 6th hour schedule based on the AA

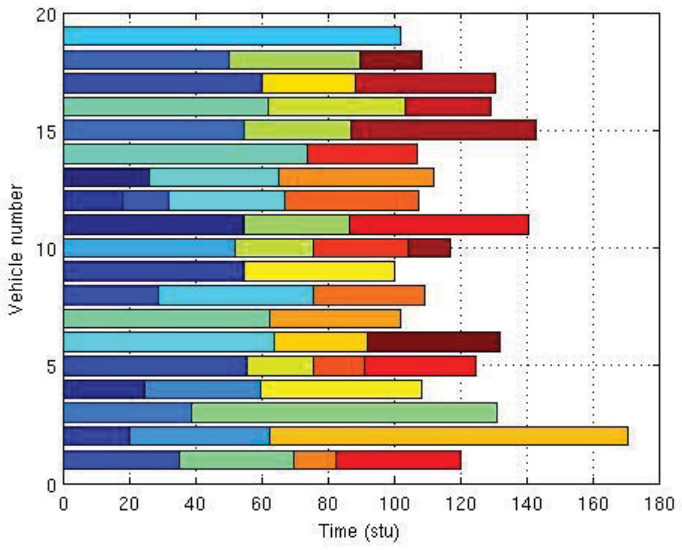


Figure 7-17: Gantt chart of the 6th hour schedule based on the SA algorithm

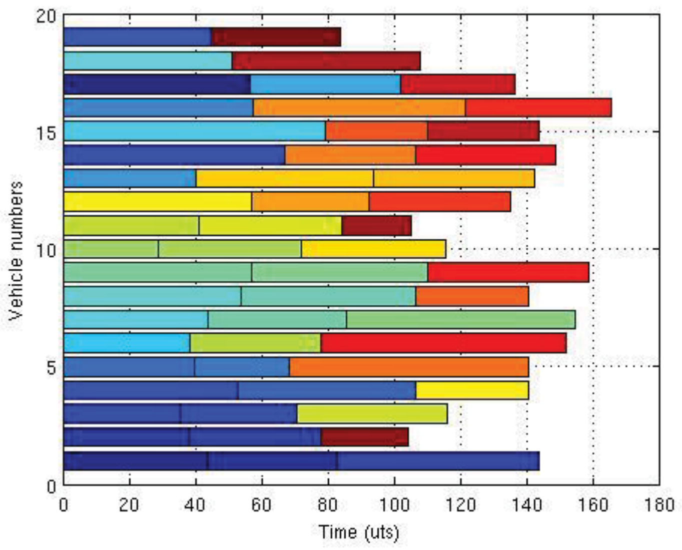


Figure 7-18: Gantt chart of the 6th hour schedule based on the ACO algorithm

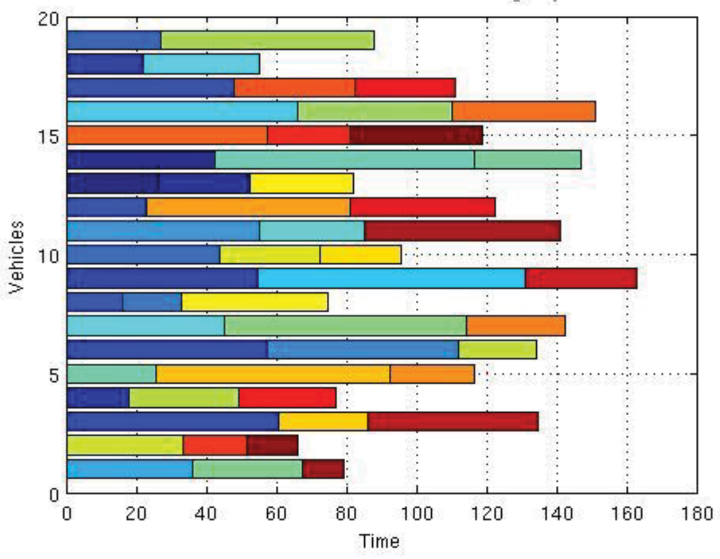


Figure 7-19: Gantt chart of the 6th hour schedule based on COF rule

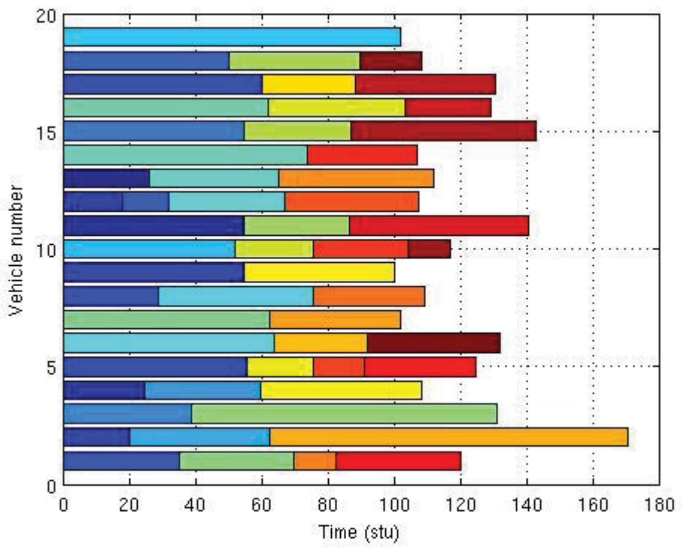


Figure 7-20: Gantt chart of the 6th hour schedule based on FCFS rule

In the 6^{th} hour schedule, The AA out-performed all the other approaches in giving a better makespan.

Of all the schedules, AA usually out-performed the two meta-heuristics and dispatching rules. As far as computational times were concerned, AA was able to generate better results with less computational cost than two meta-heuristics. Furthermore, the FCFS rule generated the schedules within a lesser time than all the other approaches and the results it generated with this computation time were reasonable to acceptable.

However, there is always a trade-off between the quality of the solution and the computational time spent to generate solutions. It was possible to improve the solution quality if more time was spent for search. As previously highlighted, if this were tested in the distributed architecture discussed in Chapter 6, he computational times could be cut down considerably.

7.4.2 Second Simulation Study

In the second simulation study, the schedule of the eight-hour period was performed using two dispatching rules while maintaining the task priority list intact. The schedules from these two dispatching rules were represented in Gantt charts (Figure 7-21 and Figure 7-22). The main objective of this simulation was to investigate the behaviour of the proposed approach in a large scale problem setting.

Of the two dispatching rules, as in the other problem scenarios, the FCFS rule again outperformed the COF rule. The makespan of the FCFS rule was nearly half of the makespan of the COF rule. Further, the task distribution among the vehicles also showed that the FCFS rule gives better results than the COF rule since most of the vehicles had the same number of tasks to perform and all the vehicles were going to finish their respective tasks at approximately the same time. The AA and the other two meta-heuristic approaches could not be tested in this problem scenario due to the excessive computational times required to generate the results.

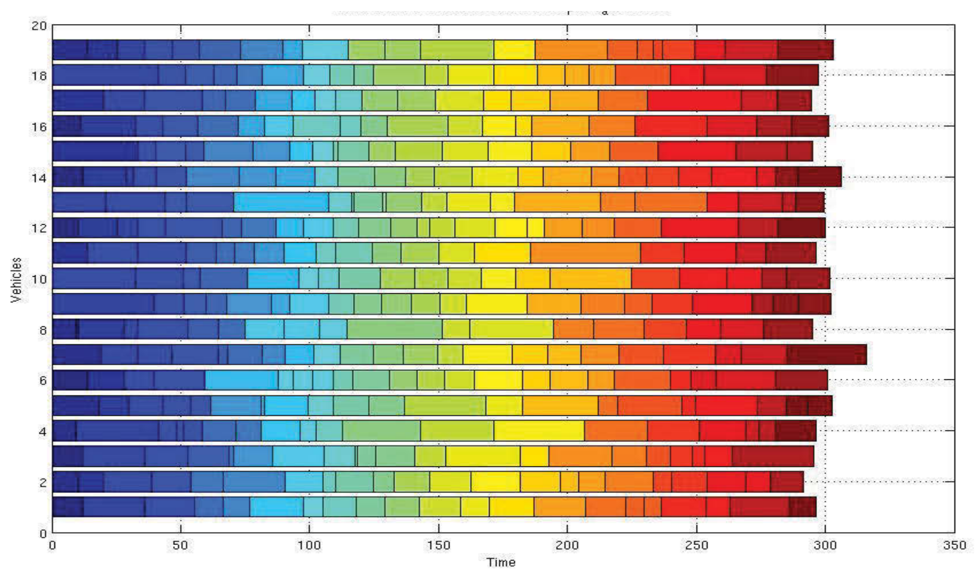


Figure 7-21: Gantt chart of overall schedule for 8 hours based on FCFS rule

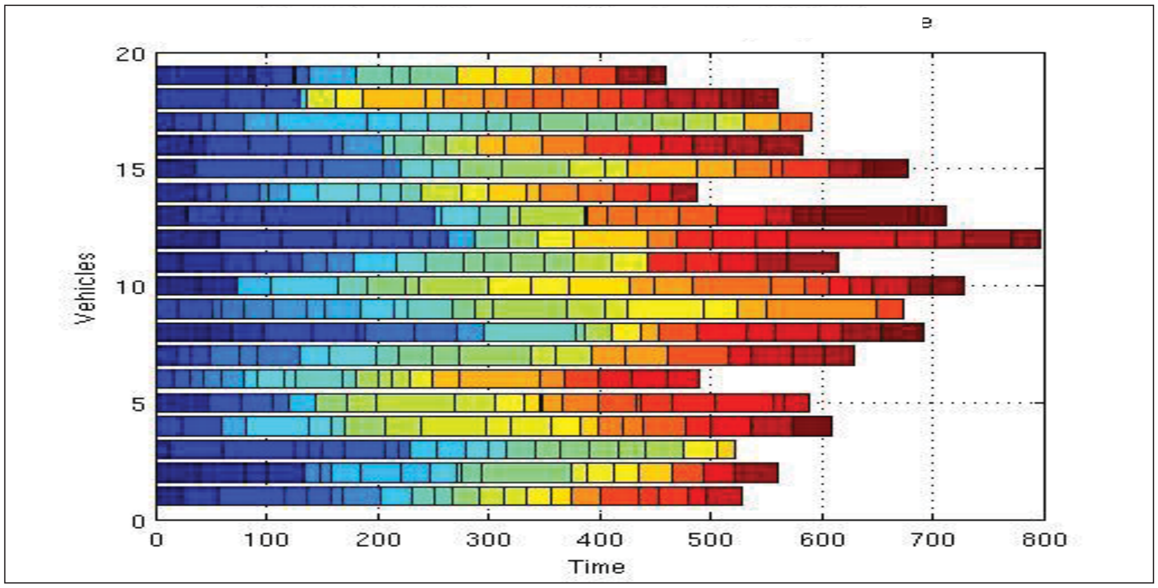


Figure 7-22: Gantt chart of overall schedule for 8 hours based on COF rule

7.5 Discussion

Based on the case studies done with different problem scenarios, it can be admitted that the STAMC approach can be used even for large scale outdoor material handling environments. Furthermore, the AA-based results show that the solution quality can be improved by sacrificing the computational time. However, this burden can be improved by adopting the distributed architecture presented in Chapter 6. The hardware platform used for the simulation studies in Chapter 6 has some limitations. In the distributed computational architecture, each workstation was allocated for path and motion planning of a straddle carrier. Because the number of workstations available is less than the number of straddle carriers in the case studies in Chapter 7 testing on this platform was not possible in this case.

Once the scale of the map (number of nodes and number of connections) increases, the computation time increases drastically. This is mainly caused by the path planning component in the STAMC approach. The Dijkstra algorithm is used in the SiPaMoP algorithm for collision-free path planning in the STAMC approach. The Dijkstra algorithm searches all the available nodes when it generates a path in the network of the map. Therefore it is very time consuming. In order to overcome this problem, other comparatively faster path planning algorithms such as A*can be used with the SiPaMoP algorithm in the proposed STAMC approach. Since A* is based on heuristic information and does not search all the nodes in the map it helps to reduce the computational time considerably. However, the collision-avoidance mechanism consumes a reasonable time out of the overall time taken to plan one path since it looks for possible collisions at each node of the selected path. Furthermore, all these simulations were done in a MATLAB software environment, which is quite slow compared to other programming software such as C/C++.

In these case studies, the proposed STAMC approach could not be compared with the currently used scheduling method in the terminal due to numerous reasons. One main reason is that many of the features and constraints in the terminal could not be incorporated into the problem scenarios due to the restrictions imposed on the information of the terminal. Some features are hard to model in the problem due to the complexity. Furthermore, for some situations in the terminal, human intervention is still being used.

Chapter 8

Conclusion

8.1. Introduction

This thesis presented a novel approach to the complex multiple autonomous vehicle task allocation and motion coordination problem. The rest of this chapter presents the summary of contributions, methodologies and techniques proposed, limitations of the proposed approaches and future extensions of this research.

The main objective of this research was to develop a novel approach to solve the complex multiple autonomous vehicle task allocation and motion coordination problem, which is categorized as NP-hard in mathematical terms. As stated in Chapter 2, this problem is an important problem in many material-handling environments. This aspect has been highlighted in a number of research papers and reviews Qiu et. al., (2002), Vis and Harika (2004), and Vis (2006).There is a scarcity of approaches which solve the integrated problem of scheduling and routing for a large fleet of autonomous vehicles in complex and dynamic material-handling environments.

A novel STMAC approach was developed to solve the multi-vehicle task allocation and motion coordination problem in this research. This approach was then extensively investigated in two scenarios: static task allocation and dynamic task allocation. Meta-heuristic and evolutionary techniques were tested for task allocation in the STMAC approach. A new collision-free pathfinder, the SiPaMoP algorithm, was developed for simultaneous path planning and motion coordination in the STAMC approach. A number of simulation studies were conducted to test and validate the proposed STAMC approach along with different task allocation mechanisms. Furthermore, the proposed STAMC approach was tested with widely used dispatching rules.

The STAMC approach was first implemented in centralized computational architecture and then in distributed architecture in order to reduce the computational time. Simulation studies were performed on the distributed architecture with different task allocation problem settings in a computer cluster environment. Finally, the STAMC approach was tested in a large-scale material handling environment, an automated container terminal where all the container movements were done by the autonomous straddle carriers.

8.2. Research Outcomes

The research outcomes achieved include:

(1) Simultaneous task allocation and motion coordination approach

This is a novel approach that integrates the three main components of autonomous vehicle scheduling and planning in material handling: task allocation, path planning and collision avoidance in constraint environment with bi-directional path topology. Itcan be used with meta-heuristic algorithms such as SA and ACO and with AA to generate near–optimal solutions. Furthermore, this approach can be used with general dispatching rules such as FCFS and COF. The results of the simulation and case studies reveal that the STAMC approach can be used to solve the task allocation and motion coordination problem of a large fleet of autonomous vehicles deployed in a constraint environment with bi-directional path topology.

(2) Simultaneous path and motion coordination algorithm

In order to coordinate the motion (path planning and collision/deadlock avoidance) of a large fleet of autonomous vehicles in the multiple vehicle task allocation and motion coordination problem, a novel simultaneous path and motion coordination algorithm (SiPaMoP) was developed. The Dijkstra algorithm was used for small-scale maps and the A* algorithm was used for large maps for the shortest path search. The connection weights were changed dynamically based on traffic condition in order to avoid collision and deadlock among the autonomous vehicles.

(3) Dynamic task allocation and motion coordination with re-planning strategy

The STAMC approach was extended and evaluated in a dynamic task allocation scenario where new tasks arrive at different times. In order to accommodate new tasks, ,the STAMC approach was applied at fixed time intervals thereby previously assigned yet not started tasks are reassigned along with newly arrived tasks. Furthermore, the STAMC approach uses meta-heuristic techniques to improve the quality of the schedule considerably. This approach was further experimented in situations where re-planning was essential due to unexpected events such as vehicle breakdowns and path blockages.

(4) Distributed computational architecture for the STAMC approach
The STAMC approach was implemented in a distributed architecture using high performance computer cluster in order to reduce the high computational time. This implementation was tested with case studies; results showed that the computational time could be considerably reduced by adopting this architecture.

Finally, the STAMC approach was tested with a fully automated container terminal's simulation model, in order to investigate its applicability to large-scale material handling problems. Overall results show that the proposed approach can be used in large-scale automated material handling environments to improve efficiency.

8.3. Limitations and Future Opportunities for Research

Previously stated outcomes were achieved with some limitations. Areas for further improvement are presented below.

- Planning and motion coordination under uncertainty and with incomplete information. The travelling times were calculated in this research deterministically. However, travelling times cannot be predicted very accurately due to numerous unforeseen circumstances in a real world environment. It would be better to accommodate uncertainty in the STAMC approach.

- In order to cater to quay cranes and yard cranes efficiently in container terminal queues of vehicles can be maintained. This can be further streamlined by queuing techniques. Thus, the real systems can be modelled in a more realistic way, and accurate and just in time delivery / pick up mechanisms can be developed.

- The autonomous operations of the vehicles was beyond the scope of this research. However, this is an interesting area for investigation since most of the processes in supply chain management are rapidly being automated with the help of information technology.

- It will be worthwhile to investigate effective approaches and algorithms for multi-objective optimisation based task allocation and motion coordination.

- The distributed/parallel architecture can be further extended to a large fleet of autonomous vehicles.

- Incremental path planning
  The paths were planned for the whole journey from pick-up node to finish node at once by considering possible collisions. This increases the overall computation time considerably and sometimes it is not worth doing in situations where re-planning situations arise. However, if path planning is done in an incremental mode, computational time taken to find a collision-free path can be reduced.

REFERENCES

Achuthan, N. R., Caccetta, L. & Hill, S. P. (1997) 'On the vehicle routing problem' *Nonlinear Analysis,* 30**,** 4277-4288.

Aguilar, L., Alami, S., Fleury, S., Herrb, M., Ingrand, F. & Robert, F. (1995) 'Ten autonomous mobile robots (and even more) in a route network like environment' *Proceedings of the IEEE Conference on Robotics and Automation,* 260 - 267.

Asef-Vaziri, A. &Laporte, G. (2005) 'Loop based facility planning and material handling' *European Journal of Operational Research*, 164, Issue 1**,** 1-11.

Bagchi, T. P. (1999) *Multi objective scheduling by Genetic Algorithm* Kluwer Academic, Boston.

Baker, B. M. &Ayechew, M. A. (2003) 'A genetic algorithm for the vehicle routing problem' *Computers & Operations Research,* 30**,** 787-800.

Baldomero, J. F. (2001) 'Message Passing under MATLAB.' Paper presented at the *Advanced Simulation Technologies Conference,*SeattleWashington, 2001.

Barbarosoglu, G. &OzguR, D. (1999) 'A tabu search algorithm for the vehicle routing problem.' *Computers & Operations Research,* 26**,** 255-270.

Bertsekas, D. P. (1978) Auction Algorithms, http://citeseer.nj.nec.com/374581.html

Bertsekas, D. P. (1979) 'A distributed algorithm for the assignment problem.' *Laboratory for Information and Decision Systems* working paper, Massachusetts Institute of Technology.

Bertsekas, D. P. (1986) 'Distributed relaxation methods for linear network flow problems.' Proceedings of the *25th IEEE Conference on Decision and Control,* 1986, 2101-2106.

Bertsekas, D. P. (1991) 'The auction algorithm for shortest paths.' *SIAM Journal on Optimization,* 1**,** 425-447.

Bertsekas, D. P. &Castanon, D. A. (1989) 'The auction algorithm for transportation problems.' *Annals of Operations Research,* 20**,** 67-96.

Bianchi, L. (2000) 'Notes on dynamic vehicle routing - the state of the art.' Proceedings of *ANTS 2000 From Ant Colonies to Artificial Ants: Second International Workshop on Ant Algorithms*,59-62.

Bish, E. K., Chen , F. Y., Leong, Y. T., Nelson, B. L., Ng, J. W. C. &Simchi-Levi, D. (2005) 'Dispatching vehicles in a mega container terminal.' *OR Spectrum* 27, Number 4, 491 - 506

Bish, E. K., Yin, T., Li, L. C. L., Ng, J. W. C. &Simchi-Levi, D. (2001) 'Analysis of a new vehicle scheduling and location problem.' *Naval Research Logistics,* 48**,** 363-385.

Bose, J., Reiners, T., Steenken, D. &Voß, S. (2000) 'Vehicle dispatching at seaport container terminals using evolutionary algorithms.' In Sprague, R. H. (ed), *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences,* IEEE Piscataway,2000, DTM-IT, 1-10.

Briskorn, D., Drexl, A. & Hartmann, S. (2006) 'Inventory-based dispatching of automated guided vehicles on container terminals.' *OR Spectrum,* 28**,**611-630.

Bullnheimer, B., Hartl, R. F. & Strauss, C. (1998), 'Applying the ant system to the vehicle routing problem.' In Voß, S., Martello, S., Osman, I. and Roucairol, C. (eds) *Meta-heuristics: advances and trends in local search paradigms for optimization,* Kluwer, Boston.

Buriol LS, Ressende MGC, Thorup, M (2003) Speeding up Dynamic Shortest Path Algorithms. At&T Labs Research Technical Report TD-5RJ8B

Cargonews (2006) February 2006. http://www.worldcargonews.com/

Chabini, I., (1997),"A new algorithm for shortest paths in discrete dynamic

Networks", "8th IFAC/IFIP/IFORS Symposium on Transportation systems, Tech Univ Crete, Greece, 16-18 June 1997

Chabini, I. (1998), "Discrete dynamic shortest path problems in transportation Applications", Transportation Research Record 1645, 1998

Chen, L., Bostel, N., Dejax, P., Cai, J. & Xi, L. (2007) 'A tabu search algorithm for the integrated scheduling problem of container handling systems in a maritime terminal.' *European Journal of Operational Research,* 181**,** 40-58.

Choi, I.-C., Kim, S.-I. & Kim, H.-S. (2003) 'A genetic algorithm with a mixed region search for the asymmetric travelling salesman problem.' *Computers & Operations Research,* 30**,** 773-786.

Colorni, A., Dorigo, M. &Maniezzo, V. (1991) 'Distributed optimization by ant colonies.' *Proceedings of the First European Conference on Artificial Life (ECAL 91),* 1991, 134-142.

Corréa, A. I., Langevin, A. & Rousseau, L.-M. (2007) 'Scheduling and routing of automated guided vehicles: a hybrid approach.' *International Journal on Computers & Operations Research,* Volume 34**,** 1688-1707.

Corréa, A. I., Langevin, A. & Rousseau, L. M. (2004) 'Dispatching and conflict-free routing of automated guided vehicles: a hybrid approach.' Combining Constraint Programming and Mixed Integer Programming Lecture, *Lecture Notes in Computer Science,* 3011 / 2004, 370 - 379

Czarnas, Z. J. C. A. P. (2002) 'Parallel simulated annealing for the vehicle routing problem with time windows.' *Proceedings of the 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing,* Canary Islands, Spain,2002, 376-383.

Das, S. K. &Spasovic, L. (2004) 'Scheduling material handling vehicles in a container terminal",*Production Planning & Control 14,* Number 7**,** 623 - 633.

Desaulniers, G., Langevin, A., Riopel, D. & Villeneuve, B. (2003) 'Dispatching and conflict-free routing of automated guided vehicles: an exact approach", *International Journal of Flexible Manufacturing Systems*, 15, Number 4, 309 - 331

Dijkstra, E. (1959). "A note on two problems in connexion with graphs", NumerischeMathematik, 1 (1), 269-271 DOI: 10.1007/BF01386390

Dhouib, K. &Kadi, D. A. (1994) 'Expert system for AGV managing in bidirectional networks: KADS methodology based approach.' I*nternational Journal of Production Economics,* 33**,** 31-43.

Dorigo, M. (1992) 'Optimization, learning and natural algorithms' *(translated from the Italian). Unpublished doctoral dissertation*, Politecnico di Milano, Dipartimento di Elettronica, Milan, Italy.

Dorigo, M. & Gambardella, L. M. (1997) 'Ant Colony system: a cooperative learning approach to the travelling salesman problem.' *IEEE Transactions on Evolutionary Computation,* 1(1)**,** 53-66.

Dorigo, M., Maniezzo, V. &Colorni, A. (1991) 'Positive feedback as a search strategy.' *Tech. Rep. 91-016*, Politecnico di Milano, Dipartimento di Elettronica, Milan, Italy.

Dorigo, M., Maniezzo, V. &Colorni, A. (1996) 'The ant system: optimization by a colony of cooperating agents.' *IEEE Transactions on Systems, Man, and Cybernetics,* 26(1)**,** 29-42.

Evers, J. J. M. &Koppers, S. A. J. (1996) ‘Automated guided vehicle traffic control at a container terminal.’ *Transportation Research-A,* 30**,** 21-34.

Ferguson, D. &Stentz, A. (2005), “The Delayed D* algorithm for efficient path Replanning”, Proceedings of the 2005 IEEE International Conference on Robotics and AutomationBarcelona, Spain, 2057-2062

Fu, L. &Rilett, L.R. (1996) ‘Expected shortest paths in dynamic andstochastic traffic networks’, Transportation Research, Part B:Methodological, vol. 32, no. 7, pp. 499-516

Gamberi, M., Manzini, R. &Regattieri, A. (2008) ‘A new approach for the automatic analysis and control of material handling systems: integrated layout flow analysis (ILFA).’ *International Journal of Advancements of Manufcturing Technologies,* DOI 10.1007/s00170-008-1466-9.

Glover, F. (1989) ‘Tabu search - part I.’ *ORSA Journal on Computing,* 1**,** 190-206.
Goasguen, S. http://falcon.ecn.purdue.edu:8080/cluster

Grossman, D.D. (1998) ‘Traffic control of multiple robot vehicles.’ *IEEE Journal of Robotics and Automation,*Vol 4**,** 491-497.

Grunow , M., Günther, H. & Lehmann, M. (2004) ‘Dispatching multi-load AGVs in highly automated seaport container terminals.’ *OR Spectrum* 26, Number 2**,** 211 - 235

Grunow, M., Günther, H. O. & Lehmann, M. (2006) ‘Strategies for dispatching AGVs at automated seaport container terminals.’ *OR Spectrum,* 28**,** 587-610.

Hartmann, S. (2004) ‘General framework for scheduling equipment and manpower at container terminals.’ *OR Spectrum,* 51 - 74

Henesey, L., Wernstedt, F. &Davidsson, P. (2003) ‘Market-driven control in container terminal management.’ *Proceedings of the 2nd International Conference on Computer Applications and Information Technology in the Maritime Industries,* 2003.

Ho, Y. C. & Liu, H. C. (2006) ‘A simulation study on the performance of pickup-dispatching rules for multiple-load AGVs’. *Computers & Industrial Engineering,* 51**,** 445-463.

Holland, J. H. (1975) Adaptation in natural and artificial systems.University of Michigan Press, Ann Arbor.

Husdal, J., 2000, "An investigation into fastest path problems in dynamic transportation networks", Unpublished, coursework for the MSc in GIS, University of Leicester.

Kennedy, J. &Eberhart, R. (1995) Particle swarm optimization, http://ieeexplore.ieee.org

Kim, K. H. & Gunther, H.-O. (eds) (2007) *Container terminals and cargo systems: design, operations management and logistic control issues.* Springer, Berlin / HeidelbergNew York.

Kim, K. H., Jeon, S. M. &Ryu, K. R. (2006) 'Deadlock prevention for automated guided vehicles in automated container terminals.' *OR Spectrum,* 28, Number 4**,** 659-679.

Kim, K. H., Kang, J. S. &Ryu , K. R. (2004) 'A beam search algorithm for the load sequencing of outbound containers in port container terminals.' *OR Spectrum,* 26, Number 1**,** 93 - 116

Kim, K. H. & Kim, K. Y. (1999) 'Routing straddle carriers for the loading operation of containers using a beam search algorithm.' *Computers & Industrial Engineering,* 36, Issue 1**,** 109-136.

Kim, K. H. & Moon, K. C. (2003) 'Berth scheduling by simulated annealing.' *Transportation Research Part B: Methodological,* 37**,** 541-560.

Kim, K. H., Won, S. H., Lim, J. K. & Takahashi, T. (2004b) 'An architectural design of control software for automated container terminals.' *Computers & Industrial Engineering,* 46**,** 741-754.

Kim, K. Y. & Kim, K. H. (2003) 'Heuristic algorithms for routing yard-side equipment for minimizing loading times in container terminals.' *Naval Research Logistics,* 50**,** 498-514.

Kirkpatrick , S., Gelatt, C. D. &Vecchi, M. P. (1983) 'Optimization by simulated annealing.' *Science,* 671--680.

Koo, P. H., Lee, W. S. & Jang, D. W. (2004) 'Fleet sizing and vehicle routing for container transportation in a static environment.' *OR Spectrum*, 26, Number 2**,** 193 - 209

Kulatunga, A. K., Liu, D. K., Dissanayake, G. &Siyambalapitiya, S. B. (2006) 'Ant colony optimization technique for simultaneous task allocation and path planning of autonomous vehicles.' *Proceedings of the IEEE International*

Conference on Cybernetics and Intelligent Systems (CIS), 7-9 June, 2006, Bangkok, Thailand, 823-828

Kulatunga, A. K., Skinner, B. T., Liu, D. K. & Nguyen, H. T. (2007) 'Simultaneous task allocation and motion coordination of autonomous vehicles using a parallel computing cluster.' *Robotic Welding, Intelligence and Automation*. 362/2007, pp 409-420, Springer, Berlin/Heidelberg

Langevin, A., Lauzon, D. &Riopel, D. (1996) 'Dispatching, routing, and scheduling of two automated guided vehicles in a flexible manufacturing system.' *International Journal of Flexible Manufacturing Systems*, 8, Number 3, 247 - 262

Lau, H. C., Sim, M. &Teo, K. M. (2003) 'Vehicle routing problem with time windows and a limited number of vehicles.' *European Journal of Operational Research,* 148**,** 559-569.

Lau, H. Y. K., Wong, V. W. K. & Lee, I. S. K. (2007) 'Immunity-based autonomous guided vehicles control.' *Applied Soft Computing,* 7**,** 41-57.

Lau, H. Y. K. & Zhao, Y. (2008) 'Integrated scheduling of handling equipment at automated container terminals.' *International Journal of Production Economics,* 112**,** 665-682.

Le-Anh, T. & De Koster, M. B. M. (2005) 'On-line dispatching rules for vehicle-based internal transport systems.' *International Journal of Production Research*, 43, Number 8 / April 15, 2005 1711 - 1728

Le-Anh , T. & De Koster, M. B. M. (2006) 'A review of design and control of automated guided vehicle systems.' *European Journal of Operational Research,* 171 (2006)**,** 1–23.

Leong, C. Y. (2001) 'Simulation study of dynamic AGV-container job deployment scheme.' Master's thesis, National University of Singapore.

Li, J.-Q., Mirchandani, P. B. &Borenstein, D. (2008) 'Real-time vehicle rerouting problems with time windows.' *European Journal of Operational Research,* due for publication 2008.

Lim, J. K., Kim, K. H., Yoshimoto, K., Lee, J. H. & Takahashi, T. (2003) 'A dispatching method for automated guided vehicles by using a bidding concept.' *OR Spectrum,* 25**,** 25-44.

Liu, C. I., Jula, H., Vukadinovic, K. &Ioannou, P. (2004a) 'Automated guided vehicle system for two container yard layouts.' *Transportation Research Part C: Emerging Technologies,* 12, Issue 5**,** 349-368.

Liu, C. I., Jula, H., Vukadinovic, K. &Ioannou, P. (2004b) 'Automated guided vehicle system for two container yard layouts.' *Transportation Research Part C: Emerging Technologies,* 12**,** 349-368.

Liu, D. K., Wu, X., Kulatunga, A. K. & Dissanayake, G. (2006) 'Motion coordination of multiple autonomous vehicles in dynamic and strictly constrained environments.' *Proceedings of the IEEE International Conference on Cybernetics and Intelligent Systems (CIS),* 7-9 June, 2006, Bangkok, Thailand**,** 204-209.

Liu, D. K. &Kulatunga, A. K. (2007) 'Simultaneous Planning and Scheduling for Multi-Autonomous Vehicles', in Dahal, K., Tan, K.C. and Burke, E. (eds) *Evolutionary Scheduling, Studies in Computational Intelligence,* Volume 49/2007, 437-464, Springer, Berlin/Heidelberg.

Liu, F.-H. F. &Shen, S.-Y. (1999) 'An overview of a heuristic for vehicle routing problem with time windows.' *Computers & Industrial Engineering,* 37**,** 331-334.

Matson, J. O. & White, J. A. (1982) Operational research and material handling. *European Journal of Operational Research,* 11**,** 309-318.

Maza, S. &Castagna, P. (2005) 'A performance-based structural policy for conflict-free routing of bi-directional automated guided vehicles.' *Computers in Industry,* 56**,** 719-733.

Meersmans, P. J. M. (2002) Optimization of container handling systems, https://ep.eur.nl/handle/1765/1855.

Meersmans, P. J. M., &Wagelmans, A. P. M. (2001) Dynamic scheduling of handling equipment at automated container terminals, *Technical Report EI 2001-33,* Erasmus University Rotterdam, Econometric Institute, http://www.eur.nl/WebDOC/doc/econometrie/feweco20011128140514.pdf.

Meersmans, P. J. M. D., R. (2001a) Operations research supports container handling http://hdl.handle.net/1765/1689

Meersmans, P. J. M. &Wagelmans, A. P. M. (2001b) Effective algorithms for integrated scheduling of handling equipment at automated container terminals, https://ep.eur.nl/handle/1765/95.

Melouk, S., Damodaran, P. & Chang, P.-Y. (2003) 'Minimizing makespan for single machine batch processing with non-identical job sizes using simulated annealing.' *International Journal of Production Economics,* corrected proof, due for publication

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. & Teller, E. (1953) 'Equation of state calculation by fast computing machines.' *Journal of Chemical Physics,* 21**,** 1087-1091.

Mohring, R. H., Kohler, E., Gawrilow, E. &Stenzel, B. (1998) Conflict-free real-time AGV routing. http:/citeseer.edu.

Moorthy, R. L., Hock-Guan, W., Ng, W.-C. & Chung-Piaw, T. (2003a) 'Cyclic deadlock prediction and avoidance for zone-controlled AGV system. *International Journal of Production Economics,* 83, Issue 3**,** 309-324.

MPI.FORUM (2005) MPI-2: Extensions to the Message-Passing Interface. Message Passing Interface Specification. November 15, 2003 ed., NSF and DARPA.

Narasimhan, A.&Palekar, U. S. (2002) 'Analysis and algorithms for the transtainer routing problem in container port operations.' *Transportation Science,* 36(1)**,** 63-78.

Narasimhan, R. &Batta, M. R. K., H. (1999) 'Routing automated guided vehicles in the presence of interruptions.' *International Journal of Production Research,* 37 Issue 3**,** 653-682.

Nishi, T., Morinaka, S. &Konishi, M. (2007) 'A distributed routing method for AGVs under motion delay disturbance.' *Robotics and Computer-Integrated Manufacturing,* 23**,** 517-532.

Nishi, T., Sakata, A., Hasebe, S. & Hashimoto, I. (2000) 'Autonomous decentralized scheduling system for just-in-time production.' *Computers & Chemical Engineering,* 24**,** 345-351.

Oboth, C. &Batta, R. K., M (1999) 'Dynamic conflict-free routing of automated guided vehicles.' *International Journal of Production Research*, 37 Issue 9**,** 2003-2031.

Ochi, L. S., Vianna, D. S., Drummond, L. M. A. & Victor, A. O. (1998) 'A parallel evolutionary algorithm for the vehicle routing problem with heterogeneous fleet.' *Future Generation Computer Systems,* 14**,** 285-292.

Paul, G. & Liu, D. K. (2006) ‘Replanning of Multiple Autonomous Vehicles in Material Handling.’ *Proceedings of the IEEE International Conference on Cybernetics and Intelligent Systems (CIS),* 7-9 June, 2006, Bangkok, Thailand

Patrick_Corporation_Website (2007) http://www.patrick.com.au/IRM/Content/default.aspx

Philippsen, R. &Siegwart, R. (2005), “An interpolated dynamic navigationfunction”, *Proceedings of the 2005 IEEE International Conference onRobotics and Automation* Barcelona, Spain, April 2005, 3793-3800

Qin, Y. , Sun, D., Li, N. & Cen, Y. (2004), “Path planning for mobile robot usingthe particle swarm optimization with mutation Operator”, *Proceedingsof 2004 International Conference on Machine Learning and Cybernetics*,2004, Volume: 4, 26-29 Aug. 2004, 2473 – 2478

Qingcheng, Z. &Zhongzhen, Y. (2008) ‘Integrating simulation and optimization to schedule loading operations in container terminals.’ *Computers & Operations Research,* doi:10.1016/j.cor.2008.06.010.

Qiu, L. Hsu, W. J., Huang, S. Y. & Wang, H. (2002) ‘Scheduling and routing algorithms for AGVs: a survey.’ *International Journal of Production Research*, 40, Number 3**,** 745 - 760

Qiu, L. & Hsu, W. J. (2001) ‘Scheduling of AGVs in a mesh-like path topology: a case study in a container terminal.’ *Technical Report CAIS-TR-01-35,*Nanyang Technological University, School of Computer Engineering, Centre for Advanced Information Systems.

Renaud, J., Laporte, G. &Boctor, F. F. (1996) ‘A tabu search heuristic for the multi-depot vehicle routing problem.’ *Computers & Operations Research,* 23**,** 229-235.

Sadeh, N. M. (1996) ‘Focused simulated annealing search: an application to job shop scheduling.’ *Annals of Operations Research,* 63(1996)**,** 77-103.

Sarker, B. R. &Gurav, S. S. (2005) ‘Route planning for automated guided vehicles in a manufacturing facility.’ *International Journal of Production Research* 43, Number 21, 4659 – 4683.

Seifert, R. W. K. & M. G. Wilson, J. R. (1998) ‘Evaluation of AGV routing strategies using hierarchical simulation.’ *International Journal of Production Research* 2003 - 2030.

Seo, Y., Lee, C. & Moon, C. (2007) 'Tabu search algorithm for flexible flow path design of unidirectional automated-guided vehicle systems.' *OR Spectrum,* 29**,** 471-487.

Shintani, K., Imai, A., Nishimura, E. & Papadimitriou, S. (2007) 'The container shipping network design problem with empty container repositioning.' *Transportation Research Part E: Logistics and Transportation Review,* 43**,** 39-59.

Singh, S. P. &Tiwari, M. K. (2002) 'Intelligent agent framework to determine the optimal conflict-free path for an automated guided vehicles system.' *International journal of Production Research,* Vol. 40**,** 4195-4223.

Solomon, M. M. (1987) 'Algorithms for vehicle routing and scheduling problems with time window constraints.' *Operations Research Letters,* 35,(1987)**,** 254-265.

Stahlbock, R. &Voß, S. (2008) 'Operation research at container terminals: a literature update.' *OR Spectrum,* 30**,** 1-52.

Steenken, D. (1992) 'Fahrwegoptimierung am ContainerterminalunterEchtzeitbedingungen.' *OR Spektrum,* 14**,** 161-168.

Steenken, D., Henning, A., Freigang, S. &Voß, S. (1993) 'Routing of straddle carriers at a container terminal with the special aspect of internal moves.' *OR Spektrum,* 15**,** 167-172.

Stentz, A. (1994), "Optimal and efficient path planning for partially-knownenvironments", Proceedings of the IEEE International Conference on Robotics and Automation, May 1994

Taghaboni-Dutta, F. &Tanchoco, J. M. (1995) 'Comparison of dynamic routing techniques for automated guided vehicle systems.' *International Journal of Production Research*. 33 Issue 10**,** 2653.

Taillard, E. D., Gambardella, L. M., Gendreau, M. &Potvin, J.-Y. (2001) 'Adaptive memory programming: a unified view of meta-heuristics.' *European Journal of Operational Research,* 135**,** 1-16.

Takeda, K., Tsuge, Y. &Matsuyama, H. (2000) 'Decentralized scheduling algorithm to improve the rate of production without increase of stocks of intermediates.' *Computers & Chemical Engineering,* 24**,** 1619-1624.

Tan, K. C., Lee, L. H., Zhu, Q. L. &Ou, K. (2001) 'Heuristic methods for vehicle routing problem with time windows.' *Artificial Intelligence in Engineering,* 15**,** 281-295.

Ulusoy, G., Sivrikaya-Serifoglu, F. & Bilge, U. (1997) 'A genetic algorithm approach to the simultaneous scheduling of machines and automated guided vehicles.' *Computers & Operations Research,* 24**,** 335-351.

Van Der Heijden, M., Ebben , M., Gademann, N. & Van-Harten, A. (2002) 'Scheduling vehicles in automated transportation systems: algorithms and case study.' *OR Spectrum*, 24, Number 1, 31 - 58

Vis, I. F. A. (2002) Planning and control concepts for material handling systems. ERIM PhD. *Series Research in Management 14, E*rasmus University Rotterdam, Netherlands. http://www.irisvis.nl/ivis/

Vis, I. F. A. &Koster, R. D. (2003) 'Transshipment of containers at a container terminal: an overview.' *European Journal of Operational Research* 147(1)**,** 1-16

Vis, I. F. A. (2006) 'Survey of research in the design and control of automated guided vehicle systems.' *European Journal of Operational Research,* available online 2 November 2004. http://www.irisvis.nl/ivis/

Wagner D, Willhalm T, Zaroliagis C (2003) Dynamic Shortest Paths Containers. Theoretical Computer Science, 92. available online http://www.elsevier.nl/locate/entcs/volume92.html

Wallace, A. (2001) 'Application of AI to AGV control - agent control of AGVs.' *International Journal of Production Research,* 39(4)**,** 709-726

Wen-Chyuan Chiang, R. A. R. (1996) 'Simulated annealing meta-heuristics for the vehicle routing with time windows'. *Annals of Operation Research,* 63(1996)**,** 3-27.

Zeng, J. & Hsu, W. J. (2008) 'Conflict-free container routing in mesh yard layouts.' *Robotics and Autonomous Systems,* 56**,** 451-460.

Zhan, F. B. ,Noon, C. E. (1996), " Shortest Path Algorithms: An Evaluation using Real Road", Transportation Science vol. 32, no. 1, pp. 65-73

Zhang, C. Y., Li, P. G., Rao, Y. Q. & Guan, Z. L. (2008) 'A very fast TS/SA algorithm for the jobshop scheduling problem.' *International Journal of Computers & Operations Research,* 35**,** 282-294.

APPENDICES

Appendix 1

Pseudo code of SiPaMoP algorithm

```
Set pick-up node as labelled
Select adjacent node
Calculate arrival time (t) at adjacent node

If labelled or adjacent node occupied at time (t)
    Update weight of labelled and adjacent node
Else
    Relaxation
End
If all nodes scanned
    If all nodes labelled
        Register the shortest path with travelling time
    Else
        Select unlabelled node
        Select adjacent node
    End
End
```

Appendix 2

Pseudo code of Simulated Annealing Algorithm

```
Step 1: Generate an initial random or heuristic solution S.
          Set an initial temperature T, and other cooling schedule parameters

Step 2: Choose randomly S' ϵ N(S) and compute $\Delta = C(S') - C(S)$

Step 3: If:
        (i)    S' is better than S($\Delta < 0$) , or
        (ii)   S' is worse than S but "accepted" by the randomization process at the
               present temperature T, i.e. $e(-\Delta/T) > \theta$, (where $0 < \theta < 1$ is a random
               number).
        Then replace S by S'.
        Else Retain the current solution S.

Step 4: Update the temperature T depending on a set of rules, including:
        (i)    The cooling schedule used
        (ii)   Whether an improvement was obtained in Step 3 above,
        (iii)  Whether the neighborhood N(S) has been completely searched.

Step 5: If a "stopping test" is successful stop, else go to Step 2.
```

Appendix 3

Pseudo code of Ant Colony Algorithm

```
Initialize Data
While (not terminate) do
    ConstructSolutions
    LocalSearch
    UpdateStatistics
    UpdatePheromoneTrails
end-while
end-procedure
```

(Procedure to initialize the algorithm)

```
Procedure
InitializeData
    ReadInstance
    ComputeDistances
    ComputeNearestNeighborLists
    ComputeChoiceInformation
    InitializeAnts
    InitializeParameters
    InitializeStatistics
end-procedure
```

Appendix 4

Pseudo code of Auction Algorithm

```
Generate task sequence

For i = 1 to total no. of tasks

    Broadcast task to vehicles
    For j = 1 to No. of vehicles
        Calculate respective task completion times using SiPaMoP algorithm
        Each vehicle bid for respective task

        If all bids received
            Select best vehicle for task
        End

        If all tasks allocated
        Then
            Display results
        End
    End
End
```

Appendix 5

Simulation 1

Variation of rescheduling intervals with tardiness, late tasks and tasks scheduled

Trail number	Schedule interval	Batch size	Makespan	Tardiness	Late tasks
1	10	12	101.55	36.45	5
2	20	12	104.06	66.62	11
3	30	12	166.72	174.45	12
4	40	12	130.99	165.51	10
5	50	12	141.49	245.51	10
6	10	16	123.44	80.02	8
7	20	16	132.76	155.75	11
8	30	16	139.88	231.79	11
9	40	16	144.63	163.68	11
10	50	16	146.09	321.34	14
11	10	20	116.10	77.78	6
12	20	20	119.25	105.56	8
13	30	20	118.08	251.33	11
14	40	20	126.64	200.53	11
15	50	20	141.74	267.47	12
16	10	24	97.66	137.89	10
17	20	24	122.91	99.51	10
18	30	24	126.09	184.54	10
19	40	24	144.26	174.02	10
20	50	24	131.58	155.52	9

Simulation 2

Completion times and tardiness of tasks up to four reschedules using dispatch rule

Task number	Task priority level	Scheduled interval	Available time (stu)	Expected completion time (stu)	Completion time (stu)	Tardiness (stu)
1	Highest	Start	At start	20	9.02	0.00
2	Highest	Start	At start	20	6.35	0.00
3	Highest	Start	At start	20	12.39	0.00
4	Highest	Start	At start	20	9.42	0.00
5	Highest	Start	At start	20	15.65	0.00
6	Highest	Start	At start	20	23.40	3.40
7	Highest	Start	At start	20	4.57	0.00
8	Highest	Start	At start	20	20.38	0.38
9	Medium	Start	At start	40	44.74	4.74
10	Medium	Start	At start	40	48.94	8.94
11	Medium	Start	At start	40	20.36	0.00
12	Medium	Start	At start	40	31.62	0.00
13	Medium	Start	At start	40	21.36	0.00
14	Medium	Start	At start	40	40.52	0.52
15	Medium	Start	At start	40	35.90	0.00
16	Medium	Start	At start	40	32.13	0.00
17	Highest	Start	At start	60	55.49	0.00
18	Low	Start	At start	60	66.02	6.02
19	Low	Start	At start	60	82.08	22.08
20	Low	Start	At start	60	62.38	2.38
21	Low	Start	At start	60	55.01	0.00
22	Low	Start	At start	60	55.78	0.00
23	Low	Start	At start	60	65.89	5.89
24	Low	Start	At start	60	88.12	28.12
25	**Low**	**1st**	20	80	**52.86**	**0.00**
26	**Medium**	**1st**	20	60	**35.19**	**0.00**
27	**Highest**	**1st**	20	40	**30.37**	**0.00**
28	**Low**	**1st**	20	80	**44.89**	**0.00**
29	**Low**	**2nd**	40	100	**66.93**	**0.00**
30	**Medium**	**2nd**	40	80	**46.54**	**0.00**
31	**Low**	**3rd**	60	120	**79.92**	**0.00**
32	**Medium**	**3rd**	60	100	**71.31**	**0.00**
33	**Highest**	**3rd**	60	80	**73.45**	**0.00**
34	**Low**	**3rd**	60	120	**83.77**	**0.00**
35	**Highest**	**3rd**	60	80	**76.97**	**0.00**
36	**Low**	**4th**	80	140	**89.64**	**0.00**
37	**Low**	**5th**	100	160	**107.97**	**0.00**
38	**Medium**	**5th**	100	140	**107.45**	**0.00**
39	**Highest**	**5th**	100	120	**111.12**	**0.00**
					Total tardiness	**82.47**

Completion times and tardiness of tasks up to four reschedules using AA

Task number	Task priority level	Scheduled interval	Available time (stu)	Expected Completion time (stu)	Completion time (stu)	Tardiness (stu)
1	Highest	Start	At start	20	11.64	0.00
2	Highest	Start	At start	20	6.35	0.00
3	Highest	Start	At start	20	25.24	0.00
4	Highest	Start	At start	20	17.06	0.00
5	Highest	Start	At start	20	33.27	3.27
6	Highest	Start	At start	20	11.05	0.00
7	Highest	Start	At start	20	4.57	0.00
8	Highest	Start	At start	20	11.86	0.00
9	Medium	Start	At start	40	57.97	0.00
10	Medium	Start	At start	40	29.99	0.00
11	Medium	Start	At start	40	62.71	2.71
12	Medium	Start	At start	40	8.48	0.00
13	Medium	Start	At start	40	22.59	0.00
14	Medium	Start	At start	40	38.83	0.00
15	Medium	Start	At start	40	55.07	0.00
16	Medium	Start	At start	40	52.34	0.00
17	Highest	Start	At start	60	87.23	0.00
18	Low	Start	At start	60	63.83	0.00
19	Low	Start	At start	60	35.64	0.00
20	Low	Start	At start	60	20.12	0.00
21	Low	Start	At start	60	42.65	0.00
22	Low	Start	At start	60	66.44	0.00
23	Low	Start	At start	60	72.65	0.00
24	Low	Start	At start	60	33.28	0.00
25	**Low**	**1st**	20	80	**50.67**	**0.00**
26	**Medium**	**1st**	20	60	**44.86**	**0.00**
27	**Highest**	**1st**	20	40	**43.98**	**0.00**
28	**Low**	**1st**	20	80	**75.14**	**0.00**
29	**Low**	**2nd**	40	100	**106.99**	**6.99**
30	**Medium**	**2nd**	40	80	**48.33**	**0.00**
31	**Low**	**3rd**	60	120	**91.38**	**0.00**
32	**Medium**	**3rd**	60	100	**87.05**	**0.00**
33	**Highest**	**3rd**	60	80	**73.56**	**0.00**
34	**Low**	**3rd**	60	120	**83.41**	**0.00**
35	**Highest**	**3rd**	60	80	**83.46**	**3.46**
36	**Low**	**4th**	80	140	**96.11**	**0.00**
37	**Low**	**5th**	100	160	**134.56**	**0.00**
38	**Medium**	**5th**	100	140	**125.96**	**0.00**
39	**Highest**	**5th**	100	120	**121.61**	**1.61**
					Total Tardiness	**5.98**

Completion times and tardiness of tasks up to four reschedules using SA algorithm

Task number	Task priority level	Scheduled interval	Available time (stu)	Expected completion time (stu)	Completion time (stu)	Tardiness (stu)
1	Highest	Start	At start	20	7.03	0.00
2	Highest	Start	At start	20	8.48	0.00
3	Highest	Start	At start	20	10.77	0.00
4	Highest	Start	At start	20	15.03	0.00
5	Highest	Start	At start	20	17.82	0.00
6	Highest	Start	At start	20	21.41	1.41
7	Highest	Start	At start	20	9.56	0.00
8	Highest	Start	At start	20	18.76	0.00
9	Medium	Start	At start	40	27.23	0.00
10	Medium	Start	At start	40	29.43	0.00
11	Medium	Start	At start	40	24.28	0.00
12	Medium	Start	At start	40	41.44	1.44
13	Medium	Start	At start	40	34.15	0.00
14	Medium	Start	At start	40	35.39	0.00
15	Medium	Start	At start	40	41.39	1.39
16	Medium	Start	At start	40	40.22	0.22
17	Highest	Start	At start	60	54.38	0.00
18	Low	Start	At start	60	51.40	0.00
19	Low	Start	At start	60	71.33	11.33
20	Low	Start	At start	60	53.73	0.00
21	Low	Start	At start	60	51.12	0.00
22	Low	Start	At start	60	79.57	19.57
23	Low	Start	At start	60	64.10	4.10
24	Low	Start	At start	60	67.83	7.83
25	**Low**	**1st**	20	80	**44.22**	**0.00**
26	**Medium**	**1st**	20	60	**45.01**	**0.00**
27	**Highest**	**1st**	20	40	**32.15**	**0.00**
28	**Low**	**1st**	20	80	**58.14**	**0.00**
29	**Low**	**2nd**	40	100	**66.11**	**0.00**
30	**Medium**	**2nd**	40	80	**49.86**	**0.00**
31	**Low**	**3rd**	60	120	**89.29**	**0.00**
32	**Medium**	**3rd**	60	100	**70.33**	**0.00**
33	**Highest**	**3rd**	60	80	**80.73**	**0.73**
34	**Low**	**3rd**	60	120	**83.97**	**0.00**
35	**Highest**	**3rd**	60	80	**77.03**	**0.00**
36	**Low**	**4th**	80	140	**90.23**	**0.00**
37	**Low**	**5th**	100	160	**0.00**	**0.00**
38	**Medium**	**5th**	100	140	**0.00**	**0.00**
39	**Highest**	**5th**	100	120	**0.00**	**0.00**
					Total Tardiness	**48.01**

Simulation 3

Planned from/To nodes information and assigned vehicles before the breakdown

Travel time between connections	From node	To node	Task number	Vehicle number
1.0	174	173	201	2
36.2	102	82	1	2
1.9	4	21	202	3
32.8	164	186	2	3
2.5	25	42	203	1
36.2	127	105	3	1
1.9	12	29	207	4
39.4	45	44	7	4
37.3	82	83	206	2
52.8	41	40	6	2
36.0	186	164	205	3
61.4	97	78	5	3
38.5	105	127	204	1
54.8	106	87	4	1
40.3	44	45	208	4
55.3	61	60	8	4

The travelling information of vehicle 1 before the breakdown instance of vehicle 4

Time between nodes	From node	To node	Task number
2.50	25	42	Reach task 3
5.20	42	61	
6.90	61	78	
8.40	78	97	
10.40	97	118	
12.90	118	137	
13.30	137	136	
15.50	136	135	
16.60	135	134	
18.00	134	133	
19.50	133	154	
21.20	154	178	
22.90	178	154	Perform task 3
24.40	154	133	
26.80	133	132	
27.80	132	131	
29.10	131	130	
30.40	130	129	
31.64	129	128	
33.78	128	127	
36.18	127	105	
38.51	105	127	Reach task 4
40.64	127	128	
41.94	128	129	
43.18	129	110	
44.44	110	111	
45.44	111	112	
46.44	112	111	Perform task 4
47.74	111	110	
49.04	110	129	
50.31	129	109	
52.47	109	108	
53.77	108	106	
54.77	106	87	

The travelling information of vehicle 2 before the breakdown instance of vehicle 4

Time between nodes	From node	To node	Task number
1.00	174	173	Reach task 1
2.00	173	167	
2.92	167	148	
4.37	148	128	
5.67	128	129	
6.90	129	130	
8.14	130	131	
9.14	131	132	
11.54	132	133	
13.04	133	154	
14.74	154	178	
16.44	178	154	Perform task 1
17.94	154	133	
19.34	133	134	
20.44	134	135	
22.64	135	136	
23.14	136	137	
25.64	137	118	
26.34	118	119	
28.34	119	120	
29.24	120	121	
30.74	121	122	
32.34	122	123	
34.34	123	102	
36.24	102	82	
37.30	82	83	Reach task 6
39.00	83	65	Perform task 6
42.10	65	64	
43.00	64	63	
45.00	63	62	
47.90	62	42	
49.82	42	41	
52.82	41	40	

The travelling information of vehicle 3 before the breakdown instance of vehicle 4

Time between nodes	From node	To node	Task number
1.90	4	21	Reach task 2
3.40	21	37	
6.10	37	57	
7.80	57	71	
9.30	71	90	
11.30	90	111	
12.00	111	131	
13.00	131	132	Perform task 2
15.40	132	133	
16.80	133	134	
17.90	134	135	
20.10	135	136	
20.60	136	137	
23.10	137	138	
23.70	138	139	
25.70	139	140	
26.60	140	141	
28.10	141	142	
29.60	142	164	
32.80	164	186	
35.99	186	164	Reach task 5
37.48	164	142	
38.98	142	141	
39.88	141	140	
41.88	140	139	
42.48	139	138	
44.98	138	137	
45.48	137	136	
47.68	136	135	
49.18	135	156	
50.18	156	157	
51.18	157	156	Perform task 5
52.68	156	135	
54.88	135	136	
55.38	136	137	
57.88	137	118	
59.79	118	97	
61.39	97	78	

The travelling information of vehicle 4 before the breakdown instance of vehicle 4

Time between nodes	From node	To node	Task number
1.90	12	29	Reach task 7
3.40	29	45	
6.10	45	64	
7.00	64	63	
9.00	63	62	
10.70	62	79	
12.20	79	98	
14.40	98	118	
16.85	118	137	
17.26	137	136	
19.46	136	135	
20.96	135	156	
22.46	156	135	Perform task 7
24.66	135	136	
25.16	136	137	
27.66	137	118	
29.76	118	98	
31.32	98	79	
32.92	79	62	
34.92	62	63	
35.82	63	64	
38.52	64	45	
39.42	45	44	
40.32	44	45	Reach task 8
43.02	45	64	
43.92	64	63	
45.92	63	62	
47.82	62	78	
49.30	78	97	
50.80	97	78	Perform task 8
52.50	78	61	
55.30	61	60	

Travelling information of the 1st vehicle after the breakdown instance of vehicle 4

Time between nodes	From node	To node	Task number
31.30	129	128	Task 3 continuation
33.44	128	127	
35.84	127	105	
38.51	105	127	Reach Task 4
40.64	127	128	
41.94	128	129	
43.18	129	110	
44.44	110	111	
45.44	111	112	
46.44	112	111	Perform Task 4
47.74	111	110	
49.04	110	129	
50.31	129	109	
52.47	109	108	
53.77	108	106	
54.77	106	87	

Travelling information of the 2nd vehicle after the breakdown instance of vehicle 4

Time between nodes	From node	To node	Task number
31.60	122	123	Task 1 continuation
33.60	123	102	
35.50	102	82	
37.30	82	83	Reach task 6
39.00	83	65	Task 6 performing
42.10	65	64	
43.00	64	63	
45.00	63	62	
47.90	62	42	
49.82	42	41	
52.82	41	40	

Travelling information of the 3rd vehicle after the breakdown instance of vehicle 4

Time between nodes	From node	To node	Task number
33.20	186	164	Continuation of task 2
34.69	164	142	
36.49	142	123	
38.44	123	102	
39.94	102	83	
41.74	83	65	
44.84	65	49	
46.00	49	54	
48.40	54	33	
50.90	33	34	
52.50	34	50	
35.99	186	164	Reach Task 5
37.48	164	142	
38.98	142	141	
39.88	141	140	
41.88	140	139	
42.48	139	138	
44.98	138	137	
45.48	137	136	
47.68	136	135	
49.18	135	156	
50.18	156	157	
51.18	157	156	Perform Task 5
52.68	156	135	
54.88	135	136	
55.38	136	137	
57.88	137	118	
59.79	118	97	
61.39	97	78	
62.79	78	97	Reach task 8
64.29	97	78	Perform Task 8
65.99	78	61	
68.79	61	60	

Appendix 6

Master pseudo code:

```
numrows=sizeof(uwvpmat)
for all vehicles(n)
MPI_Send(numrows,vehicle(n),TAG,NEWORLD)
MPI_Send(uwvpmat,vehicle(n),TAG,NEWORLD)
```

Vehicle (worker) pseudo code:

```
MPI_Recv(numrows,master,TAG,NEWORLD)
uwvpmatrix=zeros(numrows,numcols)
MPI_Recv(uwvpmatrix,master,TAG,NEWORLD)
```