A Constraint-based Solver for the Military Unit Path Finding Problem

Louise Leenen, Johannes Vorster, Willem Hermanus le Roux
Council for Industrial and Scientific Research, South Africa
Meiring Naude Rd, Brummeria, Pretoria
lleenen@csir.co.za, jvorster@csir.co.za, whleroux@csir.co.za

Keywords: Military Unit Path Finding Problem, Constraint Satisfaction Problem, Rural and Urban Military Operations, Autonomous Systems, Scenario Planning

Abstract

We describe the first phase development of a path finding simulation in a military environment. This concept demonstrator can be used for mission planning by constructing *what-if* scenarios to investigate trade-offs such as location of deployment and mode of transport.

The Military Unit Path Finding Problem (MUPFP) is the problem of finding a path from a starting point to a destination where a military unit has to move, or be moved, safely whilst avoiding threats and obstacles and minimising costs in a digital representation of the real terrain [1]. Although significant research has been done on path finding, the success of a particular technique relies on the environment and existing constraints. The MUPFP is ideal for a constraint-based approach because it requires flexibility in modelling.

We formulate the MUPFP as a constraint satisfaction problem and a constraint-based extension of the A* search algorithm. The concept demonstrator uses a provided map, for example taken from Google Earth, on which various obstacles and threats can be manually marked. Our constraint-based approach to path finding allows for flexibility and ease of modelling. It has the advantage of modelling new environments or additional constraints with ease, and it produces near-optimal solutions if solving is halted prematurely.

1. INTRODUCTION

During mission planning, military commanders have to investigate trade-offs between various constraints imposed by factors such as the location of deployment, the available modes of transport, the mission objectives and a number of other such factors. There is a requirement for constructive simulation environments to conduct military-based experiments such that human and military unit behaviour is modelled. We describe the first phase of the development of such a simulation concept demonstrator in which we address the problem of path finding for military units.

Path finding is the problem of moving an object from a starting point to a destination point, whilst avoiding obstacles and minimising costs in a digital representation of a real terrain. It has many applications ranging from computer games, transportation, robotics, networks, and others. The path finding problem has been well studied and there is a vast body of research in the literature. However, each application has its own characteristics. In our intended application, the military unit path finding problem (MUPFP) [1], we need to maintain a balance between the two main criteria, route speed and safety.

We describe the simulation environment and the different algorithms. This phase of our simulation includes three different path finding algorithms for solving the MUPFP:

- A constraint satisfaction problem (CSP) formulation with a branch and bound algorithm to solve it.
- A constraint-based extension of the well known A* search algorithm.
- A genetic hill-climbing algorithm.

To the best of our knowledge, the first two approaches have not been followed for solving the MUPFP as yet.

Our simulation concept demonstrator uses images as the basis for marking terrain features and combatants. The images may be from satellite sources, aerial photographs, or even images from Google Earth. The provided image is converted to a 2D grid. Various obstacles can be selected and marked by the user, for example: boundaries such a wall, buildings, parks, marshes, etc. Each such feature has properties that determine the ease of movement for different unit types. The user can also mark the position of a threat such a sniper, and the reach of that threat; sniper range in the case of a sniper, on the map. The concept demonstrator indicates the area of the solution space that has been explored, the best path found so far, and the current path being explored.

Section 2 contains an overview of methods for solving path finding problems. In Section 3 we discuss constraint-based solving techniques, and introduce a CSP formulation for the MUPFP. We also discuss two algorithms to solve this CSP. Section 4 describes our simulation concept demonstrator, and we conclude with Section 5.

2. PATH FINDING METHODS

Path finding algorithms can be divided into two different types, static (or global) and dynamic algorithms. In a static algorithm, the environment is known and an optimal path is calculated before the object is moved. In a dynamic algorithm, the environment may not be completely known before the object starts moving, or it may change whilst the object is moving. In this case, the path plan has to be updated while it is being executed. Our demonstrator currently only uses static algorithms.

A vast amount of research has been done on path planning techniques, and the success of a particular technique relies on the environment and the constraints that are imposed on it. In this paper, the scope is limited to a domain where a military unit has to move, or be moved, safely to a destination whilst avoiding obstacles such as threats and structures. We assume that the terrain can be presented as a 2D grid or a graph. Tarapatta [2] gives an overview of techniques for terrain representation.

The most common approaches to solving path planning problems are based on Dijkstra's shortest path algorithm and the A* search algorithm. We report mainly on the A* search-based algorithms because they are most efficient.

The A* search algorithm is a graph-based algorithm that finds the least cost path between a source and destination node by minimising the estimated cost to the goal from the current node, as well as the cost of the path so far. It is regarded as one of the most efficient graph based path finding algorithms. It is a heuristic, optimal algorithm and also optimally efficient, i.e. at least as efficient in terms of node expansion as any other optimal algorithm. Consult [3] for more information.

One of the disadvantages of the A* search algorithm is that it requires a large amount of memory space. Path-finding on large maps can be problematic but many extensions of the algorithm address this deficiency by reducing search space, and time and memory requirements.

The A* search algorithm is formulated for a static environment but there is an extension for a dynamic environment, the D* algorithm. A* search-based algorithms are used in most electronic games for path finding.

There are a number of extensions of A* search algorithm such as Beam Search, Iterative Deepening, Bi-directional Search, Hierarchical Path Finding, Multi-resolution A* Search [4; 5], and Incremental Search [6]. Consult [7] for more information.

There are many other approaches that also proved to be successful for path planning, for example, probabilistic path planning, ant colony optimization, etc. Svestka and Overmars [8] give an overview of probabilistic path planning. Examples of successful probabilistic planners are the probabilistic path planner (PPP), planners using genetic algorithms, and the randomized path planner (RPP) [9].

Mora et al. [1; 10] have adapted ant colony optimisation (ACO) algorithms for the MUPFP with great success.

Very little work has been done in terms of formulating path planning as a CSP. Both Gualandi et al. [11] and Allo et al. [12] successfully used concurrent programming to address path planning for aircraft. We formulate the MUPFP as a CSP. In Section 3 we give an overview of constraint satisfaction techniques.

3. CONSTRAINT-BASED SOLVING

The modelling of problems in terms of constraints has the advantage of a natural, declarative formulation. When a problem is defined as a constraint satisfaction problem (CSP), it is a representation of *what* must be satisfied, without specifying *how* it should be satisfied.

A constraint is a logical relation involving one or more variables, where each variable has a domain of possible values. A constraint thus restricts the possible values that variables can have.

A CSP consist of a set of variables, a set of domains for the variables, and a set of constraints. Each constraint is defined over a subset of the set of variables. There are many general purpose techniques that can be used to solve CSPs, for example, integer programming, local search, and neural network techniques, but there is a special purpose technique that is widely used: tree search in conjunction with backtracking and consistency checking.

Constraint satisfaction techniques have proved to be effective to solve over-constrained problems, i.e. sets of constraints that cannot be satisfied. Soft constraint satisfaction techniques allow the user to identify subsets of constraints that are less important to satisfy, such that a suitable solution to an unsatisfiable problem can be found.

Consult Dechter [13] and Bartak [14] for an introduction to constraint satisfaction techniques

3.1. The MUPFP formulated as a Constraint Satisfaction Problem

The advantage of following a constraint-based approach to our path planning problem is the flexibility this framework offers. The MUPFP involves an environment and goals that can effectively be represented by constraints, and new information can be added with ease.

Our CSP formulation of the MUPFP is a grid-based approach, and it is in fact formulated as a constraint satisfaction optimisation problem. The terrain map is divided into a 2D grid where each cell has an associated cost. The cost is an indication of the difficulty or danger of moving though a particular cell. For example, if there is a structure in a cell through which a soldier cannot move, this cell will have a maximum cost value. The objective is to solve the CSP whilst minimising the total cost of all the cells in the solution path.

The terrain grid has associated cost values that are different for each scenario, for example, if the soldiers are dismounted, the cost values will differ from a scenario where the soldiers are transported in a vehicle.

The terrain map is divided into a grid where the coordinate pair (x_i,y_j) represents the cell in the i-th column and the j-th row assuming $(0,0)$ is in the lower left hand corner and (k,m) is in the upper right hand corner.

The set of variables is $V = \{V_1, V_2, \ldots, V_n\}$, where each variable represents one cell in the solution path. The domain of each variable is $Dom = \{(x_0,y_0),(x_1,y_0),\ldots,(x_k,y_0),(x_0,y_1), \ldots,(x_k,y_m)\}$. The set of constraints, C, contains at least the ones listed below:

- C_1: An all-different constraint: $V_1 \neq V_2 \neq \ldots \neq V_n$
- C_2: $V_1 = S$ and $V_n = D$ where S is the starting cell in the path and D is the destination cell.
- C_3: For every V_{i+1}, $i = 1,\ldots n\text{-}1$, if $V_i = (x_i,y_j)$ exactly one of the following should hold:
  $V_{i+1} = (x_{i+1},y_j)$, $V_{i+1} = (x_{i+1},y_{j+1})$,
  $V_{i+1} = (x_{i+1},y_{j-1})$, $V_{i+1} = (x_i,y_{j+1})$,
  $V_{i+1} = (x_i,y_{j-1})$, $V_{i+1} = (x_{i-1},y_j)$,
  $V_{i+1} = (x_{i-1},y_{j+1})$, and $V_{i+1} = (x_{i-1},y_{j-1})$.
- These constraints express the fact that a solution consists of a path in the search area.

Constraints to model threats, or any additional requirements, should also be added.

Each member of the set Dom, (x_i,y_j), with i=1,..,k and j=1,..m, has an associated cost value, $c(x_i,y_j)$. The solution to the problem is an assignment of values for the variables in the set V such that $min \sum_{s=1,\ldots n} c(x_i,y_j)$ where $V_s = (x_i,y_j)$.

3.1.1. Solving the CSP formulation of the MUPFP

A CSP has a finite, known number of variables. In our application we do not know in advance what the length of an optimal or good solution path is. We thus have to decide what the value of n (number of decision variables in the CSP) is before solving our CSP. We will start by calculating the theoretical shortest length a solution path can have (e.g. the minimum number of cells that have to be traversed to reach the destination node from the start cell). The CSP is solved for the initial value of n. Then we increase the value of n by 1 and solve the CSP again. Retain the best solution. Now we repeatedly solve the CSP for an increased value of n until a pre-defined time limit is reached or a pre-defined upper bound value for n is reached.

In this formulation of the problem, we have to decide on the cost function for the grid in advance. This cost function will be different for each environment, and will also depend on the mode of transport for the military unit. For example, the threat values and terrain difficulty will vary for dismounted units and units transported in a protected vehicle.

In order to optimise the total cost of a path, we will solve the problem with a branch and bound algorithm.

The main criterion that has to be optimised is the difficulty of moving through the terrain. We therefore regard the cost associated with each cell on the grid representation of the terrain, to represent the level of difficulty that elevation or an obstacle (that can be crossed) contribute. This will require a cost matrix to be set up in advance. Obstacles that cannot be crossed will be indicated in the threat matrix and handled via a constraint. In this first attempt, we will regard the costs and threats to be static.

We model known threats by adding additional constraints. In this phase of the simulation, these constraints represent the way in which threats should be avoided.

3.2. Constraint-based A* Search Formulation

Our extended A* search algorithm maintains a list of potential (partial) solutions. From this list the best potential partial solution is selected and expanded: a new waypoint is added to the end of this chosen solution path for every direction under investigation. Any new waypoint has to satisfy all the constraints.

4. THE MUPFP SIMULATION CONCEPT DEMONSTRATOR

Each of the following aspects of our simulation is discussed in one of the subsections that follow below.

- Modelling the Environment – modelling the different aspects of the environment, the structures, terrain and barriers.
- Agents - modelling own forces and enemy forces, and agent behaviour.
- Simulation Aspects.
- Algorithms.

4.1. Modelling of the Environment

The concept demonstrator allows any image to be imported as the area of operation. This allows for the flexible integration of Geographical Information Systems (GIS), satellite imagery or even Google Earth. In the example in Figure 1[1] we imported an image from Google Earth depicting a coastal area around the Red Sea. The next step is to indicate a starting and a goal point, and to model different terrain features which may be encountered.

The types of structures we model are (see Figure 2):

- Fields and other vegetation – these structures are passable both on foot and by vehicle.
- Hilly areas – a large hill or mountain may not be passable for a vehicle, but may be on foot.
- Built-up areas such as informal housing areas, residential areas and so on.

[1] In Figures 1 to 5, "Global B&B" should read "A* extension".

Figure 1: Modelling the Area of Operation

The important variables for structures are:

- Movement cost for dismounted soldiers – the cost in terms of the speed a person on foot achieves when moving through this type of structure.
- Movement cost for vehicles – the cost of a vehicle in terms of speed.
- Shape-polygon – describes the shape of a structure.
- Colour or texture – a colour is used to display the different types of structures. For example, fields are light green, built-up areas are light grey whilst building complexes are dark-grey.

A barrier is a kind of structure that is difficult to cross. Some barriers, suburban fences for example, are difficult to cross by vehicle, but easy to cross on foot. The aim of a barrier in the simulation is to show where it would be difficult to cross for a specific type of transport. Another example of a barrier line may be the contour line of a hill where the incline becomes so steep that a vehicle cannot maintain grip.

The important variables for barriers are:

- Movement cost for dismounted soldiers – the time that it requires to cross the barrier on foot.
- Movement cost for vehicles – the time it takes a vehicle to cross the barrier.
- Shape-polygon – describes the shape of the barrier.

Our constraint-based approach provides a great deal of flexibility in modelling structures and barriers. For example, the avoidance of a structure that cannot be crossed or a sniper's shooting range can be enforced by a separate constraint for that particular obstacle/threat, while the crossing of a hilly area by a dismounted soldier can be modelled by an increased cost.

Our model's parametric variables are populated by subject matter and map experts.

4.2. Agents

This simulation models all active "elements" as agents. In that sense, this is an agent-based simulation. That is, the underlying mechanics are agent-based in that each agent has certain goals that could be modified in the course of the simulation – based on the environment and actions by other agents. Dynamic aspects have not been implemented yet, for example, in the current instantiation of the demonstrator the enemy agents are not movement active; they do not move. Neither has the ability to change costs been implemented yet.

The friendly (own) agent may be an individual, but more likely, this will be a group of individuals or may even be a large group of individuals. We use the term Friendly Combatant (FC) to refer to this group – this is more intuitive than to constantly refer to friendly forces or own agent or friendly agent.

Our demonstrator has the following characteristics:

1. We model the movement of FC units from a certain point (the starting point), through a path that consists of a number of waypoints and ends at the goal (finishing point).

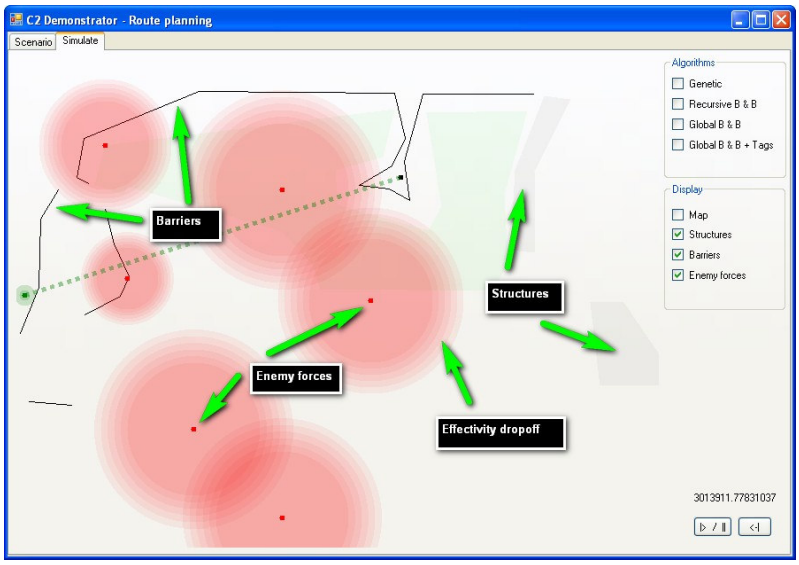

Figure 2: Modelling Structures, Barriers & Enemy Forces

2. The starting conditions are that FC units are initially in vehicles. They may however immediately abandon the vehicles and continue the planned path on foot.
3. The demonstrator is designed to investigate two questions: firstly, what path should FC units follow to the target, and secondly, at what point should the vehicle be abandoned (if at all).
4. The terrain influences movement ability, slowing down movement from the optimal speed.
5. The terrain may consist of different constructions, such as fields / woods / grasslands and also buildings / informal housing areas / building complexes, as well as barriers that may be passable by foot but not by vehicle. Barriers also take additional time to cross.
6. We want to find the fastest path, that is, the path that, given the terrain and type of transport will get FC units to the target in the shortest time.
7. Constraints: As described earlier a number of constraints must be implemented and this bounds the solution space that will be investigated.
8. For the purpose of this simulation, i.e. path planning, own agents do not carry guns and will not shoot – the aim being to find a path that will not put them in danger. However, it is quite trivial to change the behaviour of own forces agents to also include shooting ranges.

Each agent has a base position: the position where the agent is located at the beginning of the simulation. In addition, an agent also has a current position which is the location that it occurs at a certain instance in time within the simulation. Each agent also has a characteristic speed on foot. This speed is based - for example - on the type of equipment that the agent has to carry.

The prime property of an agent is that it has goals, and plans to achieve those goals. An (own) agent's basic goal is defined as a position that it wants to reach. For an enemy agent it may be a better sniping location, or to intercept FC units. In order to achieve a goal, agents will construct plans. A plan consists of a number of waypoints. The agent will move to each of the waypoints in succession.

For FC units there are also the matter of when to abandon the vehicle. Therefore, a part of a FC unit's plan is at what point along the planned path the vehicle will be abandoned.

These variables reflect the *properties of the agent*:

- Base Position – The initial position of the agent at the beginning of the simulation.
- Current Position – the position of the agent at a specific time within the simulation.
- Speed on foot.
- Colour – a representation colour for the agent. In later versions this may be replaced by an image.

The following variable is used to model the *agent's goals*: Position of goal – this is the final goal for this agent.

The following variables are used to model the *agent's planning* to achieve the goal:

- Planned path – the planned path consists of a number of positions. The agent will move to each of these positions sequentially.
- Where to abandon the vehicle – the agent's current plan is to abandon the vehicle he starts his journey with at some point along the planned

Figure 3: Modelling Active Elements

path. It may be that the agent abandons the vehicle at the starting position, implying that the vehicle is never used.

Currently, we implement only one type of enemy agent, the sniper. A sniper will be stationary in a specific location for a very long time, waiting, observing and striking at his specific target from his hidden position. A sniper has a specific shooting range. We model a sniper with a sniping range and a fall-off range of 1.5 times the sniper range. Within the fall-off range the sniper may still take a shot, but the accuracy is not guaranteed.

Visually sniper locations are indicated by red dots as shown in Figure 2 and Figure 3. The sniping range is shown as a light red circle and the fall-off range by more transparent circles as seen in Figure 2.

4.3. Simulation Aspects

There are two methods of handling such simulations; the first is a time-step based simulation where the simulation is conducted in time-increments. The second method is an event-based simulation where time-stepping is done forward to the next event. This requires an accurate prediction of the next event, which may not be easy.

In order to keep the model simple (this is the first version) the approach followed here is a time-step based simulation. The chosen time-step is one second, although this is completely arbitrary – one second will move the agents about one meter if moving on foot.

4.3.1. Agent Movements

An agent has a planned path and the agent will move along the planned path. FC units are shot at if within the gun range of a sniper. The simulation tests if FC units are moving within any of the structures. For this purpose, we use an efficient point within-polygon algorithm. Finally we also use an algorithm for testing line-segment crossing.

4.3.2. User Interface – Visualisation

The most important parts of the visual interface that have been implemented for this version of the demonstrator have already been illustrated – Figures 1 to 5. Figure 2 demonstrates some of the passive elements: barriers, structures and enemy forces with effective range circles. Figure 3 shows some of the active elements pertaining to the different solutions: the current best path, the current path under investigation, and a mesh of the search-space covered. The mesh of the covered search space is solver-dependant. The branch and bound algorithm, for example, uses a grid that can be displayed to show the searched area.

Both the best path and the current path under investigation show the path travelled by vehicle as solid lines and the part travelled by foot as dotted lines.

4.3.3. Memory Management and Prevention of Race-Conditions

The implementation of simultaneous solvers (algorithms) solving the same problem, using the underlying simulation architecture in an asynchronous fashion will lead to race conditions. There will also be race conditions between the user interface aspects and the underlying solver architectures. For the current demonstrator, three situations where race-conditions may occur are:

- The simulation itself. Each solver will invoke the simulation for each of the potential solutions. Therefore a direct race-condition exists on the simulation and care should be taken to prevent this. Three possibilities exist: use locking mechanisms to allow only a single solver access

Figure 4: The Branch and Bound Algorithm

to the simulation at a time, duplicate the simulation environment for each solver or, the third option, keep all changeable variables in the thread-stack and within the proposed solution.

- The graphical user interface. The GUI has to draw the current best solution, but during the time that it takes to draw it, it may change. In addition, if a copy is made, the copy has to be a deep-copy, that is, all underlying structures must also be copied. This simulation opts to use a lock on the current best-solution variable.
- The best solution that is shared between the different solvers. This solution must be updated within each solver and a race condition may exist if a solver makes a new best estimate simultaneous with another solver. We use locks on the update of the best solution.

4.4. Path finding Algorithms

The demonstrator also includes an architecture that allows for multiple path finding algorithms to be used simultaneously. The first algorithm is a constraint-based branch and bound algorithm (see Figure 4) and the second one is a constrained-based A* search algorithm (see Figure 5). The third algorithm, genetic hill-climbing, is not optimal but produces feasible solutions very quickly. It uses random changes in an attempt to improve on an existing solution, and can be used as an initial upper bound for the other two algorithms which allows for improved pruning of the search space when these algorithms are applied.

The architecture that was chosen for this to work is to create a solver abstract class that implements a number of standard solver functions. Each solver is then a derived instance of this class. Each solver will run its own thread. Care should be taken to synchronise the threads if multiple solvers can update the best solution found.

The main solver thread is quite simple: instantiate the solvers, start a thread for each solver, periodically check if any of the solvers have found a new best solution.

In the user interface as shown in Figure 3, the current best found solution to the MUPFP is displayed as a “green” path. The three different solvers are shown and the black dots show the search space that has been searched by the different algorithms.

The simulation produces optimal solutions or near-optimal solutions reasonable quickly.

5. CONCLUSIONS

We describe the implementation of an initial phase of a simulation concept demonstrator for military path finding. This concept demonstrator can be used to determine an optimal path for soldiers to follow during a military operation, as well as the best mode of transport. The path is optimal in terms of user-specified constraints relating to the operational terrain and the safety of the troops. The user can also get a near-optimal solution if the available time is limited, by halting the solvers prematurely. A tactical mission planner can use this demonstrator to investigate different scenarios.

The simulation uses three different path finding algorithms: a constraint-based branch and bound algorithm, a constraint-based extension of an A* search algorithm and a hill-climbing algorithm.

The main advantage of our constraint-based approach is the ease of modelling new environments or additional constraints.

We plan to extend this simulation to cater for dynamic changes in the environment.

References

[1] Mora, A.M., Merello, J.J., Millan, C., Torrecillas, J. and Laredo, J.L.T., CHAC. "A MOACO Algorithm for Computation of Bi-Criteria Military Unit Path in the Battlefield", Proceedings of the Workshop on Nature Inspired Cooperative Strategies for Optimization, 2006.

[2] Tarapata, Z., 2003. Military Route planning in Battlefield Simulation: "Effectiveness Problems and Potential Solutions". Journal of Telecommunications and Information technology, 4.

[3] Russel, S. and Norvig, P., 1995. *Artificial Intelligence: A Modern Approach*. Prentice-Hall.

[4] Wichmann, D.R. and Wuensche, B.C., "Automated Route Finding on Digital Terrains", International Image and Vision Computing New Zealand Conference, 2004.

[5] Holte, R., Perez, M., Zimmer, R. and MacDonald, A., "Hierarchical A*": Searching Abstraction Hierarchies Efficiently, Proceedings of AAAI-96, 1996.

[6] Koenig, S., Likachev, M., Liu, Y. and Furcy, D., 2004. "Incremental Heuristic Search in Artificial Intelligence". Artificial Intelligence Magazine, 25(2).

[7] Patel, A., *Patel's A* pages*. Available: http://theory.stanford.edu/~amitp/GameProgramming/

[8] Svestka, P. and Overmars, M.H., 1998. "Probabilistic Path Planning". In: J. Laumond, ed, Robot Motion Planning and Control,. Springer-Verlag, pp. 255-304.

[9] Barraquand, J. and Latombe, J.C., 1991. "Robot Motion Planning: A Distributed Representation Approach". International Journal Robotics Research, 10(6).

[10] Mora, A.M., Merelo, J.J., Laredo, J.L.J., Castillo, P.A., Millan, C. and Torrecillas, J., "Balancing Safety and Speed in the Military Path Finding Problem: Analysis of Different ACO Algorithms", Genetic and Evolutionary Computation Conference, July, 2007.

[11] Gualandi, S. and Tranchero, B., "Concurrent Constraint Programming-based Path Planning for Uninhabited Air Vehicles", Proceedings of SPIE's Defense and Security Symposium, 2004.

[12] Allo, B., Guettier, C., Legendre, V., Poncet, J.C. and Strady_Lecubin, N., "Constraint Model-Based Planning and Scheduling with Multiple Resources and Complex Collaboration Schema", AIPS, 2002.

[13] Dechter, R., 2003. *Constraint Processing*. San Fancisco: Morgan Kaufman Publishers.

[14] Bartak, R., *On-line guide to constraint programming,* http://kti.mff.cuni.cz/bartak/constraints.

Biography

Louise Leenen is a Senior Researcher in the Command, Control and Information Warfare Competency Area of the CSIR, South Africa. Her main interest is in the solution of combinatorial problems.

Johannes Vorster is a Senior Researcher in the Command, Control and Information Warfare Competency Area of the CSIR, South Africa. His main research interest is in Cyber Defence.

Herman le Roux is a Principal Researcher in the Command, Control and Information Warfare Competency Area of the CSIR, South Africa. His main area of interest is Command and Control Simulation Technologies.

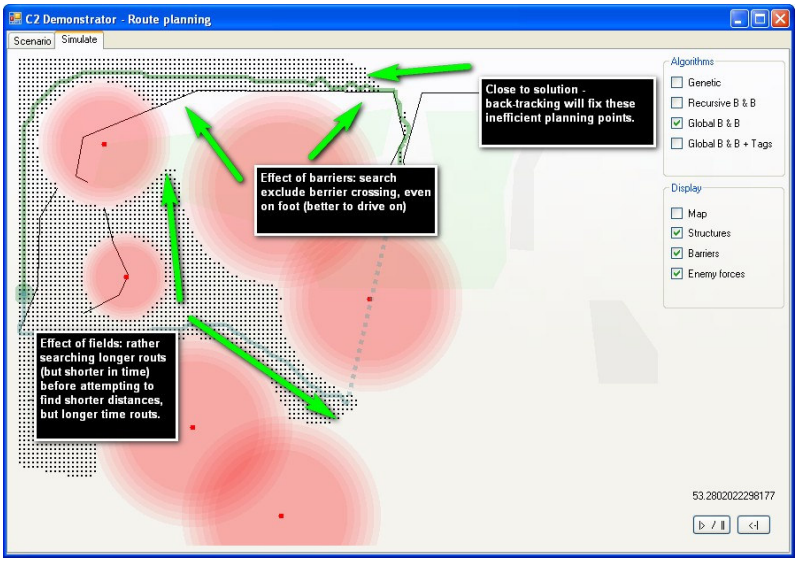

Figure 5: The A* Search Based Algorithm