Multi-Agent Path Planning for Mobile Robots With Discrete-Step Locomotion

UNIVERSITÀ DEGLI STUDI
DI GENOVA

Keerthi Sagar SOMENEDI NAGESWARA RAO

Department of Mechanical Engineering

University of Genoa, Italy

This dissertation is submitted for the degree of

Doctor of Philosophy

December 18, 2017

Acknowledgements

Firstly, I would like to thank Professoressa Rezia Molfino for giving me the opportunity to be a part of the PMAR Robotics Group. I would like to thank Prof. Matteo Zoppi (primary supervisor) who has been really supportive and has made himself available at all times for both professional and personal help. My most sincere gratitude to Prof. Dimiter Zlatanov for his very critical feedback on both my work and academic writing. I am particularly grateful to Prof. Cristiano Nattero, who provided me all the technical guidance to carry forward my research and without whom this thesis work would not have been possible. I would like to offer my special thanks to Prof. Sreekumar, for supporting me to start this Phd.

I would like to express my gratitude to Prof. Wojciech Szynkiewicz, and Prof. Włodzimierz Kasprzak for giving me the opportunity to work with the research group in Warsaw University of Technology, Poland. Their valuable and constructive feedback provided me important insights in my research work.

I am most grateful to Giovanna Naselli without whom I never would have been able to start my research in Italy. I would also like to thank Aiko Dinale for helping me navigate all the bureaucracies and aiding me to settle in Genoa. I am thankful to my fellow lab mates Cuong, Vishal, Sadiq, Ahmad, Mikka, and Giorgio for their friendship and support during my stay in the lab. In particular, my sincere thanks to Hiram Lugo for his moral support during my research work and thesis submission.

Finally, yet importantly my heartfelt thanks to my parents and my brother for supporting and believing in me.

Abstract

The "swing-and-dock" (SaD) model for realizing displacements has been invented for and is used by the mobile robotic fixtures developed in the SwarmItFix European project. This form of locomotion can be a valuable capability for material handling agents, and fixturing agents enabling simultaneous handling in a non-linear fashion and increasing manufacturing flexibility. The thesis focuses on the design of SaD path planning algorithms for the motion of the agent. Five major objectives for the SaD agent are identified, namely task allocation, minimizing the makespan (total time till last agent/robot reaches its goal), minimizing the total steps/movement of each agent (idle states), handle orientation constraints and action planning when a trajectory is provided. The contribution of this thesis are as follows: The planning problem is modeled as a graph and first single agent planning problem is addressed. Various local search techniques were employed, the results suggest that a Nearest Neighbor with a Random Insert Heuristic approach allows the generation of good solution sequences for a single agent visiting a varied size of destinations. This result can be extended to address the task allocation objective. For the multi-agent path planning (MPP) with makespan, idling, and multiple goals objectives: two Integer linear programming (ILP) formulations based on vertex and edge of a graph are developed. Computation results show that the vertex based approach proved to be superior in the SaD agent context. The vertex-based approach is extended to address the orientation constraints for the SaD agent when the agent payload is not completely symmetrical. Formulation's effectiveness is shown with measures such as time and distance optimality ratio, where experiments display solutions closer to $1.x$ of the optimal solution. Finally, constrained optimization is employed for motion planning of the agents in a fixturing environment. The formulations, tailored to the SaD system, are general enough to be applicable for many other single- and multi-agent problems over discretized networks.

Contents

List of Figures

List of Tables

List of Abbreviations

AGVs Automated Guided Vehicles. 15, 18, 19

APSP All Pair Shortest Path. 47, 52, 71, 72, 77, 78

BFS Breadth First Search. 24, 25, 27, 41

CAN Controller Area Network. 4

CGAL Computational Geometry Algorithms Library. 46

CI Confidence Interval. 34

CSP Constrained Satisfaction Problem. 22, 23

FEM Finite Element Method. 93

FMS Flexible Manufacturing System. 15

HDCC Highly Distributed Coordination Control System. 16

ILP Integer linear programming. 43, 46, 48, 58, 59, 64, 66, 67, 70, 71, 78, 81, 82, 87, 94, 109, 110

KSI Kinetic Science Inc. 16

LOCOBOT Low-Cost-Robot-Co-Workers. 16

MASMICRO Integration of manufacturing systems for mass-manufacture of miniature/micro-products. 20

MHS Material handling system. iii, ix, 4, 6, 15–25, 43, 51, 76

MHSs Material handling systems. 15, 16

MiCRoN Miniaturised Co-operative Robots advancing towards the Nano range. 20

MILP Mixed integer linear programming. 43

MPP Multi-agent path planning. 6, 43, 44, 58, 61, 62, 64, 67, 82, 87, 109, 110

MRROC++ Multi-Robot Research-Oriented Controller. 4

MRS Multi-robot systems. 3, 4, 6

NN Nearest Neighbor. vii, viii, 26, 29, 31, 35–37, 39, 40

NP Non-deterministic polynomial time. 24, 42–44, 67

PKM Parallel Kinematic Machine. 1, 85, 86

PSPACE Polynomial Space. 44

PZT Piezo-electric. 20

R Random. 29, 40

REMORA Reconfigurable mobile robot for manufacturing applications. 16

RPD Relative Percent Deviation. vii, viii, 34–40

RRT Rapidly-exploring random tree. 42

SaD Swing and Dock. vii–ix, 1, 3–6, 9, 11, 13, 15–24, 29, 40–45, 49, 50, 58, 59, 61, 62, 64, 65, 67–69, 71, 73, 75, 78, 79, 82, 85, 86, 94, 107, 109–111

SwarmItFix Self reconfigurable intelligent swarm fixtures. 1, 22, 23, 58, 78, 85, 86, 88

TSP Traveling Salesman Problem. 22–26

Chapter 1

Introduction

This thesis is based on a sub-system developed for the Self reconfigurable intelligent swarm fixtures (SwarmItFix), a mobile robotic fixture assembly project. SwarmItFix, was a project funded by the European Union (EU FP7-214678) within the 7th Framework Program. The SwarmItFix developed a mobile autonomous fixturing agents capable of periodically repositioning in order to support large dimension sheet metal workpieces in aerospace manufacturing. The mobile agents comprise of a head, a Parallel Kinematic Machine (PKM), and a mobile base. Two agents were commissioned and installed on a stationary bench. These agents were proposed as an alternative flexible solution [5, 6] for fixturing compliant sheet metals during machining operations in place of traditional mould fixtures or (multi-point positioning tooling [7] or pin-bed type with a matrix of support elements to provide support by conforming to irregular shaped sheet metals [8]). A virtual assembly of the SwarmItFix and its commissioned prototype is shown in Fig 1.1 and 1.2 respectively.

The mobile base (sub-system) developed for the SwarmItFix was used to carry an hybrid architecture PKM and accurately position the fixture heads at the points where the workpiece needed support. The mobile base acted as the major contributing factor for the reconfigurability of the fixture. A novel locomotion method was developed for the base [1], where robots swing around stationary pivot pins on the bench, was invented and patented [9] in the current project. We coin the term "Swing and dock" (SaD) [3] to refer the locomotion of these base agents.

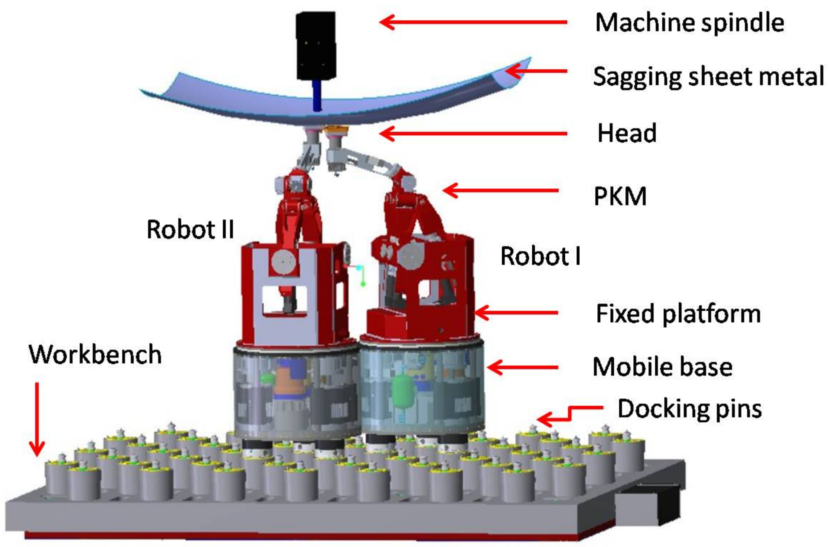


Figure 1.1: Virtual assembly of prototype: SwarmItFix

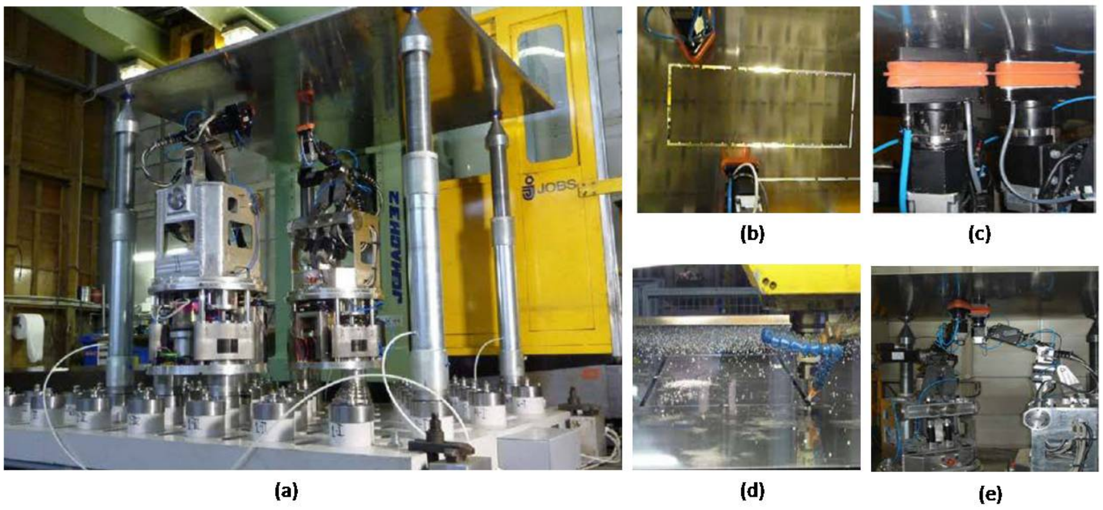


Figure 1.2: SwarmItFix Prototype (a) SwarmItFix under a five axis CNC Machine-tool; (b) Two agents supporting a workpiece (c) Close up of the heads (d) Through all-milling (e) Final Contour

The mobile base consists of three legs placed on mounting pins, in an equilateral triangle arrangement. For movement of the robot, rotations in multiples of 60° are performed around one of the legs while the other two legs are disengaged and lifted from the docking pins. Electric and pneumatic supply is available to the robot through the docking pins. A manufactured sample prototype of the base is shown in Figure 1.3. Upon investigating the architecture of the base, it is observed that these agents, even as a separate entity are valuable to the class of Multi-robot systems (MRS). By identifying the core characteristics of MRS, we can objectively state if the SaD system can be a part of the MRS class.

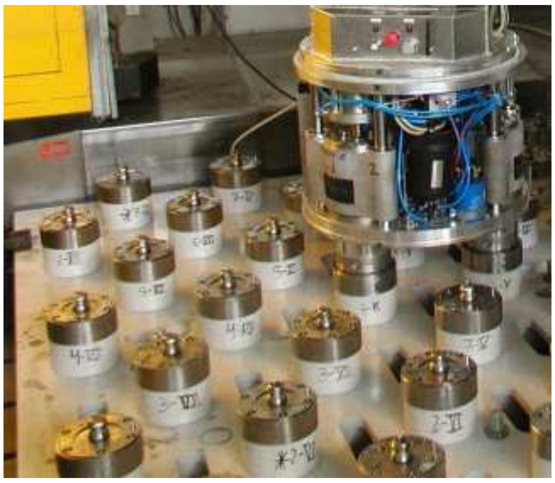


Figure 1.3: Prototype of the mobile base

1.1 Multi-robot system

MRS are becoming one of the important topic in current robotics research. The increased expertise in sensors, electronics and hardware technology has fueled the growing interest in MRS. MRS have applications in various domains such as autonomous sensor networks, building surveillance, transportation of large objects, air and underwater pollution monitoring, search and rescue operations in large-scale disasters, and several other areas [10]. In these application scenarios, MRS can often deal with tasks that are difficult, if not impossible, to be accomplished by an individual robot. Hence, implying redundancy in number of robots to be a major contributing factor towards cooperatively solving an assigned task. MRS are used when they can perform an assigned task in a more reliable, faster, or cheaper way beyond what is possible with single robots [11]. Such a collection of smaller, simpler robots are sometimes described [11] as *swarm* [12], a *colony*, or the robots may be said to exhibit *cooperative behaviour* [13]. While referring to MRS, we signify a fairly complex mobile platforms, equipped with sophisticated sensors and actuators, able to execute complex tasks rather than a simple machine/low level entity [14].

1.1.1 Characteristics

Various authors have established taxonomies in different research axes to understand multi-robot systems such as; Dudek et al. [13] presented a taxonomy that classified multiagent efforts according to communication, computational and other abilities, Cao et al. [15] provided classification based on: group architecture, resource conflicts, origins of cooperation, learning and geometric problems, and finally Iocchi et al. [16] proposed a taxonomy based on: a cooperation level, a knowledge level, a coordination level, an organization level, communication and team composition.

From literature, it is evident that the core requirement to be considered as a MRS is the availability of a control architecture which can either be centralized or decentralized, communication to enable either explicit or implicit interaction between mobile agents, and locomotion capabilities. It is also observed that homogenous multi agents result in a less complex system rather than heterogenous systems. SaD system exhibits all the mentioned characteristics, where a central control architecture has been established between the agents (on-board computer) and a central coordinator (host computer) through Controller Area Network (CAN) bus, and Multi-Robot Research-Oriented Controller (MRROC++) [17]; a dedicated robot programming framework. The host computer and the on-board computers communicate through the wireless Ethernet. The software distributed over this network is agent-based [18]. Also the SaD system possess a locomotion capability similar to the modular reconfigurable robots, where agents are homogenous in nature and can dock with each other. Hence, it is clear that the SaD base can be considered as an MRS agent. A very specific application for the SaD system for MHS is discussed in the further chapters.

1.1.2 Task Objectives

Multi-agents generally deal with different types of task objectives. Ota [19], categorizes the types of tasks based on various dimensions of the tasks and the number of tasks that need to be achieved. The word 'dimension' was used to compare the expressive form of the task goal. Ota [19], enlists three different types of dimensions of goal state: Point-reaching tasks (zero-dimension), Region-sweeping tasks (more than one-dimension), compound tasks. Point-reaching tasks express a specific configuration of the robot/point of the robot in a certain coordinate system as the goal state. This is the most common motion planning problem for multi-robot systems. Point-reaching tasks are stated to be zero-dimension due to their corresponding point in the configuration space, Eg. Motion-planning, and Pattern formation. Region sweeping tasks, as the name suggests, the goal task for the agent was to reach a specific region (considered to be multi-dimension, Eg. Sweeping, Map generation). Compound tasks were defined to be a combination of Point-reaching and Region sweeping tasks. The number of iterations of tasks that need to be performed were classified as: *one-time*, and *many-times*. When reviewing the literature on the research carried out in multi-robot systems, most involved 'one-time' and 'point-reaching tasks'. Several strategies of path planning/motion planning were developed for this particular scenario, such as the Visibility graph [20], Dynamic priority method [21], several methods which are found in this survey [22]. By observing the SaD system we can arrive at the conclusion that a point-reaching task objective would be most suitable, since the positions a SaD agent can reach is discrete in space and also the operating scenario is a known environment. Most common objective for these scenarios are (i) Generate a path/trajectory for each agent to reach their corresponding goals (ii) Avoid collisions among the robots and the environment. Apart from these, secondary objectives are addressed simultaneously during the planning such as: from scheduling literature: makespan minimization (total time till last agent/robot reaches its goal) [23, 24], minimize total number of moves of each agent, minimize overall path length [25, 26], and minimize total mission cost (energy consumption) [27, 28]. To prove the effectiveness of the method/algorithms implemented, computation times are reported.

In the further chapters, the thesis would aim to address specifically the makespan minimization and the total distance traveled by the agents.

Below, I outline briefly the contribution in this thesis and the general approach taken in the thesis to solve them. A more detailed introduction to the locomotion methodology, is given in Chapter 2, where the SaD locomotion is presented.

1.2 Contributions

The detailed contribution of this thesis are:

1. Using pins for docking is by no means new (see for example [29]); the SaD approach is novel in the way a pin is employed as a pivot and support during the motion. We identify the possible extension of the applications scenarios for SaD agents such as MHS and to act as independent MRS.

2. A new path planning framework was developed specifically for the SaD based approach. The Multi-agent path planning (MPP) problem was addressed by developing ILP formulations, tailored specifically for the SaD system. The formulations are general enough to be applicable for many other single- and multi-agent problems over discretized networks. The formulations developed performed on par with the state of the art in computation time.

3. To extend the SaD agents planning, they are proposed to use these agents mounted with static fixture heads instead of a parallel manipulator (SwarmItFix EU FP7) used in previous works. Such a modification would ease the planning complexity and also would be cost-effective alternative. Constrained optimization with aid of integer linear programming is employed to ensure accurate positioning of the agents in time to provide adequate support during machining. Multi - agent path planning w.r.t the tool trajectory, collision avoidance, and time - relevant action plan is implemented.

1.3 Thesis Structure

This thesis is organized as follows.

Chapter 1: This chapter introduces various possible multi-robot systems in manufacturing and their corresponding locomotion. The motivation towards using SaD locomotion in a manufacturing scenario and the need to develop planning algorithms is explained in detail.

Chapter 2: This chapter details the design and characteristics of SaD locomotion.

Chapter 3: This chapter discusses the possibility of using SaD agents for a material handling system (MHS) environment. Such a justification supports the extension of SaD agent application from a very narrow operating scenario such as the fixturing system to a more generic operation scenario.

Chapter 4: This chapter proposes the planning framework for SaD agents. Design of SaD path planning algorithms for a single agent planning is discussed, where the planner aims at identifying minimum agent displacements to reach multiple assigned targets. Several possible strategies and solutions are presented, elaborated and tested via simulations.

Chapter 5: This chapter presents the multi-agent path planning for SaD agents with multiple goals. Two integer linear programming formulation based on extended temporal graph is introduced: Vertex based formulation and edge based formulation, to address the path planning problem. Simulation and computational results demonstrate and compare the effectiveness of these formulations.

Chapter 6: This chapter deals with planning for SaD agents taking into account the orientation symmetry of the goal states. The agents would be required to carry tools/materials/manipulators, where the orientation in which the agent arrives at the goal location becomes of prime importance. This article deals with labeled legs describing the orientation of the agents on the mounting pins, hence taking into account the orientation of the agent w.r.t the bench. Integer Linear Programming (ILP) formulations are extended to model this particular planning problem. Simulation results are presented.

Chapter 7: This chapter discusses the motion planning for the SaD agents. A multi - agent path planning w.r.t the tool trajectory, collision avoidance, and time - relevant action plan is implemented. We propose to use these agents mounted with fixture heads instead of a parallel manipulator (SwarmItFix EU FP7) used in previous works. Integer linear programming optimization techniques are employed to ensure accurate positioning of the agents in time to provide adequate support during machining. Constrained optimization are implemented to generate the action plan. Optimization results prove the effectiveness of the formulation and also provide insight for parameter tuning, based on which design decisions for selection of geometry of the fixture head, base, and tool speed can be made.

Chapter 8: Finally, this chapter summarizes the results observed in this thesis and presents the possible future research direction.

Chapter 2

Swing and Dock Locomotion

The term "swing and dock" (SaD) was coined [3] to refer to the locomotion method of the mobile agents, where every agent docks on fixed pins on a bench and moves by swinging around one of these fixed pins. The characteristics of the SaD agents are as follows:

2.1 Locomotion Methodology

A fixed base or bench is equipped with mounting pins placed in a particular pattern. Mobile SaD agents with leg like structures are docked on these mounting pins. The legs of the agent can engage/retract from the mounting pins with the aid of linear actuation. A central harmonic drive as shown in Fig. 2.1 with a central gear and a spur gear mechanism is used to transmit motion from the central drive to the mounting legs. The legs produce a rotary action around the mounting pins. For a rotation to be executed, only one leg should be docked onto the pin. The agent rotates around the docked pin to reach adjacent pins/locations. All displacements are realized by sequences of such *swing* steps. The locomotion methodology is described in Fig. 2.2. A schematic of the working principle is shown in Fig. 2.3.

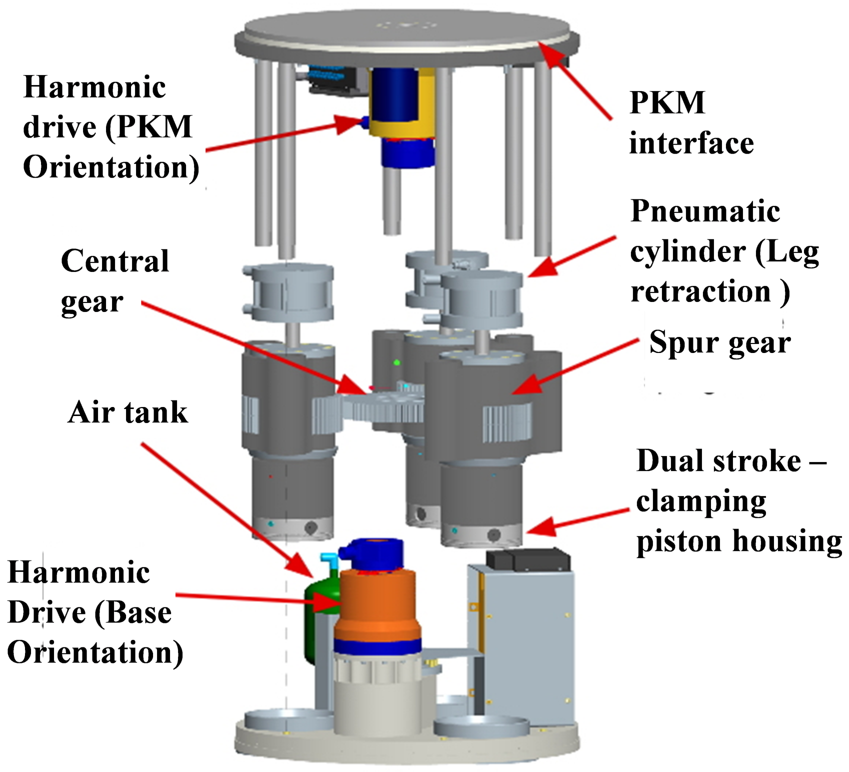

Harmonic drive (PKM Orientation)
PKM interface
Central gear
Pneumatic cylinder (Leg retraction)
Spur gear
Air tank
Dual stroke – clamping piston housing
Harmonic Drive (Base Orientation)

Figure 2.1: Locomotion components

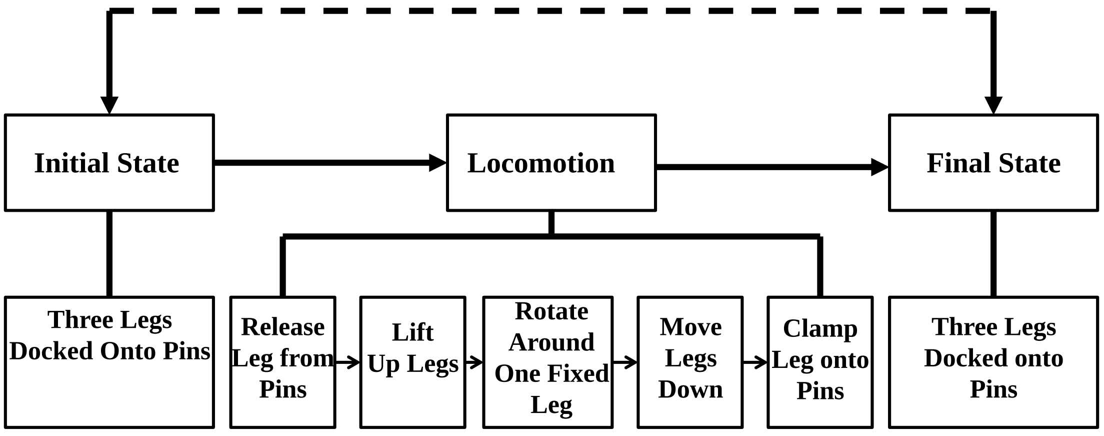


Figure 2.2: Locomotion methodology [1]

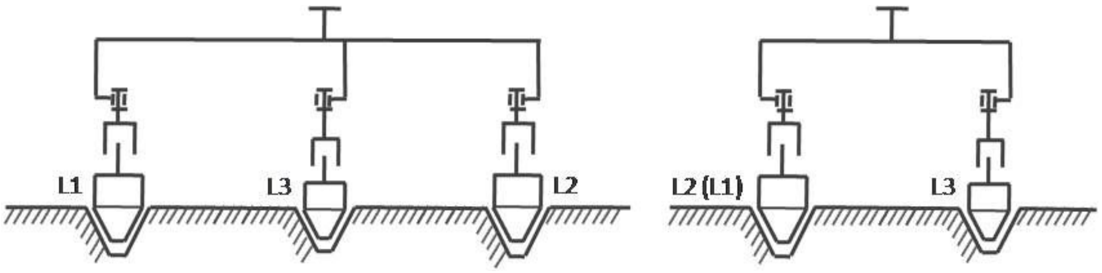
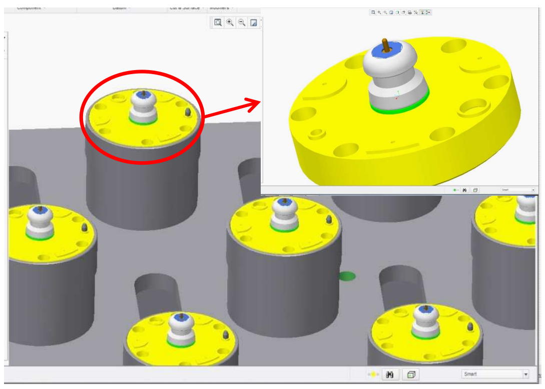
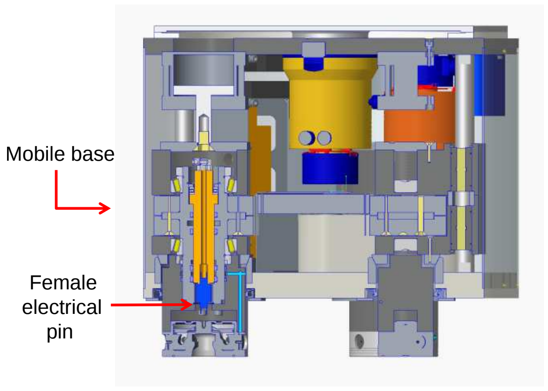


Figure 2.3: Schematic of the locomotion principle



(a) Male single pin connector



(b) Female electrical housing

Figure 2.4: SaD design elements

2.2 Design Characteristics

Two SaD agents were manufactured for prototype purpose. Two sub-systems can be identified from these agents: an active mobile agent (SaD), and a passive docking base (bench). The bench is made of steel with 52 docking pin modules. The agents are connected to the base with at least one engaged pin. The fixed bench accommodates all the components associated with providing pneumatic power to the mobile agent. And the bench pin houses a male electrical pin connector as shown in Fig. 2.4a. The active mobile agent consists of the female electrical pin connector as shown in Fig. 2.4b. As there is constant contact between the two sub-systems through the connected pin, there exists constant air and power being supplied to the active subsystem. An exploded view of the pin housing in Fig. 2.5 provides a clear view of the components.

The mobile base consists of three legs placed along the vertices of an equilateral

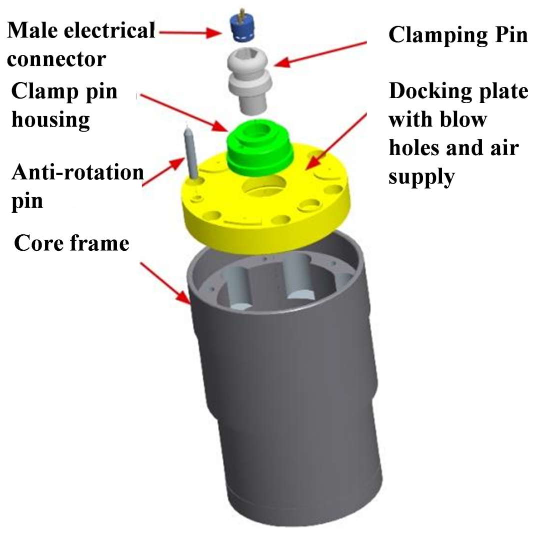
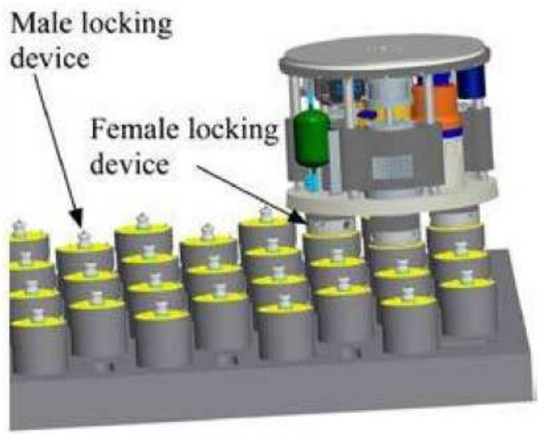

Male electrical connector
Clamping Pin
Clamp pin housing
Docking plate with blow holes and air supply
Anti-rotation pin
Core frame

Figure 2.5: Exploded view of docking pin housing

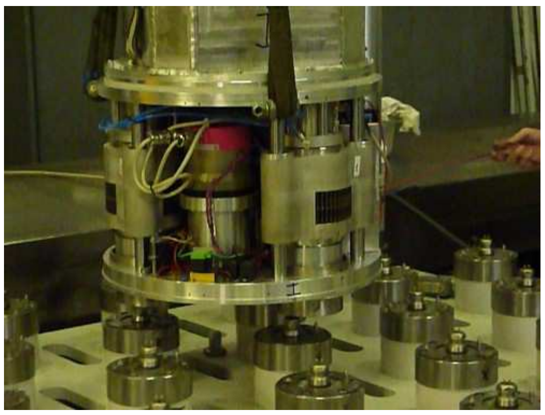
triangle. Agents rotate by $(\alpha)(60)^{\circ}, \alpha = \{1, 2, 3\}$ clockwise or counter-clockwise around the docking pin resulting in a new position of the agent. A pneumatic cylinder with a stroke of 45 mm which can be traversed in 0.5 seconds lifts the legs from the bench. Electric and pneumatic supply is available to the robot through the docking pins. This forms a cable- free environment and hence contributes to a Plug and Play type robot system. Rotation is around only one leg, the clamping force between the leg and the pin has to be greater than 10 kN. The current design enables a holding force of 75 kN. The docking pin (Fig. 2.5) components provide the necessary holding force of 75 kN and draw in force of 18 kN with accuracy close to 0.005 mm. Future requirements may demand different specifications which can be addressed by the hardware. Unlike common practice, the female electrical pin connector is placed underneath the agents to avoid accumulation of metal swarf produced during any manufacturing process. A detailed specification of the hardware is provided in Appendix A.

The bench also employs a device to blow the swarf generated during machining before the leg engages, hence maintaining a dust-free environment for the docking pins.

The SaD approach enables other possible configuration of the bench apart from flat surface (such as the ceiling or an inclined or vertical wall).



(a) Design



(b) Prototype

Figure 2.6: Mounting pins on SaD base [2]

Positional accuracy of the agents are very high as there is natural centering of the agent when docking. The harmonic drive with zero-backlashes further improves this accuracy.

The pin design also permits engaging the robot even if it tilts by an angle or with eccentricity. The base contributes for the high positioning accuracy of the robot without the need for an external sensor network and complex control system. The linear speed of SaD agents in the current design is 166.67 mm/s. Although the speed of these agents may be comparatively slow w.r.t state of the art gantry robots which achieve speeds of 2500 mm/s, they would be ideal in a multi-agent environment.

2.3 Proposed Extension

In principle, agents with two or more legs can be realized, and similarly other grid patterns on the bench could be used. Assuming an equilateral-triangle lattice, a regular-polygon agent base can be obtained for a three-leg mount (equilateral triangle) and a six-leg mount (regular hexagon) only. Various irregular polygons can be used as the base of an agent performing SaD locomotion with sixty-degree turns; some are shown in Fig 2.7.

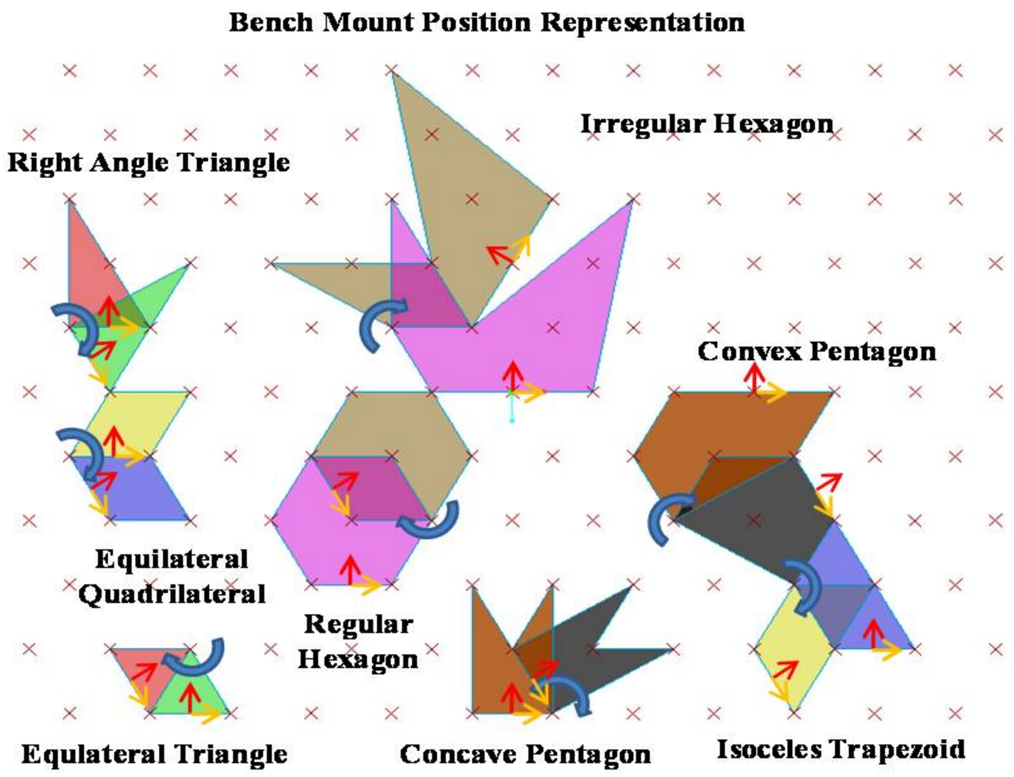


Figure 2.7: Various agent shapes on the same bench [3]

Chapter 3

Swing and Dock Locomotion for Material Handling

This chapter presents the characteristics of the SaD locomotion for the material handling system (MHS).

With rapid changes in current markets, designing for flexible manufacturing has become the norm. Technology supporting and aiding a Flexible Manufacturing System (FMS) have evolved over time with this key idea. Material handling systems (MHSs) development is an important part of this process.

Material handling plays a crucial role in many manufacturing processes. Hence, the analysis and design of systems that realize it efficiently has been of continuous research interest. Although conventional MHSs for transport and transfer of equipment remain important, there is an increasing interest in agent based flexible material handling [30]. This approach is based on the use of robots and manipulators, whose coordination and mobility play a key role in achieving flexibility. This chapter addresses precisely the agent's ability to move in a planned and coordinated manner within the manufacturing environment.

The most commonly used MHSs in the FMS environment are: (i) material-handling robots, (ii) Automated Guided Vehicles (AGVs), and (iii) Gantry robots [31]. Material handling robots/manipulators currently in use are stationary, working in a cellular layout. AGVs are typically researched and developed as transport equipment, and almost universally designed as wheeled locomotion systems. With some adaptations they are used for material handling tasks as guide-path mechanisms. They may be wire-, tape-, or paint-guided, or free ranging with a software-controlled path [32]. Gantry robots mounted on traversal beams are also common in practice.

Looking beyond these conventional agents for new approaches in flexible material handling has been an interesting research topic. Luntz and Messner [33], proposed a transfer table MHS referred to as Highly Distributed Coordination Control System (HDCC). It uses a conveyor-like array of actuators to manipulate objects in the plane while agents remain stationary. The Reconfigurable mobile robot for manufacturing applications (REMORA) [29] proposed a quadruped robot platform base with a novel locomotion locking strategy. Though wheeled robots are the norm in material handling, the efficient coordination framework established by Sugar and Kumar [34] provides a novel perspective to MHSs. A mechatronic mobile platform for flexible material handling has also been developed in [35]. The Low-Cost-Robot-Co-Workers (LOCOBOT) is a mobile robot with locomotion enabled by mecanum wheels which support longitudinal, latitudinal and rotation without steering [36]. The LOCOBOT was devised to act as a general-purpose co-worker and it can be used as a material handling device.The Kinetic Science Inc (KSI) tentacular manipulator uses a novel locomotion, mimicking tentacles, for general purpose material handling [37]. Systems aiding manual material handling based on exo-skeleton structures [38] have also been proposed. However, our focus here is on autonomous MHSs.

In this chapter, we propose the use of the SaD locomotion as the basis of a mobile-agent-based MHS. In this method, displacements are realized by 60-degree rotations about a discrete set of stationary pins. The use of a finite set of reachable locations is a novel approach diverging from more conventional solutions employing vehicles capable to continuously vary their paths. It can be asserted that this discrete locomotion ability is more than sufficient for the considered applications. Indeed, typically the displaced platform is adapted either (i) as the base on which a manipulator, with its own mobility, is mounted, or (ii) to transport materials to be used by a robotic system stationed at the final location. Hence, the locomotion system is needed respectively (i) to place the agent within reach of its next task, or (ii) to locate materials within a manipulator's workspace. Therefore, even when the system is gantry- or AGV-based, a continuous or high-density set of reachable positions is unnecessary and hence inefficient.

Thus, the proposed approach is an attractive alternative allowing fast and agile motions, for both transfer positioning and transport, with simpler and more robust control. Since the position of each pin is known a priori, SaD enables simple and precise localization without the uncertainties and complexities of odometry and external sensors. Figure 3.1 illustrates schematically the use of SaD agents for material handling in a manufacturing scenario with several machine centers.

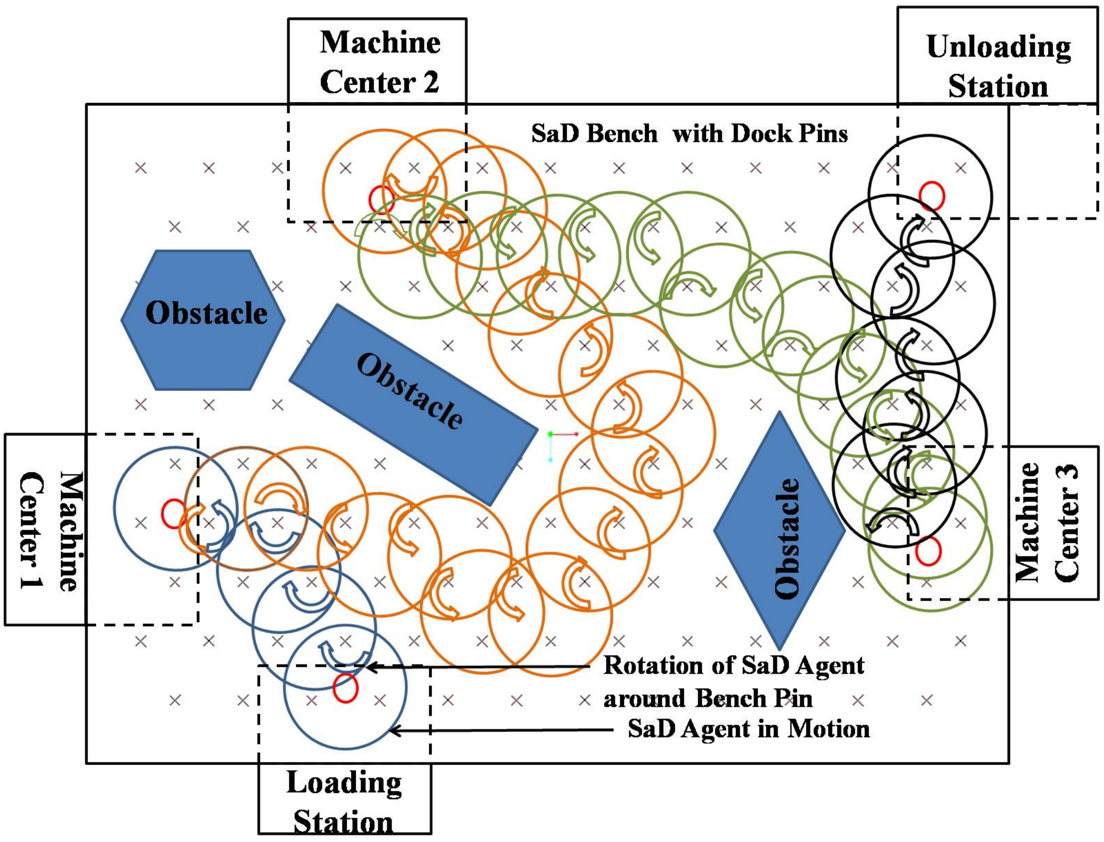


Figure 3.1: SaD agent in a Manufacturing Scenario [3]

3.1 Feasibility of the locomotion for MHS

In this subsection, the proposed model is compared with well-established approaches [39] to estimate its basic feasibility in an MHS environment. In Table

Table 3.1: Comparison of characteristics between SaD system and other MHSs

	AGVs	Gantry	Rail System	Conveyors	Robot	Forklift	SaD System
Load Type	Discrete	Discrete	Discrete	Continuous	Discrete	Discrete	Discrete
Flow Path	Bi-direction	Bi-direction	Uni-direction	Uni-direction	Stationary	Bi-direction	Bi-direction
Load Capacity	Medium	Low-Medium	High	Low-Medium	Low-Medium	High	Low-Medium
Size	Medium	Medium	Medium-Large	Small-Medium	Medium	Large	Low-Medium
Nature	Solid-Fragile	Solid	Solid	Solid	Solid-Fragile	Solid	Solid-Fragile
Speed of system	Medium	Low	High	Medium-High	Low-Medium	Medium	Low-Medium
Accumulation Requirements	No	No	No	Yes	No	No	No
Distance	Medium	Medium	High	Short-Medium	Short	High	Short-Medium
Frequency of move	Often	Low	Low	Low	Often	High	Often
Flexibility of path	High	Low	Low	Low	Low	High	Medium
Load/Unload	High	High	Medium	Medium	High	High	Medium

Table 3.2: MHS Characteristic [4]

Characteristics	SaD Model
Responsive	Changes in product being handled, work schedule, and load are met by modifications of agent geometry, number of agents and platform design. All these are controllable parameters in the current system
Flexible	Capability of transforming handling requirements is high due to SaD's flexibility in path
Autonomous	Distributed Control Architecture is possible
Highly Automatic	Capable to become highly automatic with efficient sensor network
Multi-Functional	Multi-function capability with manipulator mounting on platform
Modularized	Base platform modularization is possible resulting in different geometry lattice space for path
Multi-level	SaD can be integrated as a sub-system of a multi-level MHS cooperating with other independent material-handling equipment including AGVs and mobile robots.
Compatible	With ceiling mount, capability to interact with other MHSs is high but restricted to envelope of the agent

3.1, characteristic comparison is made between SaD and other non-manual MHSs as specified in [40]. Table 3.2 shows that the SaD system has the characteristics required for an Intelligent Manufacturing Environment [4].

Comparative ranking is performed on Table 3.1 on the basis of desirable attributes, such as having a flexible flow path, high load capacity, optimum size, solid robust nature, high speed, no accumulation requirements, high travel distance and low frequency moves, high flexibility in path, and high loading/unloading abilities. An SaD system falls in the category between robots and conveyors. The agent could be described as a type of AGVs with operational capability and without any human operator. The comparison suggests that the proposed

MHS model can offer a good solution for short-to-medium distance material handling. SaD can be particularly useful for realizing material movement in a U- or O- shaped cell layout. The system can effectively and accurately carry machine vises/materials to machining centers achieving the required MHS transfer equipment capabilities.

The SaD approach can provide locomotion capabilities for manipulators, which are similar to those of gantry-mounted robots, but with the added advantage of allowing simultaneous movement of multiple manipulators across the bench.

In principle, the SaD locomotion system can be miniaturized. For example, one could use micro spur gears for rotary motion of the legs [41], miniaturized harmonic drives for precision positioning [42], and piezoelectric (Piezo-electric (PZT)) translator/servo to enable linear translation for docking and undocking from the pins [43]. This would enable applications in the domain of desktop or micro/meso-scale manufacturing.

The locomotion setup with its regularly spaced docking points can be easily adapted to the grid-base on which micro-manufacturing units commonly operate. Precision is a very important technical aspect in the micro-factory concept [44], which SaD's priori knowledge of the exact docking positions will help to achieve. The pattern of the regularly spaced pin design can be altered to inculcate modularity [44] into the desktop manufacturing layout.

Micro-factories have an efficient utilization of space aiming to fit more machines in a limited space [44]. Hence, a transfer/handling system will need to access an increased number of destinations. Research has been carried out on carrier based transport systems such as the Automated Inter-Machine Material Handling system for the Integration of manufacturing systems for mass-manufacture of miniature/micro-products (MASMICRO) Project [45], Miniaturized Co-operative Robots advancing towards Nano Range (Miniaturised Co-operative Robots advancing towards the Nano range (MiCRoN)) [46], and many other MHSs.

3.2 Discussion

A novel “swing and dock” locomotion has been proposed for MHS agents. Comparative study between the properties of the existing MHSs and SaD indicates that the new system can act effectively as either transfer equipment or a mobile manipulator base. We propose a path planning framework for the SaD agent in the following chapter.

Chapter 4

Single Agent Planning

A path planner for SaD is quintessential to carry out material handling activities. From a material handling/positioning perspective, the primary objective is the displacement of the SaD agent towards an assigned set of transfer/machine stations. When an agent is assigned more than one goal station to visit, arriving at a decision of which order to follow becomes a task allocation problem. In essence, this is also a Traveling Salesman Problem (Traveling Salesman Problem (TSP)), where all the cities relate to the transfer/output stations and the agent represents the salesman which has to visit all output stations/cities within the shortest tour possible.

Within the SwarmItFix project, a Constraint Satisfaction Problem (Constrained Satisfaction Problem (CSP)) based motion planning has been proposed and implemented [47–49] for a pair of SaD-mobile fixturing robots supporting a machined thin sheet-metal part. Because of the many differences between the fixturing and material handling applications, efficiency requires the development of a new locomotion planner specifically targeted to the MHS conditions.

Indeed, the CSP planner is focused on its particular application, which imposes a wide variety of constraints, some of them not relevant to material handling. For instance, the locomotion of the mobile fixtures is planned on the basis of a given path/sequence of machine-tool poses [48]. Moreover, the SwarmItFix planner must guide not only displacement on the bench, but also, in an integrated way, the motion of the parallel manipulator (carried by the mobile SaD agent), whose end-effector must be precisely placed at the appropriate time to support adequately the machined part.

For the TSP-type task in material handling, we propose a planner which aims at identifying minimum agent displacements to reach the given targets. Hence, the planner is designed to consider the SaD agent as an individual entity and to plan its locomotion independently, rather than as an integrated part of the motion of a complex multi-stage and multi-agent system. In contrast, the CSP planner in [47], a complex multi-layered planning algorithm, guides two mobile SaD agents integrated with their mounted manipulators in a synchronized and coordinated manner, for a prescribed timed tool trajectory.

Unlike in the control of the SwarmItFix system, in the present MHS context one can identify two separate planning problems, with manipulator motion being performed after the agent (SaD) locomotion. This approach enables the use of simpler, faster, and more efficient algorithms. The separation of the tasks means that the planner we propose herein can be used both when the SaD agent transports materials and when it carries a manipulator.

4.1 Path Planning Model

As each SaD agent swings and docks on a bench with finite number of pins, the positions it can reach is discrete. Hence, the path planning can be modeled on a graph, $G = (V, E)$, Fig. 4.1. The graph is considered un-directed as the material handling agent executes bi-directional path movement.

The set of vertices (or nodes), V, are the possible positions of the centroid of the agent's footprint (e.g an equilateral triangle), Fig. 4.1.

The edges, E, of the graph indicate which vertices can be connected with one locomotion step, i.e, with a single rotation (of $\pm(\alpha)(60°)$, $\alpha = \{1, 2, 3\}$) around either of the legs, Fig. 4.1. In the current planning model in this chapter, we assume that there is no need to distinguish two different possible poses of the agent if they can be obtained from each other by rotation about the centroid. In some applications, this may not be the case, requiring a re-definition of G with a higher number of vertices, which will be discussed in the later chapters.

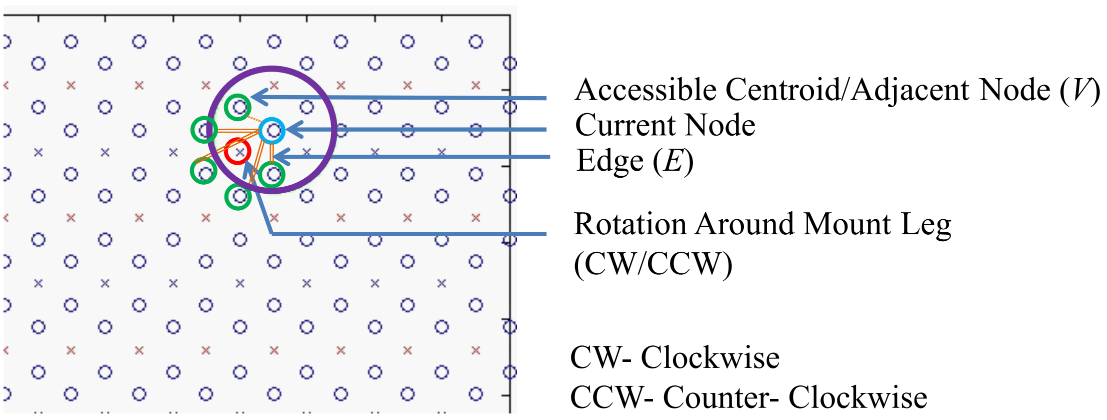

Accessible Centroid/Adjacent Node (V)
Current Node
Edge (E)
Rotation Around Mount Leg (CW/CCW)
CW- Clockwise
CCW- Counter- Clockwise

Figure 4.1: Planar representation of nodes in SaD

4.2 Problem Formulation

All notations and symbols used in this chapter will have its scope restricted to this chapter.

The MHS planning problem requires an agent to visit various target destinations with minimum total travel distance, i.e., the Non-deterministic polynomial time (NP)-hard TSP.

A Breadth First Search (BFS) is run on Graph G to determine the shortest path with the knowledge of the adjacency node list $\text{Adj}(v)$. A BFS yields the shortest path between the start node and goal node if all edges in the graph are of equal weight/un-weighted. In the current graph G, every edge is of unit length. Hence, it is convenient to utilize BFS to determine the shortest path between nodes.

BFS is run $(q + 1)$ times where q is the number of goal states, to compute the shortest path between the initial and each goal state, as well as between goal states.

The worst case time complexity for this BFS process would be similar to the All Pairs Breadth First Search complexity in an un-weighted graph, which is $O(mn)$ [50] when the number of goals equals the number of vertices in graph G (m and n denote the numbers of edges and vertices, respectively). This is comparable with the $O(n^3)$ complexity of algorithms such as the popular Floyd-Warshall for weighted directed graphs [51] and the $O(mn + n^2 logn)$ Johnson Algorithm for weighted directed graphs without negative cycles [52]. However, BFS has the advantage of computing shortest path only between source node and goal nodes unlike an all-pair comparison, which reduces both space and time complexity.

The cost distance matrix (M) for the computation of the shortest path is described in Procedure 1. The coordinates of the obstacles ($K = \{K_1, K_2, \cdots, K_\lambda\}$, where $K_i \in \mathbb{R}^2$) with respect to the bench are known; hence the nodes/centroids ($C = \{C_1, C_2, \cdots, C_{|V|}\}$, where $C_i \in \mathbb{R}^2$) in G inside the obstacle geometries are detected using the *inpolygon* function of Matlab solver which implements the Winding Number Algorithm [53]. The algorithm identifies the centroids inside/outside the obstacle geometry, where the centroids inside the obstacle are removed in the corresponding graph G, resulting in an obstacle free graph G'. The BFS is run considering each goal/source state in G' as the initial root node to find the minimum nodes to reach all other goal states in G' as described in Procedure 1 to provide both the minimum distance matrix, M, between origin $S = \{1\}$ and destinations $T = \{2, 3, \cdots, q+1\}$ and also the path sequence, P, where $P_i \in V'$ to reach the goal states. G'' is representative of the shortest distance between the source node S and $T' = \{2, 3, \cdots, q'+1\}$. TSP formulation and associated heuristics are performed on G''. The stages of graph G are represented in Fig. 4.2.

4.3 Heuristic Algorithms

The brute-force solution to the TSP compares the distances of all node permutations, calculated with the knowledge of M. It generates an optimal path sequence P.

Brute- force solving suffices when $n = |V|$ is small, e.g., 10 goal nodes with a Matlab solver. The time complexity is $O(n!)$ and so, for higher numbers of nodes heuristic algorithms are needed to obtain good solutions within a limited working time. For many large-scale MHSs, the brute-force approach is feasible. However, in some applications, such as micro-manufacturing, or when multiple orientations of the agent need to be considered, the number of nodes may be high. Moreover, when planning the coordinated displacements of several cooperating agents, the size of the equivalent TSP will grow considerably. With this motivation, in this

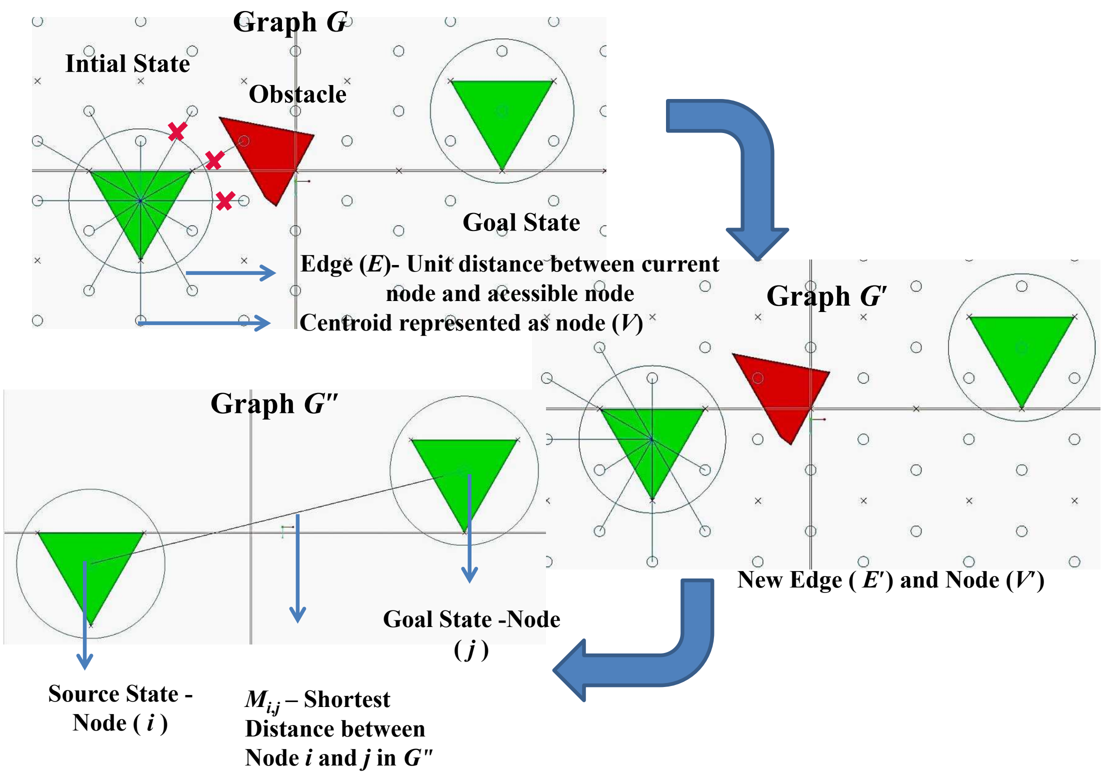

Graph G
Intial State
Obstacle
Goal State
Edge (E)- Unit distance between current node and acessible node
Centroid represented as node (V)
Graph G'
New Edge (E') and Node (V')
Graph G''
Goal State -Node (j)
Source State - Node (i)
$M_{i,j}$ – Shortest Distance between Node i and j in G''

Figure 4.2: Graph evolution [3]

and the following section we analyze the use of various local-search heuristics for this planning problem.

The Nearest neighbor (NN) heuristic to determine solutions for higher instances greater than 10 is described in Procedure 4. NN heuristic starts searching for the nearest goal state from the initial/source state. The nearness between goal states and source state is evaluated with the knowledge of Cost-Distance Matrix M. The nearest goal state is then considered as the next current state in the solution sequence φ_{sol}.

Similar iterations are done till all reachable goal states are available in the solution sequence φ_{sol}. A constructive heuristic such as NN provides a quick initial solution although the solution is greedy in nature and sub-optimal, local search heuristics can improve the incumbent solution available. Basic moves such as swap and insert have been used for local search to improve the TSP solution. Local search being compared for the TSP are Systematic Swap (AlgA′), Random

Procedure 1 Cost-Distance Matrix (M) Determination

Input Geometric Properties and coordinates of Base mount,W and agent

1. Connected Graph $G = (V, E)$
2. Get Positions P_d of initial state S & goal destinations T, where $P_d \in \mathbb{R}^2$
3. Get number of Obstacles (λ) and Coordinates of each Obstacle K_i w.r.t bench.
4. Edge Deletion in graph G based on winding algorithm [53] with knowledge of obstacle coordinates K_i and centroid coordinates C_i
5. Obstacle free graph $G' = (V', E')$
6. BFS is run $(q + 1)$ times to obtain the shortest path between nodes (S, T) in G' using Adjacency list Adj(v)
7. All pair shortest path matrix (M) represented in graph G'' and path sequence is obtained

Swap (AlgA$'$), Systematic Insert (AlgB), Random Insert (AlgB$'$), Random Swap Random Insert (AlgC) which are described in the heuristic Procedure 6.

The Swap function described in Procedure 2 and Fig. 4.3 depicts the procedure to swap two goal state positions in the solution sequence φ giving a new solution sequence φ_{new}. The Swap positions are adjacent for a Systematic Swap ($AlgA$) and random for a Random Swap ($AlgA'$). A restart of the Swap function is done if the new distance d_{new} computed by f_D in Procedure 5 improves on the best known distance d_{best}. The best distance is then updated with the new distance, $d_{\text{best}} = d_{\text{new}}$ and so is the solution sequence $\varphi_{\text{best}} = \varphi_{\text{new}}$. The restart offers a better neighborhood for the search process to continue.

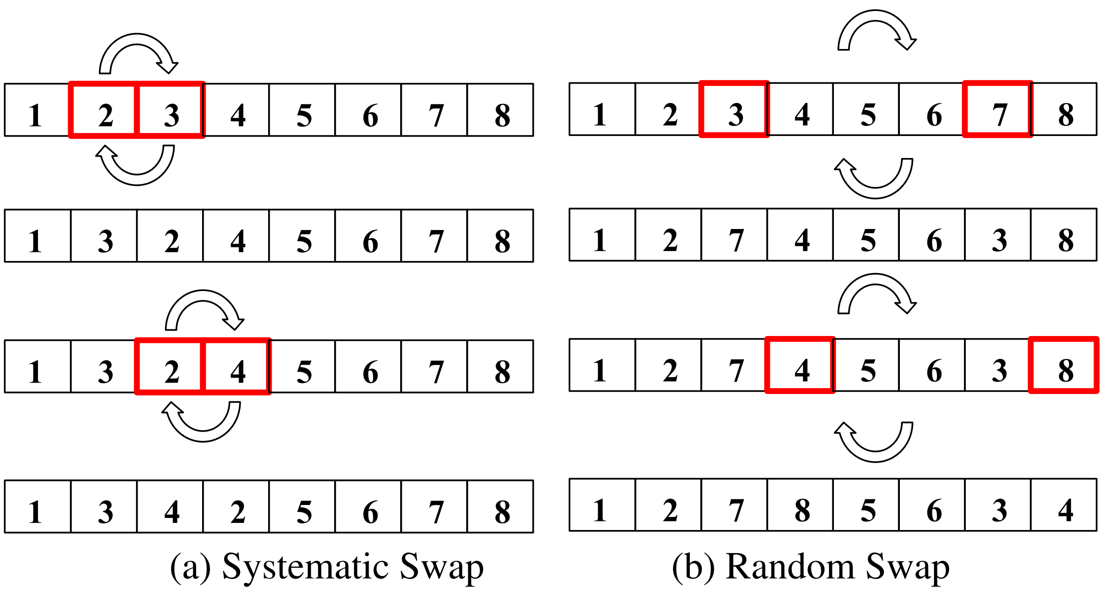


Figure 4.3: Representation of Swap function

Procedure 2 Swap Function f_{swap}

```
Input: Current solution sequence ($\varphi$),$j, k$
 1: function $f_{\text{swap}}(\varphi, j, k)$
 2:     $\varphi_{\text{new}} = \varphi$
 3:     $\varphi_{\text{new}}(k, j) = \varphi_{\text{new}}(j, k)$    ▷ Swap on $k$ and $j$ indices of $\varphi_{\text{new}}$
 4:     $d_{\text{new}} = f_D(M, \varphi_{\text{new}})$    ▷ Total Distance function $f_D$ for new solution $\varphi_{\text{new}}$
 5:     if $d_{\text{new}} < d_{\text{best}}$ then
 6:         $d_{\text{best}} = d_{\text{new}}$    ▷ Update new best distance
 7:         $\varphi_{\text{best}} = \varphi_{\text{new}}$    ▷ Update new best solution sequence
 8:         $\varphi = \varphi_{\text{best}}$    ▷ Do further swap on the new current solution sequence
 9:         break
10:     else
11:         $\varphi_{\text{new}} = \varphi$
12:     end if
13:     return $\varphi, d_{\text{best}}$
14: end function
```

The Insert function described in Procedure 3 and Fig. 4.4 is similar to the Swap function described above except that a selected Goal state is inserted into particular position rather than a swap.

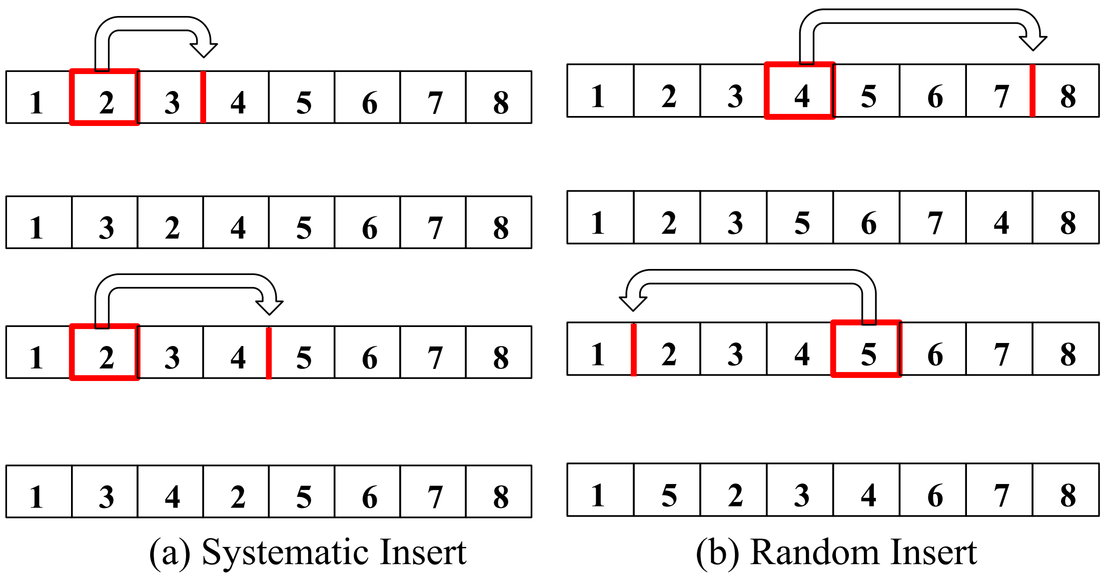


Figure 4.4: Representation of Insert function

These local searches are tested with an initial NN solution and a random solution (Random (R)). The time for one iteration t_{iter} is based on the maximum time to execute a systematic swap (AlgA) or systematic insert (AlgB). This time limit t_{iter} was obtained after repeated experiments on the data set to find an appropriate time limit t_{iter} to minimize premature termination of the systematic swap/insert.

4.4 Simulations

4.4.1 Setup

Simulations were conducted to test the effectiveness of the local searches Random Swap (AlgA′)/Insert (AlgB′), Systematic Swap (AlgA)/Insert (AlgB) and Random Swap Random Insert (AlgC) with initial solution as a Random Solution (R) and a Nearest Neighbor Solution (NN) and compare which local search heuristic provides the minimum travel distance for the SaD system.

For Random Swap (AlgA′), Random Insert (AlgB′) and Random Swap Random Insert (AlgC) heuristic, the procedures perform 10 iterations for each initial solution φ_{sol}. Mean of best distances d_{best} from the 10 iterations were represented

Procedure 3 Insert Function f_{insert}

```
Input: Current solution sequence ($\varphi$),$j, k$
 1: function $f_{\text{insert}}(\varphi, j, k)$
 2:     $\varphi_{\text{sol}} = \varphi$
 3:     if $j < k$ then
 4:         $b = \varphi_{\text{sol}(1,j)}$                ▷ Insert node $b$ into solution array to get $\varphi_{\text{new}}$
 5:         $\varphi_{\text{new}} = (\varphi_{\text{sol}}(1, 1 : k) \cup b \cup \varphi_{\text{sol}}(1, k + 1 : \mathit{end}))$        ▷ Total distance
    function $f_D$ for new solution $\varphi_{\text{new}}$
 6:         $d_{\text{new}} = f_D(M, \varphi_{\text{new}})$
 7:         if $d_{\text{new}} < d_{\text{best}}$ then
 8:             $d_{\text{best}} = d_{\text{new}}$                ▷ Update new best distance
 9:             $\varphi_{\text{best}} = \varphi_{\text{new}}$          ▷ Update new best solution sequence
10:             $\varphi = \varphi_{\text{best}}$ ▷ Do further insert on the new current solution sequence
11:             break
12:         end if
13:     else if $j > k$ then
14:         $b = \varphi_{\text{sol}}(1, j)$
15:         $\varphi_{\text{new}} = (\varphi_{sol}(1, 1 : k - 1) \cup b \cup \varphi_{\text{sol}}(1, k : \mathit{end}))$      ▷ Insert node b into
    solution array to get $\varphi_{\text{new}}$
16:         $d_{\text{new}} = f_D(M, \varphi_{\text{new}})$          ▷ Update new best solution sequence
17:         if $d_{\text{new}} < d_{\text{best}}$ then
18:             $d_{\text{best}} = d_{\text{new}}$                ▷ Update new best distance
19:             $\varphi_{\text{best}} = \varphi_{\text{new}}$          ▷ Update new best solution sequence
20:             $\varphi = \varphi_{\text{best}}$ ▷ Do further insert on the new current solution sequence
21:             break
22:         end if
23:     end if
24:     return $\varphi, d_{\text{best}}$
25: end function
```

Procedure 4 Nearest Neighbor (NN) Heuristic

```
Input: All Pair Shortest Path Matrix ($M$)
 1: Initialize array ($S \cup T$)
 2: Set starting index of $S$ and $T$, $i = S(1,1) \& j = T(1,1)$
 3: Initialize empty $\varphi_{\mathrm{sol}}$
 4: function $f_Q(i, \varphi_{\mathrm{sol}})$
 5:     while length($\varphi_{sol} \neq$ length($S \cup T$)) do
 6:         for $j = 1$ to length(($S \cup T$) $- \varphi_{sol}$) do
 7:             if $(i \neq j) \& (j \neq \varphi_{\mathrm{sol}}(:,:))$ then
 8:                 $D_{ij} = M_{ij}$
 9:             end if
10:         end for
11:         Get Minimum of $D_{ij}$ & corresponding $j$, where $j_{\mathrm{min}} = j$
12:         Update $i = j_{\mathrm{min}}$
13:         Call Function $f_Q(i, \varphi_{\mathrm{sol}})$
14:     end while
15: end function
Output: $\varphi_{\mathrm{sol}}$ is the sub-optimal solution sequence
```

Procedure 5 Total Travel Distance Evaluation f_D

```
Input: Cost-Distance Matrix ($M$) and solution sequence $\varphi_{\mathrm{new}}$
 1: function $f_D(M, \varphi_{\mathrm{new}})$
 2:     for $i = 1$ to length($\varphi_{\mathrm{new}}$)) - 1 do
 3:         $D = M_{\varphi_{\mathrm{new}}(1,i), \varphi_{\mathrm{new}}(1,i+1)}$
 4:         $d = d + D$
 5:     end for
 6:     $d_{\mathrm{new}} = d$
 7:     return $d_{\mathrm{new}}$
 8: end function
```

Procedure 6 Local Search Heuristic

```
Input: Iteration time $t_{\text{iter}}$
Input: $c$ = Choice of Heuristic
 1: $\varphi_{sol}$ = Nearest Neighbor Heuristic/ Random Solution
 2: $\varphi = \varphi_{\text{sol}}$
 3: $\varphi_{\text{best}} = \varphi$
 4: $d_{\text{best}} = f_D(M, \varphi_{\text{best}})$
 5:
 6: switch $c$ do
 7:     case A Systematic Swap (AlgA)
 8:         while $t_{\text{clock}} \leqslant t_{\text{iter}}$ do
 9:             for $j = 2$ to length($\varphi$) do
10:                 for $k = j + 1$ to length($\varphi$) do
11:                     Call $f_{\text{swap}}$
12:                 end for
13:             end for
14:         end while
15:     case A′ Random Swap (AlgA′)
16:         while $t_{\text{clock}} \leqslant t_{\text{iter}}$ do
17:             $j = \text{random}(2, \text{length}(\varphi))$
18:             $k = \text{random}(2, \text{length}(\varphi))$
19:             if $j \neq k$ then
20:                 Call $f_{\text{swap}}$
21:             end if
22:         end while
23:     case B Systematic Insert (AlgB)
24:         while $t_{\text{clock}} \leqslant t_{\text{iter}}$ do
25:             for $j = 2$ to length($\varphi$) do
26:                 for $k = 1$ to length($\varphi$) do
27:                     if $(j \neq k) \cap (j \neq (k-1))$ then
28:                         Call $f_{\text{insert}}$
29:                     end if
30:                 end for
31:             end for
32:         end while
```

```
33:     case B$'$ Random Insert (AlgB$'$)
34:         while $t_{\text{clock}} \leqslant t_{\text{iter}}$ do
35:             $j = \text{random}(2, \text{length}(\varphi))$
36:             $k = \text{random}(2, \text{length}(\varphi))$
37:             if $(j \neq k) \cap (j \neq (k-1))$ then
38:                 Call $f_{\text{insert}}$
39:             end if
40:         end while
41:     case C Random Insert Random Swap (AlgC)
42:         while $t_{\text{clock}} \leqslant t_{\text{iter}}$ do
43:             $P = \text{random}(0, 1)$
44:             switch P do
45:                 case 0 Random Swap
46:                     $c = \text{A}'$
47:                     Goto Switch Case $c, t_{\text{iter}} = t_{\text{clock}}$
48:                 case 1 Random Insert
49:                     $c = \text{B}'$
50:                     Goto Switch Case $c, t_{\text{iter}} = t_{\text{clock}}$
51:         end while
Output: $\varphi_{\text{best}}, d_{\text{best}}$
```

by (A′), (B′) and (C) and the least value of d_{best} among 10 iterations are represented as (A′A′), (B′B′) and (CC) respectively. For Systematic Swap and Systematic Insert the best distance d_{best} are recorded for a time interval t_{iter} which are represented as (A) and (B) and set of recordings when the systematic swap/insert terminates are represented as (AA) and (BB) respectively.

A number of conventional abbreviations are used in describing the simulation results in a concise way. In each name, the initial solution designation is followed by that of the local search heuristic. Thus, NNA – represents NN initial solution with Systematic Swap heuristic (AlgA), similarly RA–represents R initial solution with Systematic Swap heuristic (AlgA).

The iteration time for the while loop in the Local search heuristic Procedure 6 (t_{iter}) is assumed as 15, 30, 45 seconds. The number of nodes on the bench h_{bench} is taken as three sets as 50, 150, and 250 nodes respectively. The choice is made after repeated trials to minimize systematic swap/insert termination before completion. The upper limit on the number of goal nodes to visit is assumed to be ($\delta_{\text{out}} = 0.5h_{\text{bench}}$). Three instances of the number of outputs, with lower number $\delta_{\text{low}} = \text{random}(1, \delta_{\text{out}}/3)$, medium $\delta_{\text{medium}} = \text{random}((\delta_{\text{out}})/3, 2(\delta_{\text{out}})/3)$, and higher number of output instances $\delta_{\text{high}} = \text{random}(2(\delta_{\text{out}})/3, \delta_{\text{out}})$.

Obstacles are randomly generated $\lambda = \text{random}(0.015h_{\text{bench}})$. A sample with $n = 180$ iterations is simulated for each bench size, with 60 iterations for δ_{low} ,δ_{medium}, and δ_{high} respectively. Relative Percent Deviation (RPD) is used as a measure for the performance of the heuristics.

$$\text{RPD} = \frac{V_{\text{alg}} - V_{\text{bk}}}{V_{\text{bk}}}(100)(\%) \tag{4.1}$$

where $V_{\text{alg}} = d_{\text{best}}$ is the total distance calculated by the specific heuristic procedure for a corresponding φ_{best}, and V_{bk} is the best known minimum total distance among all heuristic procedure.

The confidence interval (Confidence Interval (CI)) of the RPD for the heuristic is calculated as follows:

$$\text{CI} = \mu + (x)(s_{\overline{x}}) \tag{4.2}$$

where x is Student's t inverse cumulative distributive function

$$x = F^{-1}(p|\nu) \tag{4.3}$$

$$p = F(x|\nu) = \int_{-\infty}^{x} \frac{\Gamma(\frac{(\nu+1)}{2})}{\Gamma(\frac{\nu}{2})} \frac{1}{\sqrt{\nu\pi}} \frac{1}{(1+\frac{t^2}{v})^{\frac{\nu+1}{2}}} dt \tag{4.4}$$

where in Equation 4.4, p lies in the range (0.025,0.975), $\nu = n$, and $\Gamma(.)$ are the desired probability, the degrees of freedom, and the gamma function respectively. The mean μ and standard error of the mean $S_{\overline{x}}$ are defined by the following equations

$$\mu = \frac{\sum_{i=1}^{n} (\mathrm{RPD})_i}{n} \tag{4.5}$$

$$S_{\overline{x}} = \frac{s}{\sqrt{n}} \tag{4.6}$$

The standard deviation, s is as follows

$$s = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} |(\mathrm{RPD})_i - \mu|^2} \tag{4.7}$$

4.4.2 Results

RPD is plotted for all heuristics versus their respective number of outputs. For 10 target nodes, the brute-force (abbreviated B.F in the figures) always provides the optimal solution. Results show that for less than 11 goal states in a 50 nodes bench, all the heuristic compared lie within 0.1% RPD from the optimal brute-force solution, with NN initial solution local searches even reaching optimality. The R initial solution with the local search heuristic starts to vary above 50% RPD for outputs lesser than 11 in a bench node size of 150 and 250 as shown in Fig. 4.9 and 4.12, whereas NN initial solution based heuristic continues to perform well within average 5% RPD for the above conditions as shown in Fig. 4.8 and 4.11. The B', A', and C with NN consistently generate optimal solution in this range of goal outputs with random insert outperforming the rest.

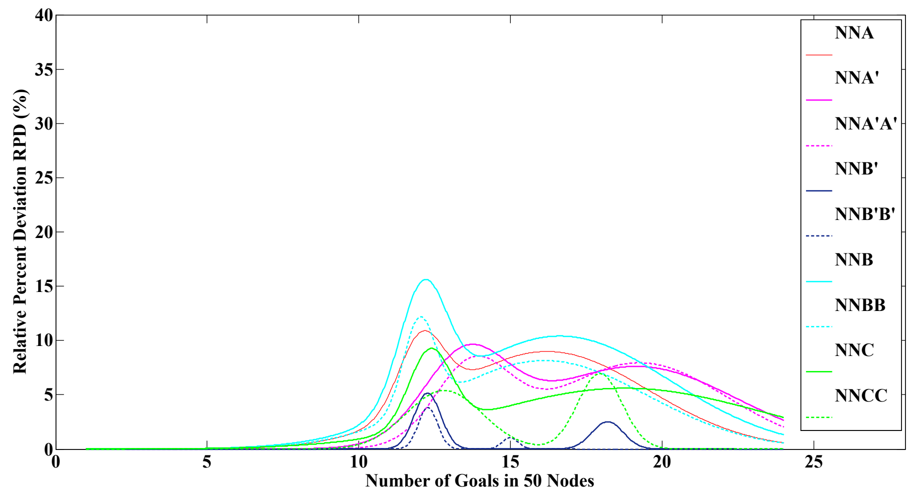


Figure 4.5: RPD with NN initial solution - 50 Nodes

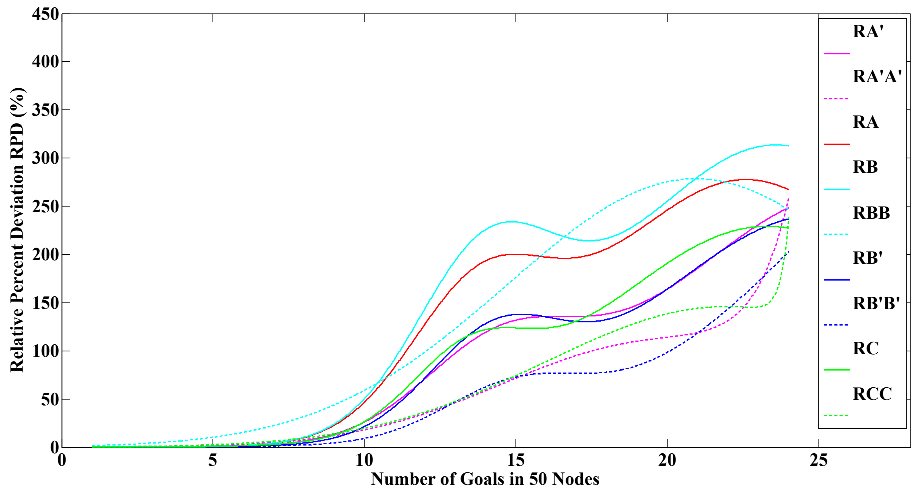


Figure 4.6: RPD with R initial solution - 50 Nodes

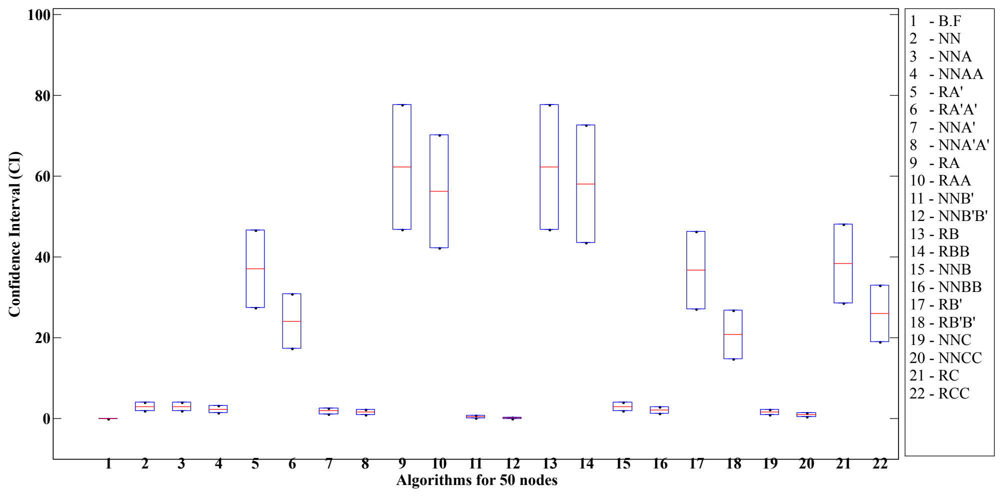


Figure 4.7: Confidence Interval of heuristics on bench size - 50 Nodes

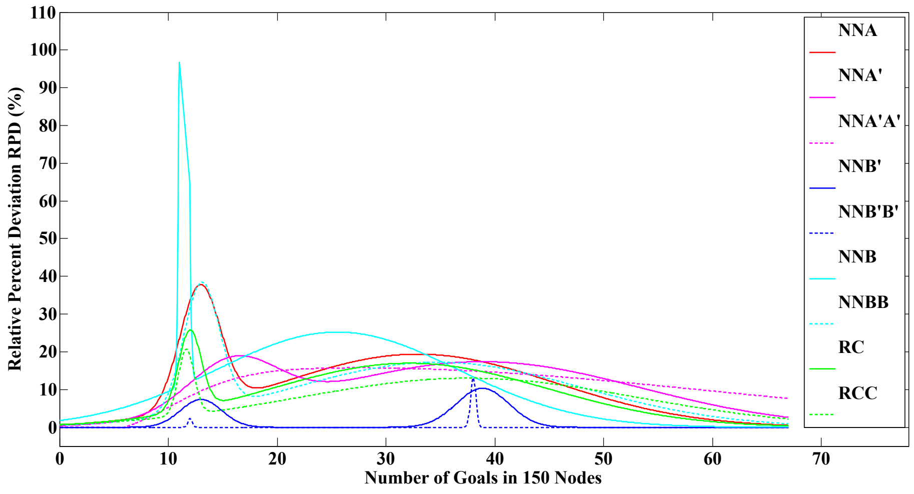


Figure 4.8: RPD with NN initial solution - 150 Nodes

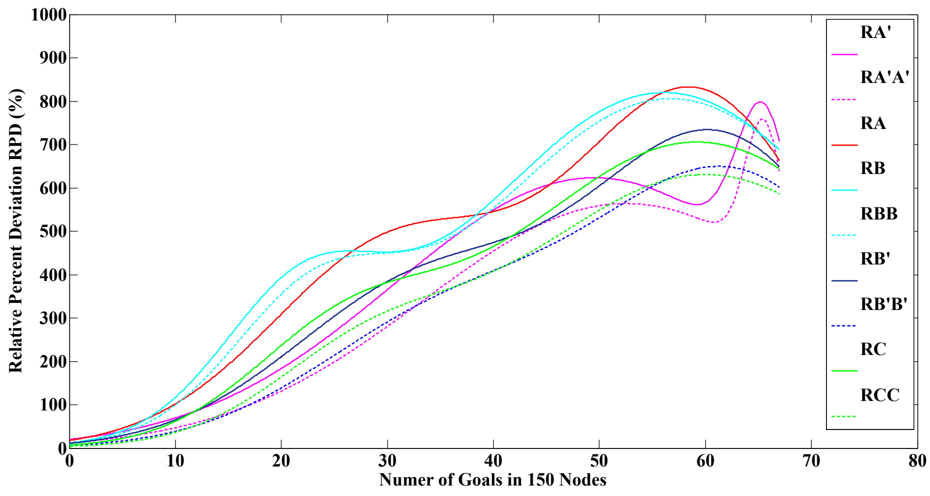


Figure 4.9: RPD with R initial solution - 150 Nodes

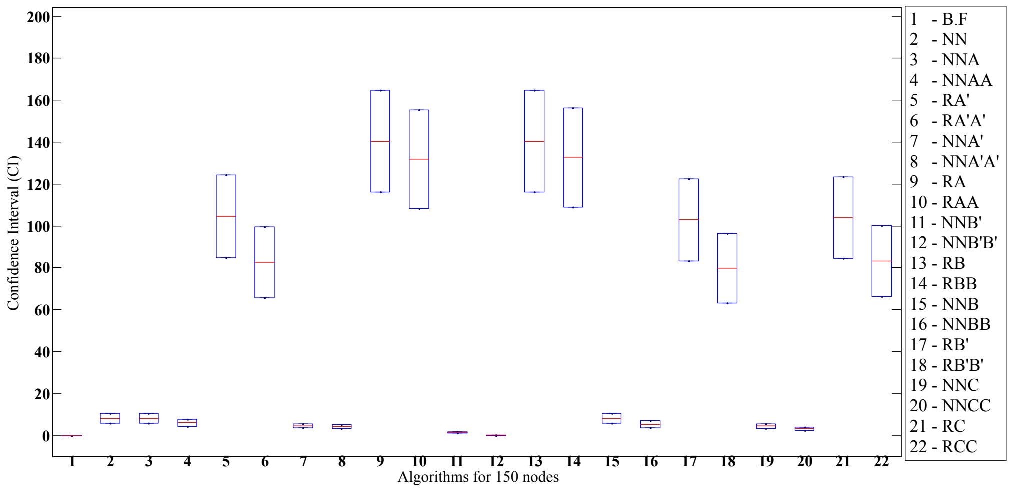


Figure 4.10: Confidence Interval of heuristics on bench size - 150 Nodes

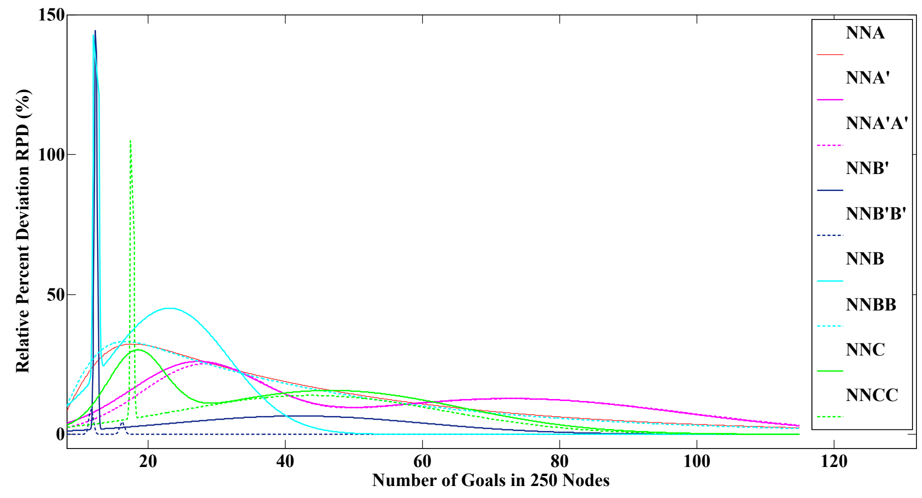


Figure 4.11: RPD with NN initial solution - 250 Nodes

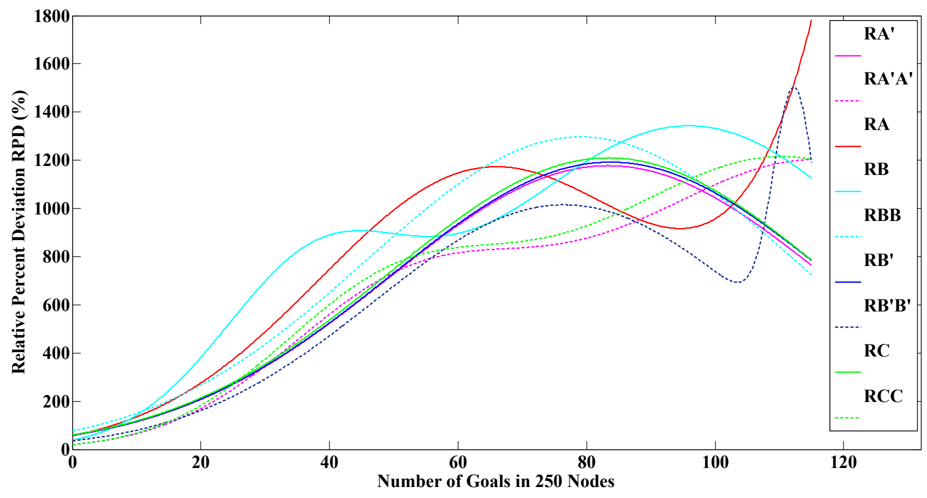


Figure 4.12: RPD with R initial solution - 250 Nodes

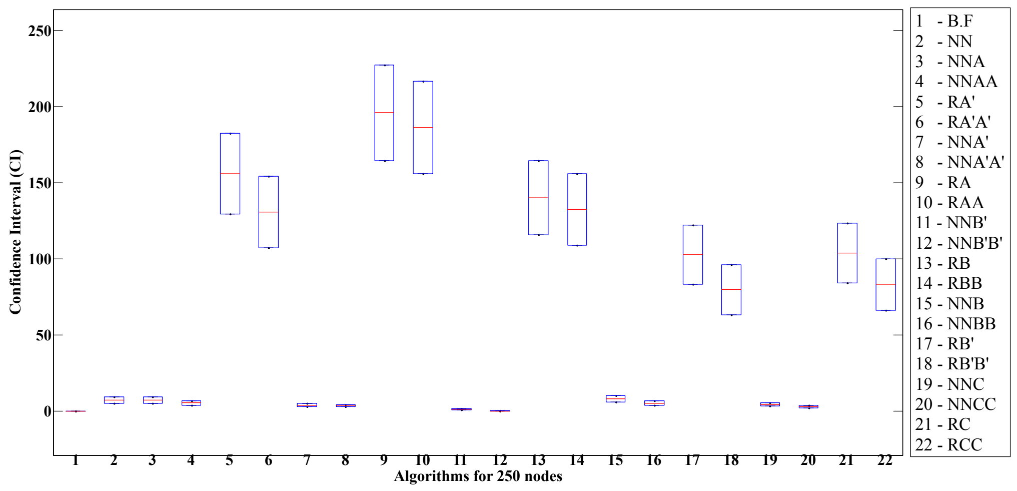


Figure 4.13: Confidence Interval of heuristics on bench size - 250 Nodes

The constructive heuristic NN generated an average RPD of $3.006\%, 8.212\%$, and 7.36% for h_{bench} nodes respectively. Local search heuristic results are tested with this NN solution. Heuristics with NN initial solution as shown in Figs. 4.5, 4.8, 4.11 have significantly lower RPD compared to the R initial solution which is depicted in Figs. 4.6, 4.9, and 4.12. For an NN initial solution, Random Insert (NNB$'$) outperforms the other heuristics with an average RPD of $0.479\%, 1.403\%$, and 1.455% as shown in Figs. 4.7, 4.10, and 4.13, respectively. The Random Insert heuristic (NNB$'$) is followed by Random Swap Random Insert (NNC) in performance with an average RPD of $1.640\%, 4.489\%$, and 4.044% for h_{bench} nodes respectively. The Random Insert Heuristic (B$'$) also performed the best among local searches with R initial solution (RB$'$) as shown in Figs. 4.7, 4.10, and 4.13 with average RPD of $24.109\%, 82.56\%$, and 131.11% for corresponding h_{bench} nodes.

Hence a Random Insert heuristic (B$'$) with an NN solution (NNB$'$) provides a good solution compared to other local search heuristics (NNA, NNA$'$, NNB, and NNC) in the given interval on the SaD based bench design with a single agent with varied obstacle sizes and geometry.

4.5 Summary

Hence, in this chapter a planning protocol for the locomotion was established based on a centroid definition of agent geometry with a graph-theoretical model for the search space. BFS was employed on the graph network for providing the shortest distance between loading and machining centers for agent movement planning. This is followed by the implementation of a Nearest Neighbor constructive heuristic. Local search moves of swap and insert were tested with the constructive heuristic incumbent solution. Simulation results show that the Nearest Neighbor heuristic provides a good head start over a random solution. Further improvement is obtained by a Random Insert heuristic, which gave better results compared to the other tested heuristics in the simulated interval node sizes. These insights can be used to efficiently define the task/goal allocation for the SaD agents during path planning. The following chapters would extend the path planning model to a multi-agent based planning.

Chapter 5

Multi-Agent Planning

This chapter introduces a new, intrinsically discrete path planning and collision-avoidance problem, with multiple SaD agents and multiple goals. Each agent must visit an array of goal positions in minimal time while avoiding collisions. The corresponding off-line path-planning problem is NP-hard. We model the system by an extended temporal graph and introduce two integer linear programming (ILP) formulations for the minimization of the makespan, with decision variables on the nodes and the edges, respectively.

5.1 Previous work

Multi-agent path planning problems have been widely researched over the years. The approaches have often been classified into coupled [54–56], where the agent path is computed in a combined configuration space, and decoupled [57,58], where the motion of the individual agents/robots is planned. Coupled solutions often use centralized planners and artificial potential field [26], while rapidly-exploring random tree (Rapidly-exploring random tree (RRT)) [59] and other heuristic methods are preferred for a decoupled planner.

Centralized planners provide near-optimal solutions but are computationally expensive. Decoupled planning is very fast but loses out on optimality and also results in many deadlock and livelock situations. Detailed reviews [60,61] describe planning algorithms for multi-robot systems; here we only refer to literature more relevant to our problem formulation.

Planning can be online or offline. The former usually requires extensive sensor systems for perceiving the environment and computing collision free paths. In the application herein, the objective is simplicity of method and equipment. Hence our focus is on offline solvers.

The time-optimal path planning for multi-robots on a graph has been shown to be NP-hard [23]. Although most applications require a feasible rather than an optimal path, in a real MHS environment saving energy and time is important. Thus, some degree of optimality is desirable especially in an industrial environment where there may be cascading effects from planning decisions.

MPP has been pursued both on discrete and continuous domains. Continuous planning [62] uses approaches such as velocity-obstacles [63], mixed integer linear programming (Mixed integer linear programming (MILP)) [64, 65] and many others.

The SaD system is, by design, an inherently graph-based discretized network. Discrete MPP has attracted a very wide research community since the abstraction of continuous to discrete domain allows a relaxed computation to be performed at a faster rate. Discretization enables the use of graphs, a very well studied research area with many guaranteed techniques for various objectives. MPP solving methodologies for discrete domains are numerous: M* [66], complete algorithms with Operator Decomposition and Independence Detection [67], Cooperative A* [68], Hierarchial Cooperative A* [68], Windowed Hierarchical Cooperative A* [68], Pebble Motion on graphs [69], subgraphs for planning [70], and Probabilistic Road Map [70].

In this thesis, the focus is on Integer linear programming (ILP) based planning methods [71]. Such methods, when implemented with advanced solvers are capable of producing very good and complete solutions in least amount of time [71]. Various integer programming approaches have been applied to MPP: ILP based on Network flow [71], MILP formulation for specified trajectory [72], and mixed integer programming [64]. After thorough study of the state of the art, ILP based methods would be very suitable to formulate the constraints of a SaD based MPP as the environment and objectives are highly constrained.

The ILP model is particularly suitable for multi-goal planning, which is our focus herein but is not common in the MPP literature. (A rare example is [73], where multiple unordered destinations are visited by the agent before it reaches its goal.)

5.2 Problem Definition

A single SaD agent multi-goal path-planning framework was discussed in chapter 4. The current chapter* would deal with the multi-agent, multi-goal planning problem.

Thus, the MPP problem herein involves $R \geqslant 1$ agents, each with N_r labelled goals, $r = 1, \ldots, R$. It is assumed that task allocation is already done and hence an assignment of ordered goals is available for each agent. This eases the planning as finding an optimal task allocation for the agents is NP-hard [74].

The problem we pose is to minimize the *makespan*, i.e., the total number of steps (individual rotations) required until the last robot reaches its goal [75]. This objective function ensures that all tasks are completed.

In MPPs, the strategies for collision avoidance are idling and detour. The SaD agent *prefers idling* to reduce wear and tear between gears.

Goal locations for different agents are not necessarily distinct. Hence, it is explicitly required that agents move away to accommodate others.

The planner takes into consideration all constraints and produces a flexible and feasible plan. Finding even a feasible solution for the MPP is Polynomial Space (PSPACE)-hard [76] even for a simplified two-dimensional case of the problem. Hence, the objective here is obtaining a near-optimal solution.

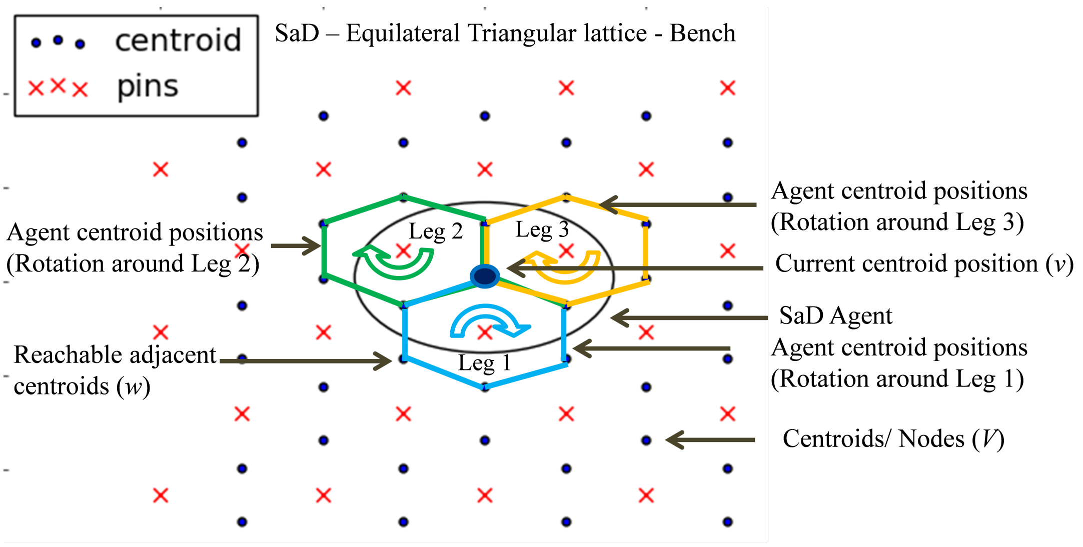
5.3 Problem Formulation*

5.3.1 Graph Representation

The bench has m pins in an equilateral triangular lattice with coordinates $P = \{p_1, p_2, \cdots, p_m\}$, $p_i = (x_i, y_i)$. A graph $G = (V, E)$ is defined with nodes (or graph

* All mathematical notations and symbols declared in this section will have its scope restricted to chapter 5.

vertices), $V = \{v_i | i = 1, \ldots, |V|\}$, identifying the centroids of the triangles of the lattice, i.e., the possible agent positions. (Agent orientation is ignored.) A connecting edge, $(v_i, v_j) \in E$, indicates the possibility to move between nodes v_i and v_j using only one pivot, i.e., their triangles are adjacent (share a side or a vertex). The representation of SaD agent on an equilateral triangle lattice bench is depicted in Fig. 5.1



● ● ● centroid
× × × pins
SaD – Equilateral Triangular lattice - Bench
Agent centroid positions (Rotation around Leg 3)
Agent centroid positions (Rotation around Leg 2)
Leg 2
Leg 3
Current centroid position (v)
SaD Agent
Agent centroid positions (Rotation around Leg 1)
Reachable adjacent centroids (w)
Leg 1
Centroids/ Nodes (V)

Figure 5.1: Nodes and edges in SaD [3]

The adjacency list of the vertex v is $\text{Adj}(v) \subseteq V$, where $v \in V$ and $w \in \text{Adj}(v)$ iff connecting edge $e = \{(v, w)\} \in E$. Each node has 13 adjacent nodes, including itself, Fig. 5.1. Some adjacent nodes have two pins that can be used as pivots. One node is selected at random during the decision. The initial locations of SaD agents is an injective map $a_I \colon \{1, \cdots, R\} \to V$, where $R < |V|$.

5.3.2 Obstacle Representation

The edges of G are pruned by removing nodes inside obstacles, which are considered to be stationary and convex in the graph. A Minkowski sum operation is performed between the obstacle geometry and the circular agent diameter to obtain a safe zone for collision avoidance. Practical agents can be modelled as disc-shaped. The complexity of the Minkowski sum of a polygon with n vertices with a disc is always $O(n)$ [77].

The Computational Geometry Algorithms Library (CGAL) library provides functions to perform these actions, whereas this work implements the Shapely library buffer feature in Python 2.7 to obtain the buffer zone. The point in polygon test is later performed to identify all the centroid vertices inside the obstacle [53]. They are utilized to relax the ILP formulations. The centroids inside obstacles are $K = \{k_1, k_2, \cdots, k_n\}$, $k_i \in V$.

5.3.3 Multi-goal Representation

The requirements of the current problem demand ordered task assignments for the agents. Therefore, a time-based ILP is used: G is extended to a temporal graph, or a time-expanded network, as in [78]. The network is based on a fixed discrete time horizon, T, creating T copies of the vertices of G [75]. Fig. 5.2 shows a representation of the temporal graph. To achieve sequential visiting of destinations, the goal set should be a unique list of nodes. The ordered list of tasks to be visited by agent r is $W_r = \{w_1, w_2, \cdots, w_{N_r}\}$, $w_i \in V$.

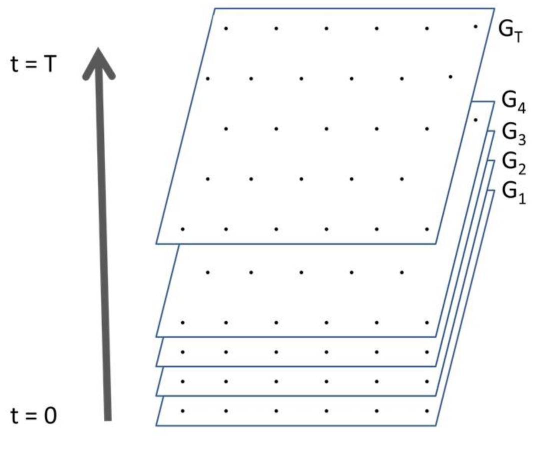


Figure 5.2: Temporal graph

Two ILP based formulations are presented for the multi-agent path planning in the following sections.

5.4 Vertex based formulation

A vertex based formulation with three indices, for the node, time, and agent, are presented. Binary variables $x_{i,t,r}$ are created with space complexity of $O(|V|TR)$: we have $x_{i,t,r} = 1$ iff agent r is in vertex i at time step t. A graphical representation of the binary variable assignment is shown in Fig. 5.3. For convenience, vertex based formulation would be referred to as Formulation I in all further text in this chapter.

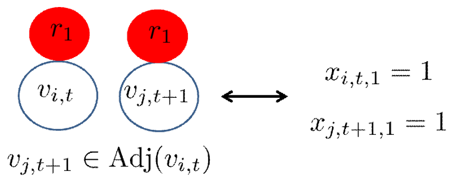


Figure 5.3: Binary variable representation for Vertex based formulation

5.4.1 Makespan minimization

The objective function is to minimize the makespan. In a time expanded network, a fixed time horizon T, is required. As T provides a suitable lower bound for the objective function, the shortest path between the agents and the task goals [75] is a suitable initial value for T. The problem being a multi-task assignment, an all-pair shortest path (All Pair Shortest Path (APSP)) is implemented with Floyd Warshall algorithm [79] with complexity of $O(|V|^3)$. APSP gives the distance matrix $d_{i,j}$ (the shortest path distance between vertices i and j; every edge is assumed with length 1), computed offline. Floyd-Warshall algorithm implementation is shown in Algorithm 7.

The shortest path between the node and itself is $d_{i,i} = 0$ in the standard implementation but the assumption of $d_{i,i} = 1$ is made for inclusion of time. For an agent r and a task-allocation list $W_r = \{w_1, w_2, w_3, \cdots, w_{N_r}\}$, the minimum number of steps D_r for agent r to visit all goal states in W_r is calculated with APSP distance matrix neglecting other agents on the graph as follows

$$D_r = d_{a_I(r),W_r(1)} + \sum_{n=1}^{N_r-1} d_{W_r(n),W_r(n+1)} \qquad 1 \leqslant r \leqslant R,\ N_r \geqslant 1 \tag{5.1}$$

Procedure 7 Floyd Warshall Algorithm [79]

```
Input: Adjacency matrix of graph $G$, Adj$(i, j)$
Let $n = |V|$                                  ▷ The number of vertices in the graph
Initialize shortest distance matrix $d_{[],[]}$
Let $d_{i,j} = 0$, for all i
Let $d_{i,j} = \infty$ if Adj$(i, j) = 0$
Let $d_{i,j} = 1$ if Adj$(i, j) = 1$           ▷ All edges of graph $G$ are of unit weight

for $k := 1$ to $n$ do
    for $i := 1$ to $n$ do
        for $j := 1$ to $n$ do
            $d_{i,j} = \min(d_{i,j}, d_{i,k} + d_{k,j})$
        end for
    end for
end for
Output $d$    ▷ The shortest path matrix between every pair of vertices in $G$
```

The maximum of all the D_r, $t_{\min} = \max(D_r)$, is the lower bound for time horizon T for makespan minimization [75]. The formulation assumes that all agents are initiated at the same instant.

5.4.1.1 Objective Function

The objective is to minimize the makespan f_1 of all the agents. Time horizon T is initially set to $t_{\min}$ and increased iteratively to find a feasible solution. The feasible makespan time solutions generated by these ILP formulations are always optimal since the graph system is discrete and has finite number of states [75].

$$\text{Minimize} \quad f_1 = \sum_{i \in V} \sum_{t=0}^{T} x_{i,t,r} \tag{5.2}$$

for some arbitrary agent r, $1 \leqslant r \leqslant R$.

subject to:

5.4.1.2 Constraints

Agent initiation

The agents' positions are initiated

$$x_{a_I(r),0,r} = 1 \quad 0 \leqslant r \leqslant R \tag{5.3}$$

where $a_I(r)$ is the initial centroid position of agent r, $a_I(r) \in V$

Goal Assignment

Single- or multi-goals are assigned:

$$\sum_{t=0}^{T} x_{j,t,r} \geqslant 1 \quad \forall j \in W_r, \forall r \tag{5.4}$$

Collision Avoidance

Similar to the "meet" and "head-on-head" scenarios in [75], we show in Fig. 5.4 two possible collisions in an SaD system. The initial position of the agents is in a collision-free state if no two agents share the same pin. A collision state occurs if either one agent moves to a vertex occupied by another, or if two agents reach a vertex at the same time step.

$$\sum_{r=1}^{R} x_{i,t,r} \leqslant 1 \quad 0 \leqslant t \leqslant T, \forall i \in V \tag{5.5}$$

Obstacle avoidance

Stationary obstacles in the bench are represented by

$$x_{i,t,r} = 0 \quad \forall i \in K \tag{5.6}$$

$0 \leqslant t \leqslant T, 1 \leqslant r \leqslant R,$ where K is centroid list inside obstacles

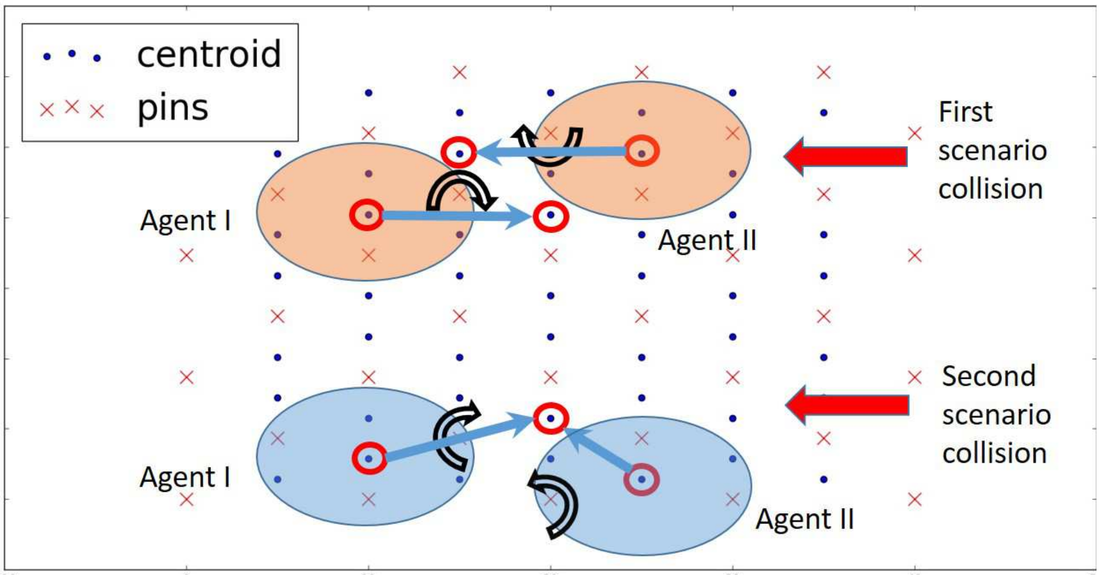


Figure 5.4: Collision scenario in SaD

Bench pin collision

The use of each pin is restricted to one agent at a time:

$$\sum_{r=1}^{R} \sum_{\forall k \in \mathrm{Adj}(i)} x_{k,t,r} \leqslant 1 \quad \forall i \in V, 0 \leqslant t \leqslant T \tag{5.7}$$

Continuity

The following constraint ensures that an agent leaves only from the node that it has entered in the previous step hence maintaining the continuity of the path. Fig. 5.5 depicts a continuity constraint violation.

$$x_{i,t,r} + x_{j,t+1,r} \leqslant 1 \tag{5.8}$$

$$0 \leqslant t \leqslant T-1, 1 \leqslant r \leqslant R, i \in V, \forall j \notin \mathrm{Adj}(i)$$

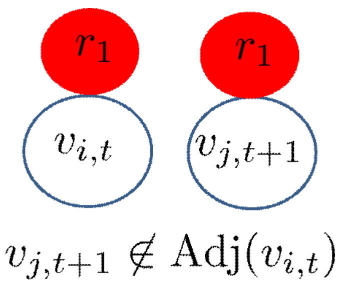


Figure 5.5: Continuity constraint violation

Enforcing Sequence

The task allocation, W_r, for every agent is done prior to the path planning. This is realisic in an MHS, where loading/unloading stations are known for each agent. Hence, the planning assumes that ordered goals are available for each agent to visit. Successive goals in the task list can only be visited after their predecessors.

$$\sum_{t=0}^{H} x_{i,t,r} \geqslant x_{j,H+1,r} \tag{5.9}$$

$$0 \leqslant H \leqslant T-1, 1 \leqslant r \leqslant R, i \in W_r(k), j \in W_r(k+1)$$

Prevent agent disappearance

It is necessary to write constraints for agents to stay in the planning graph until the last agent has reached its destination. This prevents an agent from disappearing after it reaches its goal state and makes the node it occupies available. The formulation provides the agent the freedom to move once it has satisfied all its goal states if demanded. This actively prevents deadlocks.

$$\sum_{\forall i \in V} x_{i,t,r} = 1 \tag{5.10}$$

$$0 \leqslant t \leqslant T, 1 \leqslant r \leqslant R$$

5.4.2 Idling Preference

To reduce mechanical problems, it is better for an agent to idle than to detour. Idling of agent r in node i at time step t is represented by $x_{i,t-1,r} = 1$ and $x_{i,t,r} = 1$. This constraint leads to a multi-objective formulation, where the model tries to minimize both the total makespan and the total distance traveled by each agent. The current model follows a lexicographic multi-objective optimization approach where the total makespan is minimized first, followed by maximizing the number of idling states. $D_{\max}$, the minimum total moves needed by all the agents to reach their goals provides a suitable lower bound for the objective function .

$$D_{\max} = \sum_{r=1}^{R} D_r \tag{5.11}$$

$D_{\max}$ is calculated using APSP between each agent and its corresponding goal states, and D_r is obtained from Eq. 5.1.

5.4.2.1 Objective Function

Makespan minimization is followed by the second objective (f_2) of idling maximization. This is converted into a minimization of the non-idling moves performed by the agents. These are represented by a binary variables, $c_{t,r} = 1$, for every agent r at time step t, $0 \leqslant t \leqslant T$, $1 \leqslant r \leqslant R$. Their sum, y, is to be minimized

$$\begin{aligned} & f_2 = y \\ \text{subject to} \quad & y = \sum_{r=1}^{R}\sum_{t=0}^{T} c_{t,r} \\ & y \geqslant D_{\max} \\ & c_{t,r} \in \{0,1\} \end{aligned} \tag{5.12}$$

5.4.2.2 Constraints

Idling constraints

A binary variable $b_{i,t+1,r}$ is assigned to represent all non-idle moves; $b_{i,t,r} = 1$ means that the agent was present in node i at time instant $t-1$ and not available in node i at time instant t. A grahical representation of the binary variable $b_{i,t,r}$ for the idling constraint is shown in Fig. 5.6.

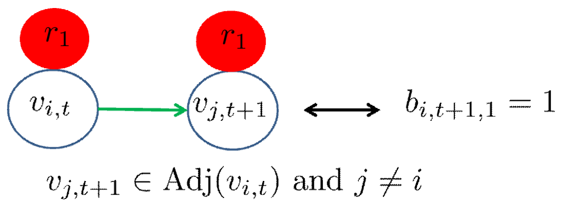


Figure 5.6: Graphical representation of binary variable $b_{i,t,r}$

$$x_{i,t,r} + \sum_{\substack{\forall j \in \text{Adj}(i) \\ \forall j \neq i}} x_{j,t+1,r} - 1 \leqslant b_{i,t+1,r} \tag{5.13}$$

$$b_{i,t+1,r} \leqslant x_{i,t,r} \tag{5.14}$$

$$b_{i,t+1,r} \leqslant \sum_{\substack{\forall j \in \text{Adj}(i) \\ \forall j \neq i}} x_{j,t+1,r} \tag{5.15}$$

$$0 \leqslant t \leqslant T-1, 1 \leqslant r \leqslant R, \forall i \in V$$

Non-idling of each agent in each time instant

$$\sum_{\forall i \in V} b_{i,t,r} \leqslant c_{t,r} \tag{5.16}$$

$$0 \leqslant t \leqslant T, 1 \leqslant r \leqslant R$$

5.4.2.3 Linear Program

The solution from the makespan function model is set as the incumbent solution for the idling function. Hence, the second objective function starts with a feasible solution.

$$\begin{aligned} \text{Minimize} \quad & f_2 \\ \text{subject to constraints} \quad & (5.12) \;\text{ to }\; (5.16) \\ & f_1 = t^* \end{aligned}$$

where t^* is the minimum makespan value obtained for f_1.

5.5 Edge based formulation

Edge based formulation is constructed with a four index binary variable $x_{i,j,t,r}$ of space complexity $O(|E|TR)$. Binary decision variable $x_{i,j,t,r} = 1$ iff agent r moves from vertex i to vertex j at time step t. The binary variable $z_{i,r}$ represents the initial position of the agent in the graph. $z_{i,r} = 1$ if agent r is in vertex i during initiation. For convenience edge based formulation would be referred to as Formulation II in all further text

5.5.1 Makespan minimization

The objective as previously formulated, is to minimize the makespan f_1 for all the agents.

5.5.1.1 Objective Function

The objective function f_1 is the makespan.

$$\text{Minimize} \qquad f_1 = \sum_{i \in V} \sum_{j \in \text{Adj}(i)} \sum_{t=0}^{T} x_{i,j,t,r} \tag{5.17}$$

for some arbitrary agent r, $1 \leqslant r \leqslant R$.

subject to:

5.5.1.2 Constraints

Agent initiation

An additional variable $z_{i,r}$ is required to initiate the agents on the planning graph, compared to the vertex based formulation.

$$\sum_{\forall i \in V} z_{i,r} \leqslant 1 \quad z_{i,r} \in \{0,1\}, 1 \leqslant r \leqslant R \tag{5.18}$$

$$z_{a_I(r),r} = 1 \quad 1 \leqslant r \leqslant R \tag{5.19}$$

where a_I is the initial centroid position of agent $r, a_I(r) \in V$

We have

$$z_{i,r} = \sum_{j \in \mathrm{Adj}(i)} x_{i,j,0,r} \quad 1 \leqslant r \leqslant R, \forall i \in V \tag{5.20}$$

Output/Goal Assignment

All goals for each robot W_r should be reached within the time planning horizon. Hence the below constraint requires the goal states to be visited at least once by its corresponding agent before T time steps is reached.

$$\sum_{t=0}^{T} \sum_{i \in \mathrm{Adj}(j)} x_{i,j,t,r} \geqslant 1 \quad 1 \leqslant r \leqslant R, \forall j \in W_r \tag{5.21}$$

Collision Avoidance

Each edge in the planning graph at any time instant, can have at-most only one robot, hence avoiding collision.

$$\sum_{r=1}^{R} \sum_{i \in \mathrm{Adj}(j)} x_{i,j,t,r} \leqslant 1 \quad 0 \leqslant t \leqslant T, \forall j \in V \tag{5.22}$$

Bench pin collision

Possible agent positions as mentioned earlier are represented as centroids. To avoid bench collision, adjacent centroids cannot be assigned to two different agents at any time step.

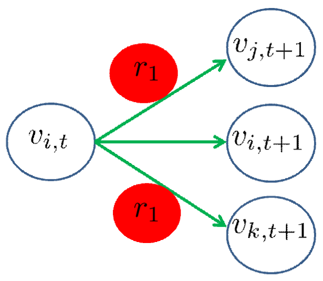
$$\sum_{r=1}^{R} \sum_{j \in \mathrm{Adj}(i)} \sum_{k \in \mathrm{Adj}(j)} x_{j,k,t,r} \leqslant 1 \tag{5.23}$$

$$0 \leqslant t \leqslant T, \forall i \in V$$

No trees/No Bifurcation

Each agent is constrained to leave to only one node. This constraint restricts branching. A depiction of the scenario is shown in Fig. 5.7

$$\sum_{r=1}^{R} \sum_{j \in \mathrm{Adj}(i)} x_{i,j,t,r} \leqslant 1 \quad 0 \leqslant t \leqslant T, \forall i \in V \tag{5.24}$$



$\{v_{i,t+1}, v_{j,t+1}, v_{k,t+1}\} \in \text{Adj}(v_{i,t})$

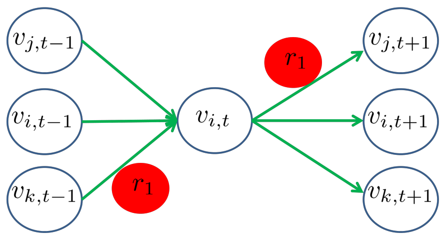
Figure 5.7: Tree/Bifurcation scenario

Continuity

Similar to the vertex based formulation, each agent can leave only from the node it has entered in the previous time step. A representation of the constraint is shown in Fig. 5.8

$$\sum_{j \in \text{Adj}(i)} x_{j,i,t,r} \geqslant \sum_{j \in \text{Adj}(i)} x_{i,j,t+1,r} \tag{5.25}$$

$$1 \leqslant r \leqslant R, 0 \leqslant t \leqslant T-1, \forall i \in V$$



$\{v_{i,t+1}, v_{j,t+1}, v_{k,t+1}\} \in \text{Adj}(v_{i,t})$
$\{v_{i,t-1}, v_{j,t-1}, v_{k,t-1}\} \in \text{Adj}(v_{i,t})$

Figure 5.8: Continuity constraint

Enforcing Sequence

As mentioned for the vertex based formulation, all ordered goals need to be visited in sequence. The constraint below ensures that the planner makes assignment to the binary variables accordingly.

$$\sum_{t=0}^{H} \sum_{j \in \text{Adj}(i)} x_{j,i,t,r} \geqslant \sum_{l \in \text{Adj}(p)} x_{l,p,H+1,r} \tag{5.26}$$

where $0 \leqslant H \leqslant T-1, 1 \leqslant r \leqslant R, i \in W_r(k), p \in$ $W_r(k+1)$, W_r is the task allocation list of agent r, $0 \leqslant k \leqslant length(W_r)-1$

Prevent agent disappearance

Till every agent has visited its corresponding assigned goals, all other agents need to be present in the planning graph.

$$\sum_{\forall i \in V} \sum_{\forall j \in \mathrm{Adj}(i)} x_{i,j,t,r} = \sum_{\forall i \in V} \sum_{\forall j \in \mathrm{Adj}(i)} x_{i,j,t,r+1} \tag{5.27}$$

$$0 \leqslant t \leqslant T, 1 \leqslant r \leqslant R-1$$

5.5.2 Idling Preference

As in the vertex-based formulation, the secondary objective is to maximize the number of idle states for all agents.

5.5.2.1 Objective function

As above, the objective is to maximize the number of idle states for each agent r. This is converted to a minimization problem, where each non-idle state of each agent at each time step is minimized. These non-idle states at each time step are represented by a binary variable $c_{t,r}$.

$$\begin{aligned} & f_2 = y \\ \text{subject to} \quad & y = \sum_{r=1}^{R} \sum_{t=0}^{T} c_{t,r} \\ & y \geqslant D_{\max} \\ & c_{t,r} \in \{0,1\} \end{aligned} \tag{5.28}$$

5.5.2.2 Constraints

At a time step t, when an agent r has moved from its current node to its adjacent node then binary variable $c_{t,r}=1$ is assigned. Hence minimizing the number of instances where $c_{t,r} = 1$ would result in the required idling maximization.

$$\sum_{\substack{\forall j \in \mathrm{Adj}(i) \\ j \neq i}} x_{i,j,t,r} \leqslant c_{t,r} \tag{5.29}$$

$$0 \leqslant t \leqslant T, 1 \leqslant r \leqslant R$$

5.5.2.3 Linear Program

The minimal value, t^*, of the makespan is set as a constraint for the secondary optimization.

$$\begin{aligned} \text{Minimize} \quad & f_2 \\ \text{subject to constraints} \quad & (5.28) \text{ to } (5.29) \\ & f_1 = t^* \end{aligned}$$

The two ILP models presented in this section are tested via simulations in the following chapter. Experimental results are presented in the following chapter to discuss the effectiveness of the formulation in computing the objective functions described in this section.

5.6 Simulations

5.6.1 Setup

Simulations are performed on an Intel Xeon Dual Core Processor 2.0 Ghz 64-bit OS with 16.0 GB RAM. Mathematical models are implemented on Gurobi 6.5.1 with Python 2.7 as interface.

The number of pins in the actual SaD bench in the SwarmItFix prototype is 52. To have a symmetrical grid for computation, 36 pins are considered for the first grid where the graph has 49 vertices and 491 edges. A second grid with 190 pins is considered with $|V| = 323$ and $|E| = 3805$.

To compare the formulations for a general scenario, the bench pin collision constraint is not taken into consideration and all agents need to visit only one goal. The results can therefore directly be related to any graph-based MPP with

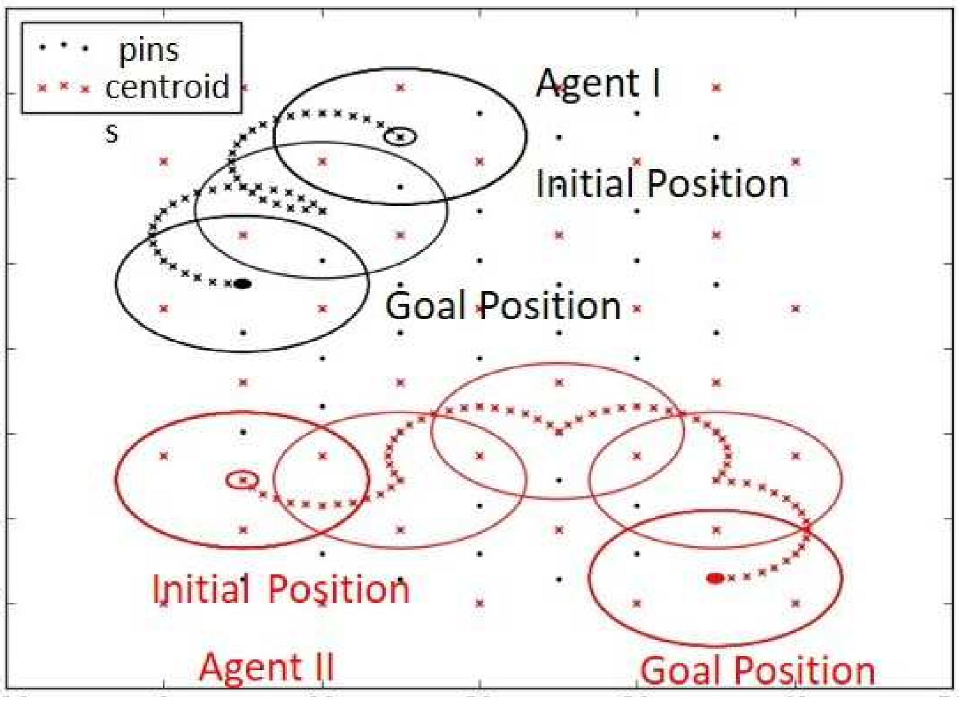


Figure 5.9: Motion trajectory of SaD agent

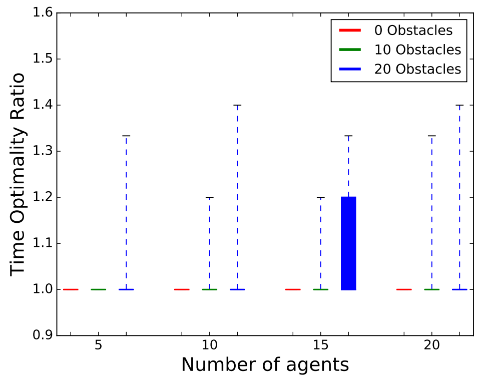
single-goal objectives. Agents are randomly initiated on the graphs with random obstacles.

5.6.2 Results

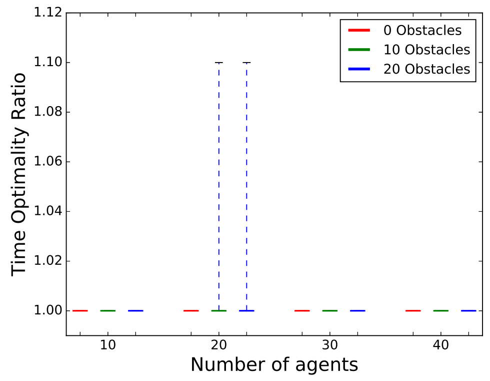
Tables 5.1 and 5.2 present the computation times of the first and second grids for 10 and 20 randomly generated initial states, respectively. Shown are the times (in seconds) for optimizing the makespan function f_1 and, in parentheses, the idling function f_2. For each computation, termination is triggered by a time limit of 1500 s or an optimality gap of 10%. The number of cases which failed to generate a feasible solution are displayed as superscipts.

To understand the performance of the ILP models time optimality [62] ratio and total number of idle states is computed for the dataset.

$$\text{Time Optimality ratio} = \frac{f_1}{t_{\min}} \tag{5.30}$$

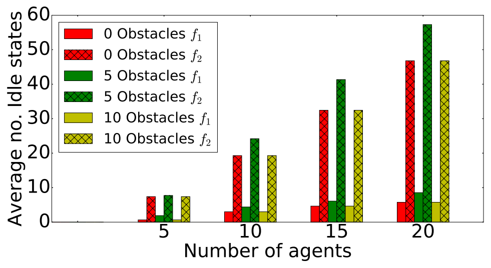


(a) First grid

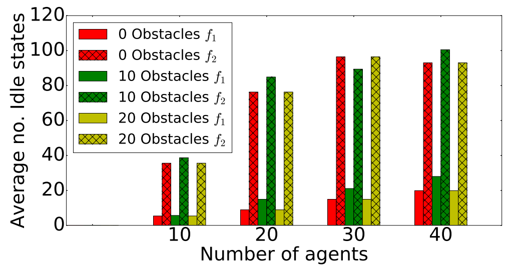


(b) Second grid

Figure 5.10: Makespan time optimality

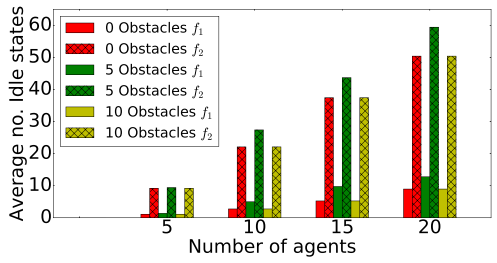


(a) First grid

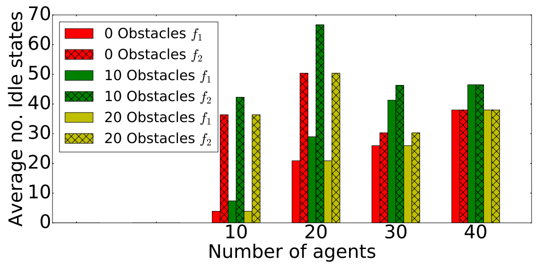


(b) Second grid

Figure 5.11: Average agent idling States - Formulation I



(a) First grid



(b) Second grid

Figure 5.12: Average agent idling States - Formulation II

Table 5.1: First grid computation time (s)

No. of Obstacles		**Number of Agents**			
		5	10	15	20
Formulation I	0	0.49 (1.05)^{0}	1.33 (3.06)^{0}	2.14 (11.10)^{0}	3.36 (49.19)^{0}
	5	0.56 (1.47)^{1}	1.53 (54.11)^{0}	2.98 (179.33)^{0}	5.68 (382.02)^{0}
	10	0.34 (1.38)^{0}	1.47 (13.01)^{1}	6.09 (669.11)^{5}	4.67 (366.09)^{1}
		5	10	15	20
Formulation II	0	0.84 (6.16)^{0}	1.35 (35.52)^{0}	2.10 (151.71)^{0}	3.52 (544.38)^{0}
	5	0.61 (34.22)^{1}	1.31 (261.17)^{0}	2.15 (581.70)^{0}	26.15 (1268.98)^{0}
	10	0.53 (34.20)^{0}	1.41 (345.50)^{1}	14.52 (997.33)^{5}	6.83 (1401.24)^{1}

Table 5.2: Second grid computation time (s)

No. of Obstacles		**Number of Agents**			
		10	20	30	40
Formulation I	0	25.00 (250.30)^{0}	140.69 (1077.59)^{0}	481.60 (1500.0)^{0}	351.33 (1500.0)^{1}
	10	29.60 (288.89)^{0}	109.90 (1500.0)^{0}	267.0 (1500.0)^{0}	329.16 (1500.0)^{4}
	20	15.45 (106.22)^{1}	65.79 (1202.8)^{0}	84.09 (1455.40)^{5}	233.375 (1500.0)^{2}
		10	20	30	40
Formulation II	0	4.430 (413.33)^{0}	11.115 (1320.90)^{0}	40.715 (1500.0)^{0}	50.45 (1500.0)^{1}
	10	4.956 (230.19)^{0}	10.850 (1489.14)^{0}	9.894 (1500)^{0}	112.32 (1500.0)^{4}
	20	4.638 (123.92)^{1}	9.894 (1205.60)^{0}	45.03 (1500.0)^{5}	405.42 (1500.0)^{2}

$$\text{Total idle states} = (f_1 R) - f_2 \tag{5.31}$$

The optimality ratio shown in the graphs are based on a 95% confidence interval. As discussed earlier, if the computation is run without a termination time, the assignment solution obtained for a makespan objective is always an optimal one. Termination time were included for these experiments for practical purpose, and we observed both the formulations obtain the same minimum makespan.

Fig. 5.10 shows that both the formulations provide near optimal solution for the single goal planning without SaD bench constraints. When computation time performance is recorded (Table 5.1 and 5.2)), it becomes evident that Formulation II is a stronger model for the current scenario. Though Formulation II required higher computation time for secondary optimization, it outperformed Formulation I in makespan computation and average idle states produced without secondary optimization (Figs. 5.11a, 5.11b and 5.12a, 5.12b). Most single goal MPP problems are generally directed towards makespan minimization; the idling function is just

Table 5.3: Multi goal no sequence enforcing

		Computation time (s)			
		Formulation I			
No. of Agents		5	4	3	2
No. of Goals		1	2	3	4
No. of Obstacles	0	669.91 (1384.86)^{11}	391.31 (1003.75)^{2}	29.58 (411.19)^{0}	2.963 (2.627)^{1}
	2	NA $(NA)^{20}$	NA $(NA)^{20}$	7.25 (44.21)^{18}	2.58 (5.92)^{10}
		Formulation II			
No. of Obstacles	0	1181.62 (1253.12)^{12}	317.85 (1389.85)^{7}	150.55 (612.049)^{0}	6.50 (5.28)^{2}
	2	NA $(NA)^{20}$	NA $(NA)^{20}$	33.5 (124.50)^{18}	7.22 (1.78)^{11}

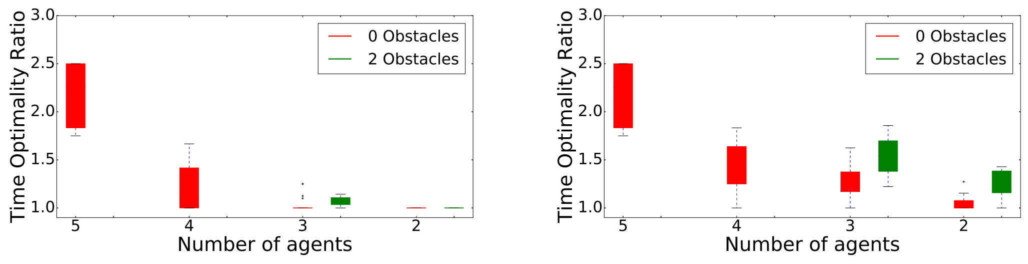
Table 5.4: Multi goal with sequence enforcing

		Computation Time (s)			
		Formulation I			
No. of Agents		5	4	3	2
No. of Goals		1	2	3	4
No. of Obstacles	0	676.85 (1276.77)^{11}	365.46 (1372.63)^{6}	255.39 (810.55)^{0}	14.21 (127.19)^{2}
	2	NA $(NA)^{20}$	NA $(NA)^{20}$	44.58 (1145.08)^{18}	235.08 (647.360)^{10}
		Formulation II			
No. of Obstacles	0	1319.14 (1265.71)^{13}	1028.91 (1500)^{8}	860.0 (1377.76)^{3}	259.05 (838.05)^{2}
	2	NA $(NA)^{20}$	NA $(NA)^{20}$	420.5 (892.5)^{18}	941.57 (1280.71)^{13}

a feature and not a mere necessity. Hence, the four index formulation is ideal for less constrained single goal MPP scenarios with bigger grids. These results can be extended to all graph structures with similar constraints.

The formulations are further extended to the SaD agent constraints: bench pin collision and multi-goal.The computational time results are provided for 20 randomly generated instances for the first grid in Tables 5.3 and 5.4. Smaller number of agents are the main scope for the SaD multi-agent path planning problem since the agents are considered to be physical entities. Convex polygonal obstacles are placed on the graph, where each polygon encapsulates 10 nodes.

Experiments were also made to study the Gurobi's task allocation capability, when no ordering sequence is provided to the goals for each agent to visit. The minimum makespan achieved by relaxing the sequence constraints is shown in Fig. 5.13a. Results show that the solver provides near optimal solution for smaller



(a) No sequence constraint **(b)** With sequence constraint

Figure 5.13: Multi-goal time optimality

number of agents at a reasonable time (Table 5.3).

Solver times in Tables 5.3 and 5.4, clearly show that Formulation I performed better: it produced usually fewer and never more failures than II.

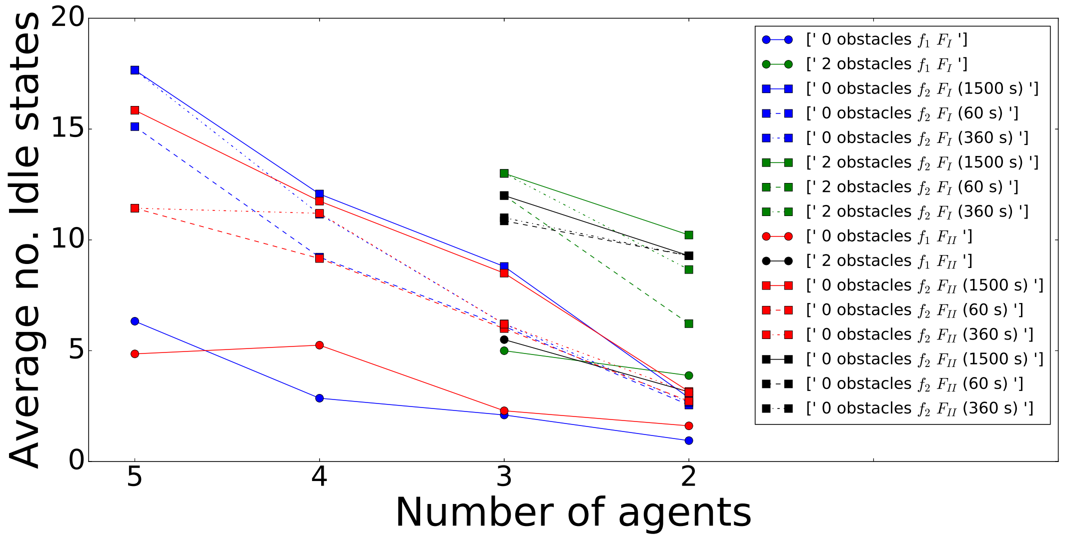


Figure 5.14: With sequence idle state comparison

For the idling function objective (numbers in parentheses), both Formulations take a long time to converge to the lower bound or do not terminate at all. The model was tested with lower termination times comparing limits of 60, 360, and 1500 seconds.

Results are summarized in Figs. 5.14 and 5.15: with the introduction of idling function f_2, Formulation F_I performs better than F_{II} with the sequence constraint. Without the sequence constraint both formulations performed nearly identically.

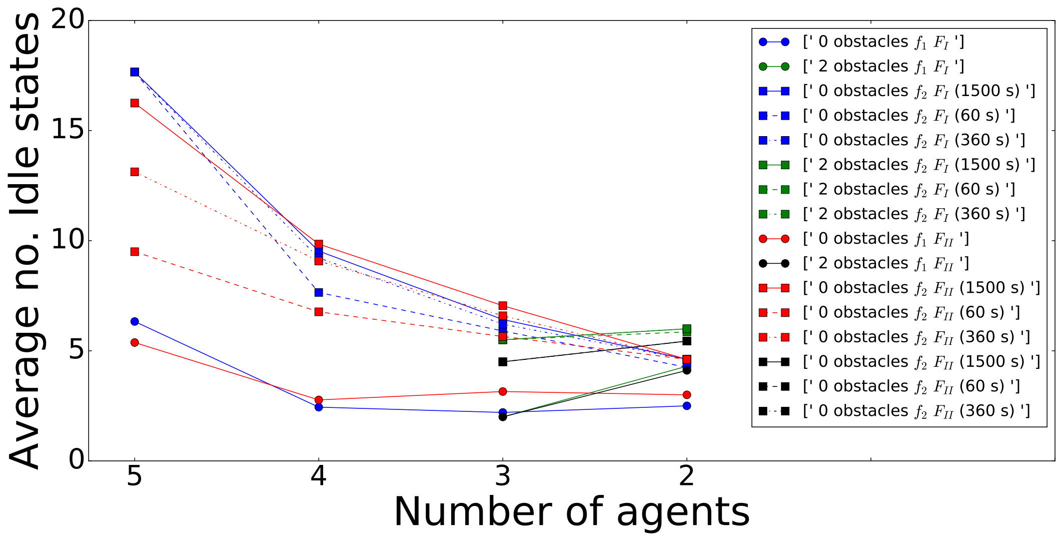


Figure 5.15: No sequence idle state comparison

Also there existed only a 10 to 15% difference between solutions with 360 and 1500 s solver time: a good tradeoff favoring reducing time.

Hence, during real-time execution of the SaD system planner, a vertex-based three-index formulation with termination time of 360 s would be a preferable choice.

5.7 Summary

A "swing and dock" locomotion system is proposed for a multi-agent system. A collision-free path-planner based on a temporal graph with integer linear programming is introduced. A lexicographic multi-objective optimization is required where minimizing of the makespan time of the agents is followed by maximizing the number of idle states. Two ILP formulations are presented, vertex-based three-index and edge-based four-index. Both actively address livelock and deadlock situations.

The edge-based approach performs best for a makespan objective in a conventional graph-based MPP with single goals yielding better computational times and higher numbers of idle states.

However, with multiple goals and the SaD agent constraints, the vertex-based approach is clearly superior. Though both formulations produced identical makespan solutions, F_I has shorter computation times, fewer failures, and performs well on the idling function. Simulation results also suggest that a smaller termination time is sufficient for the idling function to get a close estimate of the total idling states.

Hence, the three-index formulation will implemented in the SaD planner on the existing prototype.

The ILP formulations, tailored to the SaD system, are general enough to be applicable for many other single- and multi-agent problems over discretized networks.

The following chapter will address path planning with orientation constraints, which arise when the agent pay-load is not completely symmetrical.

Chapter 6

Orientation Planning

Previous chapter dealt with permutation invariant goal states to address a general case. In a real world scenario, the SaD agents would be required to carry tools/materials/manipulators, where the orientation in which the agent arrives at the goal location becomes of prime importance. This chapter deals with labeled legs describing the orientation of the agents on the mounting pins, hence taking into account the orientation of the agent w.r.t the bench. Integer Linear Programming (ILP) formulations are used to model the path planning problem. The mathematical formulations are implemented and tested using GUROBI solver. Computational results display the effectiveness of the formulations.

The previous chapter 5 dealt with accommodating multi-agent planning with multiple goals. The planning problem was expressed as a temporal graph, where centroids: agents position on a graph is used to represent the nodes, and all connection to adjacent centroids (edges of the graph): represented reachable pins by swinging around the docked leg. The legs were unlabeled in the previous work, hence not accounting for the agent's orientation definition. The current chapter deals with these specific constraints, i.e the orientation at which each agent arrives at its desired task/goal location.

6.1 Problem Definition

The MPP problem herein involves $R \geqslant 1$ agents, with $L \geqslant 2$ legs. Each agent is assigned with N_r labelled goals, $r = 1, \ldots, R$. Task allocation is assumed to be provided prior to planning, hence simplifying the planning as finding an optimal task allocation for the agents is known to be NP-hard [74].

Similar to the previous chapter 5 with multiple goals, the current objective in this chapter [†] is to minimize the *makespan*, i.e.,the total number of steps (individual rotations) required until the last robot reaches its goal [75]. Ordered goal assignments are provided for each agent, such a constraint is ideal for manufacturing scenarios. Most of the assumptions are similar to the previous chapter 5 : where the strategies for collision avoidance are idling and detour. The SaD agent *prefers idling* over detour since mechanical components in gear drives are involved. The increased wear and tear of gear components may affect performance of the agents over time, hence idling of agents would be desirable. In literature, most MPP have unique goals for each agent whereas in a practical scenario, goal locations for different agents are not necessarily distinct. Hence current work does not assume agents to have unique goals. Formulations are modeled such that agents move away to accommodate other agents to reach their goals. The planning approach would aim to achieve only near-optimal solution.

6.2 Problem Formulation[†]

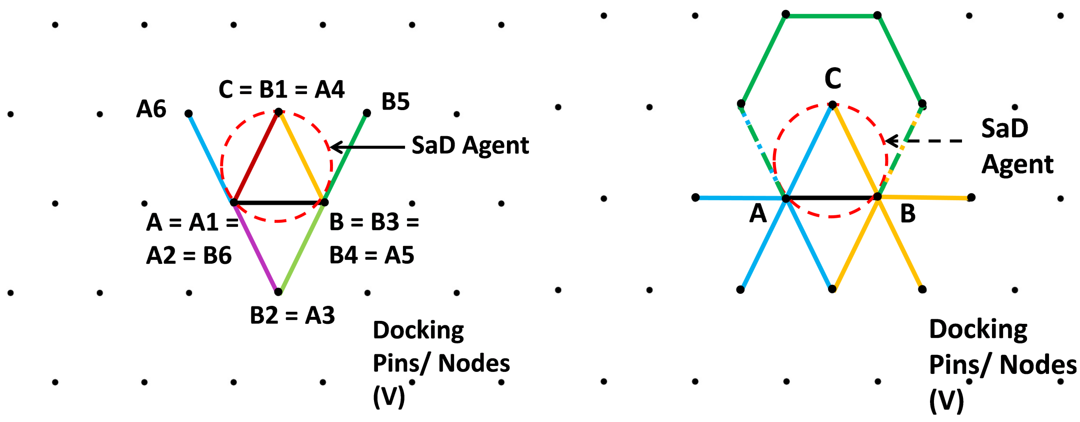
Observing the SaD design leads us to formulating the problem as a discrete MPP. Discrete MPP is a well studied research area, with numerous proven techniques such as Pebble Motion on graphs [69], Probabilistic Road Map [70], Windowed Hierarchical Cooperative A* [68], Cooperative A* [68], Hierarchial Cooperative A* [68], ILP based on Network flow [71], and many other methodologies. The results provided by [71], where the uilization of ILP methods coupled with commercial solvers proved very useful to plan large instances of multi-agents in graphs with minimal computation time. However, most research is carried out considering

[†] All mathematical notations and symbols declared in this section will have its scope restricted to chapter 6.

agents to be point objects and represented by nodes in a graph. This chapter would build upon the methodologies provided by [71] to take into account the orientation of the agents while computing a collision-free multi-agent path plan.

6.2.1 Graph Representation

In a plane, two points define a rigid body pose. We derive the orientation of a SaD agent when any two leg placement of the SaD agents on specific pins are provided. This enables us to deviate from the previous work where centroid was used to describe the location of an agent, wherein here it suffices to just provide the position of two legs of the SaD agent. Hence the graph is modified as follows.



(a) 60° rotation around pins **(b)** $60°, 120°$ and $180°$ rotations

Figure 6.1: Graph G

The bench has m pins in an equilateral triangular lattice with coordinates $P = \{p_1, p_2, \cdots, p_m\}$, $p_i = (x_i, y_i)$. A graph $G = (V, E)$ is defined with nodes (or graph vertices) $V = \{v_i | i = 1, \dots, m\}$, identifying the docking pins of the fixed bench, i.e., where an agent's leg can be mounted. The graph G in the current work differs from the previous work where the vertices V were representative of the centroids of the triangles of the lattice (agent's positions with respect to the bench pins). The graph G used in the current work aids to formulate accurately the current pose of the agent i.e. which agent's leg is on which pin. A connecting edge, $(v_i, v_j) \in E$, indicates the possibility of a leg to move between nodes v_i and

v_j by rotation around a pin in a single step. The adjacency list of the vertex v is $\mathrm{Adj}(v) \subseteq V$, where $v \in V$ and $w \in \mathrm{Adj}(v)$ iff $e = \{(v, w)\} \in E$.

A representation of the graph is shown in Fig. 6.1. In Fig. 6.1a, a SaD agent is represented by dotted red circle circumscribing an equilateral triangle ABC. The pose of the agent can be described by specifying the position of side AB in graph G. A new edge in graph G represents the various orientations side AB can achieve when the eq. triangle ABC is rotated by $\pm 60°$ around the vertices A, B, and C. Current pose of side AB is shown in black color. The new configuration of the edge AB possible are depicted in Fig. 6.1a by subscripted numbers such as eg: A_1, B_1 and different colors. All possible orientation of the side AB when rotated in multiples of 60° is shown in Fig. 6.1b. From Fig. 6.1b, we can observe that there exists 15 edges, two edges have overlapping colors. For the 180° rotation, the edges overlap over themselves indicating both clockwise and anti-clockwise rotation provide the same pose. Hence, by observation it is clear that there exists at-most 18 adjacent nodes for a given node in graph G. When considering each node is adjacent to itself, we get "at-most" 19 adjacent nodes

The centroids still play an important role in defining the agent position at a particular time instant t. A graph G' is defined to represent all possible centroids of the bench pins. A representation of G' is shown in Fig.6.2. $G' = (V', E')$, $V' = \{v'_i | i = 1, \ldots, |V'|\}$, identifying the centroids of the triangles of the lattice. Connecting edge $(v'_i, v'_j) \in E'$ when there exists a common pin between two centroids/vertices in G'. In Fig. 6.2, the centroids/vertices which lie on the hexagon (green, yellow and pink) are all adjacent to the centroid at the intersection of all the 3 hexagons. Hence by representation, it is clear that a vertex in graph G' can have at most 13 adjacent vertices (including itself).

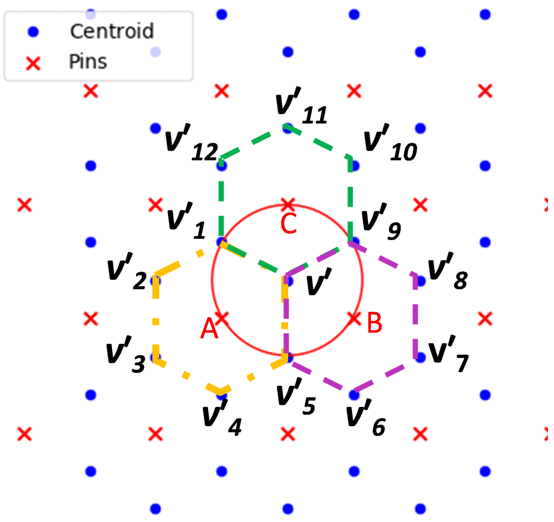


Figure 6.2: Graph G': v' represents the current centroid where the agent ABC is located on the bench. $v'_1, v'_2, v'_3, \cdots, v'_{12}$ represent the adjacent centroids/vertices of v'

6.2.2 Obstacle Representation

Obstacles are represented as convex polygons in a 2D environment. A pentagon or a hexagon which encapsulates 3 pins each, are placed at random. Similar to previous work, a Minkowski sum operation is performed between the obstacle geometry and the circular agent diameter to obtain a safe zone for collision avoidance. Also, a point in polygon test is performed to identify all pins/ vertices inside the obstacle region [53]. The ILP formulations are relaxed accordingly. The centroids inside obstacles are listed with the array $K = \{k_1, k_2, \cdots, k_n\}$, $k_i \in V$.

6.2.3 Multi-goal Representation

Each agent is allowed to have multiple goal assignments. Goals in the current context indicate the agents legs on a particular set of pins. For this scenario, it is important to introduce a centroid list which has all the permutations of the pins with respect to its corresponding legs and centroids. The centroid list for agents with L legs is represented as: $\delta = \{\delta_1, \cdots, \delta_q\}$, where $q = L!|V'|$, $\delta_i = \{\delta_{i1}, \delta_{i2}, \cdots, \delta_{iL}\}$ and $\forall i, \forall j, \delta_{ij} \in V$ and $\forall i$ every δ_{ij} is unique. For example, when an agent with $L = 3$ legs is mounted on three pins k, l, m: their corresponding centroid is represented as c , then their corresponding entry in the centroid list is as follows $\delta_1 = \{k, l, m\}$, $\delta_2 = \{k, m, l\}$, $\delta_3 = \{l, k, m\}$, $\delta_4 = \{l, m, k\}$, $\delta_5 = \{m, l, k\}$, $\delta_6 = \{m, k, l\}$.

Each element in δ represents all possible positions of the agent's legs on the fixed pin base (similar to centroids) and δ_{ij} represents the vertex/pin occupied by the j^{th} leg of an agent. Hence, $f : \delta \twoheadrightarrow V'$ is a surjective function, where all possible leg positions on pins are mapped to their corresponding centroids.

The planning problem requires visiting of ordered goals. Hence, a time-based ILP [80] is used: G is extended to a temporal graph, or a time-expanded network, as in [78]. The network is based on a fixed discrete time horizon, T where T copies of the vertices of G [75] are created. All ordered goal assignments for each agent is unique. The ordered list of tasks to be visited by agent r is $W_r = \{w_1, w_2, \cdots, w_{N_r}\}$, $w_i \in \delta$. With surjective function f we can obtain the corresponding centroids of the goal states $C_r = \{c_1, c_2, \cdots, c_{N_r}\}$, $c_i \in V'$.

The initial locations of SaD agents is an injective map $a_I \colon \{1, \cdots, R\} \rightarrow \delta$, where $R < |V|$ and all assignments in a_I are distinct.

6.3 Integer Linear Programming Formulation

An integer linear programming formulation with four indices, for the node, leg, time, and agent, is presented. Binary variables $x_{i,l,t,r}$ of size $(|V||L|TR)$ are created: we have $x_{i,l,t,r} = 1$ iff agent r's leg l is in vertex/pin i at time step t. As mentioned in section 6.1, the objective function for the ILP is to minimize the makespan and maximize the number of idle states for each agent.

6.3.1 Makespan minimization

With a fixed time horizon network approach, T provides a suitable lower bound for the objective function. The shortest path between the agents and the task goals [75] provides a suitable estimate value for T. All-pair shortest path (APSP) provides the distance matrix $d_{i,j}$ (the shortest path distance between vertices i and j in G'; every edge is assumed with length 1). For a time expanded network it is reasonable to make the assumption that $d_{i,i} = 1$. For an agent r and a task-allocation list $W_r = \{w_1, w_2, w_3, \cdots, w_{N_r}\}$, the minimum number of steps required by agent r to reach its N_r goals is calculated with APSP distance matrix neglecting other agents on the graph as follows:

$$d_r = d_{a_I(r),C_r(1)} + \sum_{n=1}^{N_r-1} d_{C_r(n),C_r(n+1)} \tag{6.1}$$
$$1 \leqslant \ r \leqslant R,\ N_r \geqslant 1$$

The maximum of all the d_r, $t_{\min} = \max(d_r)$, provides the lower bound for time horizon T for makespan minimization objective [75]. d_r also provides the information if the graph is connected, i.e there exists a path between the input state and the corresponding output goals.

6.3.1.1 Objective Function

The objective is to minimize the makespan f_1 of all the agents. Formulation starts with an initial time horizon $T = t_{\min}$ and increased iteratively until a feasible solution is found. All assumptions from the previous chapter 5 is applied to the current formulation.

$$\text{Minimize} \quad f_1 = \sum_{i \in V} \sum_{t=0}^{T} x_{i,l,t,r} \tag{6.2}$$

for some arbitrary agent r, $1 \leqslant r \leqslant R$ and some arbitrary leg l, $1 \leqslant l \leqslant L$.

subject to constraints:

6.3.1.2 Constraints

Agent initiation

The agents' positions are initiated with each leg on their respective docking pins. The formulation assumes that all agents are initiated at the same instant. The initial position of agent r with its corresponding legs l on the pins are an element of the matrix a_I, where r and l correspond to the element and index of a_I respectively.

$$x_{a_I(r,l),l,0,r} = 1 \tag{6.3}$$
$$1 \leqslant r \leqslant R, 1 \leqslant l \leqslant L$$

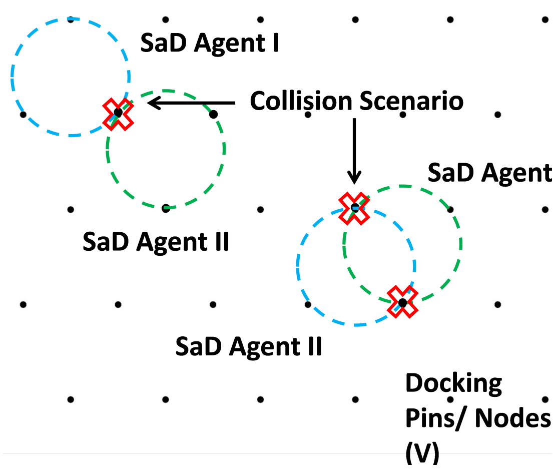


Figure 6.3: Collision scenario in SaD

Goal Assignment

Single- or multi-goals for each agent and their corresponding leg positions are assigned. This assignment enables the orientation planning:

$$\sum_{l=1}^{L} x_{W_r(k,l),l,t,r} - (L-1) \leqslant y_{k,t,r} \tag{6.4}$$

$$y_{k,t,r} \leqslant x_{W_r(k,l),l,t,r} \quad 1 \leqslant l \leqslant L \tag{6.5}$$

$$1 \leqslant k \leqslant N_r, 0 \leqslant t \leqslant T, 1 \leqslant l \leqslant L, 1 \leqslant r \leqslant R$$

The following constraint ensures that the assigned goals are reached at least once before the planning time horizon is completed.

$$\sum_{t=0}^{T} y_{k,t,r} \geqslant 1 \tag{6.6}$$

$$1 \leqslant k \leqslant N_r, 1 \leqslant r \leqslant R$$

Collision Avoidance

Collision scenario occurs (Fig. 6.3) when a pin is occupied by two different agents in a given time step t. Hence only one agent is allowed to occupy a given node.

$$\sum_{l=1}^{L}\sum_{r=1}^{R} x_{i,l,t,r} \leqslant 1 \tag{6.7}$$

$$0 \leqslant t \leqslant T, \forall i \in V$$

Obstacle avoidance

Stationary obstacles in the bench are represented by the list K. Over the time horizon, they are instantiated to not be used by the agents.

$$x_{i,l,t,r} = 0 \quad \forall i \in K \tag{6.8}$$

$$1 \leqslant l \leqslant L, 0 \leqslant t \leqslant T, 1 \leqslant r \leqslant R$$

Move/wait Constraint I

The decision to move/wait is modeled with a binary variable $z_{i,l,t,r}$, where $z_{i,l,t,r} = 1$ would represent the leg l of agent r is present both at time step t and $t+1$ on the pin i. The corresponding constraint is modeled as follows:

$$x_{i,l,t,r} + x_{i,l,t+1,r} - 1 \leqslant z_{i,l,t,r} \tag{6.9}$$

$$z_{i,l,t,r} \leqslant x_{i,l,t,r} \tag{6.10}$$

$$z_{i,l,t,r} \leqslant x_{i,l,t+1,r} \tag{6.11}$$

$$1 \leqslant l \leqslant L, 0 \leqslant t \leqslant T-1, 1 \leqslant r \leqslant R, \forall i \in V$$

Move/wait Constraint II

The decision to either rotate around one leg (move)/ all legs L to be stationary (wait) at a particular time step t is described as below. The move/wait choice are made with corresponding binary variables $u_{t,r}$ (move) and $v_{t,r}$ (wait).

$$\sum_{\forall i \in V} \sum_{l=1}^{L} z_{i,l,t,r} = u_{t,r} + L v_{t,r} \tag{6.12}$$

$$u_{t,r} + v_{t,r} = 1 \tag{6.13}$$

$$0 \leqslant t \leqslant T, 1 \leqslant r \leqslant R$$

Continuity

The following constraint ensures that an agent leaves only from the node that it has entered in the previous step hence maintaining the continuity of the path.

$$x_{i,l,t,r} + x_{j,l,t+1,r} \leqslant 1 \tag{6.14}$$

$$1 \leqslant l \leqslant L, 0 \leqslant t \leqslant T-1, 1 \leqslant r \leqslant R, i \in V, \forall j \notin \text{Adj}(i)$$

Adjacent set

Adjacent nodes in graph G on which a SaD agent can be placed is enlisted and constraints are developed accordingly. A binary variable $p_{k,t,r}$ is defined to show that all possible assignments of the legs to their corresponding pins is from the δ array.

$$\sum_{l=1}^{L} x_{\delta(k,l),l,t,r} - (L-1) \leqslant p_{k,t,r} \tag{6.15}$$

$$1 \leqslant k \leqslant len(\delta), 0 \leqslant t \leqslant T, 1 \leqslant r \leqslant R$$

$$p_{k,t,r} \leqslant x_{\delta(k,l),l,t,r} \tag{6.16}$$

$$1 \leqslant k \leqslant len(\delta), 0 \leqslant t \leqslant T, 1 \leqslant r \leqslant R$$

One of the many configurations of the legs on the pins needs to be true for every time step t.

$$\sum_{k=1}^{len(\delta)} p_{k,t,r} = 1 \tag{6.17}$$

$$1 \leqslant r \leqslant R, 0 \leqslant t \leqslant T$$

Enforcing Sequence

The task allocation, W_r, for every agent is done prior to the path planning. This is realistic in an MHS, where loading/unloading stations are known for each agent. Hence the planning assumes that ordered goals are available for each agent to visit. Successive goals in the task list can only be visited after their predecessors. γ_r is a vector containing indices of elements δ which are common between δ and W_r.

$$\sum_{t=0}^{H} p_{j,t,r} \geqslant p_{k,H+1,r} \tag{6.18}$$

$$0 \leqslant H \leqslant T-1, 1 \leqslant r \leqslant R, 1 \leqslant k \leqslant \text{length}(\gamma_r), i \in \gamma_r(k), j \in \gamma_r(k+1)$$

where W_r is the task allocation list of agent r and length(γ_r) is the size of the vector γ_r.

Prevent agent disappearance

It is necessary to write constraints for agents to stay in the planning graph until the last agent has reached its destination. This prevents an agent from disappearing after it reaches its goal state and makes the node it occupies available. The formulation provides the agent the freedom to move once it has satisfied all its goal states. This actively prevents deadlocks.

$$\sum_{\forall i \in V} x_{i,l,t,r} = \sum_{\forall i \in V} x_{i,l,t,r+1} \tag{6.19}$$

$$0 \leqslant t \leqslant T, 1 \leqslant l \leqslant L, 1 \leqslant r \leqslant R-1$$

6.3.2 Idling Preference

The maximization of idling of agents are modeled as below.

$D_{\max}$, the minimum total moves needed by all the agents to reach their goals provides a suitable lower bound for the objective function .

$$D_{\max} = \sum_{r=1}^{R} d_r \tag{6.20}$$

$D_{\max}$ is calculated with APSP between each agent and its corresponding goal states (C_r).

6.3.2.1 Objective Function

The second objective f_2 is to maximize the number of idle states for each agent of idling maximization. The formulation is converted into a minimization problem by minimizing the non-idling moves performed by the agents. The non- idle moves are a binary choice in each time step defined by $u_{t,r}$ in Eq. (6.12) and (6.13). Constraints are as follows:

$$\begin{aligned} & f_2 = y \\ \text{subject to} \quad & y = \sum_{r=1}^{R}\sum_{t=0}^{T} u_{t,r} \\ & y \geqslant D_{\max} \\ & c_{t,r} \in \{0,1\} \end{aligned} \tag{6.21}$$

6.3.2.2 Linear Program

The solution from the makespan function model is set as the incumbent solution for the idling function. Hence the second objective function starts with a feasible solution. This makes it faster for the solver to converge to a solution.

$$\begin{aligned} \text{Minimize} \quad & f_2 \\ \text{subject to} \quad & (6.21) \\ & f_1 = t^* \end{aligned}$$

where t^* is the minimum makespan value obtained for f_1.

6.4 Simulations

6.4.1 Setup

Simulations are performed on a similar environment as in chapter 5, with an Intel Xeon Dual Core Processor 2.0 Ghz 64-bit OS with 16.0 GB RAM. Mathematical models are implemented on Gurobi 6.5.1 with Python 2.7 as interface.

There are 52 pins in the actual SaD bench in the SwarmItFix prototype. To have a symmetrical grid for computation, 36 pins are considered, where the graph G has $|V| = 36$ vertices and $|E| = 461$ edges. For the given graph G, the centroid graph G' has $|V'| = 49$ and $|E'| = 491$ edges. A sample motion trajectory of two SaD agents is represented in Fig. 6.4

Agents are randomly initiated on the graphs with random obstacles. Connectivity between the initial states and the goal states are verified with APSP matrix. If connectivity does not exist between the goal nodes and the initial state of agents, the agents and obstacles are re-initiated until a connected graph is obtained.

To understand the performance of the ILP models time optimality [81] ratio and distance optimality ratio is computed for the dataset. These optimality ratios provide an insight into the gap between optimal solution and the objective solution generated.

$$\text{Time Optimality ratio} = \frac{f_1}{t_{\min}} \tag{6.22}$$

$$\text{Distance Optimality Ratio} = \frac{f_2}{D_{\max}} \tag{6.23}$$

Smaller number of agents are the main scope for the SaD multi-agent path planning problem since the agents are considered to be physical entities. Convex polygonal obstacles are placed on the graph, where each polygon encapsulates 3 nodes in graph G. It represents a highly constrained environment since 16% of the graph is now covered with obstacles when 2 polygons are placed.

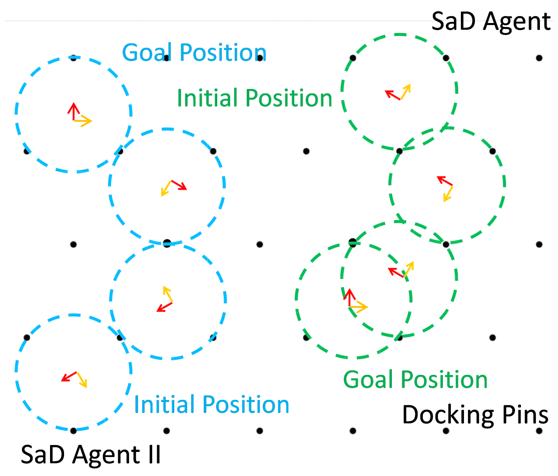


Figure 6.4: Motion trajectory of SaD agent

Table 6.1: Multi goal with sequence enforcing

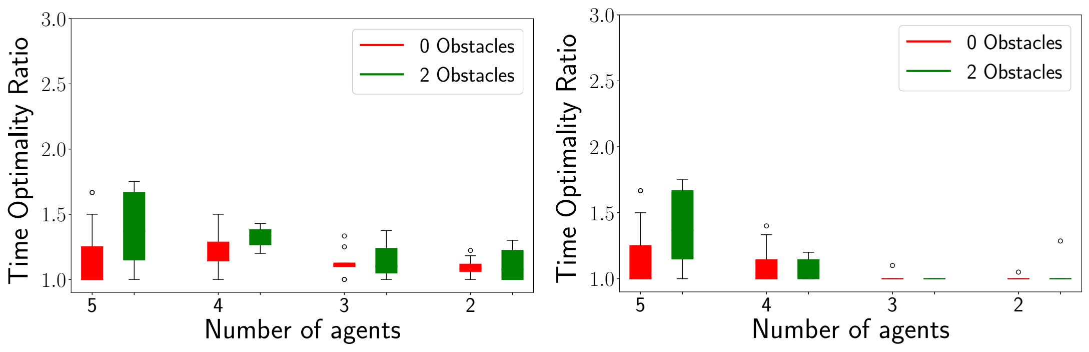
		Computation Time (s)			
		Formulation			
No. of Agents		5	4	3	2
No. of Goals		1	2	3	4
No. of Obstacles	0	69.46 (0)	528.99 (6)	885.116 (11)	649.06 (8)
	2	139.95 (12)	608.25 (17)	1112.59 (17)	685.194 (11)

6.4.2 Results

Table 5.4 present the average of the computation times of 20 randomly generated initial states, respectively. Shown are the times (in seconds) for optimizing the makespan function f_1. For each computation, termination is triggered by a time limit of 1500 s or an optimality gap of 10%. The number of cases which failed to generate a feasible solutions are displayed in parenthesis. 2 sets of computation time readings are recorded and the average displayed in the tables.

The optimality ratio shown in the graphs are based on a 95% confidence interval.

The task allocation problem where the required goal-sequence is not provided,

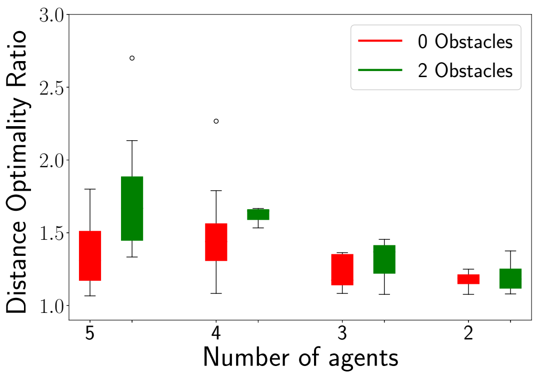


(a) With Sequence **(b)** No sequence

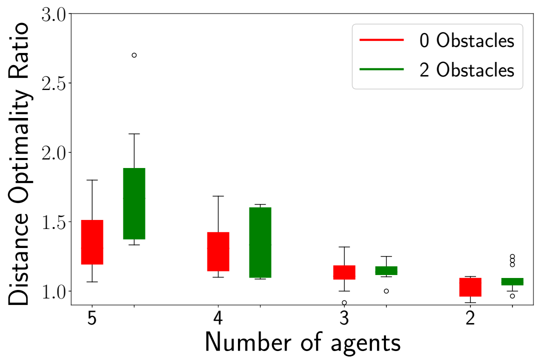
Figure 6.5: Time Optimality Ratio

Table 6.2: Multi goal no sequence enforcing

		Computation time (s)			
		Formulation			
No. of Agents		5	4	3	2
No. of Goals		1	2	3	4
No. of Obstacles	0	69.695 (0)	480.181 (2)	621.870 (3)	321.701 (0)
	2	146.875 (12)	349.097 (15)	644.78 (13)	178.631 (7)



(a) With Sequence



(b) No sequence

Figure 6.6: Distance Optimality Ratio

is also studied. Experiments were made to study the computation time when the goal-sequence constraint was relaxed for the same dataset. The minimum makespan achieved by relaxing the sequence constraints is shown in Fig. 6.5b and computation times are displayed in Table. 6.2. Results show that the solver provides near optimal solution at a reasonable time (Table 6.2).

The computation time results in the current study shows that the ILP model has shown an improvement from the previous chapter [80]. In previous chapter, experiments had failed to produce feasible solutions for many cases such as (5 agents, 2 obstacles: 4 agents, 2 obstacles) whereas the current model generates feasible solution. The computation time as observed for the multi-goal with sequence constraint seems to increase linearly with increase in number of goals. This is a clear indicator that the system becomes highly constrained. This is also evident from the multi-goal no-sequence condition, where the computation time is far lesser compared to the with-sequence constraint. These observations can be utilized to develop special heuristics to reduce the computation time. Although the computation time for an offline solver is not a very critical requirement, a better heuristic model can aid when there is limited resources and requirement for quicker solutions.

For the idling function objective, in previous chapter it was shown that a lesser termination time is sufficient to achieve results closer to 10 - 20 % of the optimal solution. Hence, the model was tested with lower termination times comparing distance optimality ratio achieved within 360 and 720 seconds. Results observed for both the time (360 and 720 seconds) were very similar. Fig. 6.6 shows the distance optimality ratio with sequence and no-sequence constraint (720 seconds). The observations made from Fig. 6.6 shows that the solutions obtained are within $1.x$ of the optimal solutions for most of the cases.

6.5 Summary

A multi-agent system with "swing and dock" locomotion system posed a unique planning problem of MPP on graphs. A collision-free path-planner based on a temporal graph with integer linear programming is developed. Constraints are defined for aiding the agents to achieve required orientation/positioning of legs on specific vertices in a graph. A lexicographic multi-objective optimization was computed where minimizing of the makespan time of the agents is followed by maximizing the number of idle states. The ILP formulations effectiveness is shown with measures such as time and distance optimality ratio, where experiments display solutions closer to $1.x$ of the optimal solution. The computational time observed were reasonable for an offline planning problem. The current work was solely focused on generalized formulations for the SaD problem, wherein future work with specific heuristics can greatly reduce the computation time.

Chapter 7

Trajectory Planning

In sheet metal manufacturing, re-configurable fixtures significantly reduce the lead time and cost. In this chapter, robotic agents with discrete-step locomotion and interchangeable supporting heads are proposed to act as fixturing locators. These agents swing (rotate) around fixed pins on a bench to reach adjacent reachable pins, such a locomotion avoids slip and provides accurate localization of the agent. Due to their simplicity in design and movement, they can be scaled to a multi-agent system. We propose to use these agents mounted with static fixture heads instead of a parallel manipulator (SwarmItFix EU FP7) used in previous works. Such a modification would enable to scale to a multi-agent systems and also would be cost-effective alternative. Integer linear programming optimization techniques are employed to ensure accurate positioning of the agents in time to provide adequate support during machining. Multi - agent path planning w.r.t the tool trajectory, collision avoidance, and time - relevant action plan is implemented. Mathematical formulations for the constrained optimization are implemented with GUROBI solver. Simulation and optimization results are presented and also they provide insight for parameter tuning, based on which design decisions for selection of geometry of the fixture head, base, and tool speed can be made.

7.1 Introduction

Sheet metals are the most commonly used raw material in manufacturing industry. The size of sheet metals are dependent on the industries: the aerospace, automobile and naval industries consume them in large dimensions. When the length to thickness ratio is too high, these sheet metals tend to sag under self-weight, since they need to undergo lot of manufacturing operations, fixture development for these large sized sheet metals has been a very widely research topic.

Specific fixtures for sheet metal/ thin walled objects have been proposed and developed such as reconfigurable modular fixturing system [82]: fixturing thin walled flexible objects subject to a discrete number of point forces , Single structure flexible fixture system (SSFFS): matrices of support ("bed of nails" system) for the sheet metal, Robotic Fixtureless assembly (RFA): sensor guided robots to cooperatively fixture the workpiece.

From observation of these system it is clear that the core components of these fixture systems are: a specially designed base plate, a locator and a clamping mechanism. Various combinations of these components generate a new class of fixtures. One of the most relevant fixture to the current work being discussed is of the pin-type fixture, where [83] several axially actuating rods/pins are placed in a grid like pattern on a mounting base to support different workpiece geometry (Flexible tooling apparatus [84], Variable contour securing systemm [85], Universal holding fixture [86], POGO Flexible tooling). The EU FP7 project (SwarmItFix) [2, 5, 6], an improved version of the RFAs, where a reconfigurable mobile fixture consisting of a parallel manipulator mounted on a mobile base was developed to cater to fixturing of large aerospace sheet/skin materials. The SwarmItFix project was successful in automating the entire fixturing option. The usage of parallel manipulators to locate and clamp the sheet metals/skins has made the planning procedure quite complex and expensive to scale to a multi-agent system. The reconfiguration time of the parallel manipulator and their intersecting workspaces limit their ability to fixture sheet metal with different tool speeds and geometry. This work proposes to extend the concept of the SwarmItFix, where the mobile base is mounted with a static fixture head. Such a modification would simplify the overall design of the system, reconfiguration time can be reduced,

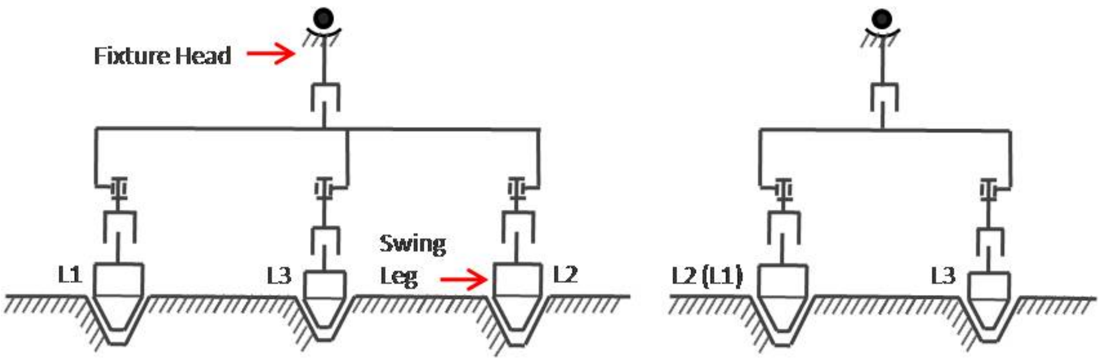


Figure 7.1: Schematic of the SaD base with a fixture head

more complex geometries can be fixtured, and the overall cost of each agent can be reduced.

7.2 Design

The SaD base is proposed to be extended to mount a static fixture head rather than an expensive PKM. This concept of having a reconfigurable locator was intially proposed by Selena [82], where a novel reconfigurable modular system for the fixturing of thin-walled, flexible objects subject to a discrete number of point forces was presented. Several other pin-type fixtures operate on a similar principle where a bench consists of fixtures placed in a grid like pattern which extend to provide support to sheet metals. These designs served as a basis during the development of the SwarmItFix fixture. A simple schematic of the proposed design shows the swinging locomotive agents carrying fixture heads which can extend and retract in one dimension to support sheet metal parts.

7.3 Problem Definition

In this chapter, we develop the high level planning of the SaD agents for the fixturing purpose. Apart from this application, such a planner would be useful to locate SaD agents at a given instant in time along an arbitrary trajectory.

7.3.1 Graph Representation

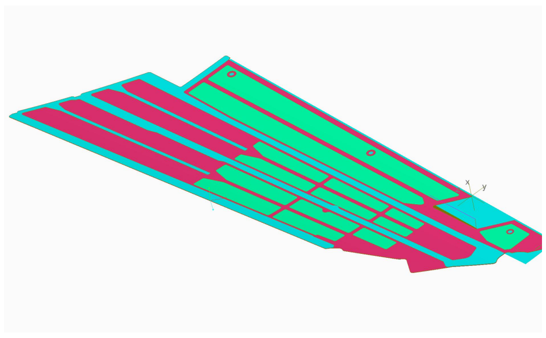
The planning graph G is similar to the graph G considered in chapter 5. The bench has m pins in an equilateral triangular lattice with coordinates $P = \{p_1, p_2, \cdots, p_m\}$, $p_i = (x_i, y_i)$. A graph $G = (V, E)$ is defined with nodes (or graph vertices), $V = \{v_i | i = 1, \ldots, |V|\}$, identifying the centroids of the triangles of the lattice, i.e., the possible agent positions. (Agent orientation is ignored.) A connecting edge, $(v_i, v_j) \in E$, indicates the possibility to move between nodes v_i and v_j using only one pivot, i.e., their triangles are adjacent (share a side or a vertex).

7.3.2 Workpiece

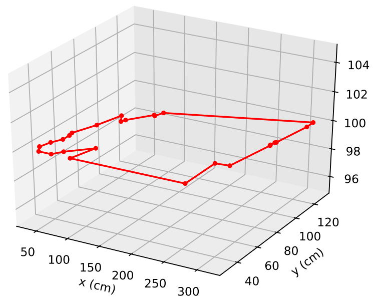
Sheet metal workpieces in the scenario are not planar, but have complex 3d contour shapes. CAD data of the workpiece to be machined is obtained prior to the planning process. They are converted into IGES format to extract the point cloud data of the surface to be fixtured. The point cloud data are easy to manipulate and use in the Python environment. Fig. 7.2a and 7.2b The distance between the SaD agent in the current prototype and the bench is 500 mm, in SwarmItFix design the maximum height the PKM could reach was 1100 mm from the mounting base. We assume that the static fixture head can reach a distance of 1000 mm from the bench. The workpiece points are projected onto the base of the SaD bench. The workpiece needs to be oriented in a manner where maximum number of mounting points can be placed. Point in polygon test is performed to identify all pins/vertices inside the projected workpiece region [53], the same method employed to identify obstacles in the bench. Hence, we can obtain the optimal placement of the workpiece for fixturing.

7.3.2.1 Workpiece Segmentation

Similar to the work carried out for the SwarmItFix planner [49], we discretize the contour points. Machine tool data is provided, where the tool speed is provided. In most conditions tool speed are assumed to be uniform. Hence, we segment the contour based on the location of the tool at every second. We assume this strategy



(a) CAD Model



(b) Projected Points

Figure 7.2: Vertical fin model

would provide a good estimate of the contour points. APSP algorithm provides the nearest possible position of the agent w.r.t the contour points.

7.4 Integer Linear Programming Formulation

The vertex based formulation presented in chapter 5 with three indices, for the node, time, and agent, are extended to formulate the current problem. Unlike the previous cases, in the current problem the planning time horizon is fixed, hence the ILP would either return a solution or would terminate stating in-feasibility of the model or violation of constraints. The environment is a known environment, hence the scenario is considered to be obstacle-free. When nodes in graph G are further away from contour points, the nodes and their corresponding edges are deleted. This significantly improves the computation time for the solver. The ILP formulation implemented in the previous chapters 5 and 6 required the time step of all agents to be uniform, i.e if an agent has to move or idle, it has to perform that action for the same amount of time. Most MPP in literature adhere to these relaxations as it is easy to model in a planning graph. But for a real world environment such a constraint would hinder the performance of the system. Hence, we model the constraints for the agent such that asynchronous movement is possible for each agent and its corresponding action.

7.4.1 Objective Function

The objective function f_1 is to maximize the support provided to the sheet metals. The cost function is defined here to be the distance between the support points and the tool trajectory. The cost-distance matrix provides the weighted coefficients $c_{j,t}$ for the minimization function. It is clear from the assumption, that only one contact point is sufficient to provide adequate support at any time instant [49].

The objective function is independent of the number of agents.

$$\text{Minimize} \quad f_1 = \sum_{j \in V} \sum_{t=0}^{T} (c_{j,t})(x_{j,t})$$

subject to constraints:

7.4.1.1 Constraints

Choice of support points

Only one support point is sufficient to provide adequate fixture support (assumption) for every time step (t). This assumption is derived from the SwarmItFix system, where when one agent was providing support the other agent was either in reconfiguration or had already reached its next desired position.

$$\sum_{j \in V} x_{j,t} = 1 \tag{7.1}$$

$$0 \leqslant t \leqslant T$$

Agent initiation

All agents are initiated at time step ($t = 0$) at required centroid j based on the cost. This constraint ensures that two agents are not in collision while the planning starts in the planning graph. No priority is assigned to any agent manually/by the user.

$$\sum_{j \in V} y_{j,0,r} = 1 \tag{7.2}$$

$$1 \leqslant r \leqslant R$$

Same support point at next time step (t)

When the tool progresses around the trajectory, it may suffice to assign an robot/agent to be positioned strategically to provide continuous support over a length of the trajectory. In such cases, the same support point j needs to be chosen for an extended time period rather than a single step. When such a scenario exists, the binary variable $s_{j,t}$. is enabled.

$$x_{j,t} + x_{j,t-1} - 1 \leqslant s_{j,t-1} \tag{7.3}$$

$$x_{j,t} \geqslant s_{j,t-1} \tag{7.4}$$

$$x_{j,t-1} \geqslant s_{j,t-1} \tag{7.5}$$

$$\forall j \in V, 1 \leqslant t \leqslant T$$

Relating agent to the support point

One of the agents are assigned for each support point. This constraint ensures that there exists fixturing agents available throughout the machining process.

$$x_{j,t} \leqslant \sum_{r=1}^{R} y_{j,t,r} \tag{7.6}$$

$$\forall j \in V, 1 \leqslant t \leqslant T$$

Collision Avoidance constraint

A centroid j can at-most be occupied by only one agent. Two/more agents occupying the same centroid will lead to agent collision.

$$\sum_{r=1}^{R} y_{j,t,r} \leqslant 1 \tag{7.7}$$

$$\forall j \in V, 0 \leqslant t \leqslant T$$

Agent move/dock at time step (t)

We declare a binary variable $m_{t,r}$ to represent if an agent r is moving at a particular time step t. $m_{t,r} = 1$, if agent r is moving/rotating at time step t. The following constraint defines that every agent r should be moving or docked in a centroid j at every time step t.

$$\sum_{\forall j \in V} y_{j,t,r} + m_{t,r} = 1 \tag{7.8}$$
$$0 \leqslant t \leqslant T, 1 \leqslant r \leqslant R$$

Ensuring agent (r) moves for Tm steps before docking

We define the time taken by each agent to reach its adjacent node to be T_m, where $T_m \in \mathbb{Z}^+$. When the agent takes the decision to move, i.e $m_{t,r} = 1$ in Eq. 7.8, the agent continuous to be in the move state in the consecutive time steps until T_m steps elapse.

$$(T_m - 1) - \sum_{t_n=t+1}^{t+T_m-1} m_{t_n,r} \leqslant \mathrm{M}((1 - m_{t,r}) + (1 - y_{j,t-1,r})) \tag{7.9}$$
$$\forall j \in V, 1 \leqslant t \leqslant T - T_m, 1 \leqslant r \leqslant R$$

Agent needs to dock after Tm moves

Agent (r) needs to dock after T_m steps if it is moving in the current time step t. This is explicitly modeled in the following constraint:

$$m_{(t+T_m),r} \leqslant \mathrm{M}(T_m - \sum_{t_n=t}^{t+T_m} m_{t_n,r}) \tag{7.10}$$
$$0 \leqslant t \leqslant T - T_m, 1 \leqslant r \leqslant R$$

Ensuring agent (r) docks after Tm moves in adjacent centroid (j)

Every agent (r) after making T_m moves, docks in one of the adjacent centroid of current centroid j. This constraint is also similar to the continuity constraint modeled in the previous chapters, where agent moves to its adjacent node in the graph.

$$1 - \sum_{\forall i \in \mathrm{Adj}(j)} y_{i,(t+T_m+1),r} \leqslant \mathrm{M}((1 - m_{t,r}) + (1 - y_{j,t-1,r})) \tag{7.11}$$

$$\forall j \in V, 1 \leqslant t \leqslant T - T_m - 1, 1 \leqslant r \leqslant R$$

Agent idle at time step (t)

A binary variable $z_{t,r}$ is declared to represent the agent idling. When $z_{t,r} = 1$ in the current time step t, agent (r) idles at centroid (j) in the next time step .

$$1 - z_{t+1,r} \leqslant \mathrm{M}((1 - y_{j,t,r}) + (1 - y_{j,t+1,r})) \tag{7.12}$$

$$\forall j \in V, 0 \leqslant t \leqslant T - 1, 1 \leqslant r \leqslant R$$

Agent moves if idling not enabled in previous time step

As described in the previous constraint Eq. 7.12, $z_{t,r}$ denotes the idling of agent r, when $z_{t,r} = 0$ at time step t, it implies that agent needs to move in the next time step $t + 1$.

$$T_m - \sum_{t_n = t+1}^{t+T_m} m_{t_n,r} \leqslant \mathrm{M}(z_{t,r}) \tag{7.13}$$

$$0 \leqslant t \leqslant T - T_m, 1 \leqslant r \leqslant R$$

Two different centroids cannot be assigned to same agent (r) in consecutive time steps (t)

Agents need to move for certain time steps before they can reach their adjacent nodes. This is considering the time to move ($T_m > 1$). The following constraint is modeled as follows:

$$y_{j,t,r} + \sum_{\substack{i=0 \\ \forall i \neq j}}^{|V|} y_{i,t+1,r} \leqslant 1 \tag{7.14}$$

$$\forall j \in V, 1 \leqslant t \leqslant T - 1, 1 \leqslant r \leqslant R$$

Assignment of docking decision variable

A docking decision variable $l_{j,t,r}$ is required in the first place to distinguish between moving and docked agent. A docked agent in centroid j i.e. $y_{j,t,r} = 1$ invariably assigns $l_{j,t,r} = 1$. Hence, the pin around which the agent moves can be clearly identified with this constraint.

$$1 - l_{j,t,r} \leqslant \mathrm{M}(1 - y_{j,t,r}) \tag{7.15}$$

$$\forall j \in V, 0 \leqslant t \leqslant T, 1 \leqslant r \leqslant R$$

Agent needs to be docked/rotating around some mounting pins

All pins on which the agents are mounted in every time step can be monitored by the binary variable $l_{j,t,r}$. Hence, we constrain the variable $l_{j,t,r}$ to be assigned to any one centroid (j) at all time steps.

$$\sum_{\forall j \in V} l_{j,t,r} = 1 \tag{7.16}$$

$$0 \leqslant t \leqslant T, 1 \leqslant r \leqslant R$$

Assignment of docking decision variable

If agent is moving in the current time step $m_{t,r} = 1$, then assign the centroid j from the previous docked time step $t - 1$.

$$1 - l_{j,t,r} \leqslant \mathrm{M}((1 - m_{t,r}) + (1 - l_{j,t-1,r})) \tag{7.17}$$

$$\forall j \in V, 1 \leqslant t \leqslant T, 1 \leqslant r \leqslant R$$

Collision avoidance for the usage of docking pins

Since each agent occupies 3 docking pins, for a given centroid j all its adjacent centroids are restricted to only one agent. This avoids collision caused due to the usage of the mounting pins.

$$\sum_{\forall i \in \mathrm{Adj}(j)} \sum_{r=0}^{R} l_{i,t,r} \leqslant 1 \tag{7.18}$$
$$\forall j \in V, 0 \leqslant t \leqslant T$$

7.5 Simulations

Simulations were performed on an Intel Xeon Dual Core Processor 2.0 Ghz 64-bit OS with 16.0 GB RAM. Mathematical models are implemented on Gurobi 6.5.1 with Python 2.7 as interface.

The number of pins in the actual SaD bench in the SwarmItFix prototype is 52. The exact dimensions of the bench was modeled in Python.

The cost function variable $c_{j,t}$ was penalized when the distance between the tool point and the agent centroid position were more than 700 mm. This constraint arises based on the dimension of the head of the fixtures.

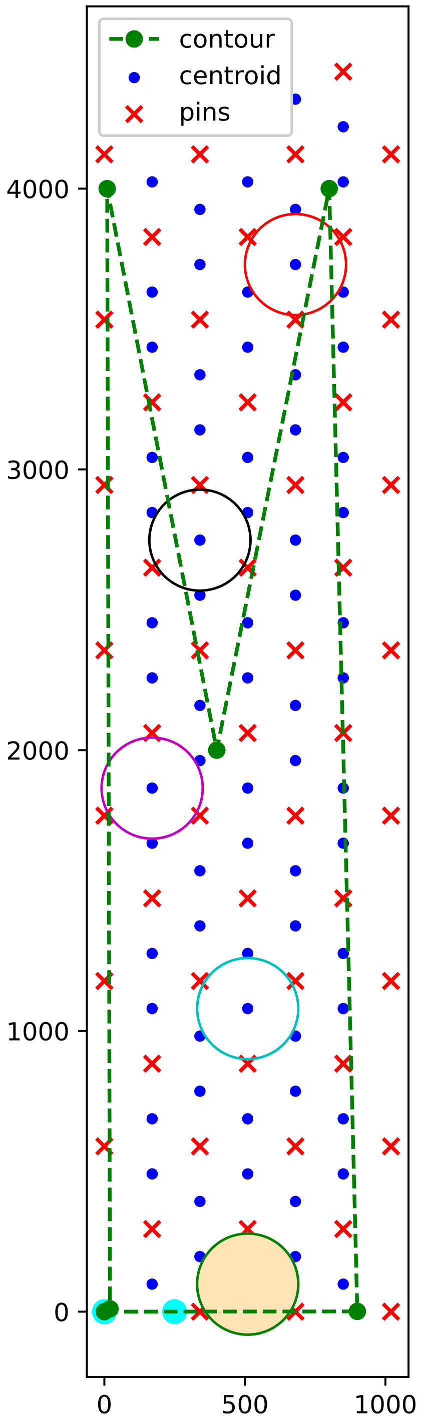
Termination time was not fixed while performing the simulation. This enables the solver to search the entire branch and bound tree to arrive at the optimal solution. Simulations were run to first understand if the ILP solver always terminates with the current constraint optimization problem. When the constraints were violated for the cost-function $c_{j,t}$, the model terminated immediately. When the number of agents were less, the solver took longer time to find a feasible solution. After the feasible solution was determined, finding a near -optimal solution (10 % gap) took long hours with the current formulation. Few instances the optimal solution were reached within minutes, whereas in few other contours and with lesser agents, the solver took days to reach the optimal solution. The solver also in a particular case took 226 hours to report infeasible model. Two major parameters that contribute towards an infeasible solution: number of agents and support function. Less number of agents may result in less time to reconfigure, whereas very high number of agents would result in no possible motion. When the support function is concerned, in the current research we have considered the geometric constraints, whereas Finite Element Method (FEM) constraints can play a major factor in this scenario. In this optimization problem we try to achieve a feasible solution by changing the parameters such as: number of agents, time to move and tool speed.

We present the simulation of a contour with 5 fixturing SaD agents in Fig.7.15. The planning horizon is determined by the tool speed. We arbitrarily chose 25 mm/s as the tool travel speed along the cutting contour. Hence, the trajectory was segmented into 51 points, representing 51 seconds for the tool to traverse the entire contour. The motion planning of the agents are displayed in the below figures. The centroids are represented as blue and the docking pins are represented as red. The contour on which the tool travels is represented in green color. All SaD agents are represented as hollow circles (light green, light blue, red, pink, and black). Cyan colored circles represent the tool position on the contour. A filled mocassin circle represents the current agent which is supporting the workpiece at that particular instant of time. From graphical representation in Fig.7.15, it becomes evident that the ILP formulation provide a feasible solution with no-collisions among agents at any time step t.

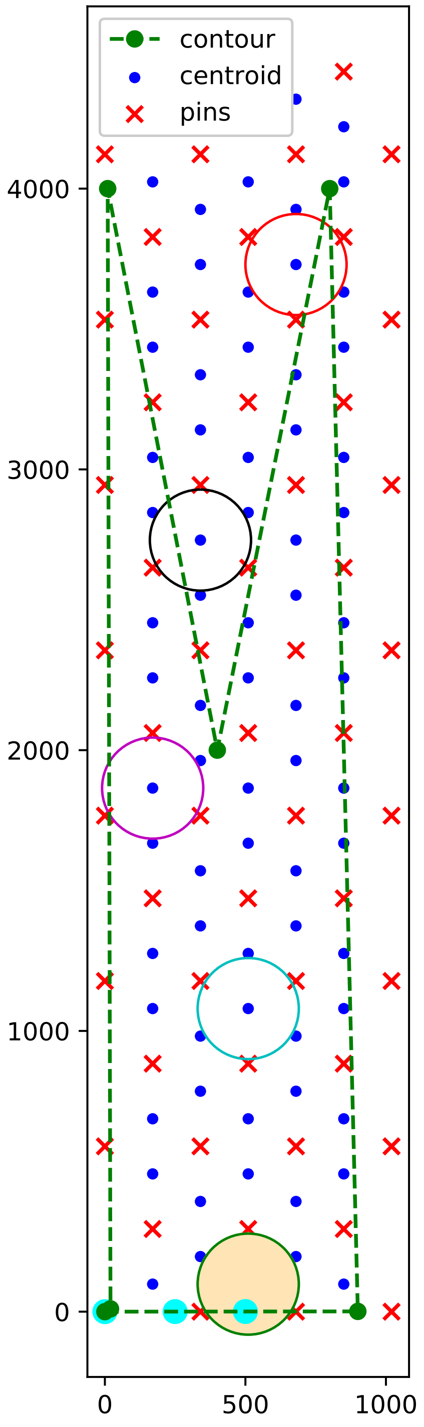
A feasible solution for the above contour was obtained within 3 hours, but to reach the optimality criterion of 10 % minimum gap, the solver took 387 hours. This shows that the model can be vastly improved from its current state by exploring in-depth the methods to search the branch and bound tree. Various heuristics can fairly reduce the computation time. This would be a potential research direction to take when this planner is implemented.

7.6 Discussion

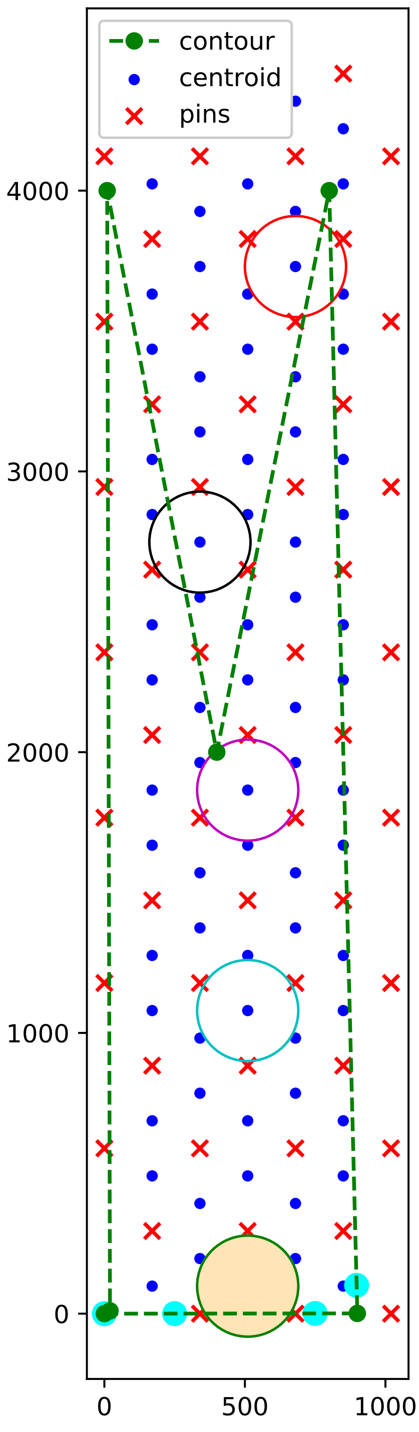
We have proposed to extend the SaD agent to act as a reconfigurable fixture with a simple fixture head. Such a solution would be easy to scale to multi-agent support and also would be a cost-effective solution. We have implemented constrain optimization techniques to locate these fixtures along the tool path trajectory . We have only included geometric constraints while modeling the mathematical models, but the optimization can easily take into account finite element constraints if required. We display the motion planning of the agents when a trajectory for a workpiece is provided. The ILP techniques ensure there is only a 10% gap between


contour
centroid
pins
4000
3000
2000
1000
0
0
500
1000

(a) t = 1

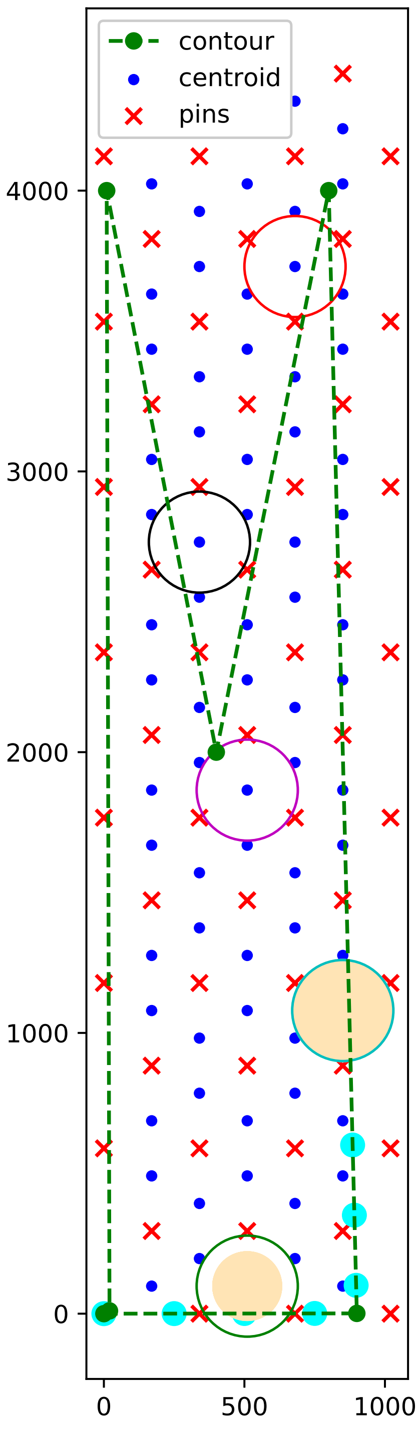

contour
centroid
pins
4000
3000
2000
1000
0
0
500
1000

(b) t = 2



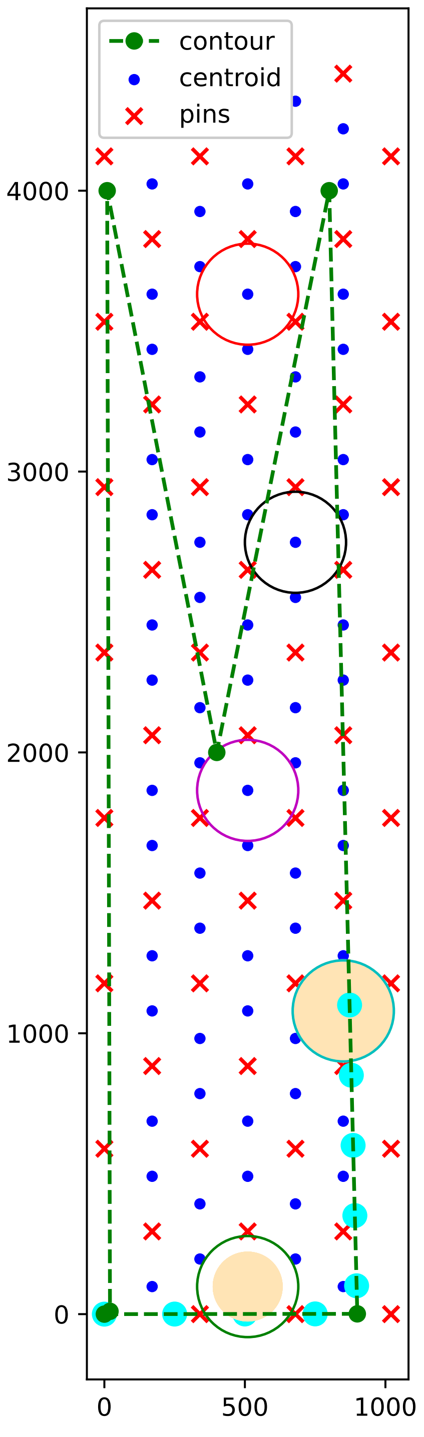
contour
centroid
pins
4000
3000
2000
1000
0
0
500
1000

(a) t = 4

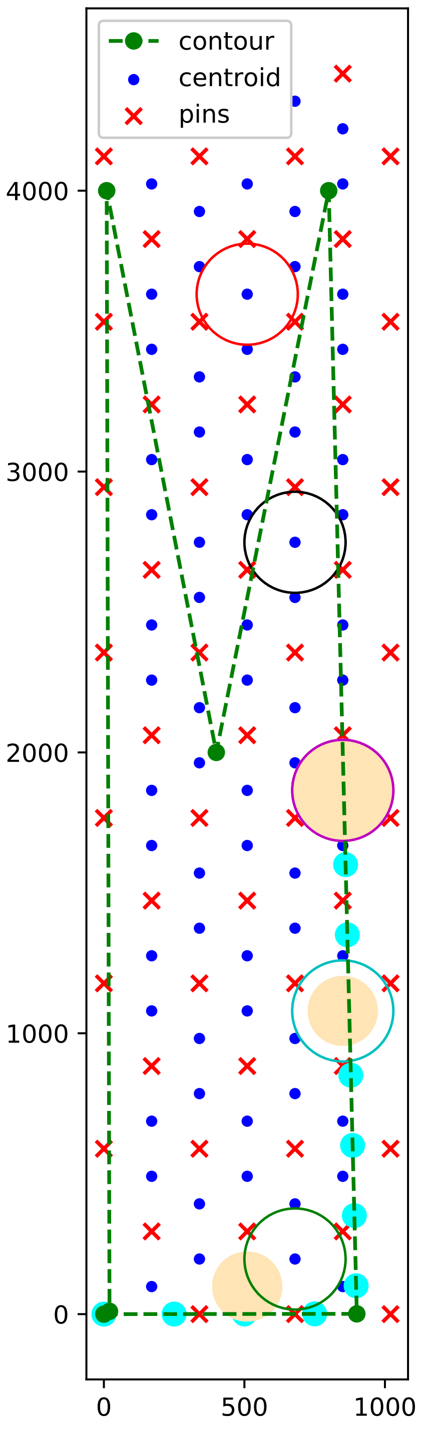


contour
centroid
pins
4000
3000
2000
1000
0
0
500
1000

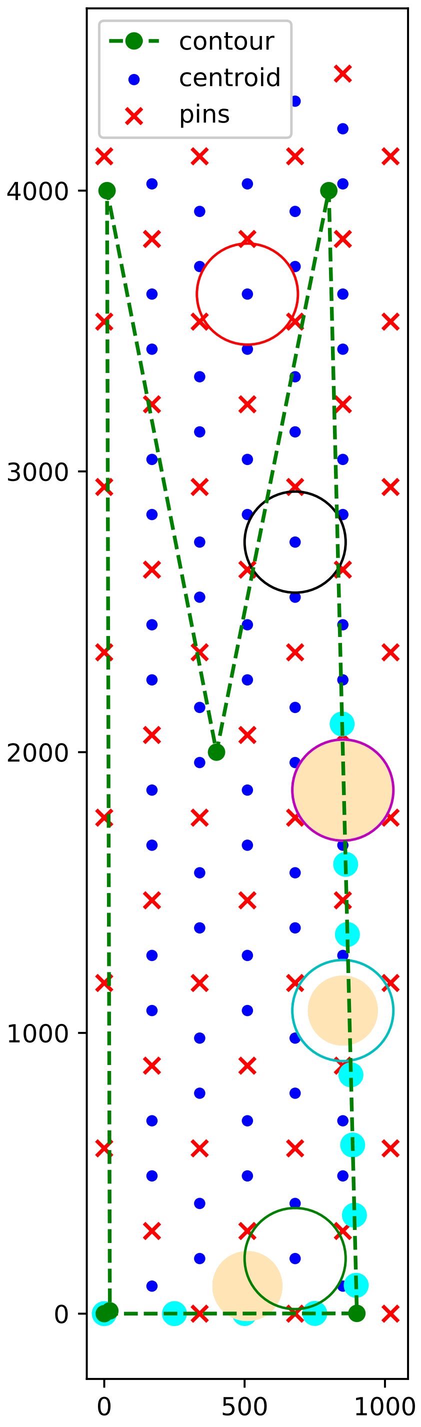
(b) t = 6


contour
centroid
pins
4000
3000
2000
1000
0
0
500
1000

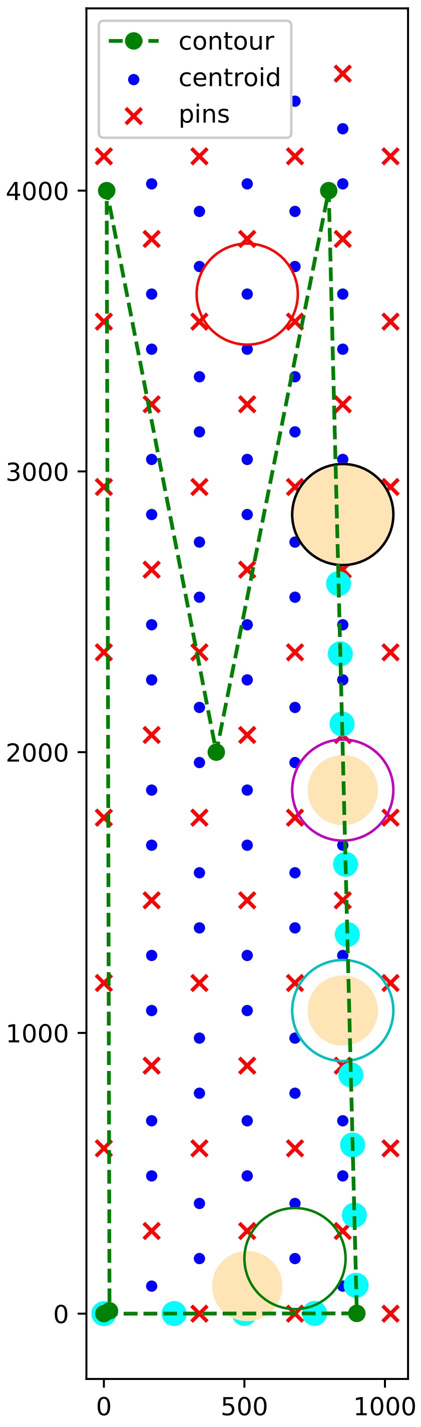
(a) t = 8


contour
centroid
pins
4000
3000
2000
1000
0
0
500
1000

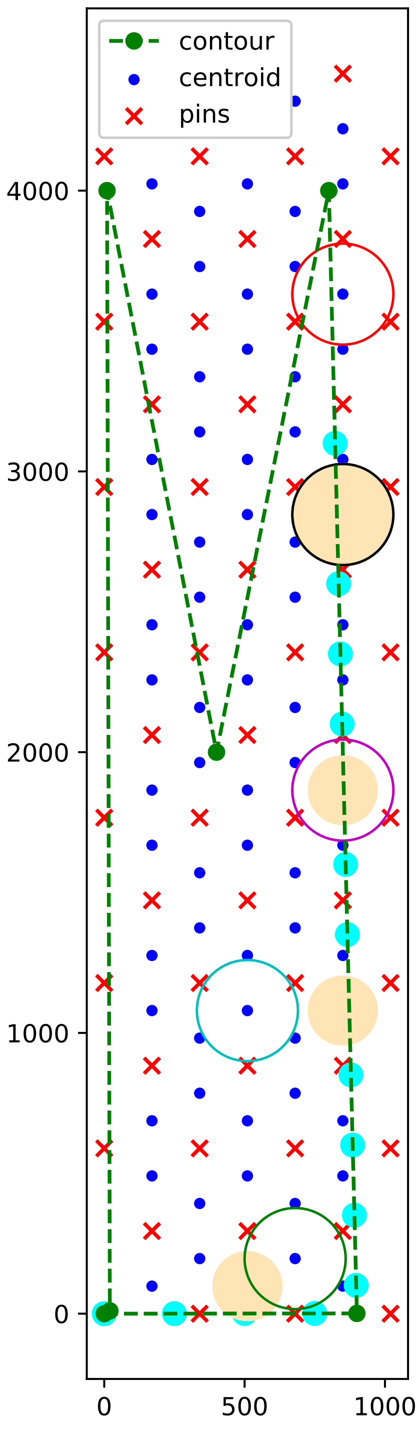
(b) t = 10


contour
centroid
pins
4000
3000
2000
1000
0
0
500
1000

(a) t = 12

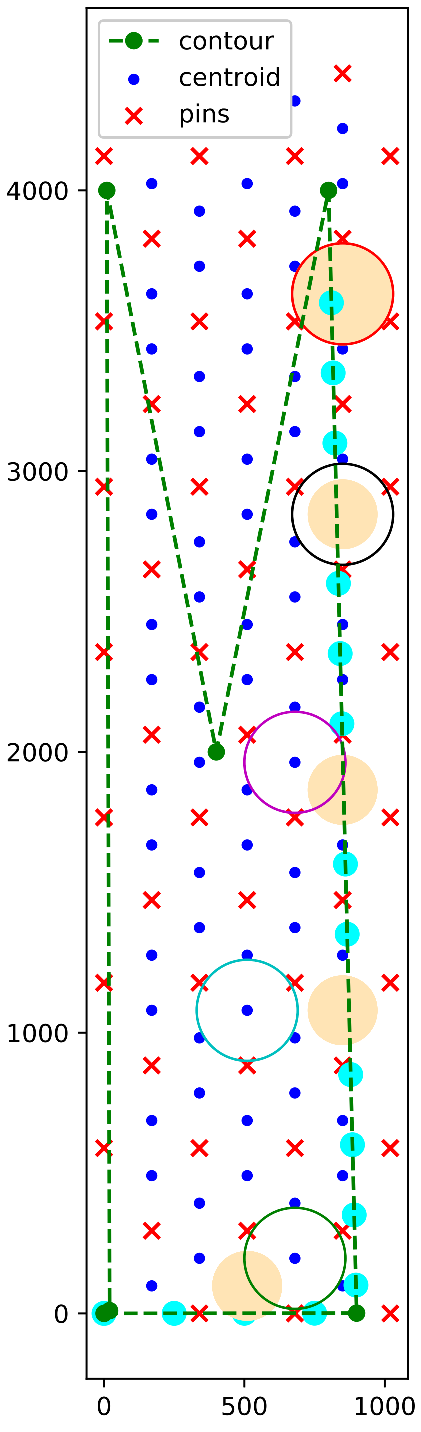

contour
centroid
pins
4000
3000
2000
1000
0
0
500
1000

(b) t = 14



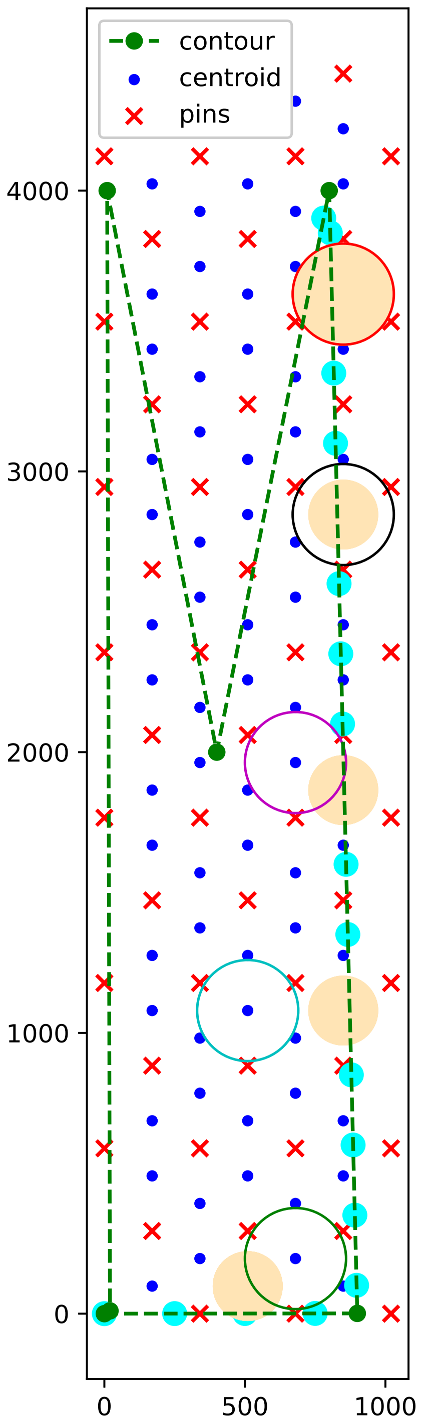
contour
centroid
pins
4000
3000
2000
1000
0
0
500
1000

(a) t = 16

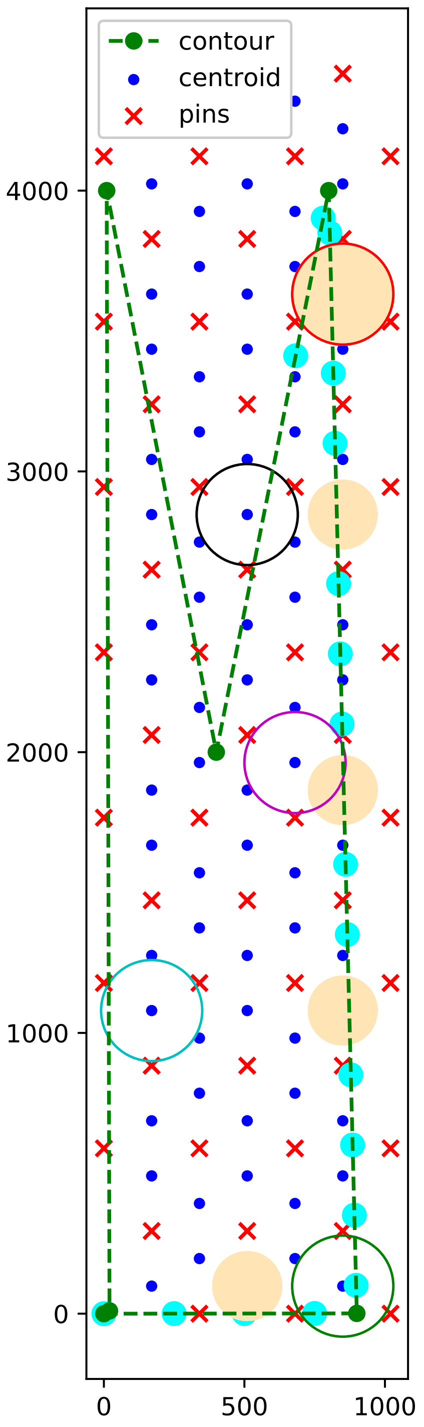


contour
centroid
pins
4000
3000
2000
1000
0
0
500
1000

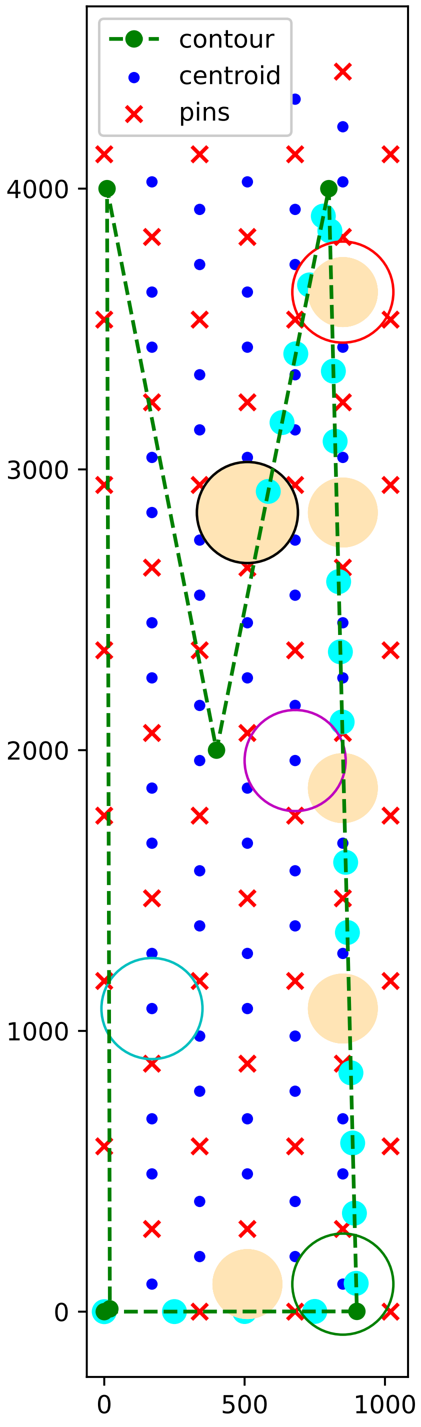
(b) t = 18


contour
centroid
pins
4000
3000
2000
1000
0
0
500
1000

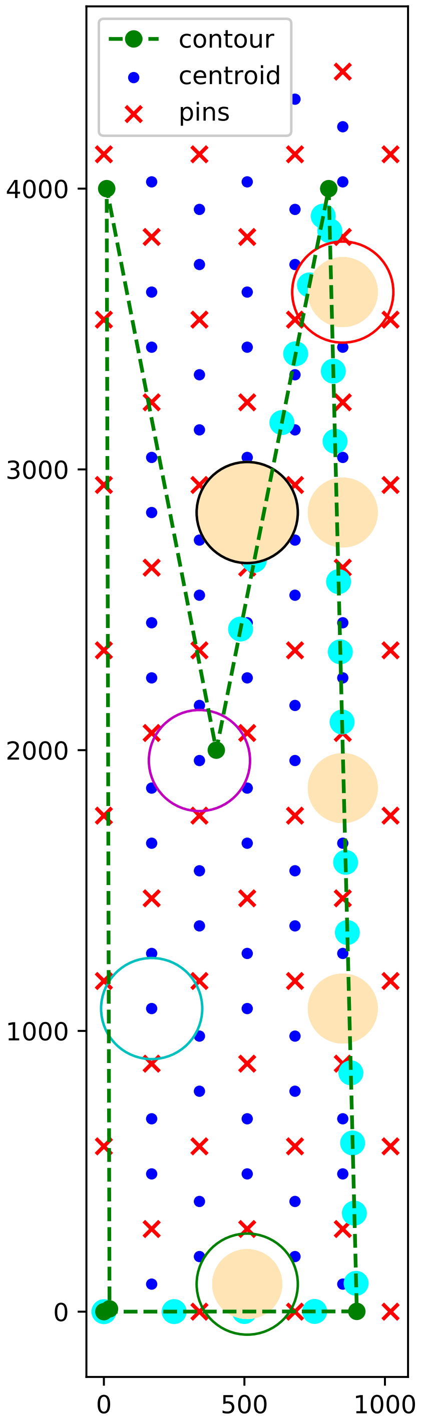
(a) t = 20


contour
centroid
pins
4000
3000
2000
1000
0
0
500
1000

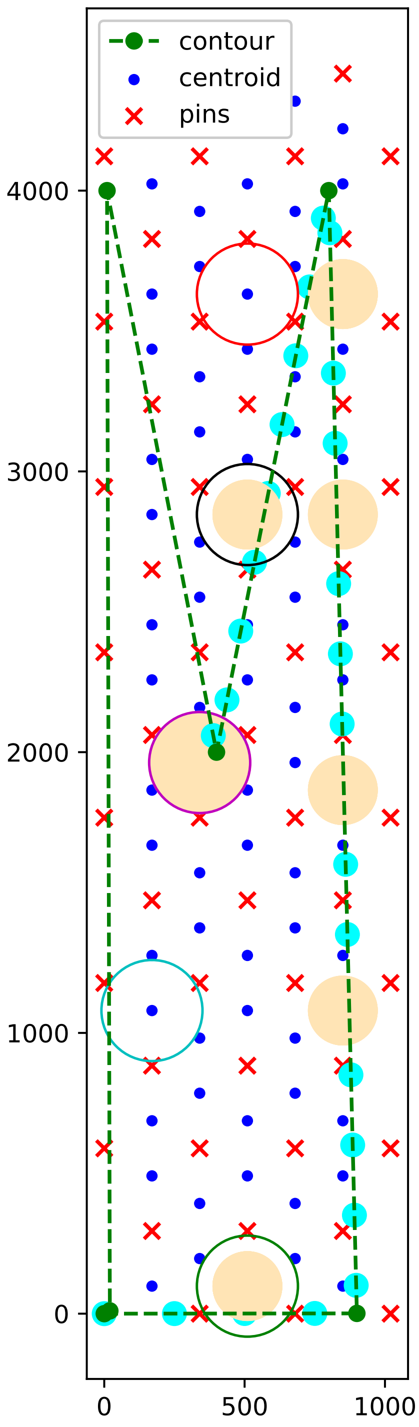
(b) t = 22


contour
centroid
pins
4000
3000
2000
1000
0
0
500
1000

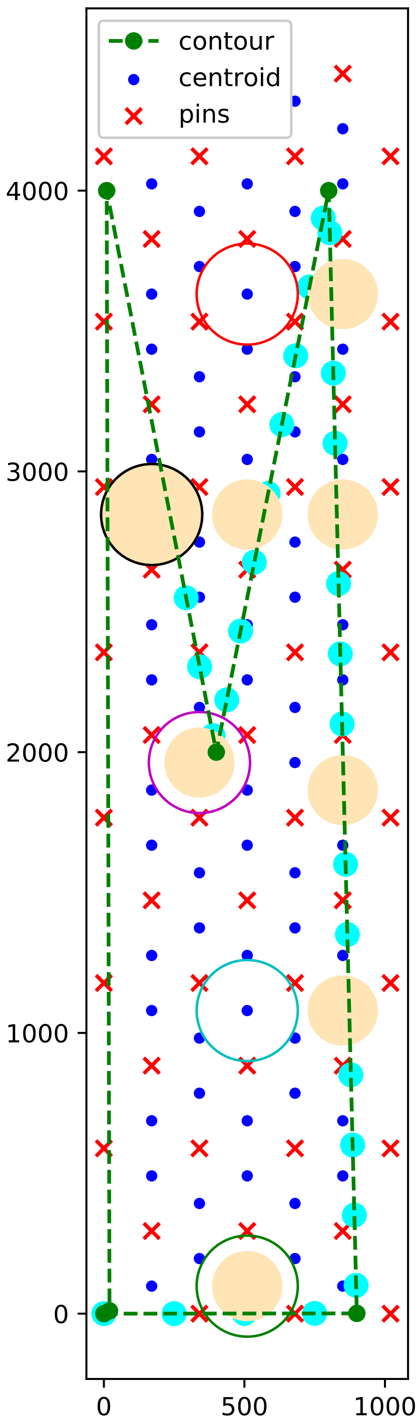
(a) t = 24


contour
centroid
pins
4000
3000
2000
1000
0
0
500
1000

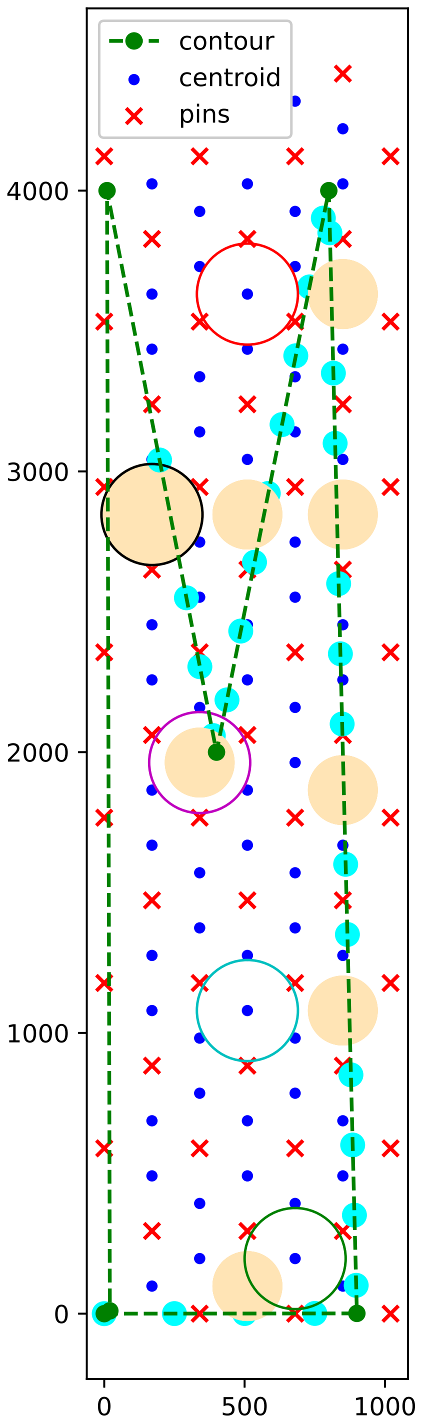
(b) t = 26


contour
centroid
pins
4000
3000
2000
1000
0
0
500
1000

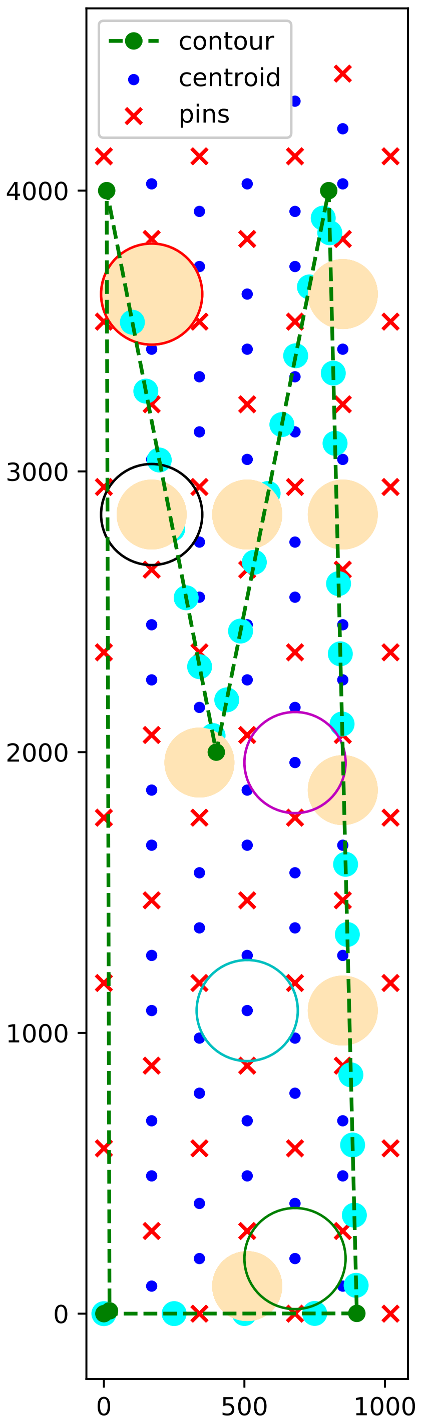
(a) t = 28


contour
centroid
pins
4000
3000
2000
1000
0
0
500
1000

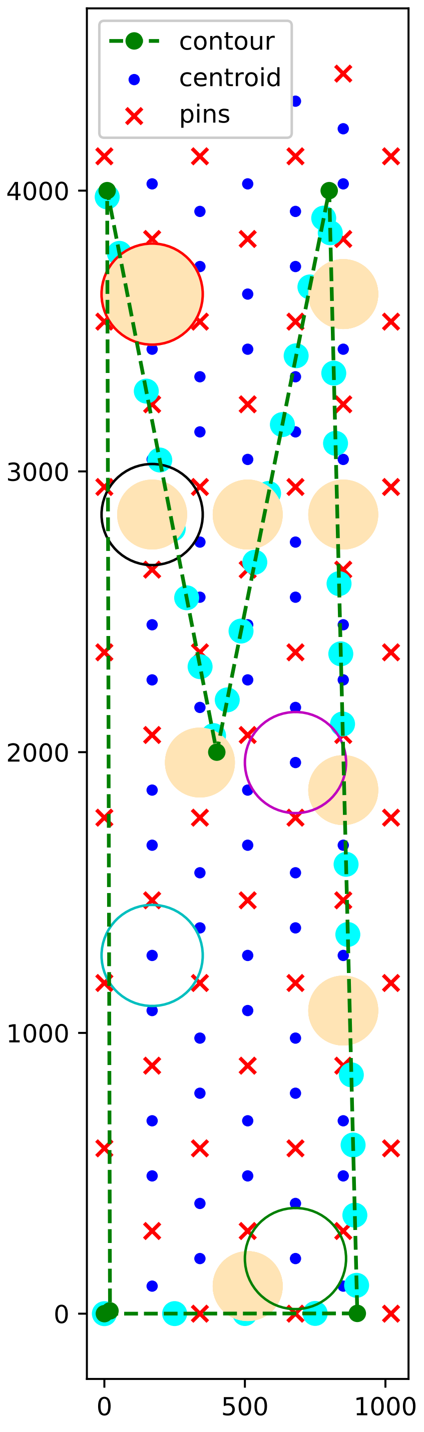
(b) t = 30


contour
centroid
pins
4000
3000
2000
1000
0
0
500
1000

(a) t = 32

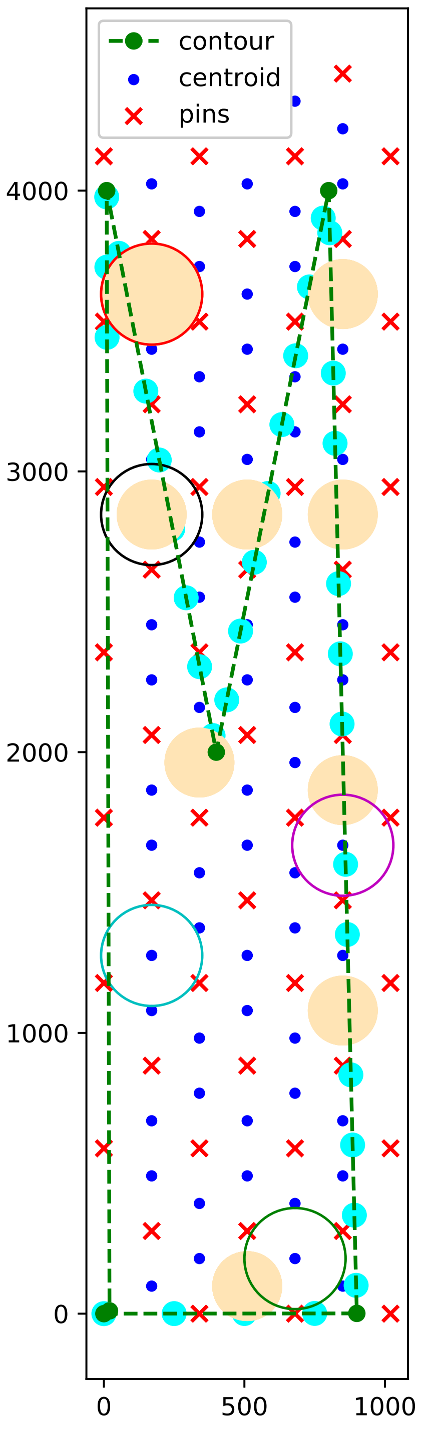

contour
centroid
pins
4000
3000
2000
1000
0
0
500
1000

(b) t = 34



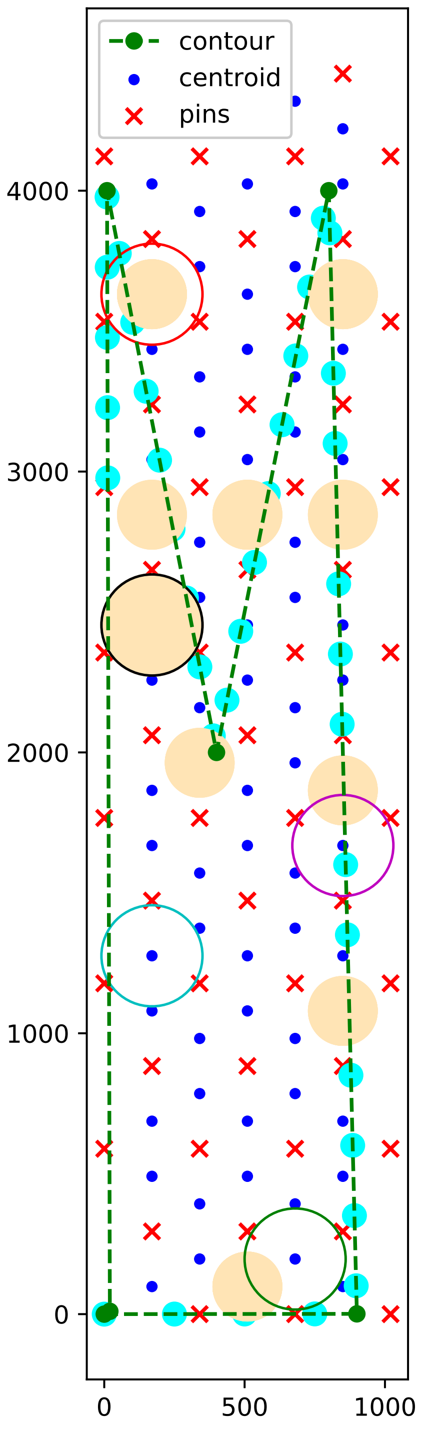
contour
centroid
pins
4000
3000
2000
1000
0
0
500
1000

(a) t = 36



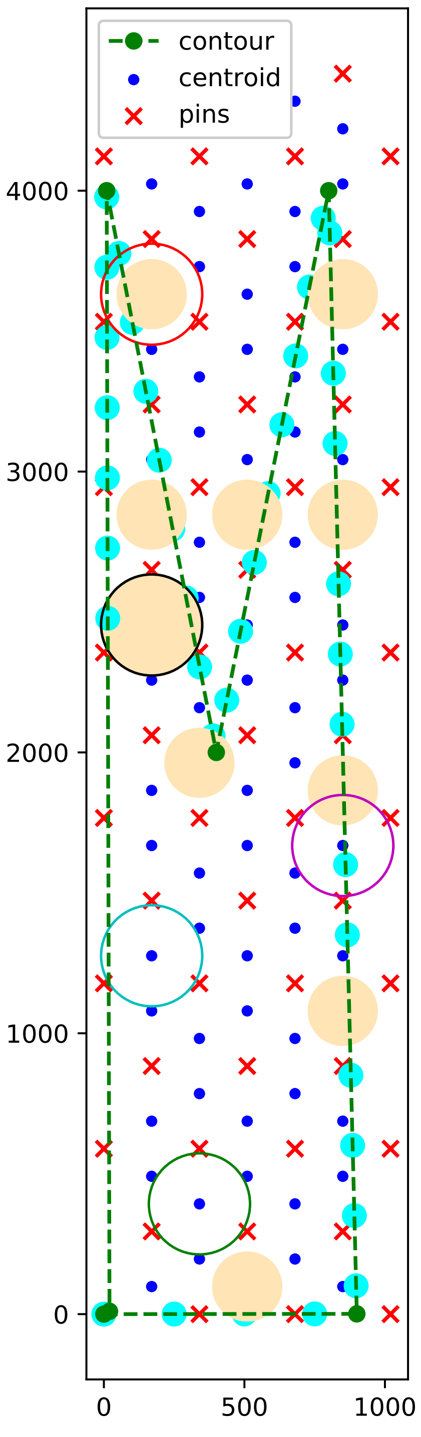
contour
centroid
pins
4000
3000
2000
1000
0
0
500
1000

(b) t = 38



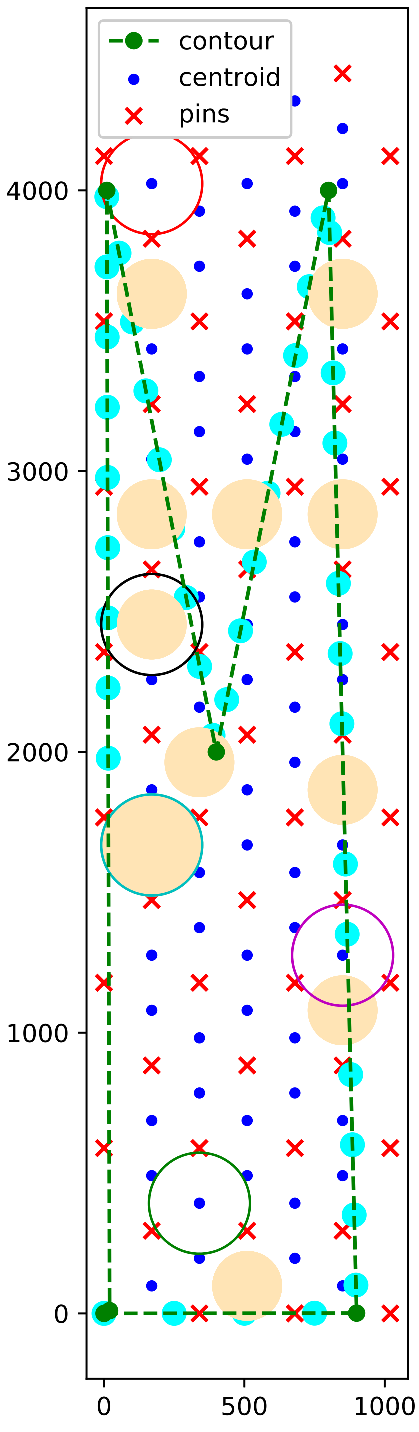
contour
centroid
pins
4000
3000
2000
1000
0
0
500
1000

(a) t = 40

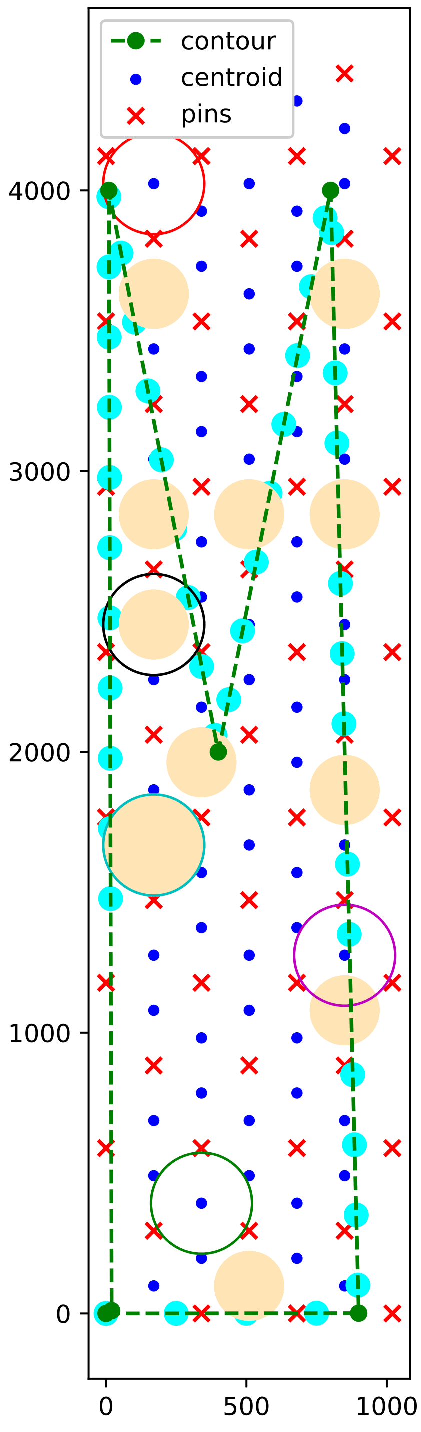


contour
centroid
pins
4000
3000
2000
1000
0
0
500
1000

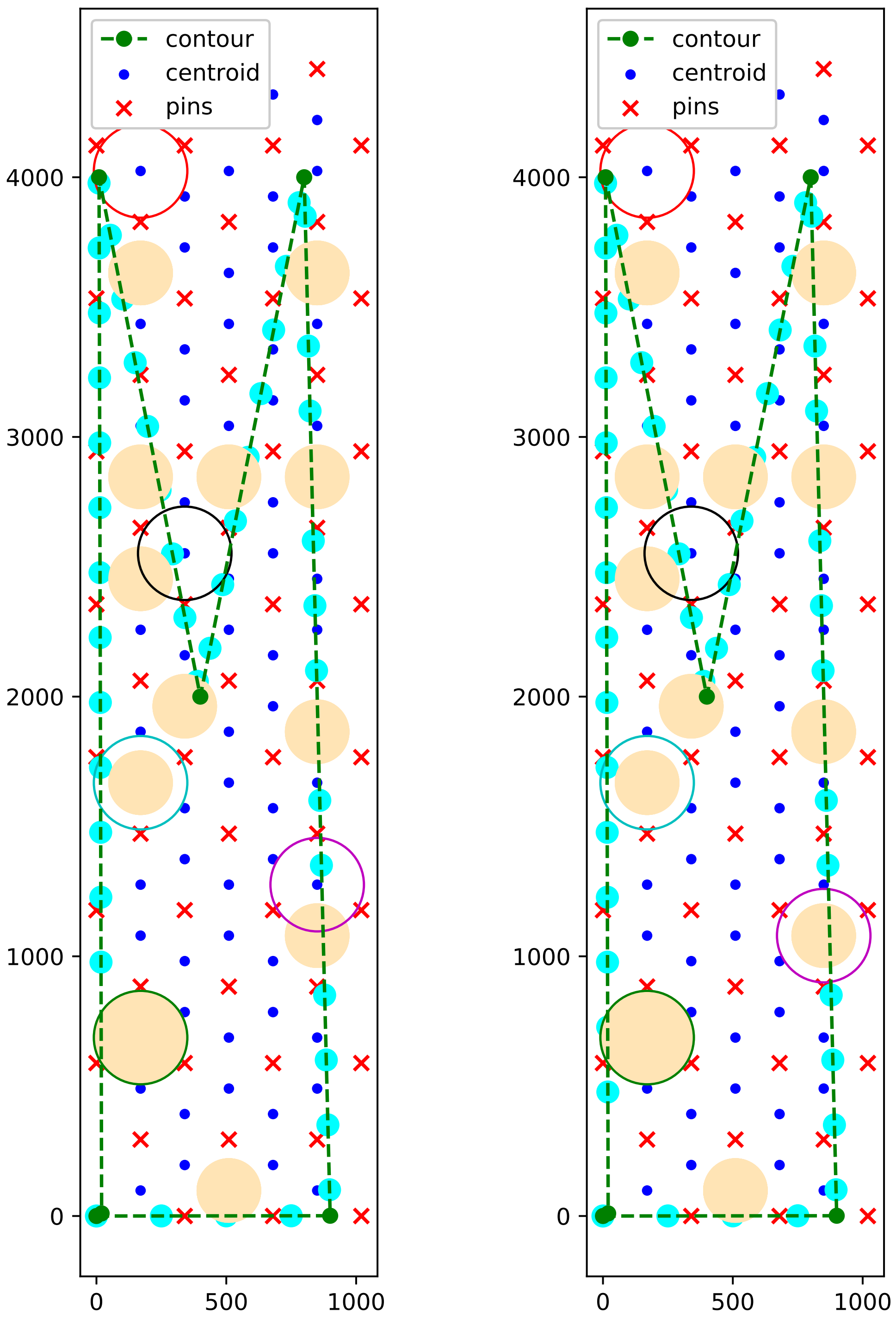
(b) t = 42


contour
centroid
pins
4000
3000
2000
1000
0
0
500
1000

(a) t = 44


contour
centroid
pins
4000
3000
2000
1000
0
0
500
1000

(b) t = 46



(a) t = 48

(b) t = 50

Figure 7.15: Motion planning of SaD agents along the contour

the obtained solution and the optimal solution. The motion planning formulations resulted in an asynchronous planning for the agents diverging from the previous chapters, where each time step had to be uniform, hence we can take into account the time required for each agent to move.

Chapter 8

Conclusions

8.1 Conclusions

We have observed that SaD locomotive agents architecture are ideal candidates for multi-robot systems. We have identified few of the potential applications for the SaD system such as: Material Handling, and fixturing agents and provided detailed analysis justifying the suitability. These specific applications required dedicated planner for generating feasible paths, hence we developed ILP techniques based mathematical models to achieve the multi-agent path planning for the SaD agents. The main focus during the MPP were to achieve two major objectives: minimize the makespan and maximize the number of idle states for each agent. Simulations were performed to test the ILP models for these specific objectives. A state of art ILP solver "GUROBI" was used to perform all the simulations. Experimental results suggested near optimal solution with the implemented models. We further extended the ILP model to address a more specific constraint of orientation for the agents, where we observed makepsan solutions close to $1.x$ of the optimal solution. These type of MPP problem are very unique to the multi-robot system community. The results and methodology developed in this work can provide good insight into similar problems in MPP community, where most traditionally only point objects are considered as agents. Apart from the path planning we also address another specific motion planning problem, where SaD agents act as fixture agents. The ILP techniques are implemented as constrained optimization for addressing this problem. Simulation results prove the effectiveness of the solution. The developed methodology is a holistic method which integrates both the motion planning and the selection of design parameters for the fixturing agents. All the mathematical models developed in this work are general and can be easily extended to address similar single-goal or MPP problems.

8.2 Future Research

There exists lots of research possibilities with SaD agent system. One of the central focus should lie upon moving towards a distributed architecture, where each agent takes decision based on its surrounding environment and task objectives. The ILP formulations presented in this thesis can be modeled with specific heuristics to achieve very fast computation.

Appendix A

Specification of SaD agents

A.1 Hardware Specification

Table A.1: SaD Hardware Specification

Specification	SaD Agent
Dimensions of SaD agent	Cylinder (⌀550 mm, height: 500 mm)
Harmonic Drive Rated motor speed Motor torque constant Motor stall torque	 3500 RPM 0.57 Nm 1.8 Nm
Controllers	Maxon EPOS2
Locking pin (required) Holding force (available) Draw - in force (available)	Locking force $>$ 10 kN 75 kN 18 kN
Interface	Multi-Robot research oriented controller (MROCC++)

References

[1] L. de Leonardo, D. Zlatanov, M. Zoppi, and R. Molfino, "Design of the locomotion and docking system of the swarmitfix mobile fixture agent," *Procedia Engineering*, vol. 64, pp. 1416–1425, 2013. vii, 1, 10

[2] L. de Leonardo, M. Zoppi, L. Xiong, D. Zlatanov, and R. M. Molfino, "Swarmitfix: a multi-robot-based reconfigurable fixture," *Industrial Robot: An International Journal*, vol. 40, no. 4, pp. 320–328, 2013. vii, 13, 84

[3] K. Sagar, D. Zlatanov, M. Zoppi, C. Nattero, and M. Sreekumar, "Path planning of a material handling agent with novel locomotion," in *ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference.* American Society of Mechanical Engineers, 2016, pp. V004T05A007–V004T05A007. vii, viii, 1, 9, 14, 17, 26, 45

[4] G. Bright and A. Walker, "Standardised framework for flexible materials handling management based on operating system primatives," in *Proceedings of the 2007 Australasian conference on Robotics & Automation*, 2007. ix, 19

[5] R. Molfino, M. Zoppi, and D. Zlatanov, "Reconfigurable swarm fixtures," in *Reconfigurable Mechanisms and Robots, 2009. ReMAR 2009. ASME/IFToMM International Conference on.* IEEE, 2009, pp. 730–735. 1, 84

[6] R. M. Molfino and M. Zoppi, "The robotic swarm concept in fixtures for transport industry," in *Proceedings of the ASME 2010 international*

design engineering technical conferences and computers and information in engineering conference. Washington, DC, 2011, pp. 8–31. 1

[7] C. Liu, F. Tan, and X. Li, “Multi-point positioning tooling technology for aircraft manufacturing,” in *Materials Science Forum*, vol. 628. Trans Tech Publ, 2009, pp. 517–522. 1

[8] D. F. Walczyk and R. S. Longtin, “Fixturing of compliant parts using a matrix of reconfigurable pins,” *TRANSACTIONS-AMERICAN SOCIETY OF MECHANICAL ENGINEERS JOURNAL OF MANUFACTURING SCIENCE AND ENGINEERING*, vol. 122, no. 4, pp. 766–772, 2000. 1

[9] M. Zoppi, R. Molfino, and D. Zlatanov, “Bench and method for the support and manufacturing of parts with complex geometry,” Jul. 30 2013, uS Patent 8,495,811. 1

[10] P. U. Lima and L. M. Custodio, “Multi-robot systems,” in *Innovations in robot mobility and control*. Springer, 2005, pp. 1–64. 3

[11] T. Arai, E. Pagello, and L. E. Parker, “Advances in multi-robot systems,” *IEEE Transactions on robotics and automation*, vol. 18, no. 5, pp. 655–661, 2002. 3

[12] G. Beni and J. Wang, “Swarm intelligence in cellular robotic systems,” in *Robots and Biological Systems: Towards a New Bionics ?* Springer, 1993, pp. 703–712. 3

[13] G. Dudek, M. R. Jenkin, E. Milios, and D. Wilkes, “A taxonomy for multi-agent robotics,” *Autonomous Robots*, vol. 3, no. 4, pp. 375–397, 1996. 3, 4

[14] A. Farinelli, L. Iocchi, and D. Nardi, “Multirobot systems: a classification focused on coordination,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 34, no. 5, pp. 2015–2028, 2004. 4

[15] Y. U. Cao, A. S. Fukunaga, A. B. Kahng, and F. Meng, “Cooperative mobile robotics: Antecedents and directions,” in *Intelligent Robots and Systems*

95.'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on, vol. 1. IEEE, 1995, pp. 226–234. 4

[16] L. Iocchi, D. Nardi, and M. Salerno, "Reactivity and deliberation: a survey on multi-robot systems," in *Workshop on Balancing Reactivity and Social Deliberation in Multi-Agent Systems.* Springer, 2000, pp. 9–32. 4

[17] C. Zieliñski and T. Winiarski, "Motion generation in the mrroc++ robot programming framework," *The International Journal of Robotics Research*, vol. 29, no. 4, pp. 386–413, 2010. 4

[18] C. Zieliński, W. Kasprzak, T. Kornuta, W. Szynkiewicz, P. Trojanek, M. Walęcki, T. Winiarski, and T. Zielińska, "Control and programming of a multi-robot-based reconfigurable fixture," *Industrial Robot: An International Journal*, vol. 40, no. 4, pp. 329–336, 2013. 4

[19] J. Ota, "Multi-agent robot systems as distributed autonomous systems," *Advanced engineering informatics*, vol. 20, no. 1, pp. 59–70, 2006. 5

[20] N. J. Nilsson, "A mobile automaton: An application of artificial intelligence techniques," SRI INTERNATIONAL MENLO PARK CA ARTIFICIAL INTELLIGENCE CENTER, Tech. Rep., 1969. 5

[21] K. Azarm and G. Schmidt, "A decentralized approach for the conflict-free motion of multiple mobile robots," *Advanced robotics*, vol. 11, no. 4, pp. 323–340, 1996. 5

[22] P. Raja and S. Pugazhenthi, "Optimal path planning of mobile robots: A review," *International Journal of Physical Sciences*, vol. 7, no. 9, pp. 1314–1320, 2012. 5

[23] P. Surynek, "An optimization variant of multi-robot path planning is intractable." in *AAAI*, 2010. 5, 43

[24] W. Wang and W. B. Goh, "An iterative approach for makespan-minimized multi-agent path planning in discrete space," *Autonomous Agents and Multi-Agent Systems*, vol. 29, no. 3, pp. 335–363, 2015. 5

[25] M. Bennewitz, W. Burgard, and S. Thrun, "Optimizing schedules for prioritized path planning of multi-robot systems," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 1. IEEE, 2001, pp. 271–276. 5

[26] P. Vadakkepat, K. C. Tan, and W. Ming-Liang, "Evolutionary artificial potential fields and their application in real time robot path planning," in *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, vol. 1. IEEE, 2000, pp. 256–263. 5, 42

[27] A. Vergnano, C. Thorstensson, B. Lennartson, P. Falkman, M. Pellicciari, F. Leali, and S. Biller, "Modeling and optimization of energy consumption in cooperative multi-robot systems," *IEEE Transactions on Automation Science and Engineering*, vol. 9, no. 2, pp. 423–428, 2012. 5

[28] M. B. Dias, R. Zlot, N. Kalra, and A. Stentz, "Market-based multirobot coordination: A survey and analysis," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1257–1270, 2006. 5

[29] H. Yang, S. Krut, C. Baradat, and F. Pierrot, "Locomotion approach of remora: A reonfigurable mobile robot for manufacturing applications," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE, 2011, pp. 5067–5072. 6, 16

[30] P. Leitão, "Agent-based distributed manufacturing control: A state-of-the-art survey," *Engineering Applications of Artificial Intelligence*, vol. 22, no. 7, pp. 979–991, 2009. 15

[31] A. Kusiak and S. S. Heragu, "The facility layout problem," *European Journal of operational research*, vol. 29, no. 3, pp. 229–251, 1987. 15

[32] M. G. Kay, "Material handling equipment," *Fitts Dept. of Industrial and Systems Engineering North Carolina State University*, vol. 65, 2012. 16

[33] J. E. Luntz and W. Messner, "Work on a highly distributed coordination control system," in *American Control Conference, Proceedings of the 1995*, vol. 4. IEEE, 1995, pp. 2838–2842. 16

[34] T. Sugar and V. Kumar, “Control and coordination of multiple mobile robots in manipulation and material handling tasks,” *Experimental Robotics VI*, pp. 15–24, 2000. 16

[35] G. Bright and A. Walker, “A mobile mechatronic platform architecture for flexible materials handling,” in *Proceedings of the Australasian Conference on Robotics and Automation, Brisbane, Australia*, vol. 10, 2007. 16

[36] A. Rovetta, “A robotic mobile platform for application in automotive production environment,” in *Advances in Mechanisms, Robotics and Design Education and Research.* Springer, 2013, pp. 239–244. 16

[37] G. Immega and K. Antonelli, “The ksi tentacle manipulator,” in *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on*, vol. 3. IEEE, 1995, pp. 3149–3154. 16

[38] H. Hemami, E. Tarr, B. Li, A. Krishnamurthy, B. Clymer, and B. Dariush, “Towards a cybernetic model of human movement,” *Mechanical Engineering Research*, vol. 6, no. 1, p. 29, 2016. 16

[39] K. E. Stecke, “Variations in flexible manufacturing systems according to the relevant types of automated materials handling,” 1984. 17

[40] R. YAMAN, “A knowledge-based approach for selection of material handling equipment and material handling system pre-design,” *Turkish Journal of Engineering and Environmental Sciences*, vol. 25, no. 4, pp. 267–278, 2001. 19

[41] K. Gupta and N. K. Jain, “On micro-geometry of miniature gears manufactured by wire electrical discharge machining,” *Materials and Manufacturing Processes*, vol. 28, no. 10, pp. 1153–1159, 2013. 20

[42] R. Slatter and R. Degen, “Miniature zero-backlash gears and actuators for precision positioning applications,” in *Proc. of 11th European Space Mechanisms and Tribology Symposium ESMATS*, 2005, pp. 9–16. 20

[43] P. Ouyang, R. Tjiptoprodjo, W. Zhang, and G. Yang, "Micro-motion devices technology: The state of arts review," *The International Journal of Advanced Manufacturing Technology*, vol. 38, no. 5, pp. 463–478, 2008. 20

[44] Y. Okazaki, N. Mishima, and K. Ashida, "Microfactory—concept, history, and developments," *Journal of manufacturing science and engineering*, vol. 126, no. 4, pp. 837–844, 2004. 20

[45] A. J. Sanchez-Salmeron, R. Lopez-Tarazon, R. Guzman-Diana, and C. Ricolfe-Viala, "An inter-machine material handling system for micro-manufacturing based on using a standard carrier," *The International Journal of Advanced Manufacturing Technology*, vol. 47, no. 9, pp. 937–943, 2010. 20

[46] J. Brufau, M. Puig-Vidal, J. Lopez-Sanchez, J. Samitier, N. Snis, U. Simu, S. Johansson, W. Driesen, J.-M. Breguet, J. Gao *et al.*, "Micron: Small autonomous robot for cell manipulation applications," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on.* IEEE, 2005, pp. 844–849. 20

[47] T. Zielinska, W. Kasprzak, W. Szynkiewicz, and C. Zieliński, "Path planning for robotized mobile supports," *Mechanism and Machine Theory*, vol. 78, pp. 51–64, 2014. 22, 23

[48] W. Kasprzak, W. Szynkiewicz, D. Zlatanov, and T. Zielińska, "A hierarchical csp search for path planning of cooperating self-reconfigurable mobile fixtures," *Engineering Applications of Artificial Intelligence*, vol. 34, pp. 85–98, 2014. 22

[49] W. Kasprzak, D. Zlatanov, W. Szynkiewicz, and T. Zielińska, "Task planning for cooperating self-reconfigurable mobile fixtures," *The International Journal of Advanced Manufacturing Technology*, vol. 69, no. 9-12, pp. 2555–2568, 2013. 22, 86, 88

[50] M. E. Newman, "Scientific collaboration networks. ii. shortest paths, weighted networks, and centrality," *Physical review E*, vol. 64, no. 1, p. 016132, 2001. 24

[51] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, “Introduction to algorithms second edition,” 2001. 24

[52] D. B. Johnson, “Efficient algorithms for shortest paths in sparse networks,” *Journal of the ACM (JACM)*, vol. 24, no. 1, pp. 1–13, 1977. 25

[53] K. Hormann and A. Agathos, “The point in polygon problem for arbitrary polygons,” *Computational Geometry*, vol. 20, no. 3, pp. 131–144, 2001. 25, 27, 46, 70, 86

[54] J.-C. Latombe, “Robot motion planning (the kluwer international series in engineering and computer science),” 1990. 42

[55] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, “Probabilistic roadmaps for path planning in high-dimensional configuration spaces,” *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996. 42

[56] S. M. LaValle, *Planning algorithms.* Cambridge university press, 2006. 42

[57] T. Siméon, S. Leroy, and J.-P. Lauumond, “Path coordination for multiple mobile robots: A resolution-complete algorithm,” *IEEE Transactions on Robotics and Automation*, vol. 18, no. 1, pp. 42–49, 2002. 42

[58] J. Peng and S. Akella, “Coordinating multiple robots with kinodynamic constraints along specified paths,” *The International Journal of Robotics Research*, vol. 24, no. 4, pp. 295–310, 2005. 42

[59] S. M. LaValle, “Rapidly-exploring random trees: A new tool for path planning,” 1998. 42

[60] H. M. Choset, *Principles of robot motion: theory, algorithms, and implementation.* MIT press, 2005. 42

[61] J.-C. Latombe, *Robot motion planning.* Springer Science & Business Media, 2012, vol. 124. 42

[62] J. Yu and S. M. LaValle, “Optimal multi-robot path planning on graphs: Complete algorithms and effective heuristics,” *arXiv preprint arXiv:1507.03290*, 2015. 43, 59

[63] K. Kant and S. W. Zucker, “Toward efficient trajectory planning: The path-velocity decomposition,” *The International Journal of Robotics Research*, vol. 5, no. 3, pp. 72–89, 1986. 43

[64] T. Schouwenaars, B. De Moor, E. Feron, and J. How, “Mixed integer programming for multi-vehicle path planning,” in *European Control Conference*, 2001, pp. 2603–2608. 43

[65] A. Richards, T. Schouwenaars, J. P. How, and E. Feron, “Spacecraft trajectory planning with avoidance constraints using mixed-integer linear programming,” *Journal of Guidance, Control, and Dynamics*, vol. 25, no. 4, pp. 755–764, 2002. 43

[66] G. Wagner, “Subdimensional expansion: A framework for computationally tractable multirobot path planning,” 2015. 43

[67] T. Standley and R. Korf, “Complete algorithms for cooperative pathfinding problems,” in *IJCAI.* Citeseer, 2011, pp. 668–673. 43

[68] D. Silver, “Cooperative pathfinding.” *AIIDE*, vol. 1, pp. 117–122, 2005. 43, 67

[69] P. Surynek, “An application of pebble motion on graphs to abstract multi-robot path planning,” in *2009 21st IEEE International Conference on Tools with Artificial Intelligence.* IEEE, 2009, pp. 151–158. 43, 67

[70] M. R. K. Ryan, “Exploiting subgraph structure in multi-robot path planning,” *Journal of Artificial Intelligence Research*, vol. 31, pp. 497–542, 2008. 43, 67

[71] J. Yu and S. M. LaValle, “Fast, near-optimal computation for multi-robot path planning on graphs,” *Intelligence (www. aaai. org)*, vol. 12, pp. 1–0193, 2013. 43, 67, 68

[72] S. Akella and S. Hutchinson, “Coordinating the motions of multiple robots with specified trajectories,” in *IEEE ICRA*, vol. 1, 2002, pp. 624–631. 43

[73] S. Bhattacharya, M. Likhachev, and V. Kumar, “Multi-agent path planning with multiple tasks and distance constraints.” in *ICRA*, 2010, pp. 953–959. 44

[74] P.-a. Gao and Z.-x. Cai, “Multi-robot task allocation for exploration,” *Journal of Central South University of Technology*, vol. 13, no. 5, pp. 548–551, 2006. 44, 67

[75] J. Yu and S. M. LaValle, “Planning optimal paths for multiple robots on graphs,” in *IEEE ICRA*, 2013, pp. 3612–3617. 44, 46, 47, 48, 49, 67, 71, 72

[76] J. E. Hopcroft, J. T. Schwartz, and M. Sharir, “On the complexity of motion planning for multiple independent objects; pspace-hardness of the” warehouseman’s problem”,” *The International Journal of Robotics Research*, vol. 3, no. 4, pp. 76–88, 1984. 44

[77] R. Wein, “Exact and approximate construction of offset polygons,” *Computer-Aided Design*, vol. 39, no. 6, pp. 518–527, 2007. 46

[78] J. E. Aronson, “A survey of dynamic network flows,” *Annals of Operations Research*, vol. 20, no. 1, pp. 1–66, 1989. 46, 71

[79] R. W. Floyd, “Algorithm 97: shortest path,” *Communications of the ACM*, vol. 5, no. 6, p. 345, 1962. 47, 48

[80] K. Sagar, D. Zlatanov, M. Zoppi, C. Nattero, and S. Muthuswamy, “Multi-goal path planning for robotic agents with discrete-step locomotion,” in *ASME 2017 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers, 2017, pp. V05AT08A033–V05AT08A033. 71, 81

[81] J. Yu and S. M. LaValle, “Optimal multirobot path planning on graphs: Complete algorithms and effective heuristics,” *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1163–1177, 2016. 78

[82] M. Sela, O. Gaudry, E. Dombre, and B. Benhabib, “A reconfigurable modular fixturing system for thin-walled flexible objects,” *The International Journal of Advanced Manufacturing Technology*, vol. 13, no. 9, pp. 611–617, 1997. 84, 85

[83] A. Al-Habaibeh, N. Gindy, and R. M. Parkin, “Experimental design and investigation of a pin-type reconfigurable clamping system for manufacturing aerospace components,” *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, vol. 217, no. 12, pp. 1771–1777, 2003. 84

[84] M. S. Soderberg, R. A. Starr, L. R. Cook, and R. J. Thomas, “Flexible tooling apparatus,” Mar. 3 1998, uS Patent 5,722,646. 84

[85] P. P. Zebus, P. N. Packer, C. C. Haynie *et al.*, “Variable contour securing system,” May 9 1978, uS Patent 4,088,312. 84

[86] W. A. Douglas and T. Ozer, “Universal holding fixture,” Aug. 4 1987, uS Patent 4,684,113. 84

[87] L. E. Parker, “Multiple mobile robot systems,” in *Springer Handbook of Robotics.* Springer, 2008, pp. 921–941.

[88] S. Verret, “Current state of the art in multirobot systems,” *Defence Research and Development Canada-Suffield*, p. 3, 2005.

[89] R. Alami, S. Fleury, M. Herrb, F. Ingrand, and F. Robert, “Multi-robot cooperation in the martha project,” *IEEE Robotics & Automation Magazine*, vol. 5, no. 1, pp. 36–47, 1998.

[90] P. Švestka and M. H. Overmars, “Coordinated path planning for multiple robots,” *Robotics and autonomous systems*, vol. 23, no. 3, pp. 125–152, 1998.

[91] D. Herrero-Perez and H. Martinez-Barbera, “Modeling distributed transportation systems composed of flexible automated guided vehicles in flexible manufacturing systems,” *IEEE Transactions on Industrial Informatics*, vol. 6, no. 2, pp. 166–180, 2010.

[92] H. Roozbehani and R. D'Andrea, "Adaptive highways on a grid," in *Robotics Research*. Springer, 2011, pp. 661–680.

[93] G. Sanchez and J.-C. Latombe, "Using a prm planner to compare centralized and decoupled planning for multi-robot systems," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 2. IEEE, 2002, pp. 2112–2119.

[94] T. Schouwenaars, B. De Moor, E. Feron, and J. How, "Mixed integer programming for multi-vehicle path planning," in *European Control Conference (ECC)*, 2001, pp. 2603–2608.

[95] N. Wu and M. Zhou, "Modeling and deadlock avoidance of automated manufacturing systems with multiple automated guided vehicles," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 35, no. 6, pp. 1193–1202, 2005.

[96] ——, "Modeling and deadlock control of automated guided vehicle systems," *IEEE/ASME transactions on mechatronics*, vol. 9, no. 1, pp. 50–57, 2004.

[97] J. Yu and S. M. LaValle, "Multi-agent path planning and network flow," in *Algorithmic Foundations of Robotics X*. Springer, 2013, pp. 157–173.

[98] T. Fukuda, Y. Hasegawa, K. Sekiyama, and T. Aoyama, *Multi-Locomotion Robotic Systems: New Concepts of Bio-inspired Robotics.* Springer, 2012, vol. 81.

[99] T. Fukuda and S. Nakagawa, "Dynamically reconfigurable robotic system," in *Robotics and Automation, 1988. Proceedings., 1988 IEEE International Conference on*. IEEE, 1988, pp. 1581–1586.

[100] G. Beni, "The concept of cellular robotic system," in *Intelligent Control, 1988. Proceedings., IEEE International Symposium on*. IEEE, 1988, pp. 57–62.

[101] T. Fukuda, T. Ueyama, Y. Kawauchi, and F. Arai, "Concept of cellular robotic system (cebot) and basic strategies for its realization," *Computers & electrical engineering*, vol. 18, no. 1, pp. 11–39, 1992.

[102] H. Asada and A. By, “Kinematic analysis of workpart fixturing for flexible assembly with automatically reconfigurable fixtures,” *IEEE Journal on Robotics and Automation*, vol. 1, no. 2, pp. 86–94, 1985.

[103] W. Lowell, “Geo-metrics ii: Dimensioning and tolerancing,” *ANSI/ASME Standard Y*, vol. 13, pp. 35–52, 1982.

[104] W. Cai, S. J. Hu, and J. Yuan, “Deformable sheet metal fixturing: principles, algorithms, and simulations,” *Journal of manufacturing science and engineering*, vol. 118, no. 3, pp. 318–324, 1996.

[105] C. Zieliński, T. Kornuta, P. Trojanek, T. Winiarski, and M. Walęcki, “Specification of a multi-agent robot-based reconfigurable fixture control system,” *Robot Motion and Control 2011*, pp. 171–182, 2012.

[106] C. Sherwood and J. Abbott, “Pogo™ flexible tooling universal holding fixture for cutting, drilling and assembly in the aerospace industry,” SAE Technical Paper, Tech. Rep., 1996.