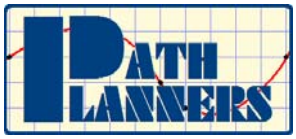


October 10, 2008

To Whom It May Concern,

This letter is to inform you of the progress of the dynamic path planning project that was proposed September 12th. Much time has been spent studying the already existing documentation and code and it has been decided that we will focus on implementing 2 improvements to the system's fitness functions, as well as a Particle Swarm Optimization (PSO) path planning method for comparison purposes.

The first fitness function improvement is a radial vision system that works to improve the already existing collision avoidance system by considering collisions for all paths (as opposed to just the current path) but placing less importance on potential collisions further down each path, and greater importance on imminent collisions.

The second fitness function improvement considers the timing aspects of a potential collision by projecting the area of an obstacle and studying the generated paths that cross through the potential collision zone to see if a collision will occur based on the velocities and the entry & exit points of the path.

Finally, based on research, a PSO algorithm could potentially require less computational complexity and therefore in implementation could increase speed and reduce power consumption. Therefore we will implement a PSO replacement for the Genetic Algorithm and study the performance of both algorithms to compare their performance in this specific scenario. It is yet to be determined if a dynamic PSO algorithm will be developed.

These three developments have just begun and will be under development simultaneously over the next few weeks.

Thank you for your time
Basil Debowski
Jo VandenDool
Kyle Binkley

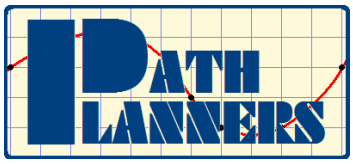
Robot Dynamic Environment Path Planning

Interim Report

Basil Debowski Jo VandenDool Kyle Binkley

Friday, October 10, 2008

University of Guelph



Executive Summary

This report documents the progress of the dynamic path planning project proposed in September. A number of revisions have been made to the proposal and the project will now focus mainly on 3 tasks.

Two of the tasks approach the fitness function currently used in the path planning system and attempt to improve collision avoidance and path feasibility. One of the tasks works by projecting the path of obstacles in the direction of movement and assessing the interactions of the projected paths with the planned path of the robot. The other assesses how imminent a collision is by considering the distance of the collision from the robots current location.

The third task will be to implement a particle swarm optimization (PSO) algorithm with which to compare to the current genetic algorithm with respect to algorithm performance as well as computational complexity. It is yet to be determined if dynamic PSO parameters will be implemented.

Currently these three tasks are simultaneously under development and testing is scheduled to commence early November.

Nomenclature

In the following report we will be using the following abbreviations and terminology.

GA: Genetic Algorithm,

A meta heuristic algorithm that works by combining solutions of known quality to generate new solutions in an attempt to improve quality.

PSO: Particle Swarm Optimization,

A meta heuristic algorithm that works by assigning solutions positions and velocities and moving them in an effort to find better quality solutions.

Table of Contents

Introduction

Problem Statement

It is not unreasonable to believe that in not too distant future, ideas like mobile robots and self-driven cars will be an everyday occurrence. These ideas both rely on the concept of mobile robot path planning in a dynamic environment. This concept deals with planning a path of motion for a mobile robot or vehicle, through an environment that consists of static and moving obstacles. The path that is planned must be short, safe, low-power for the robot to follow, and free of collisions. This type of problem is called a multi-objective problem since it deals with optimizing multiple objectives (length, safety, smoothness). Several techniques have been investigated in solving this problem and some solutions do exist. The research and existing solutions are in there early stages and much work can be done to improve existing ideas.

The project being completed by Path Planners for The University of Guelph is an improvement of an existing project. The existing project exists as a computer simulation which guides a simulated robot through a two-dimensional environment among static and moving obstacles towards a target destination. The target destination is also able to move through the environment. The current solution uses a GAP (Genetic Algorithm Planner) to find optimal paths through the dynamic environment. The GAP was originally designed to plan paths for a non-moving robot through a dynamic environment in which objects were static in motion but could be added and removed from the map, towards a static target. The GAP worked well in finding optimal solutions in this environment. Also, a Local Search algorithm was developed to possibly aid the GAP, but it was never correctly implemented. Later robot, target, and obstacle movement were added into the simulation. The GAP had

problems in this scenario such as finding in-optimal solution and colliding with obstacles. A collision avoidance system was developed but it has some design flaws. This was the current state of the project when it was handed to Path Planners.

Path Planners goals in the project are to solve the current problems with collision avoidance system, to find methods of improving the existing GAP's performance, to create and compare a new path planning algorithm, and to investigate the problem with current implementation of the Local Search algorithm and possibly correctly implement it.

Objectives

1) Investigate current collision avoidance system and address problems:

   Find all the problems with the current collision and investigate the nature of these problems. Find and understand possible causes of these problems. Either fix the problems with the current system or design and implement a new collision avoidance system that solves the problems of the old one. The system must avoid collisions when it is possible to do so, and allow the system to converge to optimal solutions.

2) Investigate methods of improving the current system performance:

   Locate bottlenecks and various problems and shortcomings in the current system. Develop solutions to these problems that will improve system performance (minimize/maximize criteria)

3) Develop a PSO (Particle Swarm Optimization) algorithm planner to compare against the existing GAP:

   Design and implement a PSO algorithm planner into the existing system. Compare the GAP with the PSO planner through a series of benchmarks in order to obtain

various performance data. Use this data to compare the performance of the two algorithms as they are used in path planning problems. Draw conclusions on the strengths and weaknesses of each algorithm when it is applied to path planning problems.

4) Investigate the LS (Local Search) algorithm implementation and possible repair it: Find the problems with the current LS algorithm and understand why it was not correctly or completely implemented. Develop solutions which discuss how to correctly implement or improve the LS algorithm. Implement these solutions and improvements if time permits it.

Background

Robot path planning is concerned with moving safely and efficiently from point A to point B. This evolving technology is proving to be more and more useful in today's society. There are many different approaches when it comes to solving the path-planning problem. These include the roadmap, cell decomposition and finally the genetic algorithm approach that will be considered in this report.

Genetic Algorithm

A genetic algorithm is a very powerful heuristic search technique used to find solutions to optimization and search problems. It's based on the evolutionary idea of natural selection that only the best-suited individuals in a population survive. The individuals that continually adapt to changing environmental factors continue to survive, and the rest die off. This same concept can be applied to the path-planning problem. In a path planning problem, it is required that the robot travel from point A to B while avoiding both static and dynamic obstacles. The algorithm generates solutions up to a given population size. Each individual in the population must then be assessed in some way to allow a quantitative comparison to be made amongst the solutions. This comparison is known as the fitness of a solution.

The fitness function is a way of evaluating the individuals based on the criteria or, objective function. In this case, the decision of feasibility is based on how smooth, direct, and costly a given path is. Each individual in the solution space is evaluated this way

and assigned a numerical value. The individuals that are deemed suitable survive and continue to evolve towards the optimal solution. Individuals evolved through mutation and crossover.

Mutation is achieved by assigning a probability to each individual. The probabilities are usually kept relatively low to ensure that good solutions are not omitted. Each gene in an individual is changed according to the assigned mutation rate. Crossover is the process where two adults trade a specified number of genes in order to form two new children solutions.

The figure below graphically displays the flow of a genetic algorithm.

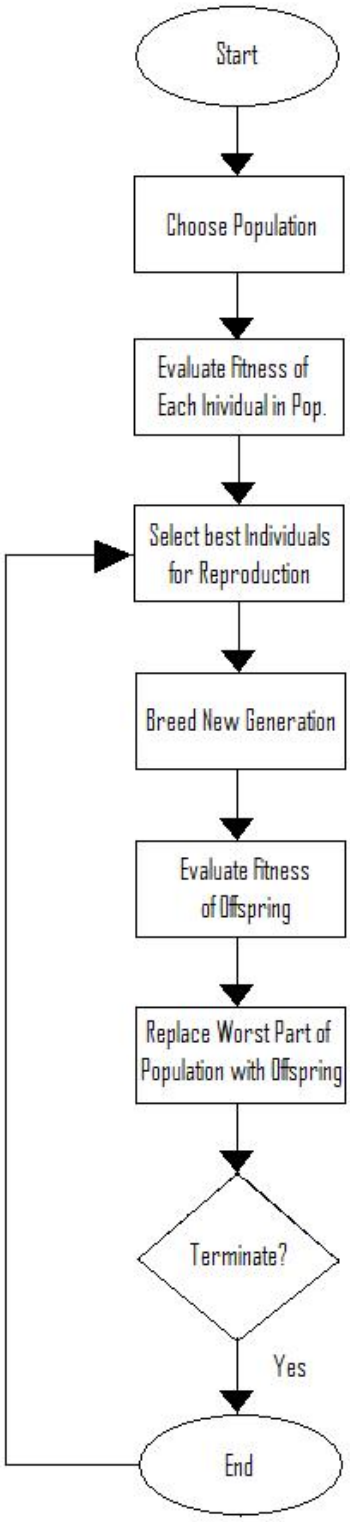


Figure 1 : Flow of a Genetic Algorithm

As mentioned previously there are many alternatives to the GA but this approach is the most attractive when considering dynamic path planning. However, no matter what approach is taken to solving a problem the criteria must be identified and optimized.

Criteria

The standards for which a solution to a problem can be based upon are known as criteria. In this particular problem safety, fitness, smoothness and distance are the primary criteria for optimization.

Safety

This refers to planning a collision free path from the source to destination. Whether objects in the environment are static or dynamic, the path must not lead the robot into a collision on route to the final destination.

Efficiency

The efficiency of a path is evaluated by considering its overall time and energy costs. Each solution is examined to assess whether it is feasible in terms of the time and energy is requires.

Smoothness

Given a robot with a fixed or limited turning radius, the smoothness of a given path then becomes important. The solution must provide changes in direction that are within the capabilities of a given robot.

The safety of a path is one of the most important factors as it ensures that no collisions occur on route to the destination. The collision avoidance portion of the algorithm is therefore a crucial part to include.

Preemptive Collision Avoidance System

Purpose of Collision Avoidance

When robot and obstacle dynamism were added into the Path Planning Problem it became clear that a collision avoidance system was needed. The GAP that had initially been developed had been designed to avoid non-moving obstacles only. Though obstacles could be added into and removed from the environment, the obstacles that were in the environment did not move around. Being so, when dynamism was added into the system the robot was unable to avoid the now moving obstacles. As a result, in-optimal solutions and frequent collisions occurred. PCAS (Preemptive Collision Avoidance System) was developed and added into the system to solve these problems.

System Overview

PCAS uses a collision detection method and a modified fitness function to avoid collision with moving obstacles. Collision detection is performed at each iteration of the GAP by assuming obstacle and robot velocity will remain constant, and for each path looking ahead into the future to see if a collision occurs. Detected collisions are used to penalize the solution that they are detected on through a modified fitness function. The modified fitness function finds the summed total of 4 weighted factors: smoothness cost, clearance cost, length, and collision cost. The collision cost is calculated as 1/timeToCollision where timeToCollision is the number of iterations into the future in which the first collision on the path will occur. Collisions occurring further into the future are penalized less heavily than collisions occurring early on.

Problems with PCAS

PCAS has one shortcoming and three critical problems.

Shortcomings:

1) The system will sometimes find in-optimal solutions when avoiding collision with moving obstacles as shown in Figure 1.01.

Critical issues:

1) The system can potentially get stuck in an endless collision avoidance maneuver as shown in Figure 1.02, thus never reaching the target destination.
2) The system guides the robot into a situation where-after collision cannot be avoided due to limitations in robot maneuverability or environmental constraints (ie. The robot falls into a trap) as shown in Figure 1.03.
3) A collision can be avoided but the system fails to do so as shown in Figure 1.04.

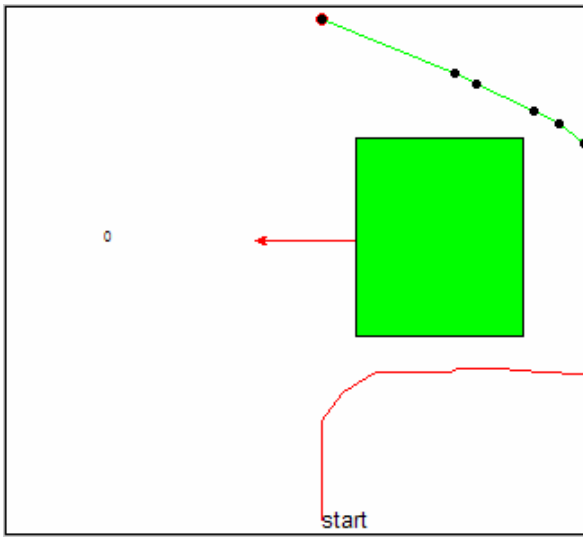


Figure 2 - In-optimal Path

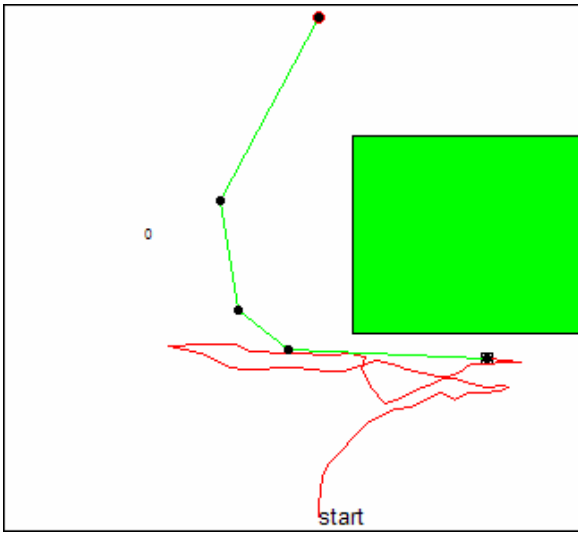


Figure 3 – Endless Loop

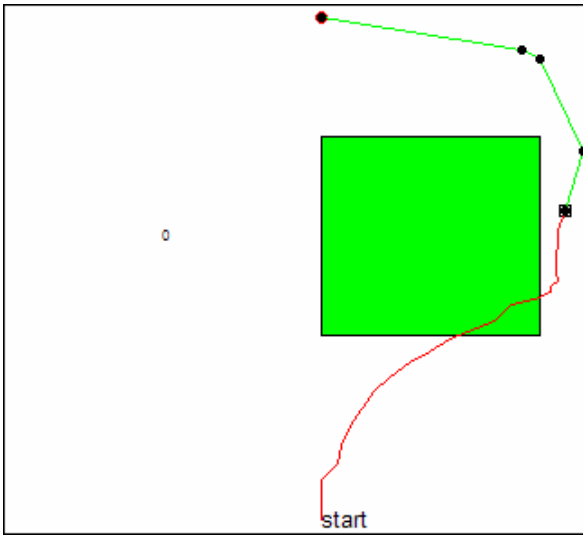


Figure 4– Imminent Collision Due to Trap

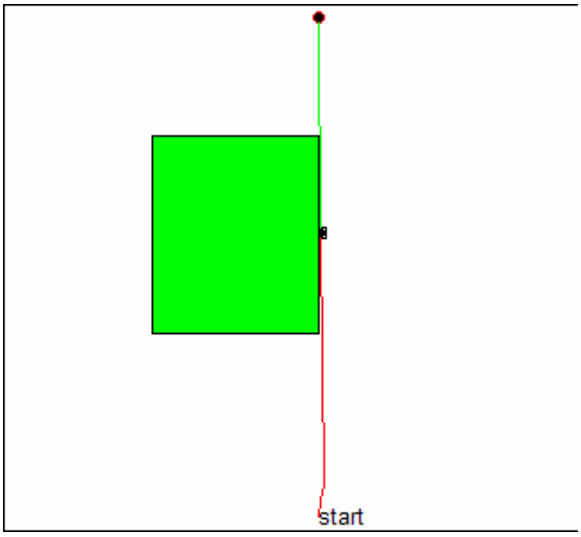


Figure 5– Avoidable Collision

Possible Causes of Problems

Much testing was performed and careful observations were made in both test runs and source code in order to determine the root cause of the problems occurring in PCAS. Thorough investigation has led to several conclusions. The collision detection method and modified

fitness function used by PCAS are in fact working correctly and are being employed for each path in the map at each iteration. The collision detection method is called right after the methods for calculating path length, smoothness cost and clearance cost are called. The collision detection method performs up to a maximum of 100 iterations and correctly returns the number of iterations that elapsed before the first collision was detected. This number is then used in the modified fitness function to penalize paths that result in collisions, with collisions occurring sooner being punished heavier than collisions occurring later. A code snippet showing where the collision detection method is called and how the value is used in the modified fitness function is shown in Figure 1.05. This code can be found in path_planning_planner_utils.c in function calculateFeasiblePathFitness(). A code snippet showing how the collision detection method works is shown in Figure 1.06. This code can be found in path_planning_dynamism.c in function get_time_to_collision().

```
double calculateFeasiblePathFitness(PATH_STRUCT* pth1)
{
   double cost;

   pth1->smthCost = calculateSmoothnessCost(*pth1);
   pth1->clrCost  = calculateClearanceCost(*pth1);

   //colission avoidance
   if (wca>0 && bot_Velocity>0){
        pth1->CollisionCost = get_time_to_collision(My_Map, *pth1);
        if (pth1->CollisionCost==-1){
            pth1->CollisionCost=0;     //no collision in near future
        }else{
            pth1->CollisionCost=1.0 / (pth1->CollisionCost + 1);  //collision is coming up! devalue path.
        }
   }else{
       pth1->CollisionCost=0;
   }

   cost = wd * pth1->length +  ws * pth1->smthCost  +  wc *  pth1->clrCost + wca * pth1->CollisionCost;

   return (cost);
}
```

Figure 6 – Fitness Function Calculation

```
//go forward in time
for (gens_ahead=0; gens_ahead<100; gens_ahead++){
    generation = cur_gen + gens_ahead;

    //move robot
    if (generation % num_generations_between_bot_move == 0){
        isEnvChanged=1;
        botposition = move_dist_on_path(bot_Velocity, &temp_path, 0);
    }
    generation++;

    //move obstacles
    if (generation % dyn_rate == 0){
        isEnvChanged=1;
        for (i = Stationary_obs_count; i< Initial_obs_count ;i++){ // i.e. only moving obstacles
            if (temp_map[i].o_type == 2)
            {
                del_x = dyn_severity * temp_map[i].move_sense * temp_map[i].lambda_x;
                del_y = dyn_severity * temp_map[i].move_sense * temp_map[i].lambda_y;
                for (j=0;j<temp_map[i].v_count;j++){
                   temp_map[i].vertcs[j].x += del_x;
                   temp_map[i].vertcs[j].y += del_y;
                }
            }
         }
    }
    if( isEnvChanged==1){
        isEnvChanged=0;
        //check for a collision
        if (check_collision(temp_map,&botposition)){
            free(temp_map);
            delete_list(&(temp_path.head));
            return gens_ahead;
        }
    }
}
```

Figure 7 – Collision Detection Method

The greatest cause for problems in PCAS is that it is unable to add any new nodes to the paths being evaluated. The only operators that add new nodes to the paths are the repair operator and the smooth operator. The repair operator adds new nodes to paths that cross through obstacles and form infeasible solutions. The smooth operator adds nodes when it replaces a sharp corner formed by a single node and two straight edges, with two nodes and three straight edges. The second greatest problem with PCAS is that the scaled collision cost

in the fitness function does not properly promote convergence to good solutions. Solutions with good genes but an early collision are heavily penalized giving the good genes little chance to survive, whereas solutions containing few good collision avoidance genes but result in a late collision are less penalized and their genes have a better chance of survival. The third greatest problem with PCAS is that it is unable to properly assess clearance cost on dynamic obstacles. The system asses obstacle clearance based on where the obstacle currently is in relation to the path, and not on where the obstacle is going to be in relation to the path. This leads to poor convergence. The consequences of these design flaws are related to each critical issue and shortcoming stated earlier. Also, different possible causes of the critical issues and shortcomings are mentioned and explained.

Possible Causes of Critical Issue 3: Collision is avoidable but the system fails to do so

This situation occurs when all paths in the population will result in collision. It occurs because either:

1) Collision is unavoidable with current number of nodes and new nodes cannot be added,

   or

2) The system cannot find a solution to avoid collision quickly enough.

In the case of 1, the path segment containing the collision is bound by two nodes which are each bound in movement either due to environment restrictions (moving a node will place it in an infeasible location and devalue the path), or the node is a start or end node and cannot be moved. New nodes will not be added by the smooth operator either because no turns or corners exist to be smoothed, or adding nodes with the smooth operator causes the path to

become heavily devalued. New nodes will not be added by the repair operator until the obstacle enters the path segment (at which point collision occurs) and causes the path to become infeasible, however at this time a collision has already occurred. This situation is depicted graphically in Figure 1.04.

In the case of 2, collision is detected for each path but cannot be avoided with the current number of nodes. The system cannot form collision avoiding paths until new nodes are added. New nodes are eventually added, but not quickly enough and not in the right areas. The repair operator will not add new nodes until an existing node is moved such that the existing path becomes infeasible. Such a move would cause the path to be devalued and the path's genes would have a low chance of surviving. The smooth operator will only add new nodes when it smoothes out corners. If no sharp corners exist, or if smoothing existing sharp corners cause poor path fitness (due to clearance, length, or collision costs) this situation also has a low probability of occurring and surviving in the population's gene pool. With low probability of survival, these genes are often quickly lost, and may not survive long enough for mutate and crossover operators to move the new nodes into areas that would avoid collision.

Both case 1 and 2 would be solved if new nodes were added to all the paths where a collision will occur so that these genes have a good chance of survival. Also, by adding the new nodes into areas where they are most needed (where the collision will occur) the system would be able to react quickly and form collision avoiding paths sooner.

Possible Causes of Shortcoming 1: System finds in-optimal solutions when avoiding collision

This situation occurs when all paths in the population will result in collision. It occurs because either:

1) The system is trapped in a local minimum where all paths have converged to an in-optimal solution or,
2) The system is unable to form collision avoiding paths soon enough or,
3) The collision cost factor in the fitness function does not properly promote convergence to good solutions or,
4) The clearance cost factor of the fitness function is unable to properly assess good solutions.

In the case of 1, limitations on the mutation operator and behaviour of the fitness function can cause the system to become trapped in a local minimum that is an in-optimal solution. This may occur because the mutation rate is too low and new genes are dying out faster than they are being introduced, or because the maximum changes allowed to be made by mutation are too small to break free of the local minimum. Also, the fitness function values may be structured incorrectly, thus preventing the solutions from breaking free of the local minimum. For example, in the situation depicted in Figure 1.01, if the smoothness operator were weighted less heavily then it may have encouraged the creation of solutions which did a 360 degree turn and went on to avoid the obstacle from the left hand side as shown in Figure 1.07.

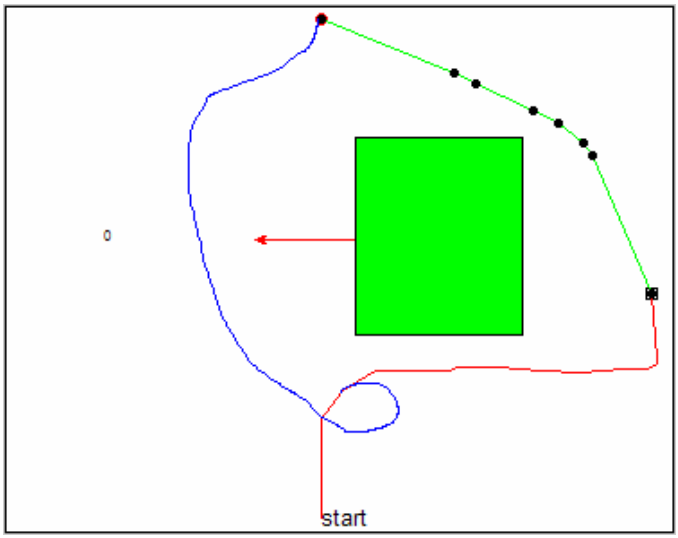


Figure 8 – Possible new solution

In the case of 2, the problem is similar to Critical Issue 3 where the system detects collisions on all paths but is unable to form collision avoiding paths soon enough or at all. In Figure 1.03 we can see that the system was unable to form collision avoiding paths until the obstacle crossed into the path of the robot and made the path infeasible, forcing the repair operator to add new nodes and allowing the system to form collision avoiding paths. Late avoidance of collisions leads to case 1 where the system becomes trapped in a local minimum that is an in-optimal solution.

In the case of 3, the problem lies in how the collision avoidance system analyzes solutions that result in collisions, and how they are penalyzed. PCAS is only able to recognize collisions with obstacles and when they will occur. It is unable to recognize which parts of the path are the major causes for collision. Also, there is a good chance that situations occur when two very similar paths are given very different fitness ratings because one path has an

early collision whereas the other has a late collision. This unbalanced fitness rating leads to poor convergence to good solutions and can be a major cause of in-optimal solutions.

In the case of 4, the problem is that when dealing with moving obstacles the system is unable to properly use the clearance cost factor in the fitness function. This factor plays an important role in converging to good solutions, and when dealing with dynamic obstacle it works very poorly if at all. The clearance cost is calculated based solely on the current location of the obstacle in relation to the path. This is clearly flawed since in fact the system should be looking at where the obstacle is *going* to be in relation to the path. A path that has good clearance now may be completely covered by the obstacle in the next few iterations. An opposite example would be a path that has bad clearance now, but may be very far away from the obstacle in several iterations. By ignoring or improperly analyzing these scenarios the convergence of the GAP is greatly reduced.

Possible solutions for case 1 would be to adaptively increase mutation rate and increase maximum distance of node position changes made by mutation when collisions are detected. Also, having an adaptive fitness function that reduces penalties on smoothness when collisions are detected could help the system escape from local minima. Solutions for case 2 are the same as those for Critical Issue 3. For case 3, a possible modification may be to calculate for how many iterations the robot is colliding with or is inside the obstacle and devalue the path fitness according to this value. In this way, paths that result in many collisions will be more devalued than paths with few collisions, and solutions will converge to optimal sooner. For case 4, a similar solution can be taken as for case 3. For dynamic

obstacles, clearance cost can be taken as an average of how close the robot comes to the obstacle at every iteration into the future. This would begin to approximate the proper behaviour of the clearance factor, but not be an exact solution.

Possible Causes of Critical Issue 2: System guides robot into region where collision will be unavoidable

This situation is an extension of Shortcoming 1, where the in-optimal path found by the system is in fact a solution that will lead to an imminent collision. The causes for this are also extensions of those for Shortcoming 1.

Solutions for these issue are the same as those for Shortcoming 1.

Possible Causes of Critical Issue 1: System is stuck in endless avoidance loop and never reaches target

This situation is a form of Shortcoming 1 where the in-optimal solution is one that never reaches the target. In this case, the system is able to escape from the local minimum that is an in-optimal solution, but does so too late and falls into another local-minimum. The system switches from local-minimum to local-minimum at the same frequency that the obstacle is switching directions.

Possible causes are similar to those for Shortcoming 1, case 1. Limitations in the mutation operator and the fitness function make it difficult for the system to break free of local minima. However, in this case the system is eventually able to do so, but not early enough. Possible causes for lateness in breaking free of local minima may be that the collision cost does not reach a high enough value until the obstacle nears a wall or another obstacle, or that the node mutation direction converges to one direction once the obstacle nears a wall.

As the obstacle nears a wall or another obstacle, the area between the two obstacles or the obstacle and the wall will grow smaller and smaller. As it does so, the possible paths will be forced to come closer and closer to the obstacle they will collide with. As this happens, the timeToCollision will grow smaller and smaller thus increasing the collision cost. Once the cost is high enough the system is able to break free of the local minimum.

When the obstacle nears a wall, the possible paths will get closer to the wall as well. When this happens, mutations that move nodes towards the wall will become infeasible as they will place the node outside of the map boundary. This will slowly force the direction of mutation to converge away from the wall, which assists the system in breaking free of the local minimum.

Possible solutions for these problems are the same as those for Shortcoming 1 as well as to restrict mutation direction on certain nodes in these types of situations.

Summary of Possible Solutions to Problems with PCAS

This section summarizes the possible solutions to problems in PCAS mentioned in the previous section. The major problems with PCAS are its inability to add new nodes to paths except via the repair and smooth operators, its improper use of the clearance cost, and its flawed calculation of collision cost. One possibility for adding new nodes would be to add a new mutation operator. This mutation operator would be responsible for adding new nodes to each path. The rate of mutation could be adaptive and also be unique to each path. Thus, paths with collisions detected could have a huge increase in this mutation rate and have a sudden increase in nodes. The downside in this method is that it would not guarantee that the nodes would be place where they are needed. A more proper calculation of collision cost could be formed by calculating the number of iterations into the future that the robot would be colliding with or be inside the obstacle. This value would then be used as the collision cost. This method would be a more accurate way of determining which solutions contain good collision avoidance genes and would promote convergence to these areas of the solution space. A better method of calculating clearance cost when dealing with moving obstacles could be formed by calculating the distance from the robot to the obstacle at each iteration into the future. These distances could be summed, averaged, and then added to the clearance cost for the rest of the path (the static objects). This would cause the clearance cost to behave more closely to how it was intended, and promote solution convergence to good solutions.

Solutions to less major problems and possible improvements for performance with PCAS are to introduce adaptive mutation rate, adaptive maximum node movement distances due to mutation, and adaptive fitness function weights such as devaluing smoothness, all to help escape local minima.

PSO

Currently the system employs a genetic algorithm to converge on a path solution it determines to be the best. It has been shown however that in some cases PSO offers a faster and better convergence towards the optimal solution. The following chart shows bench mark tests on 3 different functions (Griewank, Rastrigin & Rosenbrok) [1].

Optimum=0, dimension=30
Best result after 40 000 evaluations

30D function	PSO Type 1"	Evolutionary algo.(Angeline 98)
Griewank [±300]	0.003944	0.4033
Rastrigin [±5]	82.95618	46.4689
Rosenbrock [±10]	50.193877	1610.359

It is for this reason that we will develop a PSO algorithm and comparison to be tested against the already existing GA.

The major constraint in the development in this algorithm and comparison is that the PSO algorithm shall be developed to use the same fitness functions as the GA, however we will also maintain the number of initial particles / genes and the number of iterations. The

set of benchmark test scenarios will also be the same for both algorithms. The criteria of one being better that than the other shall be based not only on the fitness function values returned, but also on computational complexity.

Design Methodology

Scaled Portion Path Consideration for Collision Avoidance

Below is an example of a typical dynamic layout where there is a starting point, a destination and moving obstacles. The population of solutions are generated as seen below P1, P2, P3 and P4.

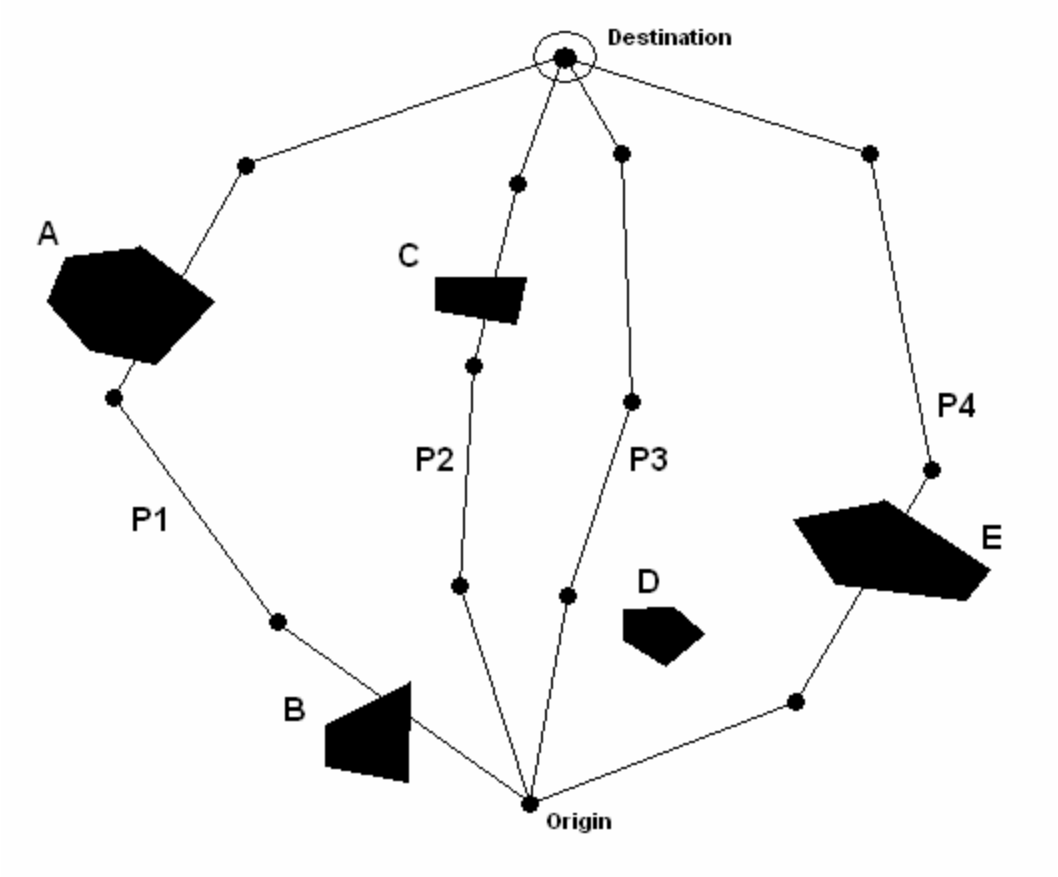


Figure 8 - Path Planning Example Layout

Implementing the current genetic algorithm in this situation would result in the fitness function evaluating each of the solutions individually, based on the following criteria:

μ - total number of intersections with obstacles

η – average number of intersections per infeasible segment

Considering paths P2 and P3 in figure 1 specifically, P3 appears to be the most feasible solution in this case. Based on the parameters above, the fitness function would support this claim. In a static environment this would hold true as it is a direct, collision free path from the origin to destination. However, in a dynamic environment where the obstacles are moving, this might not hold true. At some time t in the future, figure 1 may actually look more like this.

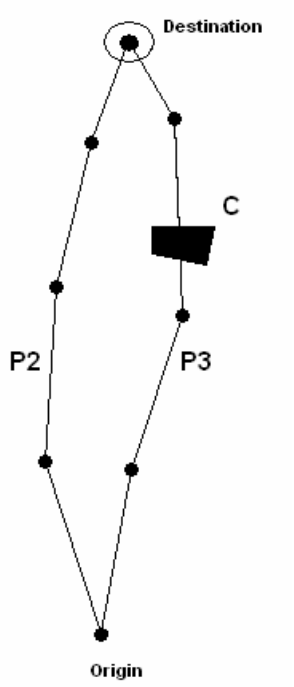


Figure 9 - Figure 8 Some Time 't' In the Future

It can be seen in figure 9 that in a dynamic environment at some time t, path 2 is no longer an optimal solution. The current algorithm would have originally assigned a poor fitness to P2 based on the fact that there is the possibility of a collision. Therefore, the

probability of this solution surviving and evolving is low. After studying this example it can be proposed that upon solution evaluation, the fitness function should take consider the location on the path where a collision occurs. Introducing a new parameter to the fitness function could assign a penalty percentage depending on where the collision occurs on the path. If the collision close to the current location of the robot, a high penalty percentage is assigned. Alternatively, if the collision is further down the path, only a minor penalty percentage would be assigned. Basically, an adjustment is made to the fitness function according to where the collision occurs in relation to the robot.

Introducing this revised fitness function would mean that path P2 would have a higher chance of survival and therefore potentially evolve into a more optimal or final solution. A potential downfall of adding a new parameter to the fitness function would be that it would effect the computation time. Since each solution is assigned a fitness function this could affect the performance of the algorithm when considering large population sizes. Upon implementation it will be possible to evaluate whether the benefits outweigh the cost of an additional parameter to the fitness function.

Radius of Vision

In the current implementation of the GA, the entire solution is considered throughout the entire evolution process. This seems like a waste of resources because only the area surrounding the robot is actually important. An alternative to this would be applying a radius of vision. In other words, only portions of paths within a specified radius of the

robot are evaluated. By doing this, the processing time of each individual in the solution space is decreased. When implemented it will be possible to pinpoint exactly how the processing time is affected. The figure below shows how the radius of vision would work.

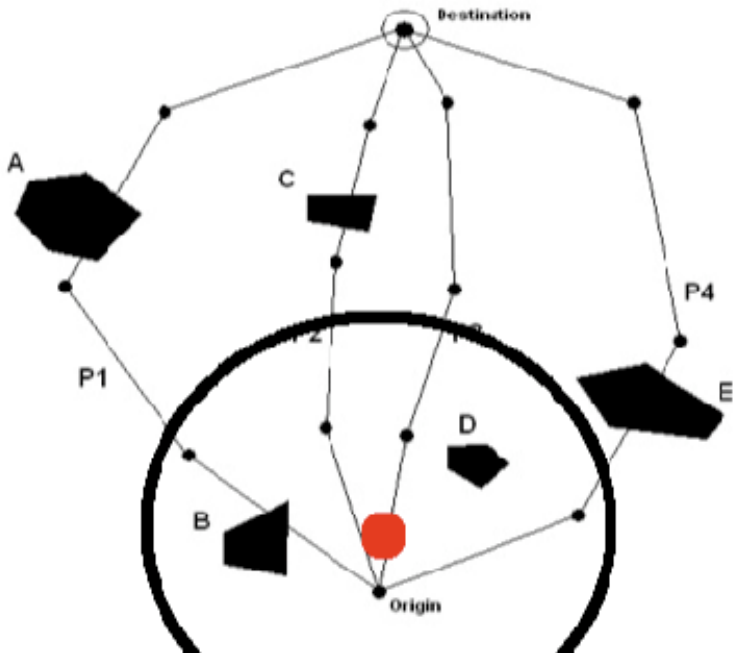


Figure 10: Radial Robot Vision

Intelligent Collision Avoidance System

In order to address the problems and shortcomings of PCAS, a new solution is being developed. ICAS (Intelligent Collision Avoidance System) will address the major causes of problems and shortcomings present in PCAS by providing methods of adding new nodes to solutions when necessary, properly assessing clearance with moving obstacles, and by removing collision detection and collision cost and instead treating collisions in the same manner that infeasible solutions are currently detected and penalized. ICAS will accomplish these achievements by mapping moving obstacles into a time domain. In this way, the GAP

will be able to see where the obstacles will be in the future and how the paths will relate to these obstacles. ICAS will allow the system to treat moving obstacles as static ones, but redraw them as each path would see them in the future. In this way, ICAS will allow the GAP to retain all of its currently tested and properly working operators to correctly and reliably add new nodes where needed and to correctly calculate the fitness function in a manner that causes predictable and desirable convergence. In this way, ICAS in its basic form will be able to outperform PCAS with solutions and improvements added. Also, ICAS will still allow for improvements such as adaptive parameters and a new mutation operator to be added later.

System Overview

ICAS attempts to see into the future in great detail and accuracy. It does so by predicting the approximate shape and position of an obstacle, as each path will encounter it as it passes through the obstacle's AOM (area of motion). The system starts by removing all moving obstacles from the standard methods of evaluation and operation. Moving obstacles are evaluated and operated on in the same way as non-moving obstacles but must be redrawn on a time-domain for each path individually before this can be done. ICAS next generates an AOM for each moving obstacle in the map. The AOM is the approximate area that the obstacle will pass through in a certain number of iterations in the future. It is approximate because it is based on the encompassing rectangle of an obstacle and not the actual obstacle itself. This approximation reduces program complexity and improves performance. An example of the AOM is displayed as the lightly shaded region in front of the obstacle in Figure 1.08 and Figure 1.09.

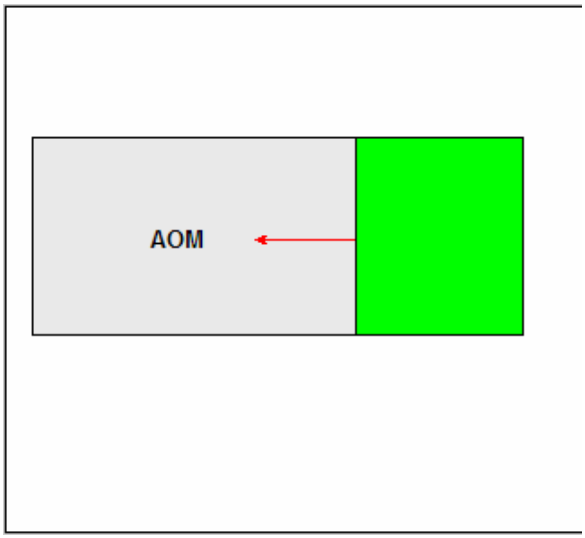


Figure 11– Area of Motion Horizontal Motion

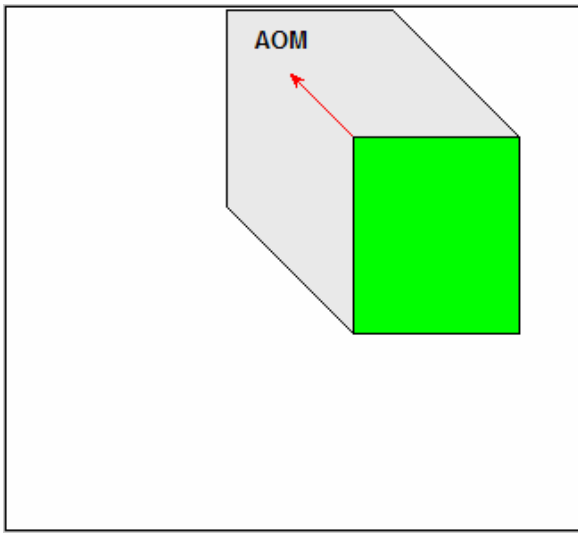


Figure 12- Area of Motion Angled Motion

The AOM is essentially a projection of the obstacle's encompassing rectangle along its vector of motion. The length of this projection is the product of the number of iterations that ICAS wishes to look into the future and the speed of the obstacle. ICAS uses the AOM to determine which paths will be evaluated according to the motion of the obstacle (ie, paths that lie outside of the AOM can be ignored to save processing power).

For each path that crosses a portion of the AOM, ICAS generates a new object unique to that path called the PDO (Path Dynamic Obstacle). The PDO is the obstacle that the GAP will use to evaluate the path's fitness and to perform its operators on (ie, repair). The path fitness as evaluated around the path's PDOs is then added to the fitness as evaluated around the static obstacles. When the repair operator is called for each path, it looks at both the static objects that the path crosses through and also the PDOs that each path crosses through.

Repairs are made accordingly. The creation of each PDO is complex and is explained in the next section.

PDO Creation

Before a PDO can be created, ICAS first determines how a path is crossing the AOM in question. There are two base cases.

Case 1: The path crosses the AOM with a single line (there are no nodes that lie inside the AOM)

Case 2: The path crosses the AOM with multiple lines (there are one or more nodes inside the AOM)

These cases are displayed in Figure 1.10 and Figure 1.11. Each case 2 type of path crossing must be partitioned into as many case 1 type crossings as necessary. The fitness evaluation function and repair operator must be performed individually on each case 1 partition and the results combined. An example of this partitioning is shown in Figure 1.12.

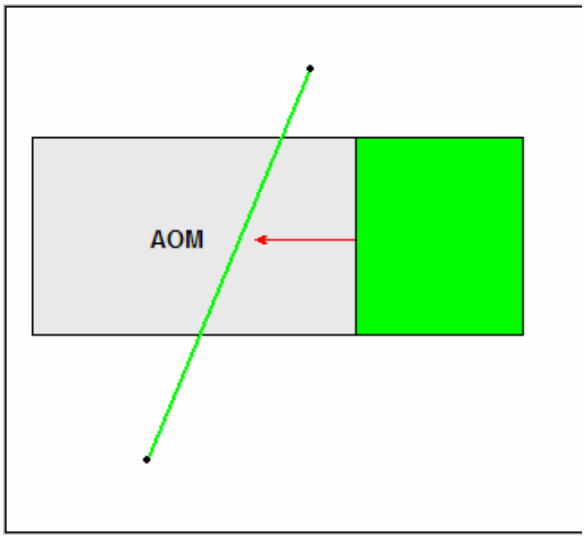


Figure 13 – Case 1 Path Crossing

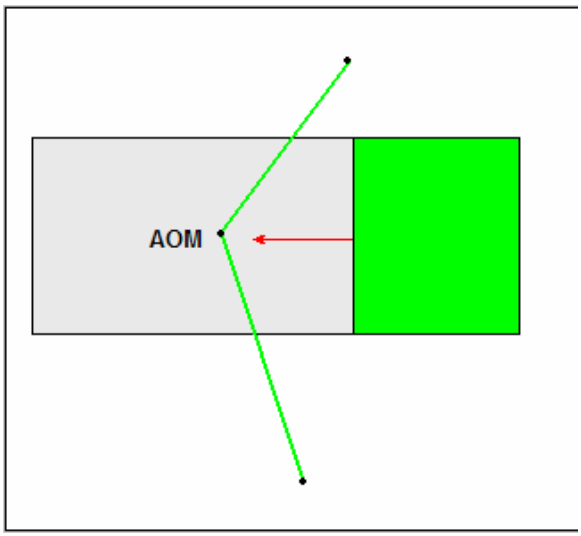


Figure 14 – Case 2 Path Crossing

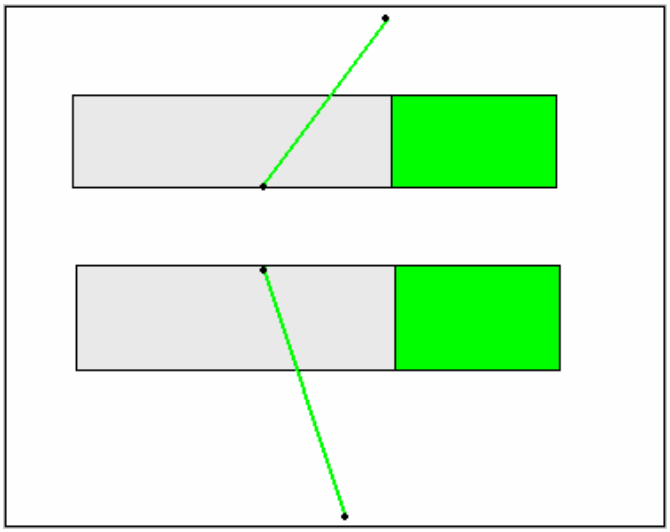


Figure 15 – Case 2 Partitioned

Each Case 1 path crossing can then be organized into one of three possible conditions:

Condition 1: Path enters AOM on one edge parallel to obstacle's DOM (direction of motion) and leaves on opposing edge parallel to object's DOM.

Condition 2: Path enters AOM on one edge perpendicular to obstacle's DOM and leaves on one edge parallel to object's DOM.

Condition 3: Path enters AOM on one edge perpendicular to object's DOM and leaves on opposing edge parallel to object's DOM.

These three possible conditions are displayed in Figures 1.13 through 1.15. Condition 2 and 3 path crossings are trimmed to become Condition 1 path crossings. This is shown in Figure 1.16.

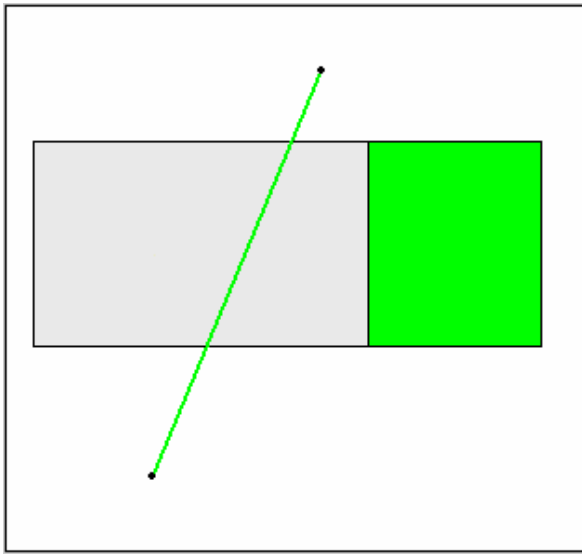


Figure 16 – Condition 1

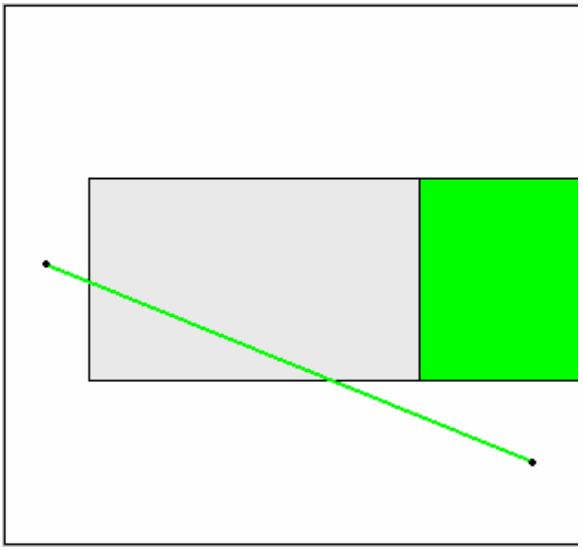


Figure 17 – Condition 2

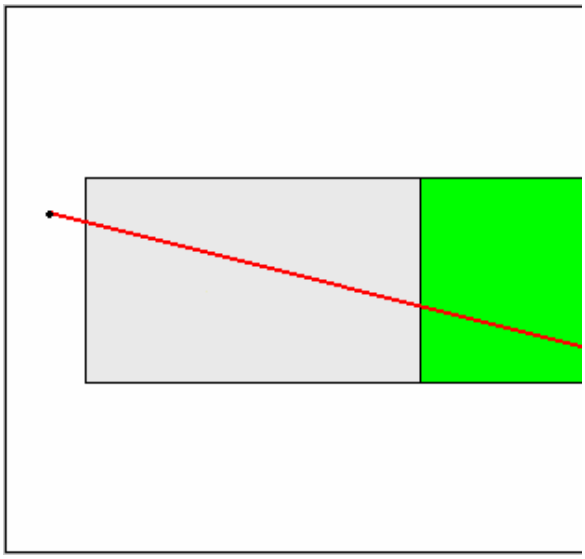


Figure 18 – Condition 3

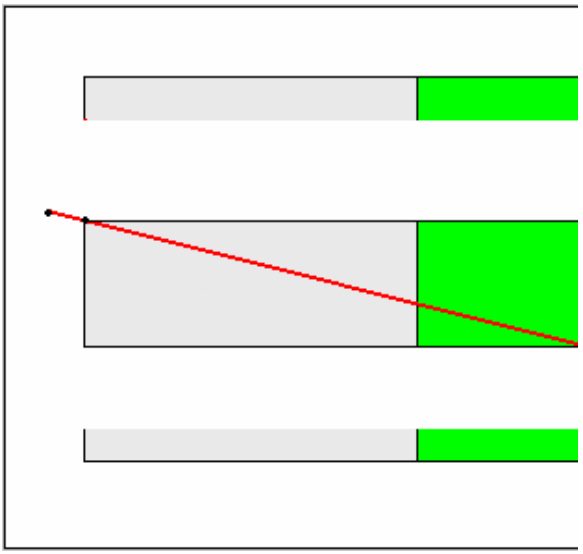


Figure 19 – Trimming to Condition 1

Next, ICAS will draw the PDO for each path at each AOM the path crosses. In this step, ICAS looks at each of the two edges where the path crosses the AOM. At each edge, ICAS determines at what iteration the robot will reach the path/edge intersection. ICAS then redraws the two obstacle vertices that are located on that edge, as they will appear at that iteration. The four newly created obstacle vertices are then joined to form the PDO. This process is illustrated in Figure 1.17.

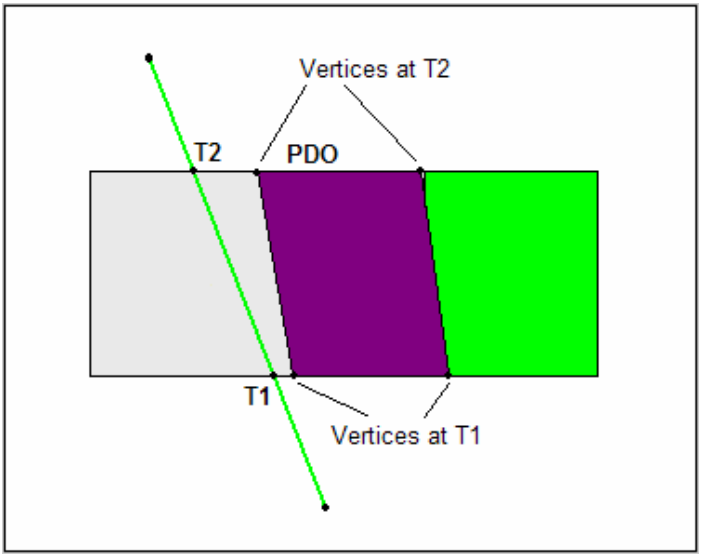


Figure 20 – PDO Creation

ICAS Benefits

The benefits of using ICAS as opposed to PCAS will be that ICAS will be able to quickly guide the GAP into regions of the solution space that not only avoid collision, but also avoid collisions in an optimum manner. ICAS overcomes the problem of generating new nodes that PCAS faces by allowing the GAP to use the repair function on the PDOs that ICAS creates. Also, ICAS allows the GAP to properly use its clearance cost factor in the fitness function which both speeds up convergence to good solutions and also allows for better solutions to be created. Rather than heavily modifying the existing fitness function and GAP operators, ICAS leaves the current system in tact and instead attempts to interface with the current system. By interfacing with existing operators and fitness function evaluation that has been extensively developed, if successfully implemented, ICAS will be a reliable and stable collision avoidance system that can later be expanded and improved on.

Possible Difficulties and Problems

Although ICAS has many benefits over PCAS, there are several possible problems and difficulties with the system. One difficulty will be apparent when dealing with partitioning Case 2 path crossings. Each partition must be evaluated separately by the fitness function and operated on separately by the repair operator. These individual operations must then be combined back into a whole such that they work with the whole solution. It may be very difficult to find a method of performing repair operators such that they do not conflict with each other or form poor solutions. One possible solution is to allow the repair operator to function only once per each set of Case 1 partitions (partitions that came from a common

Case 2 path crossing). It is unknown how well such a solution would perform. Another problem that may arise is computational difficulty. Creating a PDO for each path at each AOM that it encounters may prove to be computationally challenging and time consuming. A possible solution to this may be to only generate a PDO for the first AOM that each path crosses, and simply ignore any other AOMs the path may cross. It is unknown how such a solution may perform.

PSO:

Unlike a GA, no child nodes are created in PSO. PSO works by updating particle velocities in certain directions based on the histories of the particles best (pbest), the local best between neighbouring particles (lbest), and the global best of all particles (gbest). We will attempt to follow the standard algorithm for PSO which is as follows.

First there are 2 equations used in the algorithm that are used to update the particles velocity and position. These equations are as follows:

```
v[] = v[]+c1*rand()*(pbest[] - present[]) + c2*rand()*(gbest[]-
present[]) (a)
present[] = persent[] + v[] (b)
```

The algorithm itself follows the following pseudo code [2]:

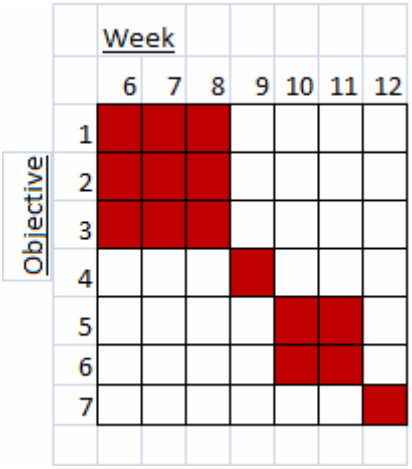
```
For each particle
    Initialize particle
END

Do
    For each particle
        Calculate fitness value
        If the fitness value is better than (pBest) in history
            set current value as the new pBest
    End

    Choose the particle with the best pBest of all the particles as the
gBest
    For each particle
        Calculate particle velocity according equation (a)
```

```
        Update particle position according equation (b)
    End
While maximum iterations or minimum error criteria is not attained
```

Ideally the development will be finished by the end of the month and testing will begin in November.

Schedule of Activity

The following Gantt chart outlines the schedule we will be following over the course of the next 6 weeks. The week numbers follow the University of Guelph semester calendar dates. The objectives correspond to the table below.



	Objective	Dates
1	Scaled Portion Path Consideration for Collision Avoidance	October 14th – 31st
2	Radius of Vision	October 14th – 31st
3	Particle Swarm Optimization Algorithm	October 14th – 31st
4	Benchmark Tests & Analysis of Results	November 1st – 8th
5	Adaptive parameter tuning for both algorithms	November 9th – 23rd
6	Local Search	November 9th – 23rd
7	Poster & Final Report	November 23rd – December 1st

Conclusions and Recommendations

PCAS, the current collision avoidance system, has several problems. The greatest of these problems are:

1) It is unable to add nodes to the possible paths or it is unable to add them early enough
2) The penalties given to paths resulting in collisions do not correctly promote convergence to optimal solutions
3) The clearance cost of paths is not being evaluated correctly and much convergence is lost due to this

These problems are difficult to resolve effectively within PCAS, and although some solutions were considered, it is unknown how well they will perform or how difficult they will be to implement.

A new collision avoidance system, ICAS, was designed and investigated. ICAS in its basic form will solve the problems faced by PCAS. ICAS will map moving obstacles into the time domain so that they may be treated, evaluated around, and operated on in the same way that static object are. ICAS will redraw the moving obstacles as static obstacles and draw them as they will be encountered by each path as the robot passes them. By interfacing with the current system's operators and evaluation techniques, ICAS will ensure proper convergence and correct behavior in the existing GAP.

By implementing a PSO algorithm we hope to observe a noticeable difference in the performance of the system, as this would have immediate implications when scaled to mobile robotics. An increase in performance would allow for more accurate

decisions in a shorter time frame, which not only saves battery power but also allows for speed increases in the robot.

The current implementation of the GA is a solid basis for improvement and testing purposes. The changes proposed in this report are justified based solely on an educated hypothesis at this point. Upon implementation these changes will be tested and the improvements will be measured and reported.

References

[1] Clerc M., Kennedy J., "The Particle Swarm-Explosion, Stability, and Convergence in a Multidimensional Somplex space", IEEE Transaction on Evolutionary Computation, 2002,vol. 6, p. 58-73.

[2] Xiaohui Hu, "Particle Swarm Optimization: Tutorial." Particle Swarm Optimization. 2006. 13 Oct 2008 <http://www.swarmintelligence.org/tutorials.php>.