

The Application of Optimization Algorithms for Path Planning of Automated Guided Vehicles



S. Ward, J. McCooey, L. Osborne
*M.Sc Intelligent Systems and Robotics
De Montfort University*

Abstract — Automated Guided Vehicles (AGV's) have been a staple of the manufacturing and logistics industries now for the past two decades. In simple terms, the function of an AGV is to automatically transport material from a start position to a target position. They are used in a wide variety of applications and offer a cost-effective, automated method for the management and transportation of goods. In recent times, developments in the disciplines of Computational Intelligence and Mobile Robotics have led to new and improved control techniques for the deployment of AGV's. Due to these techniques, several AGV functions are being continuously improved. Improvements in these functions result in maximizing the output of AGV's for industrial applications. In this paper, the Path Planning function of an AGV was investigated for manufacturing and logistics applications. The objective of the Path Planning function is to compute the optimal path from a start location to an end location. A literature review was conducted of AGV's in general along with various Computational Intelligence techniques for optimizing the Path Planning function of an AGV. Three optimization algorithms were deployed and tested to perform Path Planning within a simulation of an Industrial Manufacturing application. From the testing and analysis which was carried out, it was proven that the Ant System algorithm computed the shortest path in 60% of test scenarios and therefore was most effective. The RRT* algorithm computed the shortest path in 40% of test scenarios where a computational time limit of 300 seconds was applied but is capable of computing shorter paths with additional computational power. The GA algorithm did not plot the shortest path but did compute a result in the shortest time for most test scenarios.

Keywords—Automated Guided Vehicle (AGV), Computational Intelligence (CI), Genetic Algorithm (GA), Rapidly-Exploring Random Trees (RRT), Ant Colony Optimization (ACO), Ant System (AS)

I. INTRODUCTION

Increasingly AGV's are being used across a wide range of industry's beyond distribution and manufacturing including retail, the military and healthcare [1]. The advantages of using AGV's in the workplace include: reduced labour costs, increased safety over traditional forklift trucks, increased accuracy and modularity. Disadvantages include the high initial capital investment cost, maintenance costs and lack of the flexibility that is introduced into the work environment [1].

AGV's often integrate symbiotic hardware and software to tackle the variety of navigational and operational challenges that they have to address. AGV's use electronic hardware to aid navigation and avoid obstacles, common hardware choices are: laser sensors, vision systems and inductive sensors which are used to follow a magnetic track

[2]. This hardware is combined with algorithms that are used to calculate the path that the AGV will follow. AGV's will often be required to navigate between pick up/drop off locations as well as battery charging areas. Graphical user interfaces (GUI) will often be written which will run on a client PC to allow global control of fleets of AGV's allowing system issues to be monitored and controlled over a WLAN [2]. In one particular paper by T.Ferreira and I.A. Gorlach [3] a programmable logic controller was employed to interface between a motor drive, electronic sensors and an algorithm running on a central controller.

Currently, the majority of AGVs used in industry utilise fixed positioning techniques to navigate through environments such as warehouses. This presents some disadvantages to the end user, including a lack of path routing flexibility (i.e. paths need to be re-designed, re-installed and re-commissioned following any change to the manufacturing layout). In addition, there is usually a large cost and space burden associated with fixed positioning AGVs due to the large infrastructure required, which usually includes swim lane floor layouts or dispersed node points throughout the environment. This paper attempts to address these issues by utilising computational intelligence techniques. Computational methods are used to solve a number of real-world problems which can include the use of computer simulation. Fundamental techniques in this field include deterministic and stochastic search algorithms, which are measured by parameters such as convergence, error analysis and stability [4]. The choice of a particular method is application specific and may involve a degree of trial and error. This paper uses optimization techniques to assist with the path planning of AGV's. Optimization is useful to tackle the problem of path planning because the path needs to be as short as possible, and optimization relates to "determining the best solution to make something as functional and effective as possible by minimizing or maximizing the parameters involved in the problem" [5]. Computational intelligence techniques also open the door for future work into dynamic obstacle avoidance, which is seen as a key requirement in many modern manufacturing environments, although this is outside the scope of this study.

II. BACKGROUND

A. AGV Functionality

The principle of Automated Guided Vehicles has existed since the early 1950's. At that stage, AGV's possessed limited functionality, mainly the ability to follow fixed wired tracks while towing materials and goods. These applications were initially deployed in grocery warehouses and automotive assembly lines. Since then, the functionality of AGV's has

improved exponentially. In recent times, particularly in Europe due to Industry 4.0, AGV's have emerged as a disruptive technology. AGV's are now capable of performing functions such as Navigation, Motion Planning, Task Allocation, Fleet Management and Path Planning in real-time while simultaneously maintaining connection to various systems. As a result, the deployment of AGV's for applications within all industries has increased dramatically.

Navigation functionality for AGV's was once a single measurement of a radio signal from a wired track. Nowadays, navigation systems vary from LiDAR and Laser scanner measurements to GPS and odometry tracking systems to simultaneously map and localize (SLAM) the AGV. This data enables end users to view local and global maps of the AGV remotely and in real-time. Task Allocation is another core AGV function which involves the ordering of AGV journeys in an optimal way. This is an important function for the end user to maximize the utilization of the AGV for an industrial application. One straightforward method of allocating tasks which is commonly used is to prioritise based on which order is positionally nearest to the AGV at a given time. This method can be applied to an individual AGV or a Fleet. The management of a Fleet of vehicles can also be achieved and has become a modern day AGV core function. Fleet Management applies to larger scale logistics and manufacturing industries. Each vehicle within the Fleet can perform local actions but is managed by a master system which efficiently accommodates journey requests.

One of the key functions of AGV technology, and the function that most concerns this paper, is Path Planning. The objective of Path Planning is to find a suitable Path from the initial vehicle state (Position, Orientation) to a target state (Position, Orientation). The Path should allow for all obstacles within the scene for the AGV to successfully travel from its initial state to its target state. Traditionally, AGV's followed fixed paths and were unable to alter or optimize their Path. However, in recent times Path Planning has become a specialized discipline due to a wide range of control techniques and advances in computational intelligence. The function of Path Planning is beneficial to AGV applications for several reasons. Generating the optimal path for an AGV journey results in reduced travel time, reduced travel distance and reduced power consumption. Therefore, the Path Planning function provides an optimal solution for the end user to maximise the use of the AGV for their application.

B. AGV Control Techniques

As previously discussed, the latest generation of Automated Guided Vehicles can perform a wide range of functions, suitable for a variety of industrial applications. There are several Control Techniques that exist currently to execute the listed functions in a structured way. In general, Control Techniques for AGV's can be divided into the following two categories: 1. Centralized Systems, 2. Decentralized Systems.

A Centralized control system approach for an Automated Guided Vehicle defines a clear system hierarchy. The hierarchy consists of one high level controller, which in the case of an AGV is a single master controller/server. This master controller is solely responsible for the control and actions of several lower level components, which in this case can range from a single vehicle to a fleet. The Centralized control approach possesses advantages over other approaches when deployed on a small group of AGV's for various

straightforward "off-the-shelf" applications. The master controller of a Centralized system contains the global information of all lower-level components. As a result, this approach performs well in static environments and is easily implemented. However, these systems lack scalability due to all the AGV information and control processing occurring in one location. These systems also lack robustness and performance can degrade over time due to the master controller acting as a single point of failure. Centralized control techniques for AGV's are much more aligned with some of the earlier and traditional generations of AGV's.

A Decentralized control system approach for an Automated Guided Vehicle is where each system component (AGV) is controlled based on its own local information and is not under the control influence of any master controller. This distributed control approach allows for more complex behaviour to emerge. The local information of each AGV within the system, is directly responsible for the global behaviour of the system. This approach has several advantages. As each lower level component contains its own local information to make decisions, a Decentralized approach is much more flexible and suitable for more difficult applications. This approach also performs well in dynamic environments as the control is based off local information as opposed to global information. Implementing a Decentralized approach for an AGV application is more difficult, especially for a fleet however the performance is more robust and scalable due to the local element of control. The field of Computational Intelligence and its application to Decentralized control has become widely researched with a lot of research still on-going. Future generations of AGV control and task execution will come because of this research. Most of the research in this area is based around applying Computational Intelligence methods such as Optimization algorithms to perform local control of lower level AGV's.

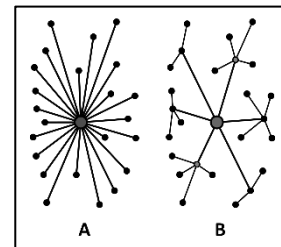


Fig. 1. A. Centralized vs. B. Decentralized Control Approach

C. Computational Intelligence

Computation Intelligence is widely used across the computer science, engineering, data-analysis and bio-medicine disciplines. The term itself is a generic one and is used to describe a broad set of component computation techniques, including fuzzy logic, neuro-computing and evolutionary computing [6]. The principal aim common to these methodologies is to replicate the actions and performance of a human being in a manner which is optimal.

The concept of an intelligent system began to crystalize in the 1980's as competing technologies such as AI, fuzzy logic, neuro-computing, evolutionary computing and probabilistic computing began to emerge [6]. Jim Bezdek used his influence in the mid-1990s to form a Computational Intelligence Society within the IEEE, which marked the official birth of computational intelligence as a discipline.

The five main areas of CI include fuzzy logic, neural networks, evolutionary computation, learning theory and probabilistic methods. This study primarily will focus on

utilising evolutionary computation techniques. Specifically the three techniques are: The Genetic Algorithm, the RRT* algorithm and a form of Ant Colony Optimisation, each of which are discussed below.

I. Genetic Algorithm (GA)

Genetic Algorithms (GAs) were first introduced in the 1960s by John Holland, and used metaheuristics inspired by the process of natural selection [7]. GAs encode a potential solution on a chromosome-like data structure and apply recombination operators to these structures in order to preserve critical information. Mutation operators are used to alter potential solutions, which are selected / disposed of depending on the resultant fitness of the mutation [8].

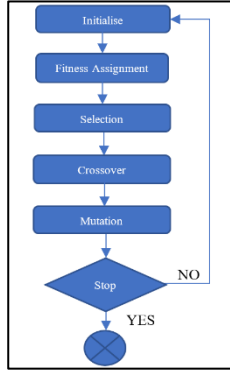


Fig. 2. Genetic Algorithm Process Flow

Error! Reference source not found. presents a flow diagram to depict the stages involved during the processing of a GA. GAs tend to be used for NP-Hard problems, where there are multiple optima, as well as for problems which have a large number of parameters and the input data is noisy. Due to their complexity, GA's are computationally expensive.

II. Rapidly Exploring Random Tree (RRT*)

Rapidly-Exploring Random Trees (RRTs) were first developed by Steven M. LaValle and James J. Kuffner Jr. in 1998 [4], primarily in response to the lack of randomised path planning techniques which were suitable to nonholonomic systems – systems whose states are dependent on the path taken in order to achieve it (i.e. non-integrable constraints). RRTs use random samples of the search space to grow the tree; connections are drawn between the sample and the nearest state, as long as the connection is feasible (i.e. no obstructions). The probability of expanding to an existing space is proportional to the size of its Voronoi Region [9]. This method of expansion means that the RRT is heavily biased towards unexplored portions of the state space [9].

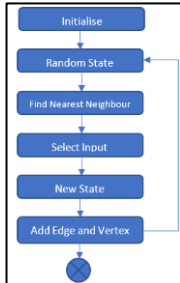


Fig. 3. RRT Algorithm Process Flow

It can be seen that the tree quickly expands to all four corners of the square. Advantages of RRT algorithms include the ability for quick exploration over an entire search space and its relative simplicity. Disadvantages include the non-deterministic nature of the algorithm, which can cause unexpected paths to be taken. Paths themselves can also be

jagged and difficult for a mobile robot such as an AGV to follow.

III. Ant Colony Optimisation (ACO)

Ant Colony Optimisation (ACO) was first proposed by Marco Dorigo in 1992 [10] and takes inspiration from the foraging behaviour of some ant species and their *stigmergic* method of communication (i.e. exchanges of information which is indirect and local) [11]. Whilst foraging, ants leave pheromones in places likely to be close to food sources. Increases in the number of pheromones deposited on a specific route will increase the likelihood of ants from the same colony using this same pathway in the future, thereby ensuring an efficient group collective effort. ACO algorithms, and more specifically, the Ant System (AS) algorithm (the original ACO algorithm developed by Dorigo), use similar terminology to their biological counterparts, where pheromone values are updated on each iteration by all m ants which have built a solution in that iteration. Pheromones τ_{ij} are associated with edges which connect nodes, and the quantity is calculated by:

$$\tau_{ij} \leftarrow (1 - \rho) \cdot \tau_{ij} + \sum_{k=1}^m \Delta \tau_{ij}^k$$

Where ρ is the evaporation rate, m is the number of ants, $\Delta \tau_{ij}$ is the quantity of pheromone laid on an edge i, j by ant k [6]. During the construction of a solution, the ants use a stochastic mechanism to decide which node is to be visited.

Similar to the previous algorithms which have been discussed, ACO algorithms are meta-heuristic and are suited to NP-Hard problems like the routing problem presented in this study. Advantages of ACOs include their inherent parallelism (can therefore be executed quickly on different cores), rapid discovery of solutions due to the inherent positive feedback and its potential to work in dynamic scenarios – it can adapt to new environment variables and update distances etc. Disadvantages include the uncertain time to convergence and the fact that the probability distribution changes on each iteration, as well as the possibility of incorrect convergence due to their stochastic nature.

III. PATH PLANNING FOR AGV'S

A. Fixed Positioning

As previously stated, the Path Planning function is essential for the implementation of a robust, industrial grade AGV application. The majority of AGV's which are currently deployed within industry, and particularly within the European region, do not include computational intelligence techniques for key AGV functions such as Path Planning. Only in the last five years, have AGV and Mobile Robot manufacturers begun to bring industrial grade AGV's with computational intelligence to market. As a result, the majority of existing AGV's present within the manufacturing and logistics industries possess the following Path Planning functionality centred about the concept of Fixed Positioning.

The term Fixed Positioning refers to any AGV where the Path Planning function is based on a fixed map containing a collection of previously determined points or paths. From the reviewed literature, the majority of existing AGV's contain this method of Path Planning. There are several disadvantages to this Fixed Position Path Planning technique. When the Fixed Position method is used, the nature of the application is fixed and lacks the ability to recalculate paths to account for environment changes.

1 – *Flexibility*: During initial setup, the integrator will move the AGV to all required locations within the environment. The reason for this is to essentially teach the collection of points or paths which are required to execute the required application tasks.

2 – *Infrastructure*: To successfully implement an AGV solution utilizing the Fixed Position method, the infrastructure within the applications environment needs to be adjusted. Specified lanes must be created, separate to where people are present. Without this, if an AGV encounters a human or other object while executing a task, its onboard sensors will detect this and come to a halt.

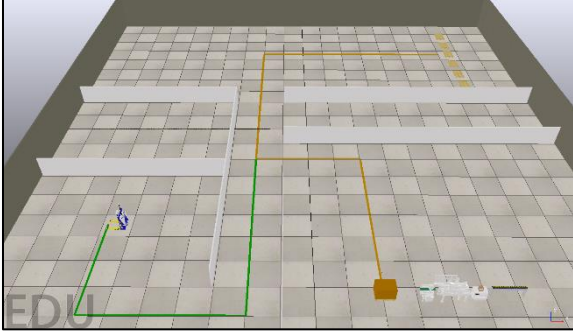


Fig. 4. Fixed Paths Implemented in VREP Environment

B. Obstacle Avoidance

When discussing the field of Path Planning related to AGV's, it is just as important to consider the function of Obstacle Avoidance. Obstacle Avoidance refers to an AGV's ability to encounter objects within its environment and perform specified actions as a result. In the current market for AGV's, there are multiple different approaches to Obstacle Avoidance. AGV's designed prior to the last five to ten years have limited capability when it comes to Obstacle Avoidance. In most cases, these earlier model AGV's come to a strict halt once an object or person is detected within a specific radius by its boundary sensors.

More recently, industrial AGV manufacturers are now incorporating more intelligence with functions such as Obstacle Avoidance. This function allows for environments where the surroundings change often. Obstacle Avoidance refers to an AGV's ability to recalculate its Path during travel once an object is detected within a specified boundary. These systems are highly sought after by end users and form part of the emerging field of flexible manufacturing systems. A level of computational intelligence is required to execute Obstacle Avoidance and it is a relatively straightforward function to implement within an already functioning Path Planning system.

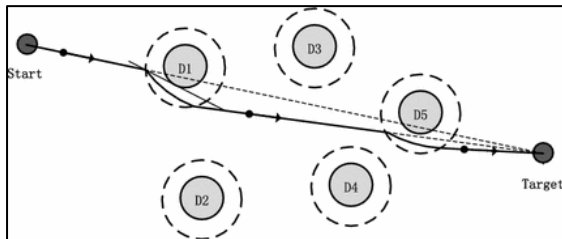


Fig. 5. Path Planning with Obstacle Avoidance

IV. AGV'S AND OPTIMIZATION

A. Genetic Algorithm

Genetic Algorithms are a form of evolutionary algorithms which were popularised in the mid 1970's by John Holland [19]. Historically, genetic programming was slow to take off as it required significant computing power which was not readily available in the 70's apart from the use of expensive super computers. This changed with the advent of powerful personal computers, and applications of these algorithms include circuit design, data sorting and searching, and quantum computing [20]. The pseudo code for Genetic Algorithm's vary in complexity but generally conform to the following structure:

```

Algorithm: GA( $n, \chi, \mu$ )
// Initialise generation 0:
 $k := 0$ ;
 $P_k :=$  a population of  $n$  randomly-generated individuals;
// Evaluate  $P_k$ :
Compute  $fitness(i)$  for each  $i \in P_k$ ;
do
{
  // Create generation  $k + 1$ :
  // 1. Copy:
  Select  $(1 - \chi) \times n$  members of  $P_k$  and insert into  $P_{k+1}$ ;
  // 2. Crossover:
  Select  $\chi \times n$  members of  $P_k$ ; pair them up; produce offspring; insert the offspring into  $P_{k+1}$ ;
  // 3. Mutate:
  Select  $\mu \times n$  members of  $P_{k+1}$ ; invert a randomly-selected bit in each;
  // Evaluate  $P_{k+1}$ :
  Compute  $fitness(i)$  for each  $i \in P_{k+1}$ ;
  // Increment:
   $k := k + 1$ ;
}
while fitness of fittest individual in  $P_k$  is not high enough;
return the fittest individual from  $P_k$ ;

```

Fig. 6. GA Algorithm – Pseudo code

The main sections of a Genetic Algorithm are the following:

1 – *Computing Fitness*: Calculating the fitness of each individual is essential for any GA as it is a prerequisite for crossover or mutation. This is because it allows a 'survival of the fittest' approach where the weakest individuals can be easily eliminated and the strongest put forward for crossover and mutation procedures. Computing fitness is achieved by using a fitness function, f , which represents a chromosome as a scalar value [22].

$$f: \Gamma^{Nx} \rightarrow \mathbb{R}$$

Where Γ = data type of elements of Nx -dimensional chromosome.

2 – *Crossover*: Crossover is where an offspring is produced from one or more parents, with a combination of their genes. The main categories of crossover operations are:

- Asexual – Offspring is generated from one parent.
- Sexual – two parents produce up to two offspring.
- Multi-recombination – more than two parents are used [22].

An example of parametrized uniform crossover is shown below, this is a sexual type of crossover.

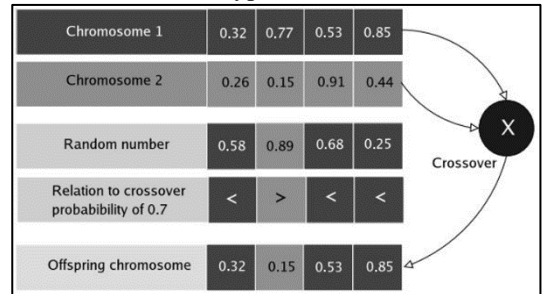


Fig. 7. Parametrized Uniform Crossover [23]

3 – *Mutation*: Mutation is sometimes carried out after crossover and ensures that fresh genetic material is added to an individual. This goes some way to ensuring that the full range of genes is available to an individual. A mutation probability p_m is used to determine if mutation will happen and is often ≤ 1 to ensure that good genetic material is not destroyed. [22] As with crossover, many mutation schemes exist, but for illustration purposes, two mutation schemes (random and inorder) are illustrated below:



Fig. 8. Before Mutation



Fig. 9. After Random Mutation



Fig. 10. Before Mutation



Fig. 11. After Inorder Mutation

B. RRT* Algorithm

As previously discussed, RRT stands for Rapidly-Exploring Random Tree. The RRT algorithm was created by Steven M. LaVelle and James J. Kuffner Jr. in 1998 and was proposed as a new tool for path planning [14]. The RRT algorithm constructs a tree from random sampling within a search space. The tree is initialized with a start state x_{init} and iteratively expands generating a path towards a target state x_{goal} . For each iteration of the RRT algorithm, a random state x_{random} is sampled within the search space. If the random state x_{random} falls within an obstacle free zone, then a node $x_{nearest}$ is searched within the tree according to a predefined step size. The tree is then expanded by connecting the nodes x_{random} and $x_{nearest}$. If the node $x_{nearest}$ is not found during the search, then a new node x_{new} is generated by using a steering function which results in the expansion of the tree by connecting the node x_{new} to $x_{nearest}$. When a path is found within the tree connecting x_{init} to x_{goal} , the search continues until a predefined number of iterations or time period is executed. This results in a very useful algorithm for generating a path from an initial location in space to a goal location avoiding obstacles in between. However, the RRT algorithm will not be the optimal path. The expansion of nodes within the RRT algorithm is shown in Figure 3 below.

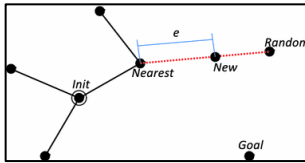


Fig. 12. Demonstration of RRT node expansion

The RRT* algorithm is an optimized version of the RRT algorithm which was popularized by Sertac Karaman and Emilio Frazzoli in 2011 [15]. As previously stated, the RRT algorithm generates a path, however this path is not necessarily the optimal path that could be found for the given journey. The RRT* algorithm contains all the properties of RRT, with two additional functions in order to optimize the found path. The first additional function is the nearest neighbour search, which finds the best parent node for the new node x_{new} before it is inserted into the tree. This function is performed within the area of a circle with a predefined radius and eliminates outliers which yield a less optimal path. The second additional function is the tree rewiring function, which rebuilds the tree within the above predefined radius to ensure that the tree with the minimum cost value is produced. The pseudo code for the RRT* algorithm is shown in Figure 4.

RRT* Algorithm	
1	$T = \text{InitializeTree}();$
2	$T = \text{InsertNode}(0, x_{init}, T);$
3	for $i = 1, \dots, n$ do
4	$x_{random} = \text{Sample}(i);$
5	$x_{nearest} = \text{Nearest}(T, x_{random});$
6	$x_{new} = \text{Steer}(x_{nearest}, x_{random});$
7	if $\text{ObstacleFree}(x_{nearest}, x_{new})$ then
8	$x_{near} = \text{Near}(T, x_{new}, V);$
9	$x_{min} = \text{ChooseParent}(x_{near}, x_{nearest}, x_{new});$
10	$T = \text{InsertNode}(x_{min}, x_{new}, T);$
11	$T = \text{Rewire}(T, x_{near}, x_{min}, x_{new});$
12	return T

Fig. 13. RRT* Algorithm – Pseudo code

The lower the cost value of the found path within the tree, the more optimal the resulting path. This optimization equates to a shorter path length which in turn can aid in the optimization of AGV use for applications.

C. ACO Algorithm

ACO algorithms are inspired by the biological processes which some species of ants use to optimise the process of foraging for food. These processes rely on ants' *stigmergic* method of communication (i.e. exchanges of information which is indirect and local) [11] on colony-wide basis, given that ants are mostly blind and therefore require guidance by other means. The underlying concept is based on a positive feedback mechanism which involves individual ants depositing pheromones during their journey to and from a successful destination. Consequently, the more pheromones left on a particular route, the more inclined subsequent ants would be to follow that route. As a result, routes which are shorter, and therefore quicker with an increased amount of traffic, would have a greater number of deposited pheromones. Whilst new routes are fortified, old routes are discouraged due to pheromone level decay. **Error! Reference source not found.** outlines the concept of a typical ant colony algorithm, which uses a discrete time period to assign probability (or number of pheromones) to individual edges. It can be seen that the ants initially diverge in equal numbers when they first approach an obstruction. Due to the shorter distance of path BCD in comparison to path BHD, the ants reach their target (E) in a shorter time. As a result, more pheromones τ are deposited on the shorter BCD path, causing more ants to take this route on the return

journey. This cycle repeats itself, and after several repeat journeys, the entire ant population will select the shortest route.

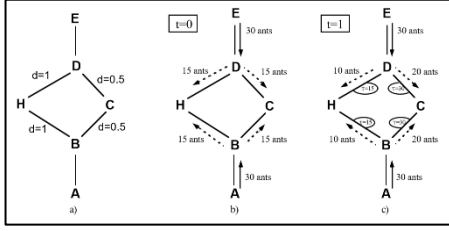


Fig. 14. The Ant System Mechanism [16]

The specific ACO algorithm-inspired algorithm which is being trialled for this study is the Ant System (AS) algorithm. This is an adapted ACO algorithm, which incorporates ‘ant’ memory, discrete time bands and some level of environmental sensing (as opposed to tradition ant colony algorithms which assumes the ‘ants’ are completely blind).

```

AS Algorithm
1 Initialise();
2 for k = 1, ... m do
3   Si ← ProbStepwiseConst[k];
4   Sicost ← Cost(Si);
5   if (Sicost ≤ Pbestcost) do
6     Pbestcost ← Sicost;
7     Pbest ← Si;
8   EndIf
9   Candidates ← Si;
10 End
11 DecayPheromone(Pheromone, ρ);
12 for Si ∈ Candidates
13   UpdatePheromone(Pheromone, Si, Sicost);
14 End
15 Return(Pbest)

```

Fig. 15. Ant System Algorithm – Pseudo code

Figure 2 above provides the pseudocode listing of the AS algorithm. Following initialisation of the algorithm variables, a probability stepwise calculation is performed on each ‘ant’ in the colony. For each edge taken by an individual ant, the probability for selection is calculated using historical information (pheromone value for that edge) and problem-specific heuristic information. The probability is given by:

$$P_{i,j} \leftarrow \frac{\tau_{i,j}^\alpha \times \eta_{i,j}^\beta}{\sum_{k=1}^c \tau_{i,k}^\alpha \times \eta_{i,k}^\beta}$$

where $\eta_{i,j}$ is the desirability of the edge i,j (typically $1/d_{i,j}$), β is the heuristic coefficient, $\tau_{i,j}$ is the pheromone value for the edge i,j , α is the history coefficient, and c is the set of usable components. The fitness of the solution is assessed and compared with existing best-fit solutions; if the cost function is smaller, the current solution is set as the best-fit route. Once the above steps have taken place for each ant in the colony, the potential candidate solutions are updated, a decay factor is applied to the pheromone levels of each edge in the search space (usually with a decay factor ρ set to 0.5), and the pheromone levels of each edge are updated using the following formula:

$$\tau_{ij} \leftarrow (1 - \rho) \cdot \tau_{ij} + \sum_{k=1}^m \Delta \tau_{ij}^k$$

Where $\tau_{i,j}$ is the pheromone value for the edge i,j and ρ is the evaporation rate of the pheromone. The AS algorithm is well suited to non-differentiable routing problems similar to the AGV path optimisation problem outlined in this study. It’s positive feedback mechanisms ensure rapid identification

of optimal routes (for smaller problems), and also inherently allows for routing in dynamic environments, although this is outside the scope of this study. The algorithm also provides inherent parallelism, allowing for the option of concurrent processing. Despite the many advantages offered by the AS algorithm, drawbacks can include an uncertain convergence time (although convergence is guaranteed), and the large number of parameters which require tuning – many of which are selected using empirical processes rather than through theoretical deduction. As a result, incorrect selection of parameters can lead to convergence to local optima and stagnation. In addition, due to the time complexity being equal to $O(n*(n-1)*m*T/2)$, AS algorithms are inefficient for dealing with large-scale combinational problems [18].

V. SPECIFIC USE CASE

A. Problem Definition

The problem tackled in this paper is very much a classic optimization problem, concurrent with the definition of optimization problems given elsewhere in this text. In the simplest terms, the problem is determining the shortest path for an AGV to follow between two points. The application is a manufacturing/logistics operation where an AGV needs to travel from a start point, to a conveyor to pick up a pallet, and then take the pallet to a pallet loading area.

Not only must the AGV find the shortest path, but it must also detect collisions with static objects within its environment and avoid them on future runs until the optimal path is reached. The problem will be tackled using three different algorithms, RRT*, GA, and AS, to each determine separate paths, upon which the paths will be compared to find the shortest.

B. Simulation Approach

Along with clearly defining the problem many AGV end-users encounter, specifically within the manufacturing and

logistics industry, we have clearly defined our approach to accurately simulate these applications through the use of optimization algorithms. The created simulation environment

includes a typical manufacturing plant layout which contains an AGV charging station (Start Location), a manufacturing line (Pallet Pick-up Location) and a warehouse storage facility (Pallet Drop-off Location). This environment was created using the Virtual Robot Experimentation Platform (V-REP) software. To simulate this industrial application, two main paths were generated. The start position of the AGV for Path 1 is the charging station and the end position is the Pallet Pick-up. The start position of the AGV for Path 2 is the Pallet Pick-up and the end position is the Pallet Drop-off. The design of the simulation environment along with Path 1 and Path 2 are shown in Figure 16 below.

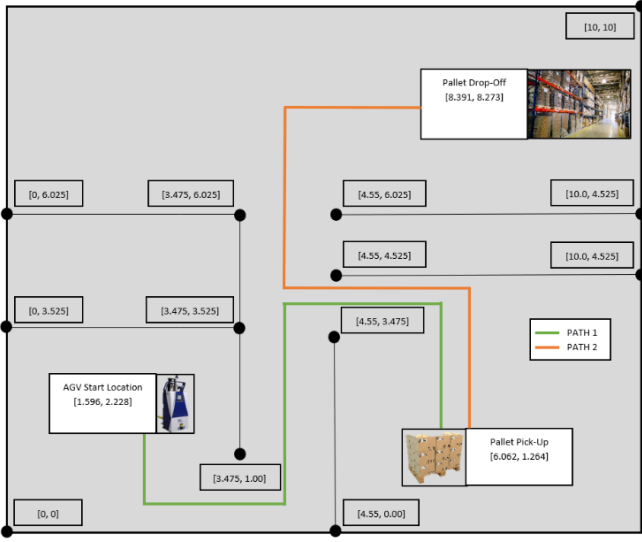


Fig. 16. V-REP Simulation Environment – Industrial Application

The V-REP Simulation Environment was used to model an industrial application. In order to apply the three optimization algorithms to the V-REP scene, the MATLAB development environment was integrated with V-REP using the MATLAB Remote API. The V-REP scene acts as the server, with the three optimization MATLAB scripts acting as clients.

VI. EXPERIMENTAL DESIGN & OBJECTIVES

A. Experimental Design

I. VREP Simulation Environment

The ‘manufacturing site’ as described in the problem definition, is recreated in a piece of simulation software called VREP developed by Coppelia Robotics. The ‘scene’ as it is referred to in VREP is constructed using a grid of a fixed size (10m x 10m) with the origin (0,0) being in the very bottom left-hand corner of the grid. There are four main objects of interest within the scene which are shown below in Figure 10.

- 1 – AGV parked at its start position.
- 2 – First location – Pick up a fully loaded pallet from conveyor belt.
- 3 – Static obstacles (sections of wall)
- 4 – Final location – Pallet loading point.

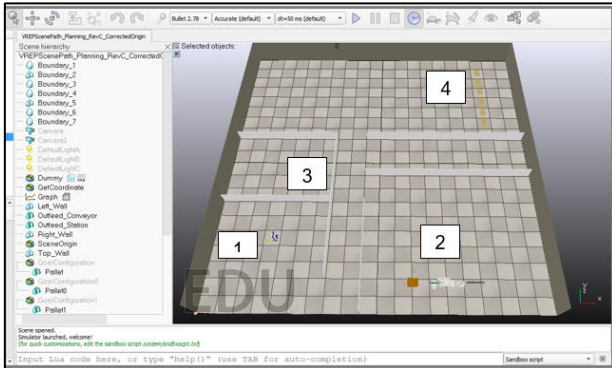


Fig. 17. V-REP Simulation Scene

II. MATLAB Development Environment

The code for each of the three algorithms used for path optimization was written and executed in MATLAB, this is for two main reasons:

1 – MATLAB integrates with VREP via an API allowing the algorithms written in MATLAB to control the robot which is navigating through an environment running in VREP.

2 – MATLAB code is ubiquitous particularly for computer science applications and so the existing code base provides for a good start point for the authors’ initial simulation/proof of concept.

To aid the presentation of the simulation results, code was written to have the AGV render a line behind itself so the results from each of the algorithms could be rapidly compared.

B. Optimization Objectives

Prior to comparing techniques, algorithms were tuned independently in order to find their optimal parameter configuration. This was done on a trial-and-error basis, where each algorithm was tasked with optimising a route from a baseline V-REP scenario. Each algorithm was run fifteen times, during which each run consisted of a slightly modified parameter set. Results from each run were compared in order to find an optimal configuration (i.e. the configuration which resulted in the shortest path from source to target location). Although it is recognised the parameter tuning process traditionally consists of an undefined number of test re-runs, the arbitrary fifteen number limit was set in this study to allow for experimental consistency and for accurate comparison between algorithms. The results of the individual tuning process can be found in Tables 2, 4 and 7.

Once optimal parameter configurations were determined, each algorithm was tasked with optimising a path from ten separate V-Rep scenarios, where each scenario consisted of a modified AGV source location. For consistency and experimental controllability, the source location was the only modifiable variable throughout the testing process; the V-Rep map scene and all static obstacles were left unchanged throughout. The coordinates of each scenario can be found in Table 1. The algorithms were assessed primarily using the success rate and path duration (i.e. the algorithm producing the shortest path being the more successful one), although secondary factors, such as number of computational time and number of tuneable parameters were considered and recorded in the results for discussion. The comparison of the three algorithms across the ten scenarios can be found in Table 9, and a discussion on the results in Section VII.

Scenario No.	Source Location (x,y)	Destination Location (x,y)
1	(0.50, 3.00)	(6.01, 1.26)
2	(0.50, 0.50)	(6.01, 1.26)
3	(1.75, 0.50)	(6.01, 1.26)
4	(2.75, 3.00)	(6.01, 1.26)
5	(1.60, 2.23)	(6.01, 1.26)
6	(6.01, 1.26)	(8.40, 8.25)
7	(5.25, 1.26)	(8.40, 8.25)
8	(7.25, 1.26)	(8.40, 8.25)
9	(6.01, 2.50)	(8.40, 8.25)
10	(8.25, 2.50)	(8.40, 8.25)

Table 1. AGV Route scenarios within V-REP

VII. EXPERIMENTAL RESULTS

A. GA Results

For the individual testing, the start position was: [1.596, 2.228] and the end position was [6.062, 1.264]. For V10 code only mutation probability of 0.2, 0.4, 0.5, 1.0 allowed the code to run so only these values were used. 1 is the maximum recommended value for mutation probability as discussed in literature review. This is to prevent too many good genes being lost by random mutation, weakening the gene pool. Furthermore timing of code implemented using tic/toc inbuilt MATLAB function which may not be the most accurate way of timing. The entire code was timed including VREP Remote API Code.

The values for the GA variables that were tested e.g. 10, 50, 100, 150, 200, 250 were chosen arbitrarily. It was decided to not test the GA with variables with values above 250 due to the excessive processing time required so the author does recognise that shorter paths could have been achievable given more experimentation with higher values. It is also recognised that the author did not test with differing combinations of values for each parameter. For example, 10 chromosomes with 50 generations and a population size of 100 was never simulated.

Test no.	No. of Chrom	Generations	Pop. Size	Mut. Prob.	Path Length [m]	Computation Time [s]
1	10	10	10	0.2	-	-
2	50	50	50	0.2	10.41	14.04
3	100	100	100	0.2	10.35	51.61
4	150	150	150	0.2	10.70	103.32
5	200	200	200	0.2	9.99	153.20
6	250	250	250	0.2	10.60	309.50
7	10	10	10	0.4	12.90	13.65
8	50	50	50	0.4	12.13	11.85
9	100	100	100	0.4	10.28	40.00
10	150	150	150	0.4	-	-
11	200	200	200	0.4	9.29	154.31
12	250	250	250	0.4	10.87	239.50
13	10	10	10	0.5	10.87	1.60
14	50	50	50	0.5	11.38	11.06
15	100	100	100	0.5	11.56	39.27
16	150	150	150	0.5	10.59	87.51
17	200	200	200	0.5	12.43	151.91
18	250	250	250	0.5	11.63	237.87
19	10	10	10	1.0	11.63	1.67
20	50	50	50	1.0	11.63	10.77
21	100	100	100	1.0	9.42	39.70
22	150	150	150	1.0	-	-
23	200	200	200	1.0	10.86	152.86
24	250	250	250	1.0	10.25	279.89

Table 2. GA Algorithm – Calibration Table

Scenario no.	Source Location [x, y]	Destination Location [x, y]	Path Length [m]	Computation Time [s]
1	(0.50, 3.00)	(6.01, 1.26)	12.43	40.28
2	(0.50, 0.50)	(6.01, 1.26)	11.19	40.37
3	(1.75, 0.50)	(6.01, 1.26)	11.34	41.07
4	(2.75, 3.00)	(6.01, 1.26)	11.50	199.08
5	(1.60, 2.23)	(6.01, 1.26)	12.19	40.56
6	(6.01, 1.26)	(8.40, 8.25)	12.05	40.88
7	(5.25, 1.26)	(8.40, 8.25)	16.67	40.47
8	(7.25, 1.26)	(8.40, 8.25)	14.81	39.89

9	(6.01, 2.50)	(8.40, 8.25)	10.54	201.10
10	(8.25, 2.50)	(8.40, 8.25)	11.37	40.03

Table 3. Test Scenarios – Shortest Path

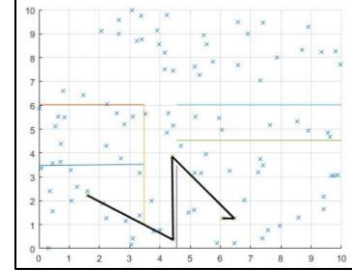


Fig. 18. GA Path 1 Visual Result

B. RRT* Results

As previously stated, the RRT* algorithm was rigorously tested with ten different path scenarios. For each scenario, fifteen test cases were performed in order to obtain a deeper understanding of the main algorithm parameters and the corresponding values which yield optimized path results. The following three parameters were adjusted within their bounds for each scenario:

1 – **Step size [m]**: This is the maximum size a branch can extend from the state x_{random} to the next state x_{nearest} . This value is set in metres.

2 – **Number of States [iterations]**: This is the number of states that are randomly plotted within the bounds of the environment. The algorithm runs until this parameter is met.

3 – **Search radius [m]**: This is the radius around the state x_{random} where the next nearest states x_{nearest} can be found within. This value is set in metres.

Each of the above parameters were varied for each of the test cases related to each V-REP path scenario. The main output to be analysed for each test case was the length of the path from the source location to the destination location in metres. The other result to be analysed was the time taken for the main loop of the RRT* algorithm to execute. The ideal result for each test case was to find the shortest path from source to destination with the shortest possible time to execute the algorithm. In order to calibrate the RRT* algorithm parameters for the V-REP scene, the parameter values displayed in Table 4 were tested for each scenario in order to arrive at the best results.

Test no.	Step Size [m]	No. of States [Iterations]	Search Radius [m]	Path Length [m]	Computation Time [s]
1	10.0	750	10.0	N/A	N/A
2	5.00	750	5.00	N/A	N/A
3	1.00	750	1.00	N/A	N/A
4	5.00	1000	5.00	N/A	N/A
5	2.50	1000	2.50	N/A	N/A
6	1.00	1000	1.00	N/A	N/A
7	0.50	1500	2.50	N/A	N/A
8	5.00	2500	5.00	N/A	N/A
9	2.50	2500	2.50	N/A	N/A
10	1.00	2500	1.00	N/A	N/A
11	0.25	2500	2.50	N/A	N/A
12	5.00	5000	5.00	N/A	N/A
13	2.50	5000	2.50	N/A	N/A
14	1.00	5000	1.00	N/A	N/A

15	0.25	5000	2.50	N/A	N/A
----	------	------	------	-----	-----

Table 4. RRT* Algorithm – Calibration Table

The test case results for each V-REP path scenario are shown in the below tables, with Table 5 displaying the shortest Path Length with no Computational Time limit and Table 6 displaying the shortest Path Length with a Computational Time limit of 300 seconds. The analysis of both sets of results are discussed in detail in Section VIII.

Scenario no.	Source Location [x, y]	Destination Location [x, y]	Path Length [m]	Computation Time [s]
1	(0.50, 3.00)	(6.01, 1.26)	9.851	2980.99
2	(0.50, 0.50)	(6.01, 1.26)	8.840	2993.37
3	(1.75, 0.50)	(6.01, 1.26)	7.893	2713.75
4	(2.75, 3.00)	(6.01, 1.26)	8.388	2754.33
5	(1.60, 2.23)	(6.01, 1.26)	8.410	2728.84
6	(6.01, 1.26)	(8.40, 8.25)	10.130	2085.17
7	(5.25, 1.26)	(8.40, 8.25)	9.762	2217.52
8	(7.25, 1.26)	(8.40, 8.25)	10.587	97.24
9	(6.01, 2.50)	(8.40, 8.25)	9.057	2069.58
10	(8.25, 2.50)	(8.40, 8.25)	11.049	124.42

Table 5. Test Scenario's – Shortest Path [No Limit]

Scenario no.	Source Location [x, y]	Destination Location [x, y]	Path Length [m]	Computation Time [s]
1	(0.50, 3.00)	(6.01, 1.26)	10.883	76.79
2	(0.50, 0.50)	(6.01, 1.26)	9.667	210.03
3	(1.75, 0.50)	(6.01, 1.26)	8.084	253.94
4	(2.75, 3.00)	(6.01, 1.26)	10.156	187.88
5	(1.60, 2.23)	(6.01, 1.26)	9.844	140.96
6	(6.01, 1.26)	(8.40, 8.25)	10.883	76.79
7	(5.25, 1.26)	(8.40, 8.25)	10.551	169.72
8	(7.25, 1.26)	(8.40, 8.25)	10.587	97.24
9	(6.01, 2.50)	(8.40, 8.25)	9.118	134.83
10	(8.25, 2.50)	(8.40, 8.25)	11.049	124.42

Table 6. Test Scenario's – Shortest Path [Limit]

In support of the above tabulated results in Table 5, Figure 18 below show two generated paths from source location to destination. The left path is for Path 1 and the right path is for Path 2 of the simulation environment.

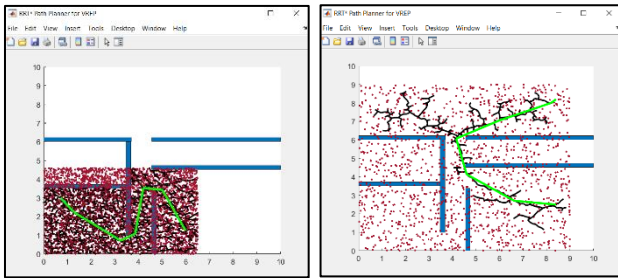


Fig. 19. RRT* Path 1 & Path 2 Visual Results

C. AS Results

For calibration, 15 test runs were carried out for one scenario (Source Location (1,2); Target Location (6.1,1)), during which four parameters were tuned to optimise the algorithm performance. Two parameters were associated with the path construction: the number of random nodes dispersed around the map, which represent the possible locations which an ant can travel to; and the maximum number of segments which can be used to construct a path.

The remaining two parameters; the number of ants and the maximum number of iterations are associated with assessing the fitness function of the available paths. The suitability of parameters was assessed via trial and error, using the primary measure of path length, and the secondary measure of application running time. Table 7 presents an overview of the tuning process and the results:

Test no.	No. of Nodes	Maximum No. of Segments	Max. Iteration	No. of Ants	Path Length /m	Run Time /Secs
1	100	3	20	50	N/A	N/A
2	500	3	20	50	N/A	N/A
3	2500	3	20	50	N/A	N/A
4	5000	3	20	50	N/A	N/A
5	2500	4	20	50	10.53	114.66
6	2500	5	20	50	9.97	98.56
7	2500	6	20	50	10.12	102.70
8	100	5	20	50	N/A	N/A
9	500	5	20	50	N/A	N/A
10	5000	5	20	50	9.93	257.12
11	2500	5	50	50	10.44	140.84
12	2500	5	100	50	9.83	160.84
13	2500	5	20	100	10.22	150.74
14	2500	5	20	20	10.32	99.29
15	5000	5	10	10	10.23	170.23

Table 7. AS Algorithm – Calibration Table

It can be seen from the table that reducing the number of any of the four parameters has a positive effect on the application run time, although reducing any of these parameters too much will negatively impact the path length performance. In the case of the maximum number of segments, reducing this value too much will result in a 'no result', as can be observed in test runs 1 – 4. This is to be expected, especially for the more complex path patterns. The results also show that increasing the maximum number of sections does not reduce path length, and in some cases can result in an increased running time. To account for the more complex path planning, it was determined that the optimal maximum number of segments would be set to five.

With regards to the number of random nodes, again, it can be observed that setting this value too low results in a 'no result', as can be observed in test runs 8 and 9. Increasing this value too much results in a far longer application running time, as can be observed in test run 10. The results demonstrate that setting a longer running time is not justified, with the path length not significantly improving in comparison to test run 6 when half the number of nodes were used. This implies the node number hits a point of diminishing returns; as a result, the number of nodes was set to 2500.

The number of ants and the maximum number of iterations had less influence on performance. This can be explained by the fact that these parameters form the fitness function assessment section of the application, and as a result, a set of suitable paths are already down selected by this stage. Despite this, different values were trialed, and a figure of 20 maximum iterations and 50 ants were deemed as suitable parameters for the main testing.

Scenario no.	Source Location [x, y]	Destination Location [x, y]	Path Length [m]	Computation Time [s]. Note 1.
1	(0.50, 3.00)	(6.01, 1.26)	10.60	138.70
2	(0.50, 0.50)	(6.01, 1.26)	10.66	199.52
3	(1.75, 0.50)	(6.01, 1.26)	9.74	145.02
4	(2.75, 3.00)	(6.01, 1.26)	10.87	230.85
5	(1.60, 2.23)	(6.01, 1.26)	19.99	146.50
6	(6.01, 1.26)	(8.40, 8.25)	9.72	92.02
7	(5.25, 1.26)	(8.40, 8.25)	9.21	134.37
8	(7.25, 1.26)	(8.40, 8.25)	10.42	212.61
9	(6.01, 2.50)	(8.40, 8.25)	8.94	126.90
10	(8.25, 2.50)	(8.40, 8.25)	9.64	250.67

Table 8. Test Scenario's – Shortest Path. Note 1: Testing was performed on an Asus ZenBook 14 Intel Core i7 Processor @ 4.6GHz 8GB

Table 8 presents the results of the algorithm performance when ran against the ten test scenarios. In the majority of cases, reasonable path lengths were found for each scenario. Scenario 5 represents an exception to this rule, where it can be seen the path length is far greater than what was anticipated. This anomaly can be explained by the random allocation of nodes and paths prior to the ant system fitness function, meaning that for slightly more complicated routes where there are less alternatives (sometimes as low as a one path alternative), the down selected paths can sometimes be suboptimal.

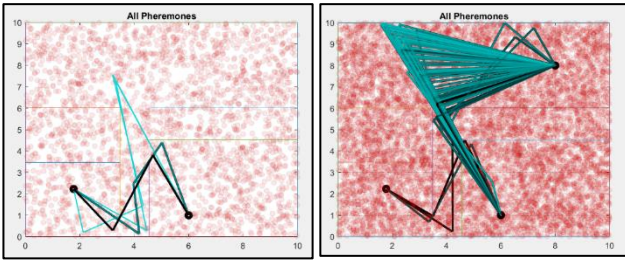


Fig. 20. AS Path 1 & Path 2 Visual Results

VIII. ANALYSIS OF RESULTS

A. GA Analysis

Strengths:

Firstly, it must be noted that certain hardware restrictions existed which affected the simulation speed of the GA and VREP scene. All testing was completed on an Acer Laptop with an Intel Core i5-4200U CPU 1.60GHz, 6GB RAM.

The genetic algorithm proved to be very well suited to the application of path planning for AGV's in an industrial environment. GA's are highly modular in structure and therefore lend themselves to continuous improvement and if required, adaptations to meet new workplace challenges.

The fitness function carried out towards the beginning of the code execution, allows for the code to rapidly decide the best candidates to put forward to become parents. This makes the algorithm ideally suited for optimization problems as the code quickly evaluates the distance between any given chromosome and each location in the path. Ultimately the effect of this is a very efficient algorithm which can quickly determine a suitable path from the start point to the end point

Weaknesses:

One noticeable flaw in the authors (LO) implementation of the GA code in MATLAB, is that there isn't any hysteresis. Because of this, the code isn't learning what an optimal path

looks like beyond collision avoidance. This means that each time the author changed variables and re-ran the simulation, the code wasn't able to learn from any previous iteration and was starting from scratch each time to determine a path.

Another weakness is in the implementation of the crossover and mutation operators. These could have been experimented with to see if different crossover operators or mutation procedures could have yielded shorter path lengths. Furthermore, the code doesn't monitor if the mutation helped to increase the effectiveness of the GA or was hindered. Much of the strength of the GA was reliant on considerable trial/error testing by the author. The mutation operator would lend itself to being decreased in value as the generation number increased, this would prevent mutation having too much of an effect on the strongest of the offspring. The V10 GA code had a mutation operator that was constant in value throughout each cycle albeit different values were experimented with. It is highly possible that the constant mutation rate through either 10 or 50 iterations of the code had a negative impact on the path length achievable. However, testing with a consistently low mutation rate, in the order of 0.1 or less, would mean that potentially the GA could converge around a local minimum too quickly and not find the ideal solution.

Suggested Improvements:

The number of iterations that the main 'for' loop was executed was arbitrarily chosen to be 10 initially. For some combinations of variables as shown in the testing section, this number of iterations wouldn't allow the code to run successfully and plot a path. Therefore, the number of iterations was increased to 50 to allow a path to be plotted successfully. It is recognised that for the sake of efficiency and reduced testing time, the number of iterations of the code could be dynamically decided and only equal to the exact number required by the algorithm to plot a path from the start point to the finish point.

Parameter Adjustments:

The parameters or variables that were adjusted to finely tune the performance of the algorithm were the number of 1) chromosomes, 2) generations, 3) population size and 4) mutation probability. All of the parameters except for the mutation probability were increased in line with each other.

Overall, 48 simulations were run as can be seen from the individual testing results for the GA in Table 2. There are some interesting outcomes from this testing to note. The longest path length was generated in test #0.07 (50 iterations, 12.90M). This is logical as the value of each parameter was only set to 10, this doesn't allow the algorithm to extensively explore the area or generate many offspring. In essence, the points used to plot the path, won't be largely dissimilar to what is randomly generated at the beginning.

The shortest path length simulated was test #0.17 (10 iterations, 9.18M). It would be expected that a test with parameter values of 200 would have a short path. However, it is worth noting that the result, for test #0.18 in which the parameters are set to 250, is worse than that of test #0.17. This suggests an idea that the value of parameters can only be increased so much, before increasing them anymore has little to no positive effect on reducing the overall path length. However, to fully substantiate this theory, each individual test would need to be run multiple times to spot reliable trends,

due to the stochastic nature of GA's it is unwise to make strong assumptions based around a limited dataset.

B. RRT* Analysis

In this section, the performance of the RRT* algorithm is analysed as a tool for Path Planning for AGV applications. As previously stated in Section VII, there were ten path scenarios implemented in the V-REP environment. For each scenario, the RRT* algorithm parameters of Step-size, Number of States and Search radius were varied fifteen times. Also shown in Section VII are the tabulated results of the algorithm performance for each test case. The performance of the RRT* algorithm was analysed by the Path Length [metres] and the Computational Time [seconds]. This was done to evaluate the RRT* algorithm as a solution for AGV Path Planning in Industrial Applications. However, when analysing the results, it is important to note that the testing for each algorithm was performed on a separate PC. Therefore, the Computational Time on each individual PC used varied as a result. For the RRT* algorithm, an Acer Aspire A315-22 PC was used with an AMD A9-9420e Radeon R5 processor (1.80GHz) and 8GB of RAM.

Overall, it is clear from the results that the RRT* algorithm is a very good option for path planning due to its exploratory nature. Analysing the results in detail, the effects of each algorithm parameter is considered below:

1 – **Step size [m]**: From the results, it was clear that as the step size value for the RRT* was decreased the path length also decreased. This is because a smaller step size results in smaller increments as the tree explores. This leads to a higher resolution and a more precise overall tree which explores more of the environment. When a step size of larger than 1m was used, the tree had a lower resolution due to the environment size [10m x 10m]. A step size of 0.5m and lower yielded the shortest paths with a more precise tree, however the number of iterations had to be increased in order to allow the tree to reach the path destination.

2 – **Number of States [iterations]**: From the results, it was seen that the number of states was an important algorithm parameter to tune. When the number of states was less than 500, the RRT* algorithm for generating the path was unreliable and was unable to reach the path destination. Due to the random nature of the tree exploration, less than 500 states were deemed unusable. When the number of states was increased to 750, the path was resolved in all test cases. Once the number of states was 750 or greater, the most reliable results were obtained. In general, the greater the number of states the shorter the resulting path. The best results were seen when the number of states was 5000, with an increased number of available random states for the tree to expand towards. However, this was computationally expensive taking up to 2981 seconds to resolve the path in some cases. In summary, the path length was shortened each time the number of states was increased however in practice for industrial applications, the number of states would need to be balanced with the computational time.

3 – **Search radius [m]**: The search radius parameter of the RRT* algorithm was directly related to the step size, setting the maximum allowable radius for the next path point. Similar to the step size, the lower the search radius

the more states required in order to find the optimal path. The search radius was aligned to the step size value as having a search radius value greater than the step size provided no additional benefit as the next step is limited.

Overall, the best results were generated using a step size of 0.5m or less, a number of states greater than 2500 and a search radius aligned to the step size. Using these parameter values for the RRT* algorithm yielded the best results and the greatest tree density i.e. the ability of the algorithm to fully cover the environment it is exploring. The RRT* algorithm is suitable for deployment to AGV's for industrial applications. To successfully deploy, a balance is required in finding the optimal path versus the computational time taken to generate the path. In order to do this, a balance between the number of states and the available computational power of the AGV's hardware to compute the path as fast as required for the application. As previously stated, the RRT* algorithm does have a randomness to its exploratory tree and therefore its path generation. This may not be applicable for all industrial applications but can be adjusted accordingly, resulting in a shorter path than traditional AGV models.

C. AS Analysis

Table 8 show the AS algorithm performed relatively well in terms of path length performance when compared to the RRT* and GA algorithms, with it achieving a superior path length for 6 of the 10 predefined scenarios. This was in comparison to the four superior path lengths achieved by the RRT* algorithm, and the zero achieved by the GA algorithm. The computational time was similar to that of the RRT* algorithm, the other algorithm with comparable path length performance to the AS programme. As discussed in the *AS Results* section, there was a clear trend in the path length results; the AS algorithm saw an increased performance across Scenarios 6 – 10 (i.e. path 2 from the pallet pickup position to the drop off position), whereas the RRT* performed better across Scenarios 1 – 5 (i.e. path 1 from the AGV start location to the pallet pickup position). This signifies that the AS algorithm in this study is better suited to simpler routing tasks such as path 2 which only has one doorway to navigate through, rather than path 1 which necessitates the navigation through two doorways. This can be explained by the cap on path segments, which was set to five for the final testing to ensure the computation cost was not too great.

The results underpin the original hypothesis that an Ant Colony type algorithm would perform strongly for this study, taking advantage of the known strengths of this family of algorithms; including robust, strong global optimisation ability, as well as the ability to perform quick convergence due to the algorithms inherent positive feedback mechanism. Despite the encouraging results demonstrated by the AS algorithm, the results also present the susceptibility of the algorithm to incorrectly convergence due to the random assignment of paths. This is demonstrated by the path performance in Scenario 5, where the algorithm achieves a performance of 19.99m; a far poorer result in comparison to the other algorithms. ACO-type algorithms are traditionally associated with the issue of incorrect local optima convergence due to their probabilistic nature [18]. Such problems have been dealt with in comparable studies by combining an ACO-based algorithm with another search

space algorithm which has stronger exploratory behaviours. For example, in 2019, Dai X. et al combined an Ant Colony Algorithm with the A* Heuristic method for mobile robot path planning, which saw an improved performance and reduced the likelihood of poor local convergence [24]. Zhou et al utilised an ACO algorithm in a similar way to the AS algorithm in the study, where a separate algorithm was used for creating the initial paths to avoid a blind search of the ant colony [25]. Whilst the AS algorithm in this study performed this initial path search randomly, Zhou's study utilises a deterministic search method; this was found to reduce the likelihood of incorrect convergence associated with stochastic methods such as ours.

D. Comparative Analysis

Table 9 below provides a collective overview of all the results from the three algorithms across the 10 scenarios. The lightly coloured green squares represent the superior value for that metric in the given scenario. For clarity, an example would be Scenario 2, where the RRT* algorithm achieved the superior path length, whilst the GA achieved the superior running time measurement. As previously discussed, the path length was the primary optimisation metric, whilst the running time was classed as a secondary metric. The two metrics were often shown to have an inverse relationship, and as a result efforts were made during the calibration process to ensure a reasonable balance between the two was made. The aim for each algorithm was to ensure suitable paths were constructed in a reasonable time frame (i.e. no longer than several minutes). This was highlighted in Table 5, where results were presented from an RRT* algorithm setup which required a high computational effort, with running times of up to 50 minutes. It can be seen that path lengths achieved from this setup were far superior to the results displayed in Table 6, although, for the purposes of accurate comparison, these results were discarded from the final comparison.

Scenario No.	RRT*		GA		AS	
	Path Length /m	Running Time /s	Path Length /m	Running Time /s	Path Length /m	Running Time /s
1	10.88	76.79	12.43	40.28	10.60	138.70
2	9.67	210.03	11.19	40.37	10.66	199.52
3	8.08	253.94	11.34	41.07	9.74	145.02
4	10.16	187.88	11.50	199.08	10.87	230.85
5	9.84	140.96	12.19	40.56	19.99	146.50
6	10.88	76.79	12.05	40.88	9.72	92.02
7	10.55	169.72	16.67	40.47	9.21	134.37
8	10.59	97.24	14.81	39.89	10.42	212.61
9	9.12	134.83	10.54	201.10	8.94	126.90
10	11.05	124.42	11.37	40.03	9.64	250.67

Table 9. Comparative Analysis – Algorithm Results.

It can be seen from Table 9 that the AS algorithm and RRT* algorithm achieve a high level of performance for optimising the path length; AS achieves superior path lengths on 6 occasions, whilst RRT* achieves superior path lengths on 4. One interesting point of note is the RRT* algorithm generally performs better on Scenarios 1-5, i.e. path 1: from the original AGV position to the pallet pickup position, whereas AS performs better on Scenarios 6 – 10, which

represents path 2: from the pallet pickup position to the final drop off point. Path 1 represents a slightly more complex path pattern in comparison to path 2, as the AGV is required to navigate through two doorways; whereas for path 2, the AGV is only required to navigate through one doorway before it enters an open space. This signifies that the RRT* algorithm is potentially more suited to more complex path planning tasks which require stronger exploratory features, whereas AS is suited to more simplified path planning tasks. It was noted that the GA performed relatively poorly in comparison to the other two algorithms, although it is clear that the running times were in general far superior. This suggests that a higher sacrifice of running time performance could be made in order to improve path lengths. Despite this, on the two occasions where the GA running time was similar to the other algorithm running times (Scenario 4 and 9), the path length performance still lagged slightly, suggesting the GA algorithm is less suited to path planning tasks when compared to the RRT* and AS algorithms.

IX. PROJECT MANAGEMENT

A. Group Approach

Effective project management was essential to the success of this project due to the nature of the remote working methods employed. The team that have completed this study are all based in different locations throughout the UK and Ireland. The first task completed for the project was to agree on which tools to use for communication and project management and then exchange details such as mobile phone numbers and email addresses. The main tools used to manage the project were:

I. Jira Project Tracking Software

Jira was used for task allocation, setting deadlines for individual tasks, portal for sharing documents and links. As can be seen above in Figure 20 Jira uses a virtual Kanban board with cards to represent individual tasks. The tasks can be assigned multiple parameters including but not limited to a priority, the person responsible and a status of either of to do, in progress, or done.

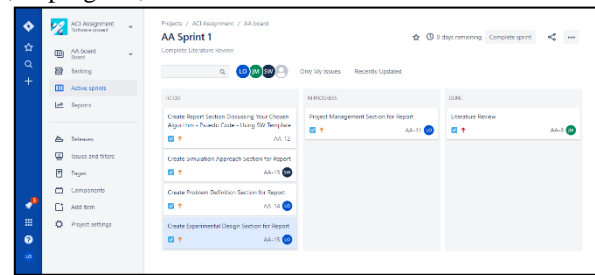


Fig. 21. Jira Project Management Software

II. Project Plan – Microsoft Excel

Excel was used for initial project timeline and stages mapped out before individual tasks were added to Jira. As shown in Figure 21 below, Microsoft Excel was used to track all tasks with their completion dates on one page, in a similar way to a Gantt Chart may be used for more complex projects or projects with larger teams.

ID	Requirements	Our Actions	Status	Checking Date	Comments
1	Facilitate Discussions	- Whatsapp Group / Weekly Skype call?	Open	Ongoing...	Weekly call agreed
2	Identify Problem Boundaries	- Review Project Summary Sheet	Completed	22/03/2020	
3	Shared Resources	- Download V-REP - Download Matlab - V-REP Source sent to all - Draft Report template sent to all for review - Draft Presentation template sent to all for review - Literature Review (2 AGV Papers, 3 on Algorithms)	Completed Completed Completed Completed Completed Completed	22/03/2020 22/03/2020 22/03/2020 22/03/2020 22/03/2020 26/03/2020	V3.6.2 PRO EDU Updated to One (Sprint 1)
4	Each person to complete	- Research optimization algorithms from List - Decide on optimization algorithms - Fill out template for Report on algorithms - Implement algorithm (MATP) - Integration with V-REP - Working solution for Path 1 - Working solution for Path 2 - Implementation complete - Report draft - Presentation draft - Report complete - Presentation complete	Completed Completed Completed Open Open Open Open Open Open Open Open Open Open	29/03/2020 26/03/2020 05/04/2020 05/04/2020 05/04/2020 12/04/2020 28/04/2020 28/04/2020 28/04/2020 28/04/2020 01/05/2020 01/05/2020	SW - RRT*, Lewis - GA, Joe - AS Submission on the 18/05/2020 Submission on the 18/05/2020

Fig. 22. Microsoft Excel Project Management

III. Project Meetings – Skype

Used to hold weekly project team video calls to discuss individual/team progress, issues experienced that week and next steps. Shown in Figure 22 below, Skype was used to hold weekly video calls between the team. This was used for long form discussion where other technologies such as IM begin to break down due to the complexity of the discussion. Skype also allowed any of the team members to share our computer screens with the other people in the call, which allowed for live demonstrations of VREP and MATLAB simulations.

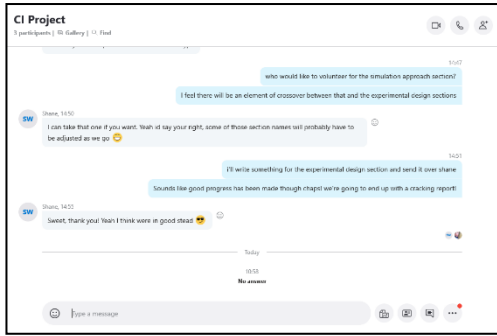


Fig. 23. Project Meetings – Skype

IV. Regular Communication – WhatsApp IM

WhatsApp was used for instant messaging to discuss issues/next steps and share thoughts and ideas. WhatsApp was the primary method of communication between the team. This is where discussions were held relating to:

- Arranging Skype/Teams video calls.
- Informing fellow member's team of progress/issues.
- Discussing differing approaches to tackle problems.
- Alerting team members before emails were sent out.

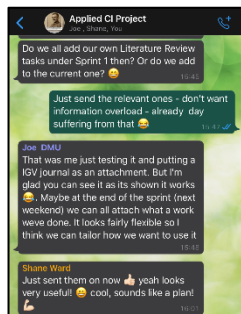


Fig. 24. WhatsApp – Instant Messaging for Mobile Devices

X. CONCLUSION

The objective of this paper was to explore the use of three popular optimization algorithms for the Path Planning function of an AGV. The use of AGV's for applications within the manufacturing and logistics industries is becoming more and more common due to increased requirements for throughput efficiency. Within Europe there has been a vast increase in the number of AGV applications due to the trends of Industrie 4.0, the Smart Factory and Flexible Manufacturing Systems. As a result, the use of Computational Intelligence will provide a key to unlock the potential for traditional automation to evolve.

For this paper, the Genetic Algorithm (GA), the Rapid-Exploring Random Tree Star (RRT*) Algorithm and the Ant System Algorithm (AS) were identified as three of the most common optimization algorithms which could improve the efficiency of the Path Planning function of an AGV to compute the shortest possible path from source to destination. As previously stated, there is a lot of research that has been conducted into the use of optimization algorithms for path planning within the field of mobile robotics. However, from the conducted literature review it is clear that there are a vast number of optimization algorithms options but no reference to their success and implementation specifically for industrial applications. This paper was used to deploy each optimization algorithm within a typical simulated manufacturing application and to compare the results of each algorithm giving a clear indication of which algorithms are suitable for obtaining the shortest path for an industrial AGV.

From the Analysis Section, in general the Ant System (AS) algorithm yielded the shortest path in 6 of the 10 test scenarios, performing best for this industrial application and making it the recommended optimization algorithm option for the path planning function of an AGV. The Rapidly Exploring Random Tree Star (RRT*) algorithm yielded the shortest path in 4 of the 10 test scenarios, performing well for this industrial application and displaying the potential to yield further improved results with additional computational power. The Genetic Algorithm (GA) did not yield the shortest path during the test scenarios but in general computed its path in the shortest time. While the nature of the Genetic Algorithm may not be suited to applications with static obstacles, it may be more suited to applications with simplified paths. When analysing the results in terms of Path 1 and Path 2, the RRT* algorithm performed best on Path 1 whereas the AS algorithm performed best on Path 2. Path 1 included a more complex route due to the location of boundary walls within the simulation environment compared to Path 2, signalling that the exploratory nature of the RRT* algorithm is more suited to complex paths and the direct nature of the AS algorithm is more suited to simplified paths.

Recommendations for Future Works:

An improvement to the Genetic Algorithm used within this paper would be to have the code run within a conditional for loop that would stop as soon as the path had been drawn. This would be more efficient than the implemented system of running the code a fixed number of times as specified by the user. An improvement to the RRT* Algorithm would be to implement a robust rewiring function for the tree in order to cover a larger density within the environment and to create more links between tree branches which would result in

additional shortening of the computed path. As mentioned in the AS Analysis, improvements to the AS algorithm used in this study would be to combine the present algorithm with another search algorithm with a stronger exploratory behaviour, and one which is less likely to stagnate in incorrect local optima. Examples of algorithms which may be suitable for combination include the A* algorithm or the simulated annealing algorithm. As well as individual improvements to each algorithm, there are also generic improvements that could be made. These include but are not limited to:

- Implement logic to handle dynamic obstacles opposed to the current static obstacles within the scene (walls).
- Implement hardware simulation using input from a proximity sensor, low battery or fault condition.
- Implement an application which would serve as a test bed for each given optimization algorithm and also allow for additional testing of other optimization algorithms.
- Implement a function to determine which algorithm to use depending on the task in hand. For example, for path planning with a single AGV, the GA would be recommended or if a fleet is in operation a SWARM algorithm would be recommended.
- Dynamic entry of target location within VREP or perhaps clicking on a particular location within VREP scene moved the AGV to that location.
- More accurate representation/movement of the AGV within the VREP scene opposed to point to point.

REFERENCES

- [1] C. Benevides, "The Advantages and Disadvantages of Automated Guided Vehicles (AGVs)," Conveyco, 27 August 2019. [Online]. Available: <https://www.conveyco.com/advantages-disadvantages-automated-guided-vehicles-agvs/>. [Accessed 28 March 2020].
- [2] A. Group, "Automated Guided Vehicles," The Muratec Group, No Date. [Online]. Available: https://www.agvegroup.com/services_item/automated-guided-vehicles/. [Accessed 28 March 2020].
- [3] T. Ferreira and I. Gorlach, "DEVELOPMENT OF AN AUTOMATED GUIDED VEHICLE CONTROLLER USING A MODEL-BASED SYSTEMS ENGINEERING APPROACH," *South African Journal of Industrial Engineering* August, vol. 27, no. 2, pp. 206-217, 2016.
- [4] T. Sorokina, "625.611 - Computational Methods," No Date. [Online]. Available: <https://ep.jhu.edu/programs-and-courses/625.611-computational-methods>. [Accessed 28 March 2020].
- [5] N. F. Johari, Z. A. Mohd and M. N. Haszlinna, "Firefly Algorithm for Optimization Problem," *Applied Mechanics and Materials*, vol. 421, pp. 512-517, 2013.
- [6] Siddique, Nazmul; Adeli, Hojjat, "Computational Intelligence: Synergies of Fuzzy Logic, Neural Networks and Evolutionary Computing", 2013.
- [7] Holland J., "Adaption in Natural and Artificial Systems", University Michigan Press, 1975.
- [8] Whitley D., "Genetic Algorithms and Evolutionary Computing" 2002.
- [9] LaVelle S. M., "Rapidly-Exploring Random Trees: A New Tool for Path Planning", Iowa State University, 1998.
- [10] Dorigo M., "*Optimization, Learning and Natural Algorithms*", PhD thesis, Politecnico di Milano, Italy 1992.
- [11] Dorigo M., Stützle T., Birattari M., "Ant Colony Optimization", IEEE Computational Intelligence Magazine, 2006.
- [12] X. Yang and C. Wushan, AGV Path Planning based on Smoothing A* Algorithm, International Journal of Software Engineering & Applications (IJSEA), Vol. 6, No. 5, September 2015.
- [13] M. D. Ryck, M. Versteyhe, Automated Guided Vehicle Systems, State-of-the-Art Control Algorithms and Techniques, Journal of Manufacturing Systems, January 2020.
- [14] Rapidly-exploring random trees: A new tool for path planning, S. M. LaValle. TR 98-11, Computer Science Dept., Iowa State University, October 1998.
- [15] Sampling-based Algorithms for Optimal Motion Planning, S. Karaman and E. Frazzoli, International Journal of Robotic Research, May 2011.
- [16] Dorigo M. & Colomi. A, "The Ant System: Optimization by a colony of cooperating agents", IEEE Transactions on Systems, Man, and Cybernetics-Part B, Vol.26, No.1, 1996
- [17] Brownlee J, "Ant Colony System", Clever Algorithms: Nature-Inspired Programming Recipes, 2011
- [18] Wang G, "A Comparative Study of Cuckoo Algorithm and Ant Colony Algorithm in Optimal Path Problems", MATEC Web of Conferences 232, 03003, 2018.
- [19] L. Scrucra, "GA: A Package for Genetic Algorithms in R," *Journal of Statistical Software*, vol. 53, no. 4, pp. 1-37, 2013.
- [20] L. W. Hosch, "Genetic Algorithm," Encyclopaedia Britannica, No Date. [Online]. Available: <https://www.britannica.com/technology/expert-system>. [Accessed 29 April 2020].
- [21] D. Bridge, "Past Teaching - Topics In Artificial Intelligence," 2007. [Online]. Available: <http://www.cs.ucc.ie/~dgb/courses/tai/notes/handout12.pdf>. [Accessed 9 April 2020].
- [22] A. P. Engelbrecht, "Genetic Algorithms," in Computational Intelligence - An Introduction, Chichester, John Wiley & Sons Ltd, 2007, p. 157.
- [23] F. J. Goncalves and M. G. C. Resende, "Biased random-key genetic algorithms," *J Heuristics*, vol. 17, no. 1, pp. 487-525, 2011.
- [24] Dai X. et al, "Mobile Robot Path Planning Based on Ant Colony Algorithm With A* Heuristic Method", METHODS ARTICLE Front. Neurobot., 2019.
- [25] Zhao, J., Cheng, D., and Hao, C., "An improved ant colony algorithm for solving the path planning problem of the omnidirectional mobile vehicle", *Math. Prob. Eng.* 2016, 1–10. doi: 10.1155/2016/7672839, 2016.

APPENDICES

A. GA Algorithm – Literature Review

No.	Format	Title	Authors	Source	Year	Impact
1	Academic Journal	Development of An Automated Guided Vehicle Controller Using A Model-Based Systems Engineering Approach	T. Ferreira & I.A. Gorlach	Department of Mechatronics Nelson Mandela Metropolitan University, South Africa	2016	Medium
2	Academic Journal	Vision-Based Intelligent Forklift Automatic Guided Vehicle (AGV)	Luyang Li, Student Member, IEEE, Yun-Hui Liu, Fellow, IEEE, Mu Fang, Zhizeng Zheng, and Hengbo Tang	Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong	2015	Low
3	Book	Computational Intelligence – An Introduction	Andries P. Engelbrecht	University of Pretoria – South Africa	2007	High
4	Academic Journal	Multi-AGV path planning with double path constraints by using an improved genetic algorithm	Zengliang Han, Dongqing Wang, Feng Liu, Zhiyong Zhao	College of Automation and Electrical Engineering, Qingdao University. Department of Industrial Engineering, University of Texas at Arlington	2017	High
5	Academic Journal	Designing an efficient method for simultaneously determining the loop and the location of the P/D stations using genetic algorithm	REZA ZANJIRANI FARAHANI ^{*y} , BEHROOZ KARIMI ^y and SARA TAMADON ^z	Department of Industrial Engineering, Amirkabir University of Technology. Supply Chain Management Research Group, Tehran	2007	High



B. RRT* Algorithm – Literature Review

No.	Format	Title	Authors	Source	Year	Impact
1	Academic Journal	AGV Path Planning Based on Smoothing A* Algorithm	Xie Yang and Cheng Wushan	Published: International Journal of Software Engineering & Applications (IJSEA)	2015	Aims to reduce distance of travel by optimizing the path of AGV using Smoothing A*, particularly as the AGV avoids obstacles. Uses simulation to demonstrate theory with practical results. Includes motion model of AGV with differential drive plus steering wheel. Compares path length and maximum deviation from path in experimental results.
2	Academic Journal	Automated Guided Vehicle Systems, State-Of-The-Art Control Algorithms and Technique	Matthias De Ryck*, Mark Versteyhe	Published: Journal of Manufacturing Systems	2020	Aims to review state-of-the-art control algorithms for AGV's. Provides very good introduction to AGV's and uses. Discusses wide range of AGV tasks (• Task Allocation • Navigation • Path Planning • Motion Planning • Vehicle Management). Provides detailed description of the most common control techniques for each of the above tasks.
3	Academic Journal	Real-Time Randomized Path Planning for Robot Navigation	James Bruce, Manuela Veloso	Published: IEEE/RSJ International Conference on Intelligent Robots and Systems	2002	Explores the use of RRT in a Real-Time scenario. Provides good description and explanation of RRT algorithm. Analyses performance of path and time to reach the target through waypoints cache and beta search. Covers step sizes in analysis of results. First ever paper using RRT for Path Planning in Robot Navigation
4	Academic Journal	An RRT-based Navigation Approach for Mobile Robots and Automated Vehicles	Luis Garrote, Cristiano Premebida, Marco Silva and Urbano Nunes	Published: 2014 12th IEEE International Conference on Industrial Informatics (INDIN)	2014	Very good experimental results demonstrated. Shows results for each test after set number of iterations. Quantifies the Mean Lateral Error, Mean Square Error, Control Effort and Smoothness Variation. Uses static and dynamic obstacle avoidance.
5	Academic Journal	UAV Path Planning System Based on 3D Informed RRT* for Dynamic Obstacle Avoidance	Jiawei Meng, Sebastian Kay, Angran Li, and Vijay M. Pawar	Published: International Conference on Robotics and Biomimetics	2018	Problem definition includes 3D space. Experimental results record the path length and also the search space. Compares algorithms % optimization rate for 1, 3 and 5 obstacles.

C. AS Algorithm – Literature Review

No.	Format	Title	Authors	Source	Year	Impact
1	Conference Paper	Path Planning and Intelligent Scheduling of Multi-AGV Systems in Workshop	Chengbao L., Li Y. and Bai X.	Proceedings of the 36th Chinese Control Conference July 26-28, 2017, Dalian, China	2017	<ul style="list-style-type: none"> Introduces concept of static and dynamic path planning approaches to AGV systems Discusses the use of Dijkstra, A*, ACO and GA's <p>The paper then goes on to describe a novel approach (unidirectional graph method alongside A* for path planning) for solving a multi-AGV path planning problem, which requires a dynamic approach to resolve collision conflicts</p>
2	Article	Optimal path in an intelligent AGV-based manufacturing system	Fazlollahtabar H. and Saidi Mehrabad M.	September 2015 Transportation Letters the International Journal of Transportation Research	2015	<ul style="list-style-type: none"> Dissects AGV Optimisation problems into three categories, (i) general path topology (ii) path optimisation (iii) specific path topologies Explains Nodes and Arcs concepts associated with AGV path planning Discusses use of Linear Regression Model to find relevance of parameters <p>Discusses potential optimisation parameters, inc. time, cost and AGV capability</p>
3	Conference Paper	A Hybrid Genetic-Heuristic Algorithm for Scheduling of Automated Guided Vehicles and Quay Cranes in Automated Container	Homayouni M. et al.	Conference: 39th Computer and Industrial Engineering (CIE39) Conference	2009	Explores the utilisation of GAs in the AGV arena, although the GAs in this instance are used to synchronise / schedule AGVs, rather than path planning.
4	Conference Paper	Path Planning for Multiple AGV Systems Using Genetic Algorithm in Warehouse	Li C., Cao C. and Gao Y.	3rd International Conference on Communications, Information Management and Network Security (CIMNS 2018)	2018	<ul style="list-style-type: none"> Utilises GA for path planning of multiple AGVs. Discusses other, more traditional path planning approaches such as Dijkstra's algorithm and A*, and explores their advantages / disadvantages <p>Utilises a local search algorithm alongside GA to provide enhanced exploitation capabilities</p>
5	Literature Review	Optimal path planning of mobile robots: A review	Raja P. and Pugazhenth S.	International Journal of Physical Sciences Vol. 7(9), pp. 1314 - 1320, 23 February, 2012	2012	<ul style="list-style-type: none"> Provides overview of path planning algorithms Review is not specific to AGVs/IGVs, but is to the more generic area of mobile robots Categorises off-line and on-line algorithms and discusses both the classical and evolutionary approaches applicable to both. <p>GAs, PSOs, ACOs and SAs are discussed, along with their advantages and disadvantages.</p>

D. Project Summary

Project Title:	CI Algorithms for Flexible Manufacturing Systems The Application of Optimization Algorithms for Path Planning of Automated Guided Vehicles
Key Points:	1. AGV's (<i>Automated Guided Vehicles</i>) have been used for decades to transport goods. They have <i>fixed paths</i> and are usually Optically guided or Laser guided (Line Following) 2. IGV's (<i>Intelligent Guided Vehicles</i>) are only reaching industry now. They have fixed paths but <i>have the intelligence to alter the path to avoid obstacles</i> .
	AGV Example https://www.youtube.com/watch?v=0VEVKWwZ-Nk IGV Example https://www.youtube.com/watch?v=9IGGibHmpMc
Schaeffer	<div data-bbox="501 353 742 580">  </div> <div data-bbox="841 387 879 400">Agilox</div> <div data-bbox="922 353 1195 580">  </div>
Project Objective(s):	1. Each pick an Optimization Algorithm from List (Designing/Implementing IGV Intelligence for an AGV) 2. Implement this Optimization Algorithm to generate shortest paths for IGV within VREP scene 3. Path 1 is from "Charging Point" to "Production Line" [Pickup Pallet] 4. Path 2 is from "Production Line" to "Warehouse" [Drop-off Pallet] 5. Simulate some obstacle avoidance we have time? 6. 18-24 Page Report [70%] 7. 30-40 minute Presentation [30%]

E. Project Schedule

ID	Requirements	Our Actions	Status	Closing Date	Comments
1	Facilitate Discussions	- Whatsapp Group / Weekly Skype call	Open	Ongoing...	Weekly call agreed
2	Identify Problem Boundaries	- Review Project Summary Sheet	Closed	22/03/2020	
3	Shared Resources	- Download V-REP - Download Matlab - V-REP Scene sent to all - Draft Report template sent to all for review - Draft Presentation template sent to all review - Literature Review (2 AGV Papers, 3 on Algorithm)	Closed Closed Closed Closed Closed Closed	22/03/2020 22/03/2020 22/03/2020 22/03/2020 22/03/2020 29/03/2020	V3.6.2 PRO EDU Upload to Jira (Sprint 1)
4	Each person to complete	- Research optimization algorithms from List - Decide on optimization algorithm(s) - Fill out template for Report on algorithm(s) - Implement algorithm (MVP) - Integration with V-REP - Working solution for Path 1 - Working solution for Path 2 - Implementation complete - Report draft - Presentation draft - Report complete - Presentation complete - Final Team Review of Report & Presentation - Submission of all Material to Blackboard	Closed Closed Closed Closed Closed Closed Closed Closed Closed Closed Closed Open Open	29/03/2020 29/03/2020 05/04/2020 05/04/2020 12/04/2020 12/04/2020 19/04/2020 19/04/2020 26/04/2020 26/04/2020 01/05/2020 01/05/2020 12/05/2020 18/05/2020	SW - RRT*, Lewis - GA, Joe - A* Submission on the 18/05/2020

F. ACI Project – Member Contributions (Report)

Section	Section Heading	Sub-Section	Sub-Section Heading	Author
N/A	N/A	N/A	Abstract	S. Ward
I	Introduction	N/A	N/A	L. Osborne
II	Background	A	AGV Functionality	S. Ward
II	Background	B	AGV Control Techniques	S. Ward
II	Background	C	Computational Intelligence	J. McCooley
II	Background	C (I)	Computational Intelligence (GA)	J. McCooley
II	Background	C (II)	Computational Intelligence (RRT*)	J. McCooley
II	Background	C (III)	Computational Intelligence (AS)	J. McCooley
III	Path Planning for AGV's	A	Fixed Positioning	S. Ward
III	Path Planning for AGV's	B	Obstacle Avoidance	S. Ward
IV	AGV's and Optimisation	A	Genetic Algorithm	L. Osborne
IV	AGV's and Optimisation	B	RRT* Algorithm	S. Ward
IV	AGV's and Optimisation	C	ACO Algorithm	J. McCooley
V	Specific Use Case	A	Problem Definition	L. Osborne
V	Specific Use Case	B	Simulation Approach	S. Ward
VI	Experimental Design & Objectives	A	Experimental Design	L. Osborne
VI	Experimental Design & Objectives	B	Optimisation Objectives	J. McCooley
VII	Experimental Results	A	GA Results	L. Osborne
VII	Experimental Results	B	RRT* Results	S. Ward
VII	Experimental Results	C	AS Results	J. McCooley
VIII	Analysis of Results	A	GA Analysis	L. Osborne
VIII	Analysis of Results	B	RRT* Analysis	S. Ward
VIII	Analysis of Results	C	AS Analysis	J. McCooley
VIII	Analysis of Results	D	Comparative Analysis	J. McCooley
IX	Project Management	A	Group Approach	L. Osborne
X	Conclusion	N/A	General Conclusion	S. Ward
X	Conclusion	N/A	Recommendations for Future Work	L. Osborne
N/A	References	N/A	N/A	S. Ward
N/A	Appendices	A	GA Algorithm – Literature Review	L. Osborne
N/A	Appendices	B	RRT* Algorithm – Literature Review	S. Ward
N/A	Appendices	C	AS Algorithm – Literature Review	J. McCooley
N/A	Appendices	D	Project Summary	S. Ward
N/A	Appendices	E	Project Schedule	S. Ward
N/A	Appendices	F	Member Contributions (Report)	S. Ward
N/A	Appendices	G	Member Contributions (Presentation)	S. Ward

G. ACI Project – Member Contributions (Presentation)

Slide Number	Slide Heading	Author
1	Project Title	S. Ward
2	Research Area	S. Ward
3	Problem Definition	S. Ward
4	Existing Literature	S. Ward
5	Current Methods	J. McCooley
6	Proposed Solution	J. McCooley
7	Methodology	J. McCooley
8	Experimental Design	J. McCooley
9	Implementation and Execution – GA Algorithm	L. Osborne
10	Experimental Results – GA Algorithm	L. Osborne
11	Implementation – RRT* Algorithm	S. Ward
12	Execution – RRT* Algorithm	S. Ward
13	Experimental Results – RRT* Algorithm	S. Ward
14	Implementation – AS Algorithm	J. McCooley
15	Execution – AS Algorithm	J. McCooley
16	Experimental Results – AS Algorithm	J. McCooley
17	Project Management	L. Osborne
18	Conclusions	L. Osborne
19	Conclusions (continued)	L. Osborne
20	Conclusions (continued)	L. Osborne
21	Future Work – Algorithm Specific	L. Osborne
22	Future Work – Project Specific	L. Osborne
23	References	S. Ward