Continuous Target Monitoring using Multiple Unmanned Aerial Vehicles

Beulah H. Moses

B.Engineering (Electronics and Communication Engineering)., M.Tech (InformationTechnology)., MBA (Computer Systems)

Submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy in Computer Systems Engineering

December 2009

School of Electrical and Information Engineering



University of
South Australia

University of South Australia

UNIVERSITY OF SOUTH AUSTRALIA
LIBRARY

Table of Contents

Abstract iv

Declaration v

Acknowledgements vi

List of Figures vii

List of Tables x

List of Authors publications xi

1 Introduction and Background 1

1.1 Motivation of this research 1

1.2 Application Domain 2
1.2.1 Unmanned aerial vehicles (UAVs) 2
1.2.2 Unmanned ground vehicles 2
1.2.3 Unmanned underwater vehicles 3

1.3 Applications 3
1.3.1 Continuous target monitoring in military 3
1.3.2 Bushfire monitoring 4
1.3.3 Landmine detection 5
1.3.4 Unmanned vehicle control for security 5
1.3.5 Reconnaissance 5
1.3.6 Surveillance tasks 6
1.3.7 Tracking items in a warehouse or factory 6
1.3.8 Tracking people in search and rescue 6

1.4 Background 7
1.4.1 Multi-agent Simulation toolkit (MASON) 7
1.4.2 Mobile ad hoc networks and Path finding 7
1.4.2.1 Ad hoc On - Demand Distance Vector (AODV) 12
1.4.2.2 Dynamic Source Routing Protocol 16
1.4.2.3 Optimised Link State Routing Protocol 18
1.4.2.4 Zone Routing Protocol 20
1.4.2.5 Comparison of Routing Protocols 21

1.5 Combinatorial Optimization Techniques 22
1.5.1 Hill climbing algorithm 22
1.5.1.1 Hill climbing characteristics 24
1.5.2 K means clustering 25
1.5.3 Reinforcement Learning 27
1.5.4 Genetic Algorithm 28

1.6 Co-operation and co-ordination 30

1.7 Thesis statement 34

1.8 Thesis Organisation 34

2 **Annealing Algorithms** **36**
2.1 Annealing 36
2.2 Boltzmann Annealing Algorithm 37
2.3 Simulated Quenching 38
2.4 Simulated Annealing 38
2.5 Cooling Schedule 40
2.5.1 Selecting cooling schedule 41
2.5.1.1 Starting and Final temperature 41
2.5.2 Types cooling schedules 41
2.5.2.1 Johnsons cooling schedule 42
2.5.2.2 Lundy- cooling schedule 43
2.5.2.3 Adaptive cooling based on system scale of time and energy 45
2.5.2.4 Simple time schedule 46
2.5.2.5 Exponential schedule 47
2.6 Conclusions on annealing algorithms and cooling schedule 48
3 **Path Finding and Search Algorithms** **49**
3.1 Networks and Graphs 49
3.1.1 Shortest Path 50
3.2 Shortest Path Algorithms 54
3.2.1 Applications of the shortest path algorithms 55
3.2.1.1 Transportation Planning 55
3.2.1.2 Knapsack problem 55
3.2.1.3 Travelling sales man problem (TSP) 56
3.2.1.4 Production lot sizing 56
3.2.1.5 Routing in telecommunication networks 57
3.3 Pathfinding and search algorithms 57
3.3.1 Distance vector algorithm 58
3.3.2 Dijkstra's Algorithm 60
3.3.3 Floyd-Warshall's algorithm 62
3.3.4 Johnson's algorithm 63
3.3.5 Best-first search 64
3.3.6 Breadth-first search (BFS) 65
3.3.7 Depth-first search (DFS) algorithm 66
3.3.8 A star algorithm 67
3.4 Conclusion and comparison of shortest path algorithms 69
4 **Co-operative target observation and new simulated annealing** **72**
4.1 Networked Team Control 73
4.1 Co-operation, co-ordination and collaboration in unmanned vehicles 75
4.1.1 Co-operation 75
4.1.2 Collaboration 77
4.2 Related work 79
4.3 Simulated annealing (SA) 80
4.3.1 New simulated annealing algorithm 81
4.4 Model description 82
4.4.1 Principle of operation and block diagram 84

5 Temporal Difference Learning and Simulated Annealing for Optimal Control 90

5.1 Temporal Difference Learning 91

5.2 Simulated Annealing 93

5.3 Details of Simulation Environment and Algorithms 94

5.4 Combination algorithm 95

5.5 Softmax or Boltzmann distribution Action Selection. 99

5.5.1 The ε greedy Action Selection with SA. 99

5.6 Improvements 100

6 Findings, conclusion and future work 101

6.1 Findings and conclusion 101

6.2 Conclusion 109

6.3 Future work 109

References 111

Appendix 1 Abbreviations 125

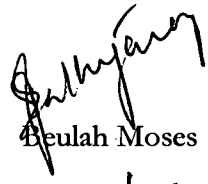
Abstract

Cooperative target observation and target tracking can be used to study multi-agent coordination, planning, and robot control from, dynamic multi-agent interaction and its emergent behaviour. These systems could be used for: unmanned vehicle control for security, reconnaissance, and surveillance tasks; tracking items in a warehouse or factory; tracking people in search and rescue. This research focuses on the various techniques used to search and monitor targets. Simulated Annealing (SA) is motivated by an analogy to annealing in solids. Metropolis's algorithm simulated the material as a system of particles. The SA algorithm simulates the cooling process by gradually lowering the temperature of the system until it converges to a steady, frozen state. Simulated annealing is a Monte Carlo approach for minimizing evaluation functions without getting stuck to the local minima like the Hill Climbing algorithm. A new simulated annealing algorithm is developed from the simulated

Teams of cooperating UAVs have the potential to: accomplish the missions in a shorter period, simultaneously accomplish many goals, cost less, be less detectable and more survivable than a single large vehicle; damage to a single UAV does not necessarily cause the entire mission to fault. The higher degree of coordination helps us to achieve better overall efficiency even though the Unmanned Aerial Vehicles usually have limited sensor capabilities relative to their large counterparts. The accomplishment of goal therefore relies heavily on cooperation between appropriate neighbours. Cooperative target observation is a very good example for study of multi agent cooperation. We use a new simulated annealing for coordination and cooperation of unmanned aerial vehicles to continuously monitor a group of moving targets. We compare with Hill Climbing algorithm and the new simulated annealing algorithm and find that the new simulated annealing algorithm is superior for almost all target speeds, UAV sensor ranges and various group sizes. The fact that the Hill Climbing Algorithm is stuck up in local minima is indeed overcome by new simulated annealing algorithm. We found that the UAV is placed as close to as many targets as possible so that the UAV can move the target whenever the need arises.

Declaration

This thesis contains no material which has been accepted for the award of any other degree or diploma in any university or other institution and affirms that to the best of my knowledge, the thesis contains no material previously published or written by another person, except where due reference is made in the text of thesis.



Beulah Moses

5/05/2010

Acknowledgements

I would like to express my gratitude to all those who gave me the possibility to complete this thesis. My sincere heartfelt thanks and gratitude go to my principal supervisor Professor Lakhmi Jain. This research would not be possible without his immense support and encouragement.

I am deeply indebted to my co-supervisors Dr. Anthony Finn and Dr. Sam Drake, Defence Science and Technology organisation for their continual guidance, stimulating suggestions and encouragement they had given me in completing the thesis. This research would not be possible without their extremely valuable contribution. I am indeed privileged to be one of their research students.

I am thankful to the University of South Australia and commonwealth government for the support they had provided me with the prestigious International postgraduate research scholarship funded by the and for providing me with the living allowance which made it possible for me to complete my thesis. I am grateful to our Dean of Research and Head of School for providing monetary help for my twenty month old son to be in child care.

I wish to thank all my friends Stephen Franklin, Jinsong Leng, Gwendolin Doss, Tess Manego and Joy Milne for their encouragement during the much stressed period of my PhD.

Finally, I would like to express my deepest gratitude to my family members- mother Hannah Charles, my late father Arnold Charles who passed away in early 2007, my sisters Esther, Rachel, Hephzibah and my brothers Emmany and Abe for their immense support, encouragement and love. I am so thankful to my wonderful boys Joshua and David for their wonderful co-operation.

List of Figures

Figure 1:1 Co-operative group of unmanned aerial vehicles 3

Figure 1:2 UAVs in bushfire monitoring 4

Figure 1:3 Simple Network 8

Figure 1:4 Bandwidth and time delay in ad hoc network 11

Figure 1:5 Propagation of RREQ 13

Figure 1:6 Propagation of RREP 13

Figure 1:7 OSLR Packet Transmissions 19

Figure 1:8 OSLR Neighbours 19

Figure 1:9 ZRP Routing 20

Figure 1:10: Hill climbing algorithm 23

Figure 1:11 Hill climbing algorithm issues 23

Figure 1:12: Hill climbing algorithm for co-operative target observation 24

Figure 1:13 K means clustering demonstration 25

Figure 1:14 Genetic algorithm components 29

Figure 1:15 Radio leashing for communication 31

Figure 1:16 Network without connectivity constraints 31

Figure 1:17 Network with connectivity constraints 32

Figure 2:1 Issues in Hill climbing algorithm 39

Figure 2:2 Probability of acceptance 40

Figure 2:3 Cooling schedule 42

Figure 2:4 Johnson's cooling schedule 43

Figure 2:5 Lundy's cooling schedule 44

Figure 2:6 Lundy's average cooling schedule 44

Figure 2:7 Adaptive cooling 45
Figure 2:8 Simple schedules 46
Figure 2:9 A simple cooling schedule 46
Figure 2:10 Exponential schedule 47
Figure 3:1 A simple cyclic network 49
Figure 3:2 An example network 50
Figure 3:3 Directed graph 51
Figure 3:4 Various paths from source to destination 52
Figure 3:5 Weighted graph 53
Figure 3:6 Adjacency matrix 54
Figure 3:7 Path finding matrix 54
Figure 3:8 Travelling sales man directed and undirected 56
Figure 3:9 Telecommunication network as a graph 57
Figure 3:10 Bellman Ford algorithm 59
Figure 3:11 Dijkstra algorithm 61
Figure 3:12 Floyd Warshall algorithm 63
Figure 3:13 Johnson's algorithm 63
Figure 3:14 Breadth-first search 65
Figure 3:15 Depth-first search 66
Figure 3:16 A star algorithm 67
Figure 3:17 Comparison of computation time 69
Figure 3:18 The time for search increases as the number of nodes increases 70
Figure 4:1 Co-operation and collaboration 75
Figure 4:2 Co-operation of UAVs for target observation 76
Figure 4:3 Importance of decentralised observation 77
Figure 4:4 Collaboration with human intervention 78
Figure 4:5 Tasks in successful collaboration 78

Figure 4:6 Simulation environment for co-operative UAVs 83

Figure 4:7 Basic building block for new simulated annealing in target observation 85

Figure 4:8 Sequence diagram for the sequence of events in co-operative target observation 87

Figure 4:9 Exponential cooling schedule used in simulation 87

Figure 4:10 Cooling schedule with various values of α 88

Figure 5:1 Basic simulation annealing algorithm 94

Figure 5:2 Ball Interception problem 95

Figure 5:3 Sarsa (λ) with modified simulated annealing algorithm 97

Figure 5:4 Cooling schedule used in this algorithm 97

Figure 5:5 Comparison of combined algorithm with epsilon greedy algorithm 98

Figure 6:1 Low target speed and varying group size 102

Figure 6:2 High target speed and varying group sizes 103

Figure 6:3 Small group size 104

Figure 6:4 Large group size 104

Figure 6:5 Low target speed and varying sensor range 105

Figure 6:6 High target speed and varying sensor range 105

Figure 6:7 Small group size and varying sensor range 106

Figure 6:8 Centralised group with varying sensor range 106

List of Tables

Table 1:1 End to end delay in ad hoc networks due to delay..... 11

Table 2:1 Simulated Annealing Analogy 40

List of Authors publications

[1] Moses, B., Jain, L.C., Finn, A. and Drake, S. Multiple UAVs Path Planning Algorithms: A Comparative Study, Journal of Fuzzy Optimization and Decision Making, Kluwer, Volume 7, 2008, pp. 257-267.

[2] Moses, B. and Jain, L.C., Cooperative Target Observation of UAVs using Simulated Annealing, International Journal of Intelligent Defence Support Systems, Volume 1, No 2, 2008, pp. 116-129.

[3] Leng, J., Sathyaraj, B.M. and Jain, L.C., Temporal difference Learning and Simulated Annealing for Optimum control: A Case study, Lecture Notes in Artificial Intelligence, LNAI 4953, KES-AMSTA 2008, Springer-Verlag, Germany, 2008, pp. 495-504.

[4] Sathyaraj,B.H and Doss, R.C., "Route maintenance using mobility prediction for mobile ad hoc networks" IEEE MHWMN'05 Nov 2005.

1 Introduction and Background

Autonomous mobile vehicles have caused a lot of impact in the field of Artificial intelligence recently. This is due to the wide range of applications and the saving of human lives. The applications are numerous in both civil and military field. The unmanned vehicles are used in the hostile regions in the context of military surveillance, border security, target monitoring, attack in war field, information collecting resources, rescue operations etc. The major objective of such exploration is for continuous monitoring of targets. In our case, the basis for target observation is based on the location, speed and type of targets involved. In the case of dynamic objects the changes over time also need to be tracked [2]. These essentially constitute localization, mapping, moving object tracking for both trajectory and speed estimation. The sensing range of unmanned vehicles vary, but when it works in a team or groups it is able to perform much better than if a single big counterpart would perform.

1.1 *Motivation of this research*

Automation is taking over the other conventional methods as it involves less human resources and makes smart decisions. The skilled human resources are falling in the defence forces and is expected to fall still further in the coming era [5]. The need to maintain operational effectiveness means that some work will require unmanned ground vehicles, unmanned aerial vehicles and unmanned ground vehicles to act autonomously by collaborating and cooperating. When a single skilled human supervisor, the performance in decision making is improved to a great extent. than with all vehicles with humans, or with no human intervention.

Having an accurate position of the targets are very important in decision making. The decision making can be human assisted one or can be autonomous when there is no communication involved. Position of the target and the speed of movement of the target are some of the factors that influence decision making

and it is used in the various applications such as air traffic control [12], military command and control operations [12, 15].

1.2 *Application Domain*

1.2.1 Unmanned aerial vehicles (UAVs)

UAVs are aircrafts with no onboard pilots. UAVs may be either remotely controlled or others fly autonomously based on pre-programmed flight plans using more complex dynamic automation systems. The autonomy of the UAV varies from simple radio to sophisticated artificial intelligence optimization techniques. The UAVs are cheaper than the normal aircrafts as there are no humans and therefore can be smaller and does not need life supporting systems. UAVs work in a team to achieve a goal. The advantages of coordinating and collaborating UAV teams are: to accomplish the missions in a shorter period, to simultaneously accomplish many goals, teams of small aircraft can be both cheaper and less detectable than a single large vehicle; damage to a single UAV does not necessarily cause the entire mission to fault.

1.2.2 Unmanned ground vehicles

The ground vehicles with no human are unmanned ground vehicles. These UGVs may be remotely operated or autonomous. These vehicles can be used in co-ordination with the UAVs to do any mission. These UGVs are used not only in military applications, but any hostile regions without the fear of human life. The remotely operated UGVs are controlled by a remote supervisor. However the autonomous UGVs are designed for decision making on its own. When it is not possible for the supervisory station to be in continuous contact with the UGV, then the UGV should be able to make decisions and actions.

1.2.3 Unmanned underwater vehicles

The UUVs are used in both military and civil applications. The military applications include: o monitoring targets under water, monitoring other UUVs, for security of border of a nation from intruding objects from foreign nations. They are also used in other applications such as marine life observation.

1.3 *Applications*

1.3.1 Continuous target monitoring in military

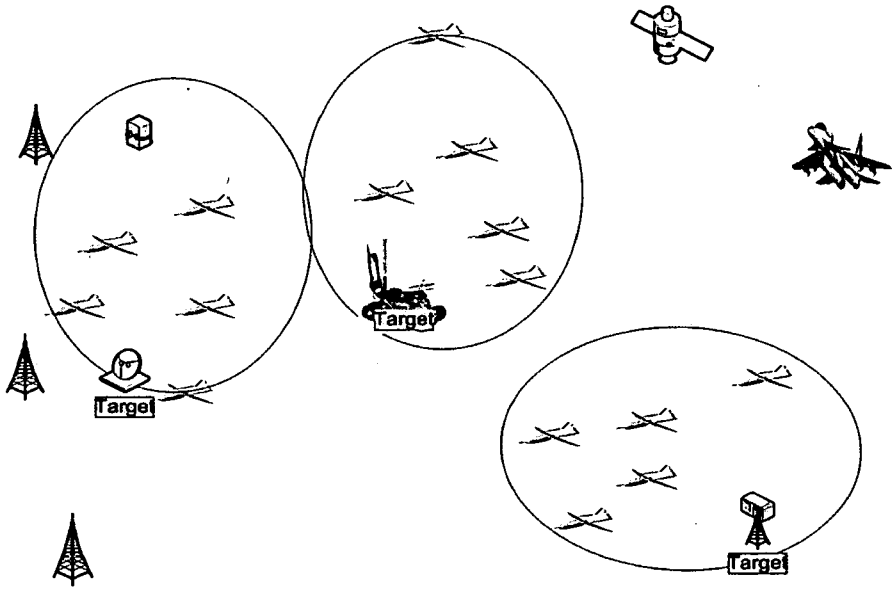


Figure 1:1 Co-operative group of unmanned aerial vehicles

Co-operation is desired collective effect through execution of individual actions of the system components. Co-operative monitoring increases: the reliability, robustness and distributed sensing, stationary and moving targets monitoring, continuous monitoring, optimised position, overall efficiency, cost saving, time saving, survivability, and detectable range.

1.3.2 Bushfire monitoring

Catastrophic bushfires has sparked researches for the development of a fully integrated airborne fire surveillance and response system using UAVs to provide early detection of hotspots to avoid the loss of many lives and properties. This could facilitate the water bombing aircraft in smoke obscured conditions. Many researches are being done in this area for early detection to avoid and reduce damages.



Figure 1:2 UAVs in bushfire monitoring

The larger region is divided into a grid or smaller region. Each small region has a monitoring point. These monitoring points are unmanned vehicles which survey the area. Some monitoring points need to be surveyed more often than others because they are more fire-prone. Some regions where lakes or rivers are found are not monitored as much as the dry, bush-filled gullies in high lightning-strike areas.

The aim of employing UAVs and supervising points are to survey all the points as close to the desired frequency as possible. Each UAV has one or more surveying region. Although central control of UAVs is feasible in the supervisory point, in real terrain communications may be too unreliable for this to be effective so the UAVs have to decide and take actions accordingly by themselves.

1.3.3 Landmine detection

UGVs and UUVs in co-ordination can be used in landmine detection[84. They can use various sensing devices which can differentiate landmines from the other terrain rocks or any similar objects. When the landmine detection is done with unmanned vehicles, many human lives are saved. This application involves large extent of autonomy with built in computer algorithms with the input data from various sensing devices. Unmanned vehicles are also used in the recovering, clearing and destroying of landmines.

1.3.4 Unmanned vehicle control for security

Border agents [84] on patrol, stationery surveillance equipments are used as traditional method of border security of a nation. However, UAVs with larger sensor range is a very significant asset when compared to the traditional surveillance methodologies. If an illegal border entrant attempts to transit through borders, through ocean, dense woods or mountainous terrain, UAVs would have a greater chance of tracking the intruder with thermal detection sensors. The extended range and endurance of UAVs can be achieved by employing them in groups or teams. This significantly reduces the burdens on human resources and human lives at the borders.

1.3.5 Reconnaissance

Reconnaissance is a military application. It is the active seeking to determine a hacker's or enemy's intentions to collect information about the environmental conditions, border safety, terrain condition, etc. This generally employs soldiers or combats trained for critical or emergency situations. However, UAVs are employed replacing the soldiers to save human resources and also human lives. The intelligence, sensor range and sensing quality of UAVs plays a vital role in reconnaissance.

1.3.6 Surveillance tasks

Coast watch programs [163] and aerial surveillance is gathering information or payload from the sensors capturing visual imagery, video from airborne vehicle like the UAVs or other traditional helicopters. Many computer algorithms and sophisticated hardware in the UAVs enable to capture images and videos and other information. When UAVs are involved in surveillance human resources is not wasted. The loss of human lives are also reduced specially in the hostile regions where humans cannot go.

1.3.7 Tracking items in a warehouse or factory

Automated guided vehicles (AGVs) [164] increase the overall performance and also reduce the costs as the system is automated for a warehouse. AGVs can track items, relocate items, carry loads or tow objects with trailers attached to them when needed. The trailers can be used to move raw materials to the final product. AGVs are utilized in many industries, including, pulp, paper, metals, newspaper, and general manufacturing.

1.3.8 Tracking people in search and rescue

UAVs are employed in tracking missing people in military and civil applications. When the terrain is not safe for human or dogs to explore, then UAVs are the only other option. It is better than the human patrols or dogs when the search is a life saving one. It can use tele-operated or autonomous vehicle to track people in rescue operations.

1.4 *Background*

1.4.1 Multi-agent Simulation toolkit (MASON)

MASON [30] is a single-process discrete-event simulation core and visualization toolkit. The simulator is written in Java and designed to be flexible enough to be used for a wide range of simulations. Some of the various simulations include: network intrusion and countermeasures, co-operative target observation using unmanned aerial vehicles, ant foraging, anthrax propagation in the human body and wetlands: a model of memory and primitive social behaviour.
The features of MASON include:

- speed
- portable
- capable of checkpointing
- restarting models with or without visualization
- able to migrate models across platforms
- capable of producing guaranteed results independent of platform
- good GUI with 2D and 3D using Java 3D
- screenshots and movies can be generated
- graphs can be generated from the simulation results

1.4.2 Mobile ad hoc networks and Path finding

Ad hoc wireless networks have mobile nodes which are interconnected by multihop communication paths. The ad hoc networks do not have fixed network infrastructure or any administrative support whereas the conventional wireless network does have fixed network infrastructure. When the mobile nodes join the network or depart the network, the radio links between the links becomes unstable and therefore, the topology of the network changes dynamically. The ad hoc wireless networks are self creating, self organizing, and self managing by itself. The technique of effective routing is one of the great challenges for networking using ad hoc while there are numerous challenges than can be overcome by using it. This includes effective routing, mobile management, power management,

channel access, quality (QoS) issues , issues pertaining to delays, security and bandwidth management. [122]

The nodes in the ad hoc networks communicate directly with each other in a peer - peer fashion in the absence of the fixed infrastructure networks. In the example given below, between nodes A and F or A and C, there is no direct radio channel. In order to communicate between A and F & A and C, the nodes B, D, E or G serves as an intermediate router. The ability of nodes to function on the basis of routers in demand is the distinguishing feature of ad hoc networks.

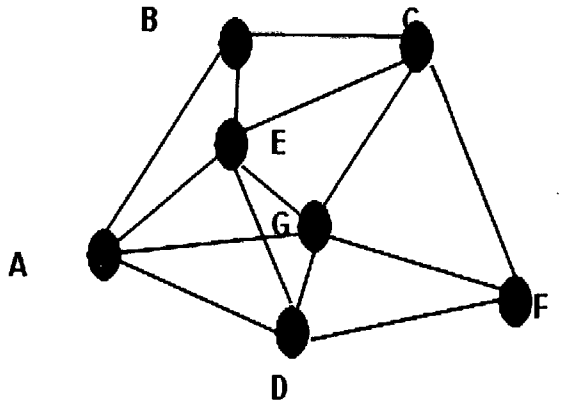


Figure 1:3 Simple Network

A mobile ad hoc wireless network varies from conventional wireless networks. Some of the characteristics of ad hoc networks [73] are

- Infrastructure-less
- uses multi-hop links for connecting
- Distributed routing as there is no centralized authority for network control
- Frequent changes in paths and topology due to mobility
- Mobile hosts require more intelligence to route the message
- Time synchronization is difficult as it consumes larger bandwidth

The mobile ad hoc networks are multihop wireless networks which changes dynamically and it consists of group of mobile nodes established on a shared wireless channels. It does not have fixed infrastructure, and is characterized by dynamic topologies, variable capacity links, limited physical security, and bandwidth and energy constraint operations. The study of routing in ad hoc networks has been carried out in the past few years [112-114] and many proposals of routing protocols have been given in ad hoc networks. Most routing protocols are designed for homogenous mobile networks, and the nodes have similar characteristics like same transmission power (range), transmission data rate, reliability, processing capability and security level. The homogenous mobile networks has poor scalability, and it has been demonstrated by using theoretical analyses, simulation experiments, and test bed experiments which proves that when the nodes in the network increases, the performance is degraded[108]. In many ad hoc realistic networks, the nodes are heterogeneous in nature [116-120]. For an example, in a battle field, the soldiers carry the wireless networks, where as the communication devices are carried by vehicles, tanks, and satellites; these nodes or devices have different characteristics in communication with respect to the transmission power, processing capability, data rate, reliability etc. Therefore, it is more appropriate to model these network elements in a more realistic way as there are different types of nodes. There are many advantages in modeling the nodes as different types in order to design more efficient routing protocols. In the overall performance; the communication and control systems, the latencies due to routing, and channel access play major roles in the ad hoc networks. In UAV networks, the task of finding paths and maintaining routes in ad hoc networks are non trivial because of the mobility of the vehicle and it results in unpredictable changes in topology. In order to reduce the communication overhead for global updates, due to the mobility, the practical routing algorithms for mobile ad hoc networks should be localized. As a result of this, more localized routing algorithms have been proposed recently. [82-84].

MANETS is a self organising mobile networks and it does not require a pre - existing infrastructure. Each node acts as a sender, a receiver and a router at the same time by itself. The devices that uses wireless connectivity such as mobile, laptops, PDAs , pocket pc are widely used. The two nodes that are in the same transmission range can communicate each other directly or else, they can

communicate using multi - hop routes. The range of applications of MANETS includes battlefields, disaster relief, and crowd control. In MANETS, routing and multi casting are most challenging tasks. The network topology changes unpredictably, often when the nodes in the network keep moving. MANETs is the key in the evolution of the wireless networks. The ad hoc networks communicate using wireless links without central control as these networks comprises of nodes that are equal. However, wireless ad hoc networks are noted to have the problems of traditional and mobile communications like configuration advertising, self - routing, ad hoc addressing, and discovery and maintenance.

The routing protocols have been researched for the mobile ad hoc networks recently. The research is carried to find a feasible route from source to destination in which network traffic and application requirements are not taken into account. It has been predicted that there might be overloading of traffic and the network performance cannot be improved in any current network condition. This might be considered for data transfer but the real time applications prefer to have Quality of Service (QoS) from the networks. The QoS can be obtained by the finding the route according to the system requirements or giving application, the network feedback when the requirements are not met. A judicious routing protocol is needed to support the ad hoc wireless mobile computing networks more efficiently.

Consider the nodes A, B,C,D, E, F, G and H in a group. Consider the available path from E to H. The three routes available are Path1, Path 2 and Path3. If suppose a packet flow from nodes E to F, that require minimum bandwidth and minimum delay, then the path 2 or path 3 is taken by the packet depending on the availability. If a packet flow from E to F needs minimum delay then path 3 is taken by it. The QoS support in mobile ad hoc networks faces challenges like lack of central coordination, mobility of the hosts and very limited resources availability. Table 1 shows the path available from E to H, Path 1: E ----> B ----> C ----> D ----> H, Path 2: E ----> D ----> H, Path 3: E ----> G ----> H. Suppose bandwidth guarantee of 8 Mbps is required, and then path 2 is the available route.

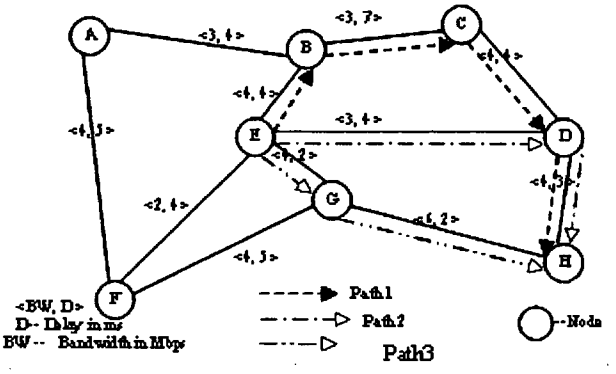


Figure 1:4 Bandwidth and time delay in ad hoc network

Path No	Route	Hop Count	Bandwidth (Mbps)	Delay(ms)
1	E→B→C→D→H	4	15	18
2	E→D→H	2	7	7
3	E→G→H	2	10	4

Table 1:1 End to end delay in ad hoc networks due to delay

- Throughput is the ratio of the data packets delivered to the destination to those generated by the sources.
- Average end -to- end delay of the packets includes all the possible delays caused by buffering using route discovery latency, queuing at the interface queue, retransmission delays at the MAC and propagation and transfer time
- Normalised routing overhead - this metric has two variants: pocket overhead is the number of routing packets transmitted per data packet delivered at the destination.
- Average end -to- end bandwidth usage for the transmission of a packet from a node to the other

1.4.2.1 Ad hoc On - Demand Distance Vector (AODV)

AODV [69] evolved from Destination - Sequenced Distance Vector Protocol (DSDV) is a reactive protocol. DSDV is called as a proactive protocol based on Bellman - Ford algorithm as each node establishes routes to every other node at all times. The hop counts are the routing metric and the sequence numbers facilitates to discriminate the previous routes. The AODV needs broadcasts for the network when a route is determined and when the previous information on topology is not known to the node. When more information is needed to be transferred from one node to different nodes, AODV supports multicast and broadcast communication (for example: multimedia, such as motion pictures). AODV uses two types of routing tables namely, unicast and multicast routes. The route entry has destination, next - hop and distance (equal amount of links between source and destination, in hops). To prevent loops, the route entries created by the destination consist of age and sequence numbers (ensures that the previous paths are not misinformed to the nodes). The sequence numbers is the key element of AODV which always helps to create new routes. The age helps to alter the entries that have not been used and it may cause problems because of movement in nodes. If the entry is used at any given time; then the age is set to zero.

1.4.2.1.1 Route Discovery

The route discovery process denotes the source as the node that initiates the process and the destination is that the node that the source is intending to reach. In a AODV network, if a route to the destination is not determined, then the source has to initiate the "route discovery process" to send message from source to destination. The process is initiated by sending a "route request" (RREQ) to its neighbours. The neighbouring nodes send RREQ to its neighbours and so on until it reaches destination node, where it can send the reply back "route reply" (RREP) to the sender node (source) by finding fresh routes. The sequence numbers with every path or routes, RREQ and route entry helps to determine the fresh or new routes.

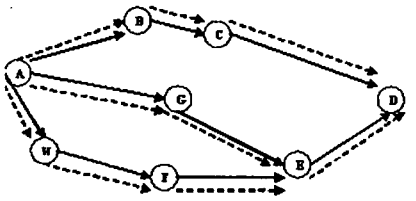


Figure 1:5 Propagation of RREQ

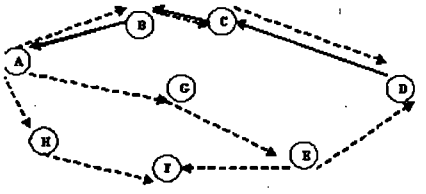


Figure 1:6 Propagation of RREP

The following fields are contained in the RREQ message of the source node:

- Source node's IP address
- Source node's broadcast ID
- Current sequence number of the source node
- Destination IP address
- The last known sequence number of the destination
- Hop count

1.4.2.1.2 RREQ Message Format

Each time when a node initiates a route discovery process and sends RREQ, the broadcast is incremented and a timer is started. To check the message that has been sent, RREQ received by each intermediate node checks the combination of IP address and the broadcast ID. RREQ sets up a reverse route entry which has sequence number, next hops address back to source, and the source of RREQ, for every node that is not seen. The intermediate node identifies the number of hops to the source.

If the node is seen, then the intermediate node drops the message. The route will be deleted from the routing table if it is not being used in a given time. When the

intermediate node that receives RREQ; after finding out the destination, a fresh route can send back the RREP by unicasting to the source node. The destination node can take control of the intermediate node, if the message traverses thus far. If the route entry is not fresh, then the RREQ has to be forwarded to the each neighbouring node with an increase in the message hop counts. The failure of the application can be notified when the destination node is not found and the timer in the source node expires, after trying for various times. Besides the controlling of the new (fresh) routes, the sequence number helps to avoid loops in the routing.

The RREP contains the following fields:

- Source's node (originator of the RREQ) IP address
- Destination IP address
- Sequence number that the destination generated for this route
- Hop count
- Life time

The intermediate nodes updates and alters the pointers for the node from where it receives RREP, the source and destination timers, and keeps record of the last sequence number for destination when it returns back to the source. If the intermediate node receives multiple RREQs, then the copy of the previous ones are forwarded first; provided, the hop counts are smaller than the previous ones.

1.4.2.1.3 Path Maintenance

To confirm the link active condition, hello messages are sent, when there are no downstream messages coming through the neighbourhood. The hello messages are not set to be forwarded as it has the TTL value of 1. The sender's ID and unincremented current sequence number are specified in the hello packet. If the link is broken and is failing to get allowed_hello_loss consecutive hello messages, a RREP message is sent by the node upstream of the broken link to the "active" neighbouring nodes which are affected along the route. The hop count is set to ∞, when RREP is unsolicited and it has new sequence number other than the broken link route. The routing table of each intermediate node has a listing of the active

neighbouring nodes. The neighbours of the intermediate nodes are the active neighbours, and the packets are sent to the destination actively according to the entry in the routing table. The packets originated or sent to the destination are called as active in the last active_timeout period. The active neighbour that receives the RREP sends it to their active neighbours in the broken route until it reaches the start nodes of each route.

Node 4 checks its routing table for ensuring the forwarded packets to Node 5 by Node 2 and causing an unsolicited RREP to be sent to Node 2 during the last active_timeout. Node 2 sends RREPs and checks the same for Node 3 and 1. Therefore, each node that used Node 4 as an intermediate node to send packets to Node 5, a destination node, knows the path and it does not work. Node 1, 2 or 3 that still in need to send packets to Node 5, has to initiate the route discovery process. AODV must be used in symmetric links [70] network as the RREPs are unicast and sends back to the sender, this condition has been tested for forwarding the message. The message forwarded is checked for the hello message is being received in the last allowed_hello_loss period over this link. In asymmetric links, the two nodes can be seen at the each other's vicinity, though only one node's radio range can be able to reach the other node because of the depleted batteries. Due to the route discovery process the whole AODV network, can be affected and each node may receive RREQs and this process could even be carried on after finding the destination. To eliminate this problem a method called "expanding ring" search is being implemented. It works by using values of 'time-to-live' in the RREQ (where TTL is decremented by the forwarder and when the counter reaches 0, the message would be discarded) which is less than the network diameter which is the longest path in the network.

If the RREQ is not been received by the source, then TTL is increased in the consecutive RREQ. If the source has reached the destination recently, then the TTL can be determined accurately and therefore, the knowledge of path length between source and destination when the routing was functioning last time can be obtained. This is questionable, and whether the remedy is effective, though the guesses cannot work. For example, if the TTL is set short in time. In this instance, the route discovery process is started and therefore, the process of finding the destination is doubled. If the destination is out of the range of the perimeter of

the first message's reach, then it should be more effective to propagate the message further than that.

1.4.2.2 Dynamic Source Routing Protocol

DSR [71] is a reactive protocol which uses Route Discovery and Route Maintenance mechanisms to discover, configure and search the routes. A discovery packet sent by a node joins itself to the route which is accumulated with the packet header. A distinguishing feature of DSR is that it supports the unidirectional links if the data link protocol supports it. Contrary to AODV, DSR supports multiple routes and therefore, nodes answer to all the received discovery packets. DSR is a source initiated and it is called as on-demand routing protocol. Each packet header has the lists of the route from the sender and identifying the forwarding next node "hop" using the address to send the packet in its way to the destination. If a mobile node needs to send a packet to the destination, then it first checks for the route cache whether the route is specified in lists or not; it sends packets by using the same route if at all it used before. If not, it broadcasts a route request packet and initiates route discovery. The node adds its address to the route request packet when a request packet is received and it rebroadcasts the query to the neighbouring nodes if the request packet is not received. If the query can be obtained from the route cache, then it sends the reply packet to the source instead of propagating the query packet any further. Therefore, any node that gets involved in route discovery can get knowledge from the sending the packets and collect information on routes from the route cache.

1.4.2.2.1 Route Discovery Process

When the sender that has the route entry in its routing table for the destination, it copies the route in the header of the packet. The first intermediate node in the route entry would have the knowledge of where to forward the packet to the destination. The sender when it does not have the entry information of the destination initiates the route discovery process. The sender broadcasts the message that has request ID, sender and receiver nodes and a list of an

intermediate node in a route discovery issued by it. The list of an intermediate node contains the neighbouring nodes information and hopes that the destination is at the next hop, and it broadcasts the Route Request again. Therefore, when it reaches the destination, it sends back the Route Reply back to the sender which has sent the Route Request. The intermediate node can stop the discovery process and add up the rest of the hops in the header for the accumulated route and sends Route Request to the sender as soon as it obtains the entry of a route. Only the valid routes are needed. Moreover, the Route Reply can be sent back to source in different number of ways. In IEEE 802.11, the every next hop of the link is possible and the route is bidirectional; and it takes the logical approach to reverse the route seen in the Route Request. Therefore, in the routing table, if the Route Request is not known for the destination node and a new route found is unidirectional, then destination sends its new Route Request. There is a risk for starting a new loop for the second Route Request while the first Route Request did not find the destination route and therefore, the Route Reply back to the originator is impossible. Hence, a piggybacked form of message must be taken as a precaution. The Route Request from the destination has to be piggybacked with the response. The buffer used to keep the addresses in the routing table for the data messages till the Route Reply message is received when the nodes are not found. The node can confine an exponential back - off waiting period when the destination is not found and therefore, it cannot send Request Routes. Consider A sends request to E. Intermediate nodes appends its ID, sends Route Request to E and E knows its route back and sends Route Reply back to A. If the intermediate node C has the details of the route in the route cache, it sends the detail to A before, the entire route is found.

1.4.2.2.2 Route Maintenance

This is a mechanism the source route is defined and intact in the way described as: The intermediate node is responsible to ensure that the data packet is received by the next hop neighbour. It can be carried out in different ways. The passive hop-by-hop acknowledgement occurs when the lower layer protocols supports the neighbouring nodes forwarding messages. The DSR at network level is used at the link layer for an active acknowledgement. If it is failed to acknowledge back to the sender after repeats, a Route Error occurs and it removes the route to the

destination from the routing table and look for another route if possible. When the route is not known then the new Route Discovery Message is sent. Piggybacking is done for route errors just like the route transmission. When Route Request is initiated, many nodes have routes to the requested node and it has to acknowledge to the request. A counter is taken as it may use more bandwidth. When a node tends to send Route Reply from the Route Cache, then it takes time for back - off which corresponds to the length of the route and the originator of the Route Request. The answering node has network interface and looks for a sign from the source, if it has taken a shortest route. The Route Reply would be sent when there is no packet is received.

1.4.2.3 Optimised Link State Routing Protocol

The Optimised Link State Routing Protocol (OLSR)[114] is a proactive protocol and is table driven in which it exchanges the information regularly about topology with the nodes in the network. It is the optimised version of the classical link state routing algorithm with the requirements modified of the ad hoc networks. The advantage of the proactive protocols is when the routes are needed, then the latency is shortened and the disadvantage is the increase of traffic control. The authors of OLSR have reduced the LSP's by sending to only a subset of neighbours. In OLSR, the LSP's are called as Topology Control (TC) messages. The Multi point Relay Selectors (MPR) is the nodes assigned for forwarding messages for TCs. The other neighbours just read the message but do not forward it. The age of TC is known by its sequence numbers. The control messages are sent with high frequency to find out the movements and have new neighbours as the node moves. The condition of node n in MPRs is that the links are bidirectional to each node, which is 2 hops from n. The condition is for a complete broadcast. When the routes are found, then the MPRs are the intermediate routers for the receiver. Nodes that have information of other node's MPRs, then it will update the tables to find the routes to the destination.

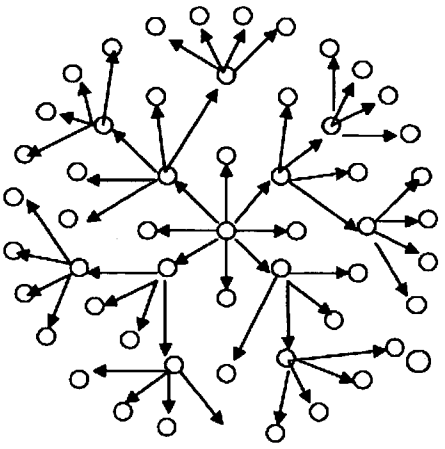


Figure 1:7 OSLR Packet Transmissions

In order to create link functionality with neighbours, OLSR uses hello messages and these messages have the neighbours of the originator. When the node receives the hello message, the node ensures the neighbour by assessing using a bi -directional link.

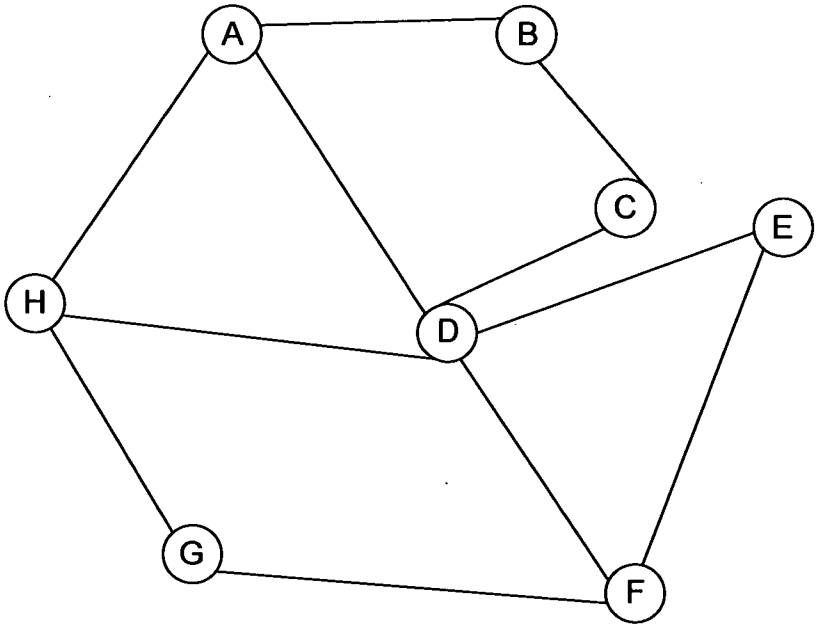


Figure 1:8 OSLR Neighbours

The sequence number for a TC is used for comparing the incoming TCs and to delete the older TC. The MPRs lists do not contain a single message but it is done over a specific and the MPRs should be given in the messages to frame a routing table. Current TC is forwarded before the next TC is scheduled, when there are alterations in MPR nodes. The TCs is withdrawn after the specified time period.

The TC information is stored in the database and then the calculation is carried out as in the classical Link State algorithm.

1.4.2.4 Zone Routing Protocol

ZRP [105,44] provides a framework for other protocols. The neighbourhood nodes when detached from the global topology employ different techniques or mechanisms and takes different features in an environment. The local neighbouring nodes are called as zones and each node could be in the multiple overlapping zones and every zone may be of different sizes. The size of the zone is not the physical measurement and the radius of the length is given as 'r' and it denotes the number of hops be made in the zone. ZRP prevents a hierarchical map and overheads in maintenance of the network by dividing network into multiple variable size zones.

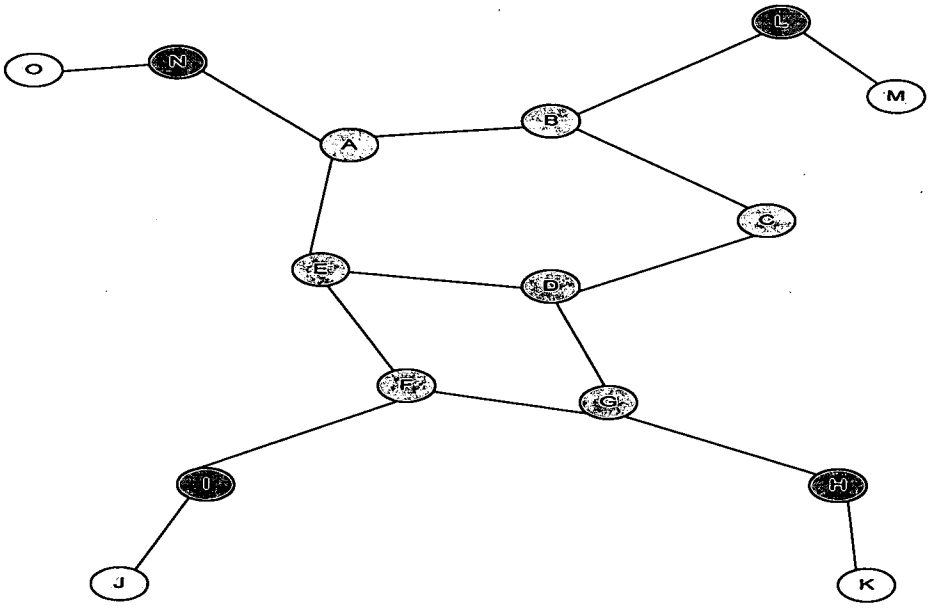


Figure 1:9 ZRP Routing

When overlapping is traced, there is possibility of route optimization and the network can be determined as a flat network. ZRP can have proactive and reactive combination approaches. From Node X, the nodes within the hop distance d, is the routing zone for Node X. Nodes at hop distance exactly at d is the peripheral

nodes in the routing zone X. It is said to be Intra - Zone routing when the routes that are proactively maintained in the source node's zone and uses on -demand protocol (DSR or AODV) to find out the routes that are outside to the zone.

Nodes outside routing zone (J, K, O and M)
Internal node (A, B, C, D, E, and F)
Peripheral Nodes (N, L, I and H)

1.4.2.5 Comparison of Routing Protocols

Pro-active protocols follow an approach similar to the one used in wired routing protocols. By continuously evaluating the known and attempting to discover new routes, they try to maintain the most up-to-date map of the network. This allows them to efficiently forward packets, as the route is known at the time when the packet arrives at the node. Pro-active or table-driven protocols, in order to maintain the constantly changing network graph due to new, moving or failing nodes, require continuous updates, which may consume large amounts of bandwidth - clearly a disadvantage in the wireless world, where bandwidth is often sparse. Even worse so much of the accumulated routing information is never used, since routes may exist only for very limited periods of time.

In contrast, reactive protocols determine the proper route only when required, that is, when a packet needs to be forwarded. In this instance, the node floods the network with a route-request and builds the route on demand from the responses it receives. This technique does not require constant broadcasts and discovery, but on the other hand causes delays since the routes are not already available. Additionally, the flooding of the network may lead to additional control traffic, again putting strain on the limited bandwidth. Either a purely pro-active or purely reactive approach to implement routing protocols for a MANET has their disadvantages.

MANET, it can safely be assumed that the most communication takes place between nodes close to each other. Changes in the topology are most important in the vicinity of a node - the addition or the removal of a node on the other side of the network has only limited impact on the local neighbourhoods. Next

Generation Networks will support real-time multimedia applications through traditional wide-area networking concepts as well as hot-spot (WLAN) and ad hoc networking concepts.

Proactive routing protocols can provide low latency and good reliability but at the same time are associated with high overhead specially with the increase of the number of nodes in the network which wastes the most valuable resource in a MANET, the bandwidth. On the other hand, reactive routing protocols efficiently make use of bandwidth as they delay the route discovery mechanism until a route is requested thus dramatically decreasing the routing overhead but enduring an apparent increase in latency. The choice of the ad hoc routing protocol to deploy in a specific MANET depends on the characteristics of this network. Factors like the degree of mobility of the nodes and the traffic load in the network among other factors are very important to take into account, but the problem is that ad hoc networks are dynamically changing and we can not forecast how the topology of an ad hoc network will look like in the future.

Hybrid adaptive ad hoc routing protocol [141] attempts to make use of the advantages of both proactive and reactive routing protocols. There already exists a variety of hybrid protocols, such as ZRP [105,144], CEDAR [150], ZHLS [151] and SHARP [151] that combine proactive and reacting routing mechanisms in various ways.

1.5 *Combinatorial Optimization Techniques*

1.5.1 Hill climbing algorithm

Hill climbing algorithm is used to solve combinatorial optimisation problems. The first position of the node is randomly selected, and the neighbours are checked at each step. The better node amongst the neighbours is selected and is named as the current node. The search continues from there on and the best solution or the optimum solution is the outcome of the algorithm.

The basic hill climbing algorithm is as given below

```
function HILL-CLIMBING(Problem) returns solution state
{
Current = MAKE-NODE (INITIAL-STATE [Problem]);
do
        Next = a highest-valued successor of Current;
if VALUE [Next] < VALUE [Current]then
return Current;
Current = Next;
        end
loop
}
```

Figure 1:10: Hill climbing algorithm

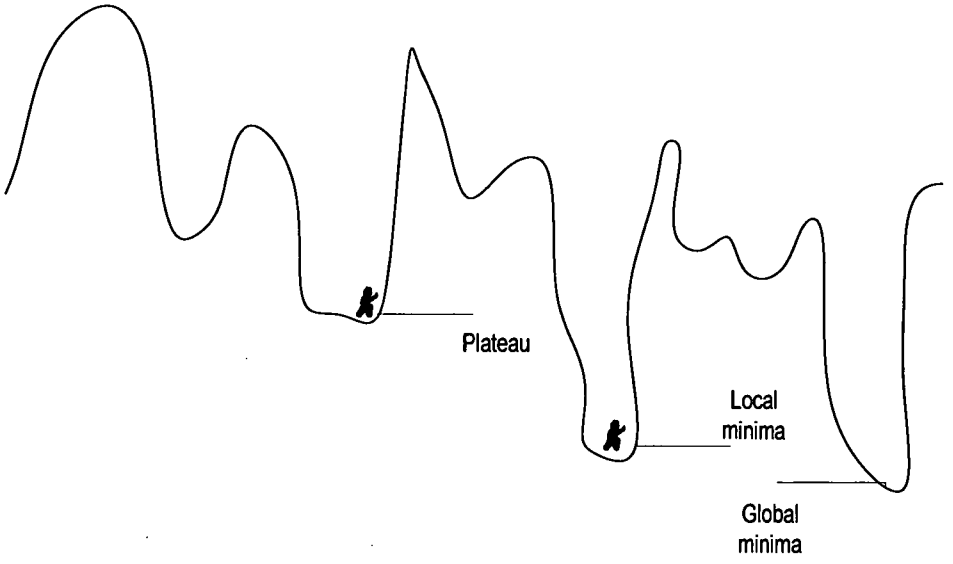


Figure 1:11 Hill climbing algorithm issues

The hill climbing algorithm works as follows:

1. Pick a random point in the search space. Name this as current state.
2. Consider all the neighbours of the current state.
3. Choose the neighbour with the best quality and move to that state.
4. Repeat steps 2-4 until all the neighbouring states are of lower quality.

5. Return the current state as the solution state.

Accept the child if it observes more targets.

$H = \sum\sum dist(m, t)$ if $R/2 < dist(o, t) < R$

$= 0$ otherwise

If $H_{child} < H_{parent}$ accept child

$G = \sum min\ dist(m', t')$ m' , t' are unobserving watchers , unwatched targets

If $G_{child} < G_{parent}$ accept child

Otherwise Reject Child

Figure 1:12: Hill climbing algorithm for co-operative target observation

The main issue with hill climbing which is called gradient descent is that we are not assured of the best solution to the search problem. For example when we go mountain climbing like the Himalayan ranges during not so good weather conditions like fog or rain. When we reach a point after which we feel we cannot get up on the range anymore we might come to a conclusion that we are at the highest peak of Himalayas, which may not be true. We are unable to see the highest peak from where we are due to the weather conditions. There may be another peak which is higher than the point we have reached. This is exactly the case in the hill climbing algorithm as illustrated in the figure above.

In the case of hill climbing algorithm if it has reached a point where it cannot observe any better neighbouring nodes it concludes that it has reached the highest peak. This is necessarily not the case. This may not be the maxima point. It might not be the global maxima. It may be the local maxima. As it cannot see any better neighbouring nodes it concludes that it has reached the best solution. Eventually there is a chance of this getting stuck in a local optimum or at a plateau.

1.5.1.1 Hill climbing characteristics

Some of the characteristics of Hill climbing algorithm are as follows:

1. As there is no backtracking it requires no memory

2. Design of the next step move is very critical. If this is not done right then the algorithm does not work well for a big search space
3. Design of the cost function is also critical as it can get stuck at local optimum rather than global optimum. It also tends to get stuck at plateau.
4. The algorithm is in efficient if the number of moves is very large.
5. The algorithm gets stuck at local optimum or plateau if the number of moves is very less.
6. Therefore optimum number of moves has to be chosen.
7. The incremental cost evaluation is easier to evaluate rather than evaluating cost from the beginning. This makes hill climbing algorithm work easier in many simple applications
8. Evaluating the local optimum is easier with this algorithm. Therefore when the problem is for evaluating the local optimum this algorithm is widely in use.
9. Hill climbing algorithm is used for maximizing problems rather than minimizing functions.

1.5.2 K means clustering

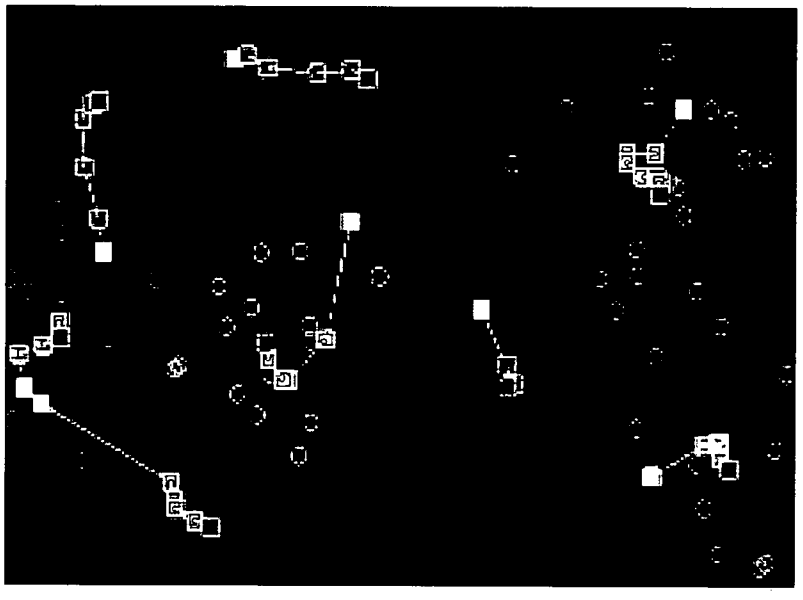


Figure 1:13 K means clustering demonstration

K means clustering is used to solve the clustering problem. The number of clusters is fixed before running the simulation. It is a simple unsupervised learning technique. Let the number of clusters be k, and then k centroids are formulated one for each cluster. Depending on the location of the nodes the centroid of the clusters are present. It is always advocated that the centroids are placed as far as possible from each other. However it depends on the number of nodes, the number of clusters, and the position of the nodes. Now each point or the nodes are related to its closest centroid. This goes on till all the nodes are related to some centroid closest to it. Then a loop is said to be formed. Next the k centroids keep moving the locations till there can be no more changes made. This K means clustering algorithm is used in minimizing an objective function

$$J = \sum_{j=1}^{k} \sum_{i=1}^{s} dist\ (x_i, c_j) \qquad [1.1]$$

where k is the number of clusters, c the cluster head or the centroid, x the nodes in the network. This objective function J has to be minimized.

The algorithm works as follows:

1. Place x points into the space represented.
2. Choose k , number of clusters.
3. Form the initial group k centroids.
4. Assign each node to the group that has the closest centroid.
5. When all objects have been assigned, recalculate the positions of the K centroids.
6. Repeat Steps 3 to 4 until the centroids no longer move. This produces a separation of the objects into groups from which the objective function to be minimized can be calculated.

Some of the issues of K means clustering are:

1. The most optimum solution may not be obtained by k means clustering algorithm.

2. The solution relies much on the random point chosen to be a cluster head.
3. The algorithm has to be run one or more times to get the most optimum solution.
4. The time for computation thus increases.

http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/AppletKM.html

1.5.3 Reinforcement Learning

Reinforcement learning is the learning mechanism which learns what to do. The situation awareness is learned and the actions are taken accordingly. Which an action to be taken when is not instructed but it learns by trial and error method. When the right actions are taken then the reward increases. When it is able to take decisions in challenging situations then the reward is immediate. There are two characteristics in reinforcement learning. These are trial and error and delayed reward. In reinforcement learning the agent has to understand the situation and take decision based on the previous knowledge of the system, rewards obtained in such situations. The main challenge in reinforcement learning is the trade-off between the exploration and exploitation of the environment or the situation.

The elements of reinforcement learning are: the policy, the reward function, the value function and the model of the environment. The policy is the way in which an agent behaves in a particular situation. The reward function defines the state of action and the desirability of that outcome. The reward function represents what is good in the immediate sense but the value function represents the long term desirability. The model of the environment represents the behaviour of the environment.

If s denotes the state before a greedy move, and s′ the state after the move, then the update to the estimated value of s, is denoted by V(s), can be written as

$$V(s) \leftarrow V(s) + \alpha\,[V(s') - V(s)] \qquad [1.2]$$

where α is a small positive fraction called the step-size, which influences the rate

of learning. This update rule is an example of a temporal difference learning method, so called because its changes are based on a difference, V(s') – V(s), between estimates at two different times. Reinforcement learning is the automated goal directed learning and it is used in problems such as searching using mobile robots, chess play, etc.

1.5.4 Genetic Algorithm

Genetic algorithm (GA) is a numerical optimization algorithm inspired by natural selection and natural genetics. The GA, is a stochastic global search method that is analogous to the metaphor of natural biological evolution or the Darwinian concept of survival-of-the- fittest. Environmental constraints doom the less fit ones whereas the fittest ones are able to survive. The descendants of the original population inherit qualities that better fit the environment. The evolution of population fits the environment as the natural adaption rather than the ones they are created from. Natural evolution operates on encodings of biological entities in the form of a collection of genes called chromosome, GAs operate on string representation of possible solutions in terms of individuals or chromosomes containing the features. The feature value, the string structure and the string structure's decoded value in case of a GA correspond to the allele, genotype, and phenotype in natural evolution.

In general, the GA consists of the following components: population of individuals, encoding or decoding mechanism of the individuals, objective function and an associated fitness evaluation criterion, selection procedure, genetic operators like recombination or crossover, mutation, probabilities to perform the genetic operations, replacement techniques and termination conditions. GA is used to solve many optimisation problems as it gives the optimal solution or near optimal solution. The advantage of GA is also it produces not only one optimum solution but near possible optimum solutions.

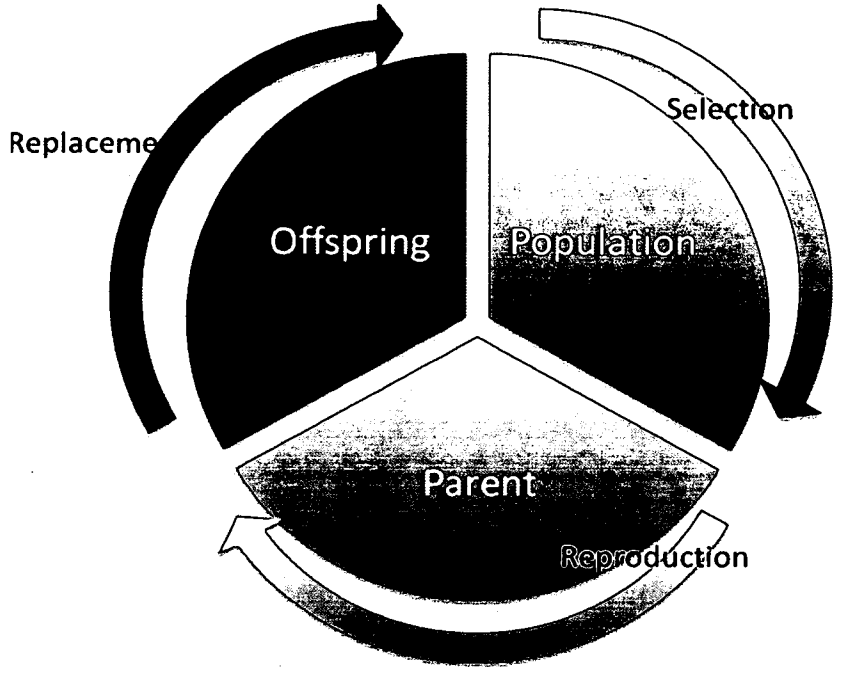


Figure 1:14 Genetic algorithm components

The chromosome f has the expected occurrence P (accept) which is given by

$$P(accept) = f_n / \sum_1^k f_s \qquad [1.3]$$

And expected occurrence e_n is given by

$$e_n = P(accept) * k \qquad [1.4]$$

The selection from the population is done by

- roulette wheel selection (RWS) and
- stochastic universal sampling selection (SUS).

There are two main operators in GA. They are the crossover and the mutation. Crossover generates new individuals that have some feature of both parents in term of genetic material. The generated off-spring strings from the crossover process might have either higher or lower fitness values than the parent strings, but the selection process will ensure that higher fitness strings have higher probability of survival. Mutation is a random process which creates an individual to produce a single new solution. This process can be implemented by randomly selecting a bit and flipping the value. By randomly altering values on genomes, this

operation helps ensure a comprehensive coverage of the search-space. In the GA, mutation is normally applied on the population with low probability.

1.6 *Co-operation and co-ordination*

The leashing problem is to physically or electronically constrain radio nodes within the network so that communication can take place between the designated end points. This problem is of interest in a range of UAV applications, where communication is required between nodes that would not otherwise be able to communicate for instance because of range constraints or line-of-site obstructions. A simple leashing scenario is an example with a ground station on the left, a sensing UAV on the right and a relay UAV above the hill. The ground station cannot directly communicate to the sensing UAV because of the line-of-site (LOS) obstruction by the mountain. However, using another UAV to relay information from the sensing UAV back to the ground station, we significantly improve the usable range of the sensing UAVs. This image also highlights how mobility can be used to improve network connectedness and performance. When the sensing UAVs are forced to block the LOS transmission to the ground station to continue tracking the target, they can call on assistance from another available UAV to act as a network relay, maintaining the vital connection. Though the figure shows a single leashing UAV team connecting a single sensing UAV to a single ground station, this process can be repeated to link a number of UAVs and UAV teams together to provide long-range communication between multiple ground stations.

The control algorithm for the leashing problem will need to take into account several network and UAV (local and neighbour) related performance parameters such as bandwidth, sensor scheduling, information relevance etc to achieve truly optimal results. It is common to see networked control systems and communication protocols [20] presented and designed independently.

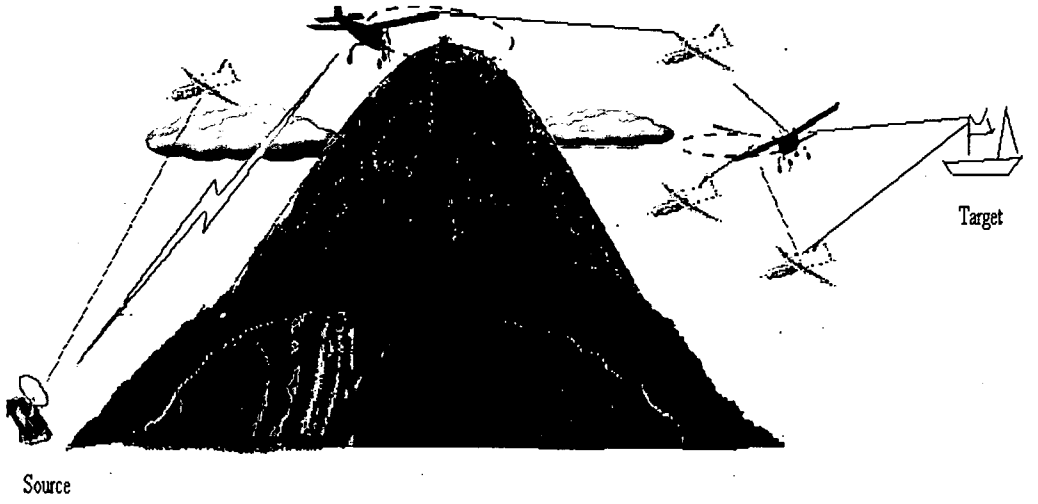


Figure 1:15 Radio leashing for communication

Consider the example network as shown in the figure. The information of the target has to reach the station which has human supervision. The target information should reach the source through any available path. For the information to reach the source, an intelligent routing algorithm has to be designed and implemented. The information reaching the source could be through any available route. The parameters that are to be considered are: the speed of the moving nodes, the link quality, the bandwidth utilization, the obstacles in-between like the hill, the sensor scheduling, the line of sight which separates the communicating nodes etc.

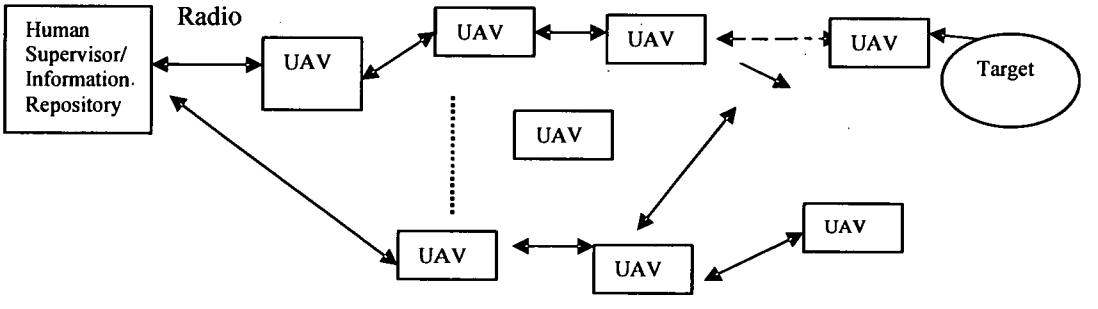


Figure 1:16 Network without connectivity constraints

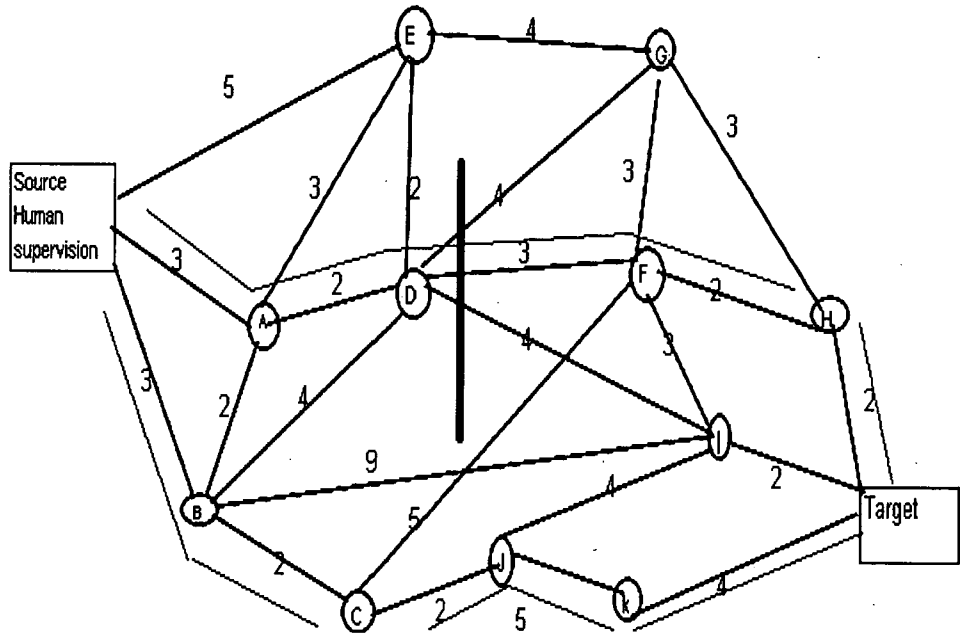


Figure 1:17 Network with connectivity constraints

There are various paths available for the information to traverse from the source to the UAVs closest to the targets. Similarly there are various paths from these UAVs to the ground station or information repository. But depending on the relevance of information the information should be routed such that the information reaches the control station in time and also uses the bandwidth more efficiently. The weight of the path depends on various factors like the distance, end-to-end time delay, reliability of the link etc. The total weight of one of the path shown is (3+2+3+2+2) = 12. The weight of each link can be predefined in the case of wired network. But in the case of a wireless network where the nodes seamlessly move, the weights can be defined dynamically using learning technique. These weights are passed on to the router which decides the route so that the target information reaches the source without any error and also in time. When there is an image of the source to be transmitted to the source from node I, then the message or the information could be divided into packets which reach the destination (control station with human supervision) through the available shortest path. The available bandwidth should be utilised efficiently. If there is an obstacle which does not allow a node to communicate with the next node, then information reaching the source may not be feasible as there is no available path.

Therefore path maintenance should be focussed on, especially when the UAV tends to get lost or looses the radio signal communication. The ultimate aim is to keep the source informed of the target information without any information being lost. Therefore increasing the throughput is an important aspect of this research. When the throughput has to be increased then the various constraints (line-of-sight, over-the-hill leashing obstruction, UAVs shot down etc) should be taken into account.

4.1 Networked Team Control

There is significant work and research in mobile ad hoc networks and in UAV team control, but little work attempts to combine these into a single framework where both systems are simultaneously designed or analyzed. Network communications between the team is required for performance enhancement of the team [78].Shared knowledge is a necessary condition for coordination [8] but little work has been done with communication as control parameter[9,48-50]. The optimal way of knowledge sharing for team collaboration needs an intelligent communication protocol [83].

In networked control systems used for UAV team control, team control is focused on ignoring the fact that individual vehicle motion would have significant impact on the overall performance of the network. Moreover it is often assumed that the underlying network is available continuously. In order to fulfil the vision of Next Generation Networks for UAV communications maintaining a real-time flow despite frequent topology changes and irregularity in movement and service further research is required [93]. While mobility has been extensively investigated in MANETs, it is generally presented as random mobility instead of controlled mobility. A networked control system[85-91] consists of numerous physical and computational agents that share information to close control loops over network resources [85]. An "agent" could be an aircraft, an on-board sensor, a control module, the ground station or an ensemble of such devices.

There are numerous technical challenges in developing viable cooperative planning and control methods for teams of UAVs. A significant issue is the level

of complexity involved in planning and coordinating a team of vehicles with varying capabilities and competing interests so that a team objective is attained in an optimal manner [79]. This is especially true for applications with unknown and rapidly changing environments, requiring UAVs to continuously react and re-plan. UAV team management and control covers a wide area of research interests including mission planning and re-planning [98], optimal sensor (UAV) placement [99], task allocation [100], formation control [101], obstacle avoidance [75], and fault tolerant distributive control [102-105]. However, only a little amount of work [80] has included network performance and network service guarantee as a team objective.

1.7 *Thesis statement*

Perform an in depth study of the various path finding algorithms in conventional networks and autonomous network. Perform co-operative target observation with the developed new simulated algorithm and compare with other existing algorithms. Combine this new simulated annealing algorithm, with the temporal difference learning to enhance the performance of the system.

1.8 *Thesis Organisation*

Chapter 2 presents a review of the existing literature on the simulating annealing algorithm and the related algorithms. The simulated quenching, the fast and the very fast annealing algorithms are studied. The various cooling schedules for each of these algorithms are presented. The various cooling schedules and how they affect the performance of the entire system performance.

Chapter 3 presents the various path finding algorithms. These algorithms are compared. of the path finding algorithms studied are; Dijkstra's algorithm, Bellman Ford's algorithm, Floyd Warshall's algorithm, A Star algorithm, Johnson's algorithm, breadth first algorithm, best-first search and depth first search algorithms. The A star algorithm preformed better than the other algorithms path finding and searching techniques.

Chapter 4, presents the modelling of the overall system. We present the mathematical model used to perform the new simulated annealing algorithm that was developed in our research for co-operative target observation. We present the methodology of implementing this algorithm. The number of targets monitored at a particular time is recorded. This algorithm performance is compared with the conventional simulated annealing algorithm and various other existing algorithms. The basic building block of the entire system is also presented.

Chapter 5, addresses a modified Sarsa (λ) control algorithm by sampling actions in conjunction with the new simulation annealing technique. A game of soccer is utilized as the simulation environment, which has a large, dynamic and continuous state space. The empirical results demonstrate that the quality of convergence has been significantly improved by using the simulated annealing approach.

In Chapter 5, we discuss the simulation results used in various scenarios and the comparison with various other existing techniques. We present the initial results and then show the effect of target monitoring and optimization. We then discuss the suitability of our approach to real applications, review the limitations, failure modes and then present some cues for future investigation. Focus is on the analysis and results of the algorithm for CTO and comparison with Hill Climbing Algorithm.

2 Annealing Algorithms

Multidimensional optimisation problem is generally non-convex function. Therefore there are more than one local minima. To find the global minima costs time and computation. To obtain the global minima for these type of non-convex functions Monte Carlo suggested the simulated annealing algorithm. The term simulated annealing (SA) derives from the analogous physical process, of heating beyond its melting point and then slowly cooling a substance to obtain a strong crystalline structure.

2.1 *Annealing*

Annealing in statistical mechanics and metallurgy is the process of heating a metal to high temperature and cooling it slowly so as to alter its structure. Annealing is done to metals to cause changes in the properties such as strength and hardness by allowing it to heat treatment. The temperature is raised to high temperatures generally above melting temperature and is held at this temperature for sometime before it is slowly cooled down to for the metallic structure to be brought to the ground or thermodynamic equilibrium state. This thermodynamic equilibrium state is reached when the metal is brought to low temperature slowly. When annealing is undergone by a metal, it not only improves the ductility but also brittleness of the metal is reduced.

In the Boltzmann's distribution, at high temperature all the states are equiprobable and is in the high energy state, however when the temperature is slowly reduced or the system is cooled when the energy also reduces until the lowest energy state which has nonzero probability. Cooling is usually done slowly through the entire range of temperatures. The system spends time at each temperature and as it reaches lower temperature more time is spent at each temperature. If this is not done then the system does not reach the equilibrium state but many defects are formed in the structure which is not stable but metastable state.

The probability distribution p_i which represents the proportion of microscopic states for each macroscopic state i is expressed as Boltzmann distribution.

$$p_i = \frac{1}{Z} e^{-\frac{Ei}{kT}} \qquad [2.1]$$

Where E_i is the energy of the state i, k is the Boltzmann constant, T is the temperature of the state. The partition function Z from the canonical ensemble can be expressed in terms of energy ΔE

$$Z = \sum \exp(-\Delta E/kT) \qquad [2.2]$$

At thermal equilibrium, the partition function encodes the statistical property of a system.

2.2 *Boltzmann Annealing Algorithm*

The Boltzmann annealing [58] is a fast simulated annealing where the cooling schedule varies from the other conventional simulated annealing. This was first studied by Monte Carlo sampling technique. The choice of the probability distribution for the conventional simulated annealing and the fast simulated annealing is probability density of state space of D parameters is

$$g(n) = \exp(|\Delta x_0|^2/T(n))\ T(n)^{D/2} \qquad [2.3]$$

and the cooling schedule is

$$T(n) = T_0 / \log(n) \qquad [2.4]$$

Any point in the space can be sampled to infinitely often annealing time, therefore we get

$$\sum_{n=0}^{\infty} g(n) = \sum_{n=0}^{\infty} 1/t = \infty \qquad [2.5]$$

For fast simulated annealing (FSA), the above parameters

$$g(n) = \quad T(n)/(T^2\text{-}|\Delta x_0{}^2|)^{(D+1)/2} = \quad T_0/t|\Delta x_0|^{(D+1)} \qquad [2.6]$$

and cooling schedule $T(n)=T_0/n$ [2.7]

$$\sum_{n=0}^{\infty} g(n) = T_0/(\Delta x_0)^{(D+1)} \sum_{n=0}^{\infty} 1/t = \infty \qquad [2.8]$$

Fast simulated annealing led to the ease of use of simulated annealing which could handle large cost values and constraints. Thus simulated quenching came into existence.

2.3 *Simulated Quenching*

Simulated quenching [58] is also simulated annealing with very fast cooling schedule. The choice of cooling schedule plays a vital role in achieving the global optimum solution. This simulated quenching has a very fast cooling schedule. The cooling schedule of simulated quenching is

$$T(n) = T_0 \ln n_0 / \ln n \qquad [2.9]$$

where n is the time parameter, and n_0 is the starting time.

$$T(n) = T_0 \exp((\alpha-1)n) \qquad [2.10]$$

where α lies between 0 and 1.

Some of the applications involve simulated quenching for obtaining the global optimum. It is faster than simulated annealing as the temperature is reduced fast.

2.4 *Simulated Annealing*

Metropolis Monte Carlo integration algorithm was first formulated by Kirkpatrick. The term simulated annealing (SA) derives from the analogous physical process, of heating up metal beyond its melting point and then slowly cooling the substance to obtain a strong crystalline structure. In simulation; minima of the cost function corresponding to the ground state of the substance. The simulated annealing process lowers the temperature by slow stages until the system freezes

and no further changes occur. At each temperature, the simulation must proceed long enough for the system to reach a steady state or equilibrium. It is a technique to find a good solution to an optimisation problem by trying random variations of the current solution. A worse variation is accepted as the new solution with a probability that decreases as the computation proceeds. The slower the cooling schedule, or rate of decrease, the more likely the algorithm is to find an optimal or near optimal solution.

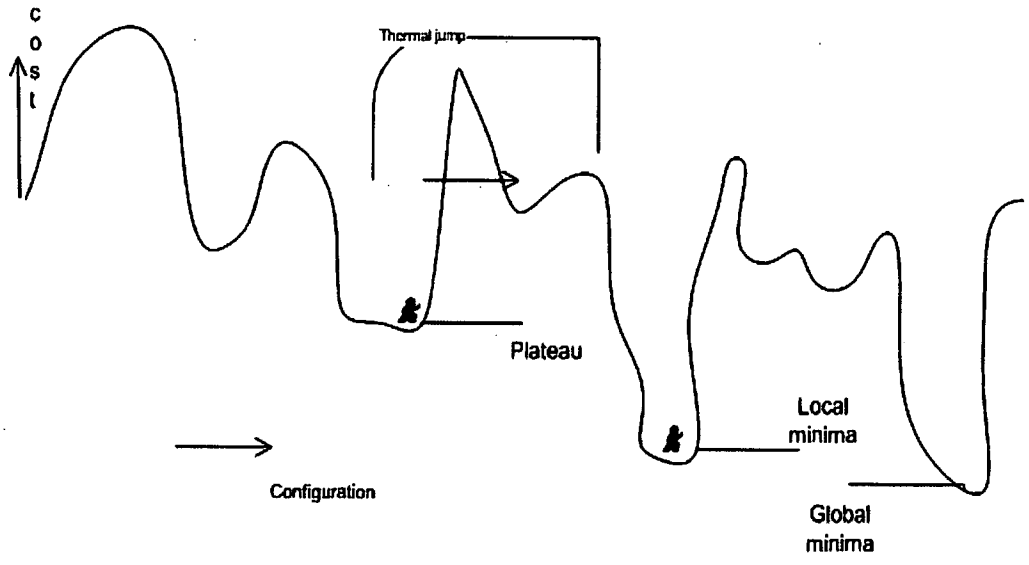


Figure 2:1 Issues in Hill climbing algorithm

The mathematical analogy between optimization problems and thermodynamic systems is simulated annealing. It searches the set of solutions for optimal position. The initial state, final states are analogous to random position, minima of the cost function respectively. Decrement factor and number of iteration per state searches are used for global minima or maxima. The applications of simulated annealing are in job scheduling, timetabling, target monitoring, etc.

The probability of acceptance is

$$P\{accept\} = \begin{cases} \{\exp(-\Delta E/\Delta(Kt)) > r & \Delta E >= 0 \\ \{1 & \Delta E < 0 \end{cases} \quad [2.11]$$

where

$$\Delta(Kt) = K \exp(-\lambda t) \quad [2.12]$$

and ΔE is change in the evaluation function,

t the current temperature,

r is a random number between 0 and 1.

Thermodynamic system	Simulated Annealing analogy
System States	All possible Solutions
Energy	Cost
Change of State	Next or offspring Solutions
Temperature	Control Parameter
Frozen State	Heuristic Solution

Table 2:1 Simulated Annealing Analogy

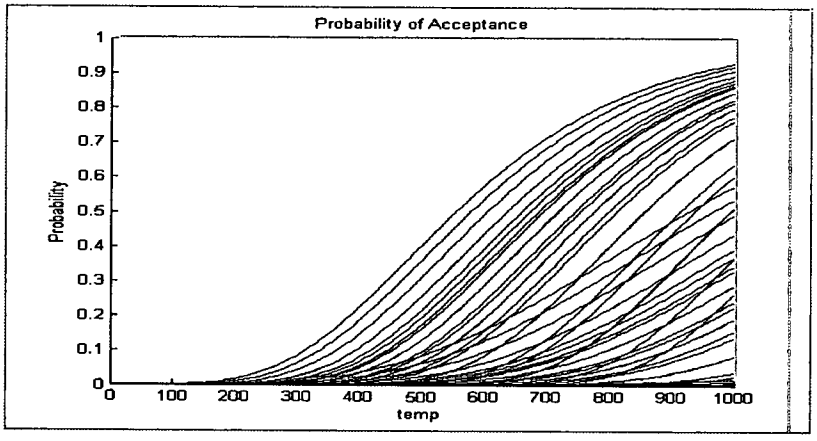


Figure 2:2 Probability of acceptance

2.5 *Cooling Schedule*

Generally there is a free parameter in the cooling schedule which determines apparently the rate of cooling of the overall system. The schedule of a combinatorial problem using annealing algorithm is a decreasing sequence of temperatures. The cooling schedule of simulated annealing algorithm depends on the starting temperature T_0, the final temperature T_∞, decrement in temperature in

each step and the number of iteration at each temperature. There are different cooling schedules being applied for various applications.

2.5.1 Selecting cooling schedule

2.5.1.1 Starting and Final temperature

Starting Temperature: must be hot enough to allow moves to almost neighbourhood state or else we are in danger of implementing hill climbing. Must not be so hot that we conduct a random search for a period of time Problem is finding a suitable starting temperature. If we know the maximum change in the cost function we can use this to estimate. Start high, reduce quickly until about 60% of worse moves are accepted. Heat rapidly until a certain percentage are accepted the start cooling. Final Temperature choosing is also important but not as the initial temperature. It is usual to let the temperature decrease until it reaches zero. However, this can make the algorithm run for a lot longer, especially when a geometric cooling schedule is being used. In practice, it is not necessary to let the temperature reach zero because the chances of accepting a worse move are almost the same as the temperature being equal to zero. Therefore, the stopping criteria can either be a suitably low temperature or when the system is frozen at the current temperature when no better or worse moves are being accepted.

2.5.2 Types cooling schedules

The cooling schedule is the most crucial factor contributing to the performance of the any simulated annealing or quenching algorithm. It is believed that if the cooling is fast then the system does not attain global minima. Cooling at a slow rate will improve the outcome of the system but takes a lot of computation time. The cooling schedule has to selected so that it does not take too much of computational power but also converges at global optimum.

We have studied intensely the various cooling schedules and how it can be used to improve the performance of the system. This is just a cooling schedule with exponential decrease in temperature for various values of α.

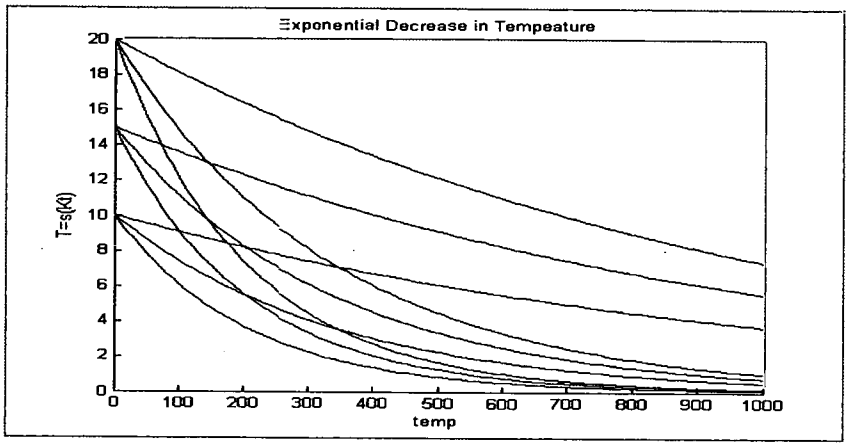


Figure 2:3 Cooling schedule

2.5.2.1 Johnsons cooling schedule

Johnson's cooling schedule [154] is widely used in many applications. The schedule is as given below.

$$T_n = T/ (1+\beta T) \qquad [2.12]$$

The cooling schedule is simple and it is linear and can be used in many applications. When the search area is not very big and does not involve many search points then it can be used. Due to its linear nature, the computation time is saved in the process of searching for the optimal position. However, the chance of not returning the optimum solution also increases.

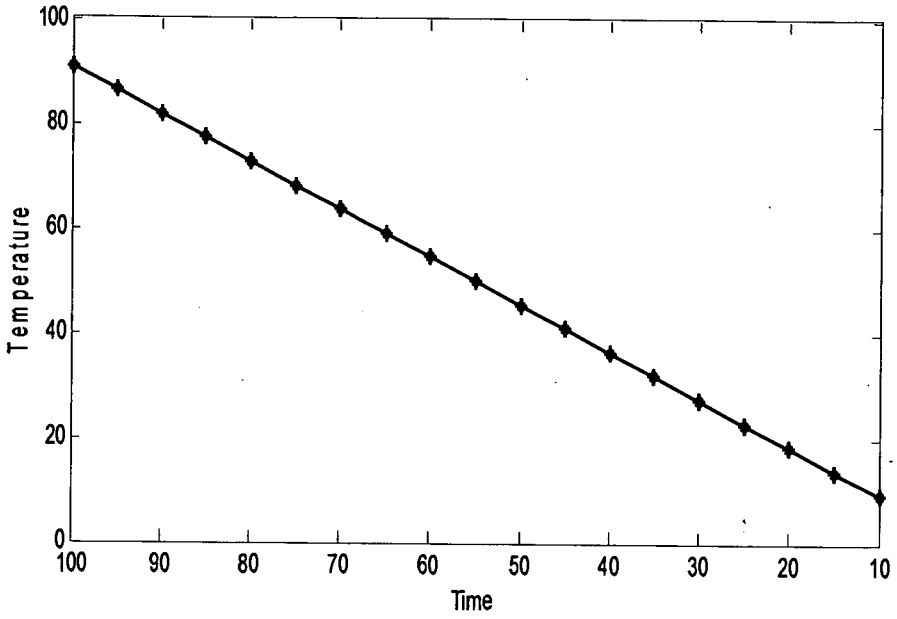


Figure 2:4 Johnson's cooling schedule

2.5.2.2 Lundy- cooling schedule

2.5.2.2.1 Classic cooling schedule

The classic cooling schedule[153] is

$$T_n = \exp(-c/T) \qquad [2.13]$$

At lower temperatures it is important that a large number of iterations are done so that the local optimum can be fully explored. At higher temperatures, the number of iterations can be less

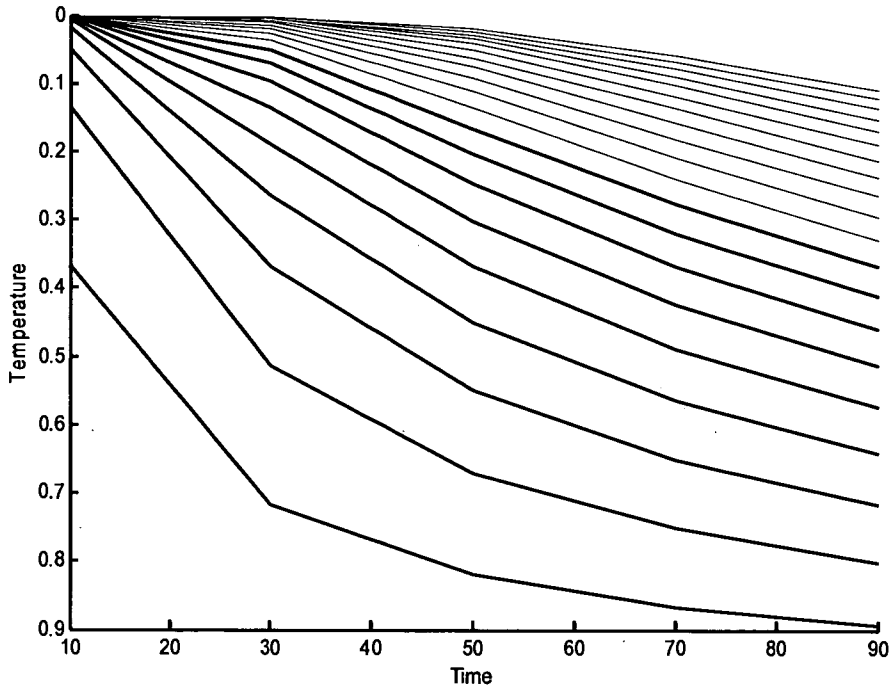


Figure 2:5 Lundy's cooling schedule

2.5.2.2.2 Average cooling schedule

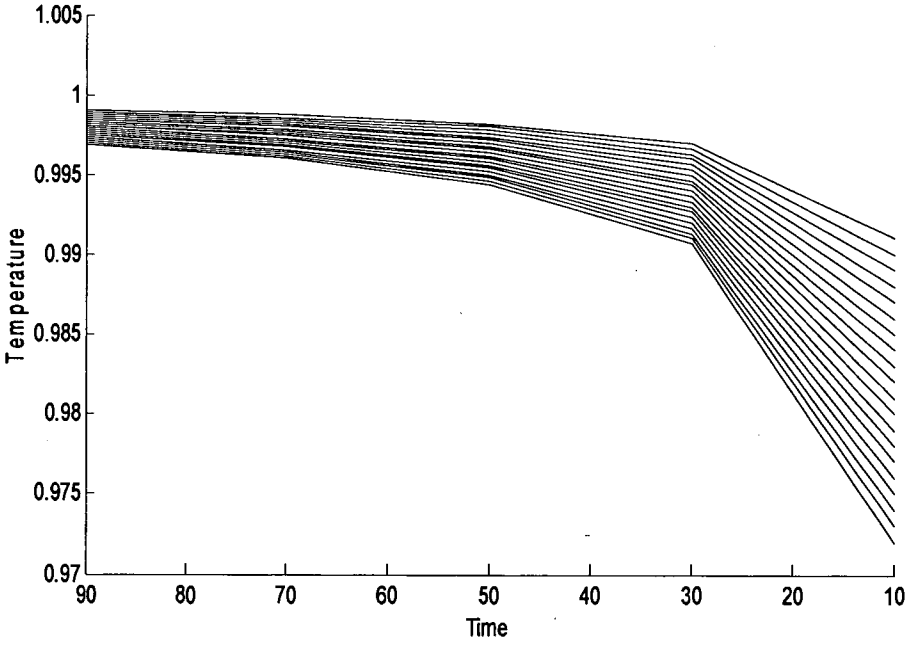


Figure 2:6 Lundy's average cooling schedule

The average cooling schedule is

$$T^n = 1 - \frac{c}{T} \qquad [2.14]$$

2.5.2.3 Adaptive cooling based on system scale of time and energy

A cooling schedule [162] which is the simplest to implement among the other adaptive schedules. Their schedule depends on the statistical behaviour of an ensemble of random walkers. The temperature is decreased by a fixed factor α and is varied in an adaptive way. At the n^{th} cooling step, the temperature is lowered from T_n to $T_{n+1} = \alpha T_n$.

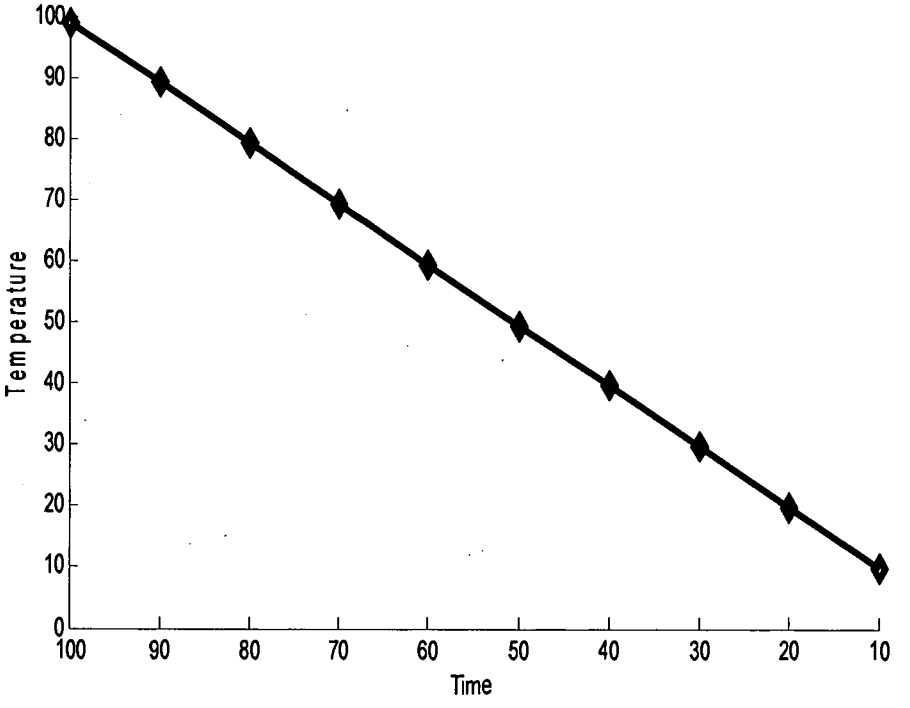


Figure 2:7 Adaptive cooling

The adaptive cooling schedule based on the energy scale of the system was developed by UC Berkeley.

$$T_{n+1} = T_n - \frac{cT^2}{\sigma_E} \qquad [2.15]$$

where $\frac{\sigma_E}{T^2}$ = rate of change of energy of the system and c is a small positive constant.

2.5.2.4 Simple time schedule

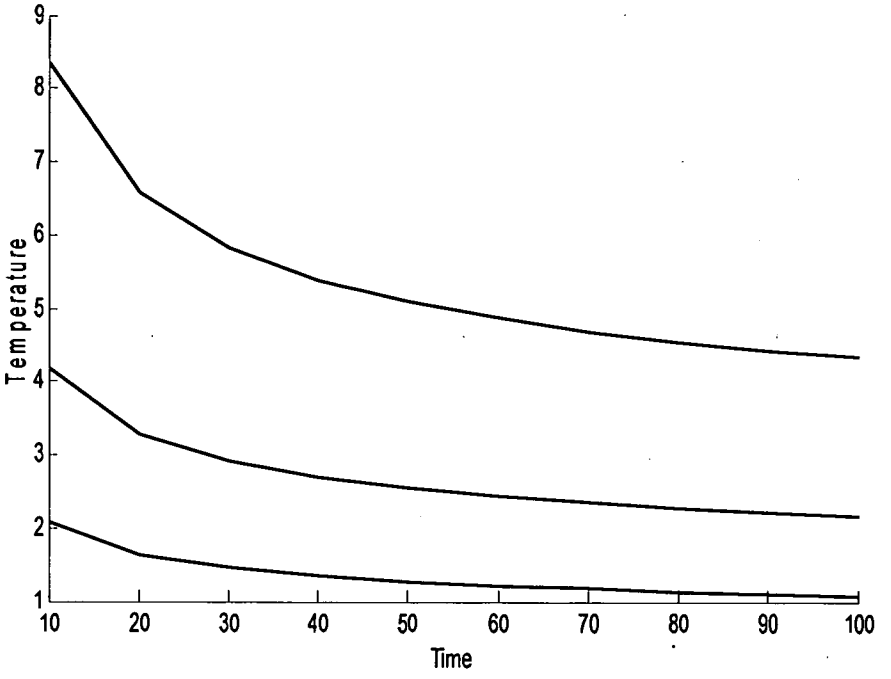


Figure 2:8 Simple schedules

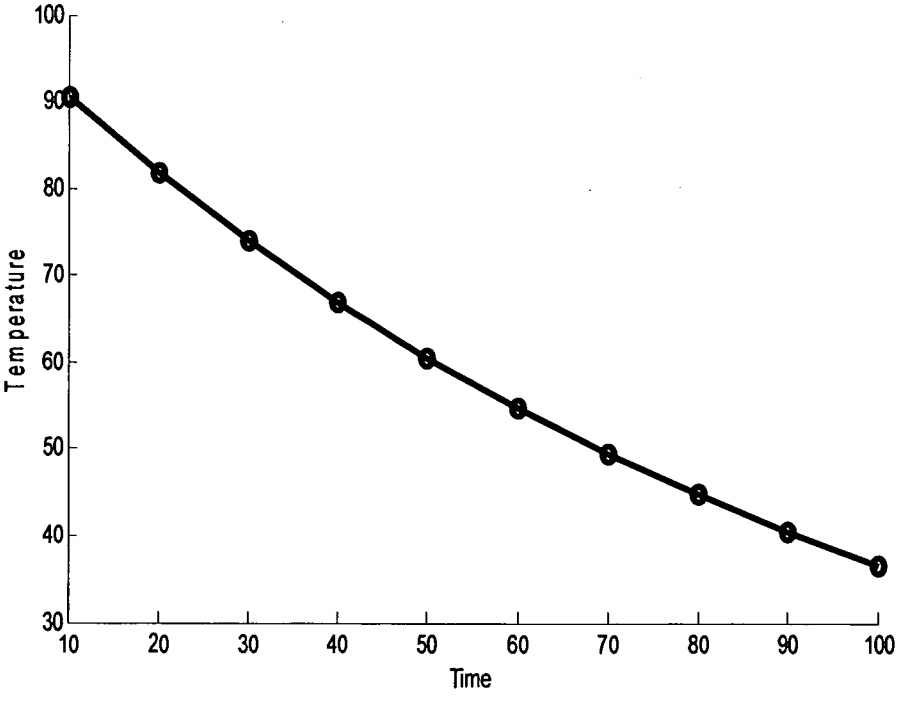


Figure 2:9 A simple cooling schedule

The graph is plotted for different values of α and c as a small positive number.

$$T(n) = \alpha / \ln (n+1) \qquad [2.16]$$

There are other simple cooling schedules [158-160] like the one stated below

$$\alpha_{n+1} = \alpha_n + c \qquad [2.17]$$

where $$\alpha = \frac{1}{T} \qquad [2.18]$$

2.5.2.5 Exponential schedule

Exponential cooling schedule [153] is used in many applications for the intense search of the global optimum.

$$T^n = T^0 \alpha^n \qquad [2.19]$$

where T^0 is the initial temperature and α is a small positive value.

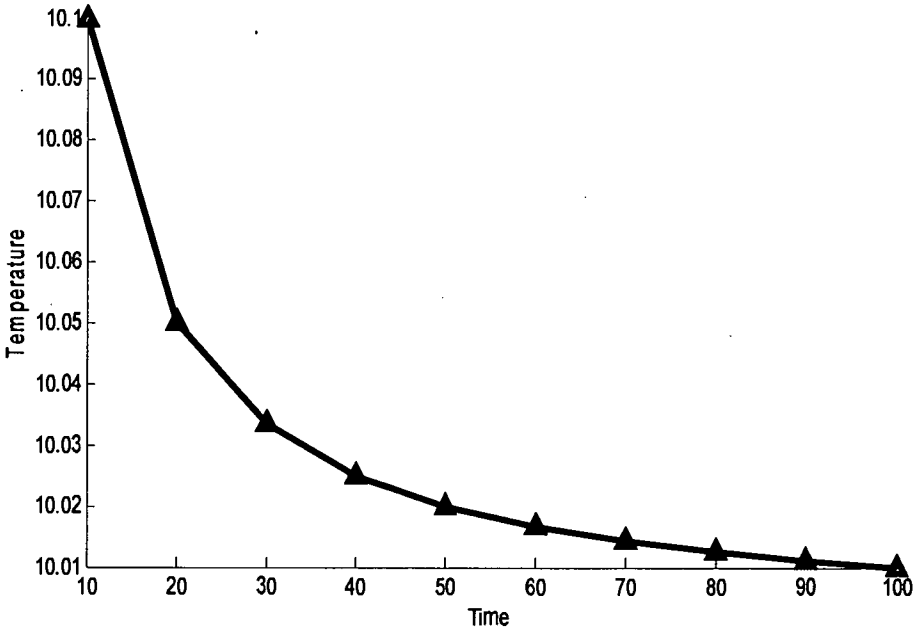


Figure 2:10 Exponential schedule

2.6 *Conclusions on annealing algorithms and cooling schedule*

Numbers of different algorithms were discussed and its various possible cooling schedules were studied in detail. The various schedules include: exponential schedule, Boltzmann's fast simulating algorithm schedule, conventional annealing schedule, adaptive schedules, time and energy variant schedules and Johnson's and Lundy's schedule. Implementing anything but simple schedules is a difficult task because of the choice of initial and final temperature. However, for our application which involves many agents, implementing an adaptive schedule gives better results.

3 Path Finding and Search Algorithms

Unmanned vehicles are used in teams to detect stationary and moving targets. Path planning is the art of deciding the path for the UAV, to manoeuvre to reach the preferred or selected goal. Path planning is to decide which route to take and act accordingly. Therefore, there are two phases involved in path planning, namely the decision and then the action phase. However, in the path planning algorithms, these phases are not explicitly known. The type of path planning varies for different types of targets. If the targets or the goals are stationary, then the entire path can be planned and subsequently the unmanned vehicles can manoeuvre along the planned path. However, in a dynamic environment, where the targets are not stationary, then prior planning of the path will not be applicable. In this case, dynamic planning of the next location move selection has to be adopted. Prior path planning and dynamically changing path planning has various algorithms. Some of the path finding algorithms studied are; Dijkstra's algorithm, Bellman Ford's algorithm, Floyd Warshall's algorithm and A Star algorithm.

3.1 *Networks and Graphs*

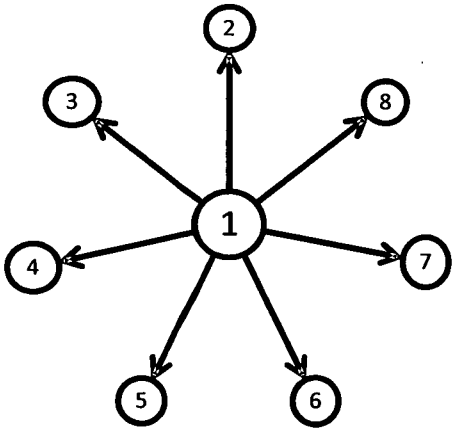


Figure 3:1 A simple cyclic network

A network can be represented as a graph. A graph has one or more nodes or vertices connected by edges or arcs. Graphs and networks can be used to model

systems such as, but not limited to communication systems, vehicle routing, transportation systems and process flow.
A graph, G= (V, E) has V number of vertices and E number of edges.

In Fig 3.1, 1, 2...8 are the nodes and the lines connecting these are the edges. The edges are directed and these directed edges are called arcs. When the edges are directed then the graph is a directed graph or a digraph. However when the edges are not directed as in case of Fig 3.2, then the graph is called as graph.

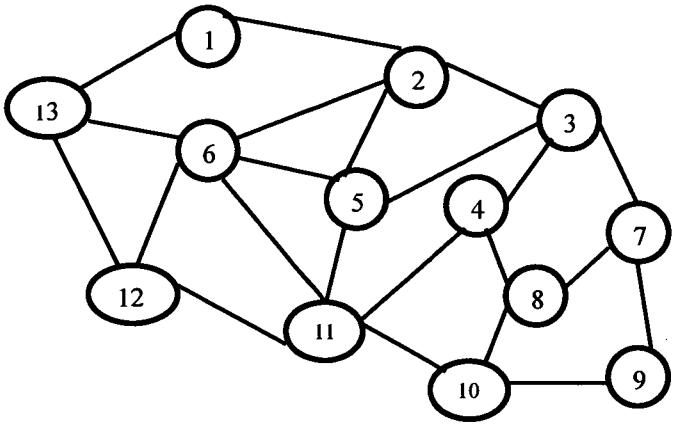


Figure 3:2 An example network

In the Fig 3.3, the weight of the arc is indicated next to the arc. Each node or the vertex is the number inside the circle. The weight indicated near the arcs is also called as cost. Therefore in Fig 3.3 to move from node a to node b, the cost is 5. This cost can rely on one or more parameters. Some of these parameters are time, monetary cost, distance, sensing range, terrain etc.

3.1.1 Shortest Path

A graph represents the topology of a network. A graph can be used to study the various network qualities and parameters. For example, if the distance between any two or more nodes in the network has to be determined, then to reach node c from f, there are three or more routes possible.

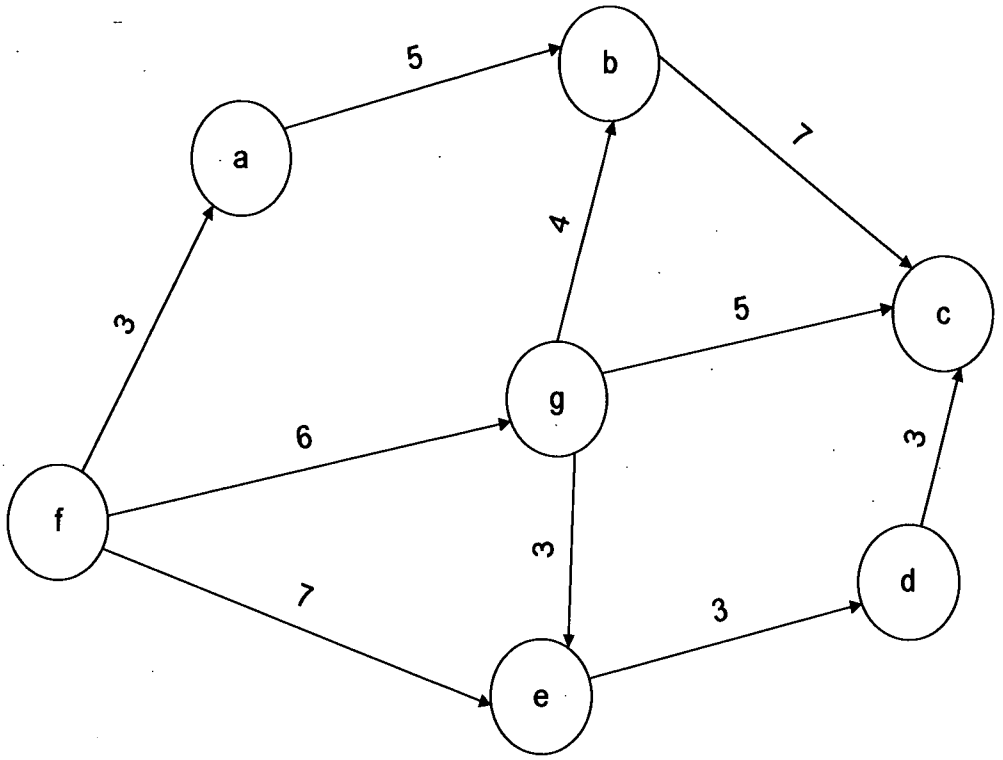
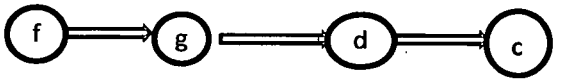


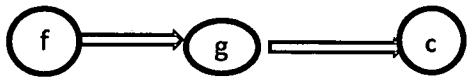
Figure 3:3 Directed graph

Some of the routes are









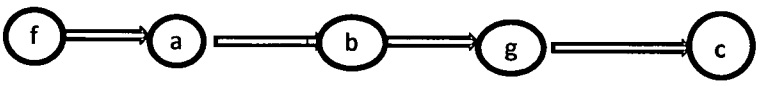


Figure 3:4 Various paths from source to destination

We consider all the possible paths or routes from f to c, to find the shortest path. For the path f, e, d, and c the total cost is 7+3+3=13.The total cost for the path f, g and c is 6+5=11. The path f, g, d, and c the total cost is 6+3+3=12. The total cost for the path f, a, b and c is 3+5+7= 15. The total cost for the path f, a, b, g and c is 3+5+4+5=17. Among all the above routes, the route f, g and c is considered to be the shortest path, not because it takes the minimum number of hops but, because the cost is minimum which is equal to 11.

In the above diagram, we have seven vertices and nine arcs. Therefore, it is easy for us to evaluate the shortest path considering all possible paths. However, when the number of nodes or vertices is large, then, it becomes impossible to evaluate manually the possible shortest path in the given time. A path through a graph is a traversal of the consecutive vertices along a sequence of edges. The vertex at the end of one edge in the sequence must also be the vertex at the beginning of the next edge in the sequence. The vertices that begin and end the path are termed the initial vertex and terminal vertex, respectively. Each vertex within the path which has two neighbouring vertices must also be adjacent to the vertex. The length of the path is the sum of the weights of edges, which are traversed along the path. The best path is the path which has the minimum sum of the weights or cost from the source to the destination. The loop is a path in which the initial vertex of the path is also the terminal vertex of the path. By removing all the loops, the path can be considered as an elementary path therefore, there will no longer be any repeated vertices.

A graph can be represented as an adjacency matrix A, in which each element (i, j) represents the edge between the elements i and j.

Aij =1, if there is an edge between node i and j and otherwise

Aij = 0.

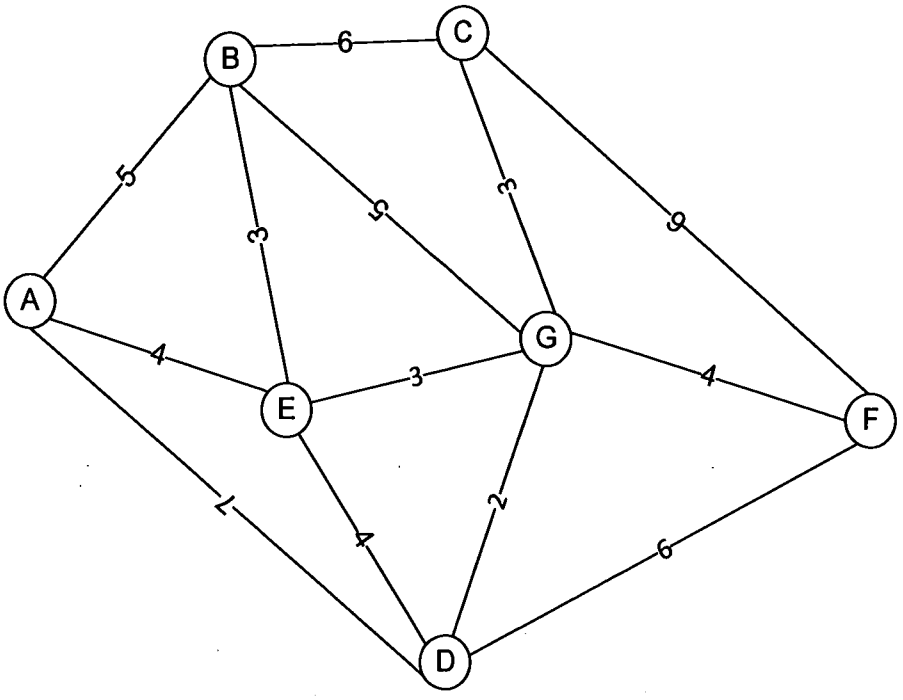


Figure 3:5 Weighted graph

For the graph shown in Fig . The adjacency matrix can be formed as given A[i][j] =1 in adjacency matrix denotes that there is a path between the two nodes. The first row denotes the path existence between A and the rest of the nodes .The second row of the matrix denotes the existence of path between the node B and the rest of the nodes. Next, the weighted cost matrix is obtained from the network graph Fig 3.5.

Here A[i][j]=actual cost denoted in the graph, for example A[1][2]=5 denotes the cost of communication between A and B .If there is no direct communication link between the two nodes then the cost is shown as 8.This matrix represents a network. It is passed to the different path finding algorithms to test the capability of the algorithm to find the shortest path or to search the destination node.

Here A[i][j]=actual cost denoted in the graph, for example A[1][2]=5 denotes the cost of communication between A and B .If there is no direct communication link between the two nodes then the cost is shown as ∞.This is a matrix which represents a network. It is passed to the different path finding algorithms to test the capability of the algorithm to find the shortest path or to search the

destination node. The shortest path algorithm is either the minimum cost algorithm or the maximum flow minimum cost algorithm.

where

$$A[i][j] = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

Figure 3:6 Adjacency matrix

and the resultant matrix can be written as

$$A[i][j] = \begin{bmatrix} \infty & 5 & \infty & 7 & 4 & \infty & \infty \\ 5 & \infty & 6 & \infty & 3 & \infty & 5 \\ \infty & 6 & \infty & \infty & \infty & 6 & 3 \\ 7 & \infty & \infty & \infty & 4 & 6 & 2 \\ 4 & 3 & \infty & 4 & \infty & \infty & 3 \\ \infty & \infty & 6 & 6 & \infty & \infty & 2 \\ \infty & 5 & 3 & 2 & 3 & 2 & \infty \end{bmatrix}$$

Figure 3:7 Path finding matrix

3.2 *Shortest Path Algorithms*

Given a graph G (V, E) with V vertices and E arcs in which any arc b between nosed A and B is associated with a weight n, then n represents the cost to traverse from A to B. There may be one or more number of paths present between any two vertices in a graph. The shortest path between any two vertices is the path with least cost. There are various shortest path algorithms present, however, we will discuss a few of those and compare their performance in a real time network.

3.2.1 Applications of the shortest path algorithms

The applications of the shortest path algorithms are not limited to , but includes transportation planning, knapsack problem, travelling sales man problem, scheduling timetables, routing in telecommunication and computer networks, production lot sizing. As the applications are so extensive, the shortest path algorithm is the more studied among the network optimisation problems.

3.2.1.1 Transportation Planning

When a heavy loaded vehicle like a truck has to travel from one state to another, it might have to consider some restrictions in each state like the prohibited load restrictions in several routes, restricted times for taking particular paths, etc. Based on all the restrictions the trucker has to plan the shortest path possible [7] to reach the destination in time.

3.2.1.2 Knapsack problem

This problem arises in resource allocation in financial constraints, resource allocation in telecommunication networks, camper's knapsack fitting, selecting experiments for space shuttle missions, etc. In the telecommunication networks, the various network parameters with constraints are bandwidth, the throughput, the traffic in the network at any particular time, the delay, the jitter, the security, the distance etc. Based on the restraints, a group of network resources are utilised at a particular time which does not exceed the actual available network resources. Each of these parameters, i has a value of d and a weight of w. Therefore the problem is

$$max \sum_{i=1}^{n} d_i x_i \qquad [3.1]$$

subject to

$$\sum_{i=1}^{n} w_i x_i \leq W \qquad [3.2]$$

where

$$x_i = 0, 1$$

The object of the knapsack problem is to calculate total weight w which should be lesser than the available weight W of the network. In the telecommunications

network example we have considered, the utilised network resources should be lesser than the available network resources of the network.

3.2.1.3 Travelling sales man problem (TSP)

Given, n cities the travelling sales man has to visit n cities in his tour, but each city has to be visited just once and gets back to the city he started from. The problem is which would be the shortest route the travelling sales person will take to cover all cities just once and completing his tour.

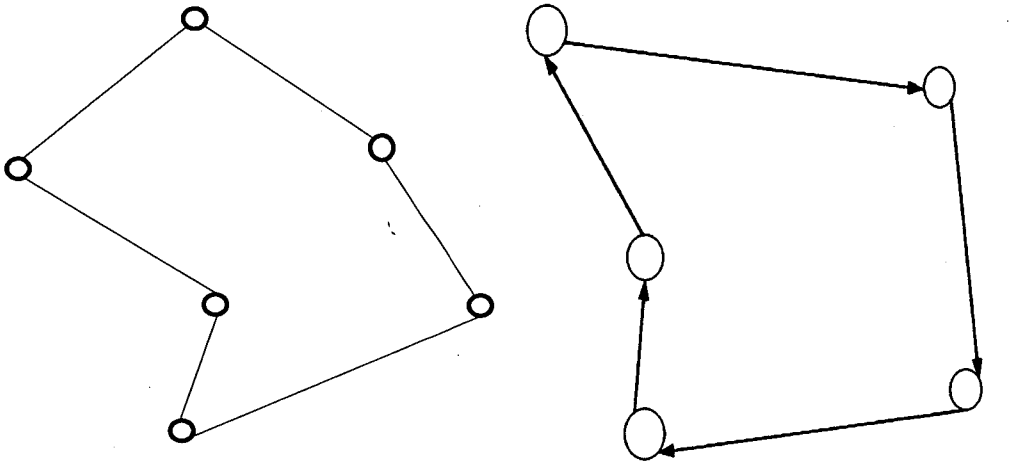


Figure 3:8 Travelling sales man directed and undirected

Some of the TSP is symmetric, when the distance between the cities A and B is the same as the distance from B to A. In this case, the lines need not have arrows. However, when this is not the case, the distance between A and B is not the same as in the reverse direction B to A, then it is not symmetric TSP therefore, arrows are present on the lines joining the two places. When there are n cities then the solution is $\frac{(n-1)!}{2}$. When the number of cities increases extensively then, the search becomes exhaustive and almost impractical.

3.2.1.4 Production lot sizing

The product demand may change deterministically over a period of time. The production has to be scheduled accordingly. Given, demands at period j are D_1, D_2 .....D_n. The total cost involved in producing is

$$W_{kj} = A_j + C_j Q_j + \sum H_j(I_j + \cdots + I_{k-1}) \qquad [3.3]$$

Where A_j is the setup cost incurred at time j, C_j is unit production cost, Q_j is the amount produced in time j, I_j is the amount held over inventory period j to j+1, and H_j is per unit holding cost. This cost has to be minimized for the production scheduling problem.

3.2.1.5 Routing in telecommunication networks

Consider a network as given below.

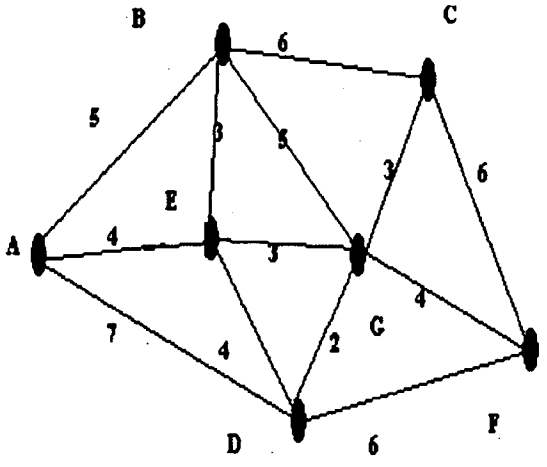


Figure 3:9 Telecommunication network as a graph

There are various paths available between any two nodes in the network. The various cost factors involved here are based on the network quality parameters involved at any particular time. Considering the traffic of the network, the bandwidth availability, the link capacity, time, monetary cost and various other factors, the weights are determined. The total cost between the source and destination should be as small as possible.

3.3 *Pathfinding and search algorithms*

Given a connected graph G = (V, E) to find a shortest path from s to each vertex v in the graph, the following parameters are passed to the algorithm to find the shortest path. G denotes the graph with weights in the form of matrix, and s is the

source node from which the shortest path to all other nodes is determined. Pathfinding helps in looking ahead and finding the shortest or the best path. Most of the path finding algorithms makes use of graphs. Some of the pathfinding algorithms we study in this chapter are distance vector algorithm, Dijkstra's algorithm, Floyd-Warshall's algorithm and A star algorithm.

3.3.1 Distance vector algorithm

Distance vector routing (DVR) algorithm also known as Bellman Ford algorithm [21]is an iterative process of exchanging information and this will continue until no more information is exchanged between the neighbourhoods. This algorithm enables each node to receive some information from one or more of its directly attached neighbouring nodes. The Bellman-Ford distance-vector routing algorithm is used by routers on internet works to exchange routing information about the current status of the network and how to route packets to their destinations. The algorithm basically merges routing information provided by different routers into lookup tables. It provides reasonable performance on small- to medium-sized networks, but on larger networks, the algorithm is slow at calculating updates to the network topology. In some cases, looping occurs, in which a packet goes through the same node more than once.

Bellman-Ford algorithm solves the single-source shortest-path problem, in the general case the edges of a given digraph can have negative weight as long as G contains no negative cycles. This algorithm, like Dijkstra's algorithm, uses the notion of edge relaxation but does not use with greedy method. The algorithm progressively decreases an estimate cost on the weight of the shortest path from the source node to each node in the network until it achieves the actual shortest-path. The computational time of Bellman-Ford algorithm increases exponentially with the increase in the number of nodes.

Routers that use this algorithm, have to maintain the lookup tables, which gives the distances and shortest path to sending packets to each node in the network. The information in the lookup table is always updated, by exchanging information with the neighbouring nodes. The amount of data in the table equals to that of all nodes in networks. The columns of table represent the directly attached

neighbours whereas the rows represent all destinations in the network. Each data contains the path for sending packets to each destination in the network and distance or time to transmit on that path or the cost. The measurements in this algorithm are the number of hops, latency, and the number of outgoing packets.

Bellman-Ford (G,w,s) G-Graph, w-weight, s-source
for each vertex ∈ Vertex[G]
dist [v]←∞ dist[s]← 0
for i←all vertices in G do
for each edge (u,v) ∈ Edge[G] do
Relax (u,v,w)
for each edge (u,v) ∈ Edge[G] do
if dist[v] > dist[u]+weight(u,v) then return *false*
return *true*

Figure 3:10 Bellman Ford algorithm

Distance vector routing protocol is used in routing information protocol (RIP) to find the shortest path. RIP works as is given below

1. Every node calculates its distance with its neighbours and stores the routing table and also passes on the routing table to its neighbours.
2. The neighbour updates its routing table from the table it received from its neighbours.
3. When all the node information is updated, then it calculates the shortest path between any two nodes in the network.

Though the basic structure of Bellman Ford's algorithm is similar to Dijkstra's algorithm, the main advantage is it can obtain the shortest path between any two nodes in the network, even when the graph has a negative edge. However, the scalability is very limited to this algorithm. When a node is added into a network, it takes time for all the nodes to update its routing table. When there is a node failure, or when a node leaves a network, then there is counting to infinity

problem that arises. Therefore it is not a scalable algorithm. When there are many nodes in the network, then the time taken for calculating the distance for the first time is very large when compared with the Dijkstra's algorithm. The time for running this algorithm is $O(VE)$.

3.3.2 Dijkstra's Algorithm

Dijkstra's algorithm [22] creates labels associated with vertices. The Dijkstra's algorithm solves the single source shortest path problem for a graph which is not directed. Given, a source node or a vertex this algorithm is able to find the distance of this node to every other node in the network. These labels represent the cost from the source vertex to that particular vertex. Within the graph, there exist two kinds of labels: temporary and permanent. The temporary labels are given to vertices that have not been reached. The value given to these temporary labels can vary. Permanent labels are given to these vertices that have been reached and their cost to the source vertex is known. The value given to these labels is the cost of that vertex to the source vertex. For any given vertex, there must be a permanent label or a temporary label, but not both.

The algorithm begins at a specific vertex and extends outward within the graph, until all vertices have been reached. More simply, Dijkstra's algorithm stores a summation of minimum cost edges whereas Prim's algorithm stores at most one minimum cost edge. Dijkstra's algorithm determines the costs between a given vertex and all other vertices in a graph. This may be useful to determine alternatives in decision making.

The algorithm begins by initializing any vertex in the graph (vertex A, for example) a permanent label with the value of 0, and all other vertices a temporary label with the value of 0. The algorithm then proceeds to select the least cost edge connecting a vertex with a permanent label (currently vertex A) to a vertex with a temporary label (vertex B, for example). Vertex B's label is then updated from a temporary to a permanent label. Vertex B's value is then determined by the addition of the cost of the edge with vertex A's value. This process is repeated until the labels of all vertices in the graph are permanent.

Dijkstra's algorithm solves the single-source shortest path problem in weighted graphs. Dijkstra's algorithm starts from a source node and in the next iteration it adds another vertex to the shortest-path spanning tree. This vertex is the point closest to the root which is still outside the tree. Watch as the tree grows by radiating out from the root. As it is not a breadth-first search; we do not care about the number of edges on the tree path, only the sum of their weights.

Dijkstra(G, w, s) G-Graph, w-weight, s-source
for each vertex in Vertex[G]
dist[v]←∞ dist[s]←0
S←empty set
while Vertex[G] is not an empty set
u←Extract_Min(Vertex[G])
S←S union {u} **for each** edge (u,v) outgoing from u
if dist[u]+weight(u,v)<dist[v]
dist[v]←dist[u]+weight(u,v)

Figure 3:11 Dijkstra algorithm

This algorithm works as follows

1. Every node is assigned a distance value. Next, set the initial node to 0 and ∞ to all other nodes.
2. All nodes are set as not visited and the initial node as current node.
3. From current node to all not visited nodes, calculate the distance.
4. When distance from the current to all other neighbouring nodes is done, then mark it as visited.
5. Next, consider the next not visited node as current, and continue from step 3 till all the nodes are visited.

This algorithm is able to determine the distance of a node to every other node in the network. Though this algorithm can find the distance of a single node to every other node in the network, the time taken for computation in determining the

distances is considerably high, especially when the number of nodes in the network increases. The time for running this algorithm is O(E+ V log V).

3.3.3 Floyd-Warshall's algorithm

The Floyd-Warshall's algorithm [24] is a graph analysis algorithm which is used to find the shortest path in a weighted and directed graph. The Floyd-Warshall's algorithm compares all possible paths in a network between any two vertices or nodes and then gives the shortest path between the two vertices as optimal solution. The Floyd-Warshall Algorithm is an application of Dynamic Programming (DP)[16] is a technique that takes advantage of overlapping sub problems, optimal substructure, and trades space for time to improve the runtime complexity of algorithms. In bottom-up dynamic programming, we start from smaller cases and store the calculated values in a table for future use, which is an effective strategy to most dependency-based problems. This avoids calculating the sub problem twice. Dynamic programming generates all enumerations, or rather, cases of the smaller breakdown problems, leading towards the larger cases, and eventually it will lead towards the final enumeration of size n, this will give the shortest distances between any two nodes, from which shortest paths maybe constructed. The time for running this algorithm is $\Theta(n3)$.

The shortest path can be defined by a recursive formula
Shortestpath(i, j, k) = min(shortestpath(i, j,k-1), shortestpath(i,k, k-1)+ shortestpath(k, j, k-1))
Shortestpath(i,j) = min(shortestpath(i,j),shortestpath(i,k) + shortestpath(k,j))

```
Floyd-Warshall (G,w,s) G-Graph, w-weight, s-source
init path[ ][ ];
for k = 1 to n
for each (i, j ) in (1 . . . n)
 path[i][j]=min(path[i][j],
path[i][k]+path[k][j]);
 end
```

Figure 3:12 Floyd Warshall algorithm

Some of the applications of Floyd-Warshall's algorithm are

1. To find shortest path in directed graphs
2. Testing if an undirected graph is a bipartite
3. Pathfinding in networks
4. Inversion of real matrices
5. Obtain maximum bandwidth paths in real time networks
6. Transitive closure of directed graphs

3.3.4 Johnson's algorithm

Johnson's algorithm [8] is an all pair shortest path algorithm unlike the Dijkstra's algorithm. It is used on graphs which are directed, sparse and weighted. It uses Bellman Ford algorithm to remove the negative edges, and then uses the Dijkstra's algorithm to work on the transformed graph.

Find a function h: $V \rightarrow$R
such that $w(i, j) \geq 0$ for all$(i, j) \in E$
by using Bellman-Ford to solve the difference constraints
$h(j) - h(i) \leq w(i, j)$, or determine that a negative-weight cycle exists
Run Dijkstra's algorithm using w from each vertex $i \in V$ to
compute $\delta(i,j)$for all $j \in V$
For each $(i, j) \in V \times V$, compute
$\delta(i, j) = \delta(i, j) - h(i) + h(j)$

Figure 3:13 Johnson's algorithm

Johnson's algorithm works as follows:

1. A new node k is added to the graph, connected by zero-weight edge to each other node.

2. The Bellman-Ford algorithm is used, when starting from the new vertex k, to find for each vertex j the least weight h (j) of a path from k to v. If this step detects a negative cycle, the algorithm is terminated.
3. The edges of the original graph are reweighed using the values computed by the Bellman-Ford algorithm: an edge from *u* to *v*, having length w (i,j), is given the new length w(i,j) + h(i) –h(j).
4. For each node *s*, Dijkstra's algorithm is used to find the shortest paths from *s* to each other vertex in the reweighed graph.

The time to run this algorithm is O (VE+ V^2 log V).

3.3.5 Best-first search

The Best-first search [23] is a search algorithm which searches the graph for the best score node. The best score node is the most promising node. The Expansion is done from this promising node. Depending on the goodness of the heuristic, this best first search is similar to the depth first search algorithm. When the heuristic is always zero, then the best first algorithm is same as the breadth first search algorithm. This best first search algorithm is a greedy algorithm; therefore it does not look into the long term consequences, instead into the short term advantages.

1. Initialize with OPEN node
2. Find the best node
3. Pick best node on OPEN

   generate successors

   for each successor DO 3
4. If generated; evaluate and add it to OPEN; record its parent
5. If not; change parent and obtain cost of obtaining any successor node
6. Continue until goal is found and no nodes are in OPEN

Best- first search is a simplified form of A* algorithm.

3.3.6 Breadth-first search (BFS)

The input given to this algorithm is a graph G, and a source vertex and the output is obtained as edges labeled as discovery and cross edges in the connected component. Breadth first search [8] algorithm traverses a connected component of a given graph and defines a spanning tree. Breadth first search starts at a given vertex, which is at level 0. Next, it traverses and visits all vertices at level 1. Then, it visits all vertices at level 2. This goes on; breadth first search algorithm stops when every vertex has been visited.

The breadth- first search algorithm finds its application in

1. Testing whether graph is connected.
2. Computing a spanning forest of graph.
3. Computing, for every vertex in graph, a path with the minimum number of edges between start vertex and current vertex or reporting that no such path exists.
4. Computing a cycle in graph or reporting that no such cycle exists.

The total computation time is O (V+E) for this algorithm.

Breadth first search (G, S)

Create a Queue Q
INSERT S into Q
While Q is not empty do
  for each vertex *v* in Q do
    for all edges e incident on v do
      if edge e is unexplored then
        let *w* be the other endpoint of e.
        if vertex *w* is unexpected then
          - mark e as a discovery edge
          - insert *w* into Q
        else
          mark e as a cross edge

Figure 3:14 Breadth-first search

3.3.7 Depth-first search (DFS) algorithm

The depth first search uses the backtracking technique. DFS algorithm starts at vertex S in graph G, which is the current vertex. Then this algorithm traverses the graph by any edge (i, j) incident to the current vertex i. If the edge (i, j) leads to an already visited vertex v, then we backtrack to current vertex u. If, on other hand, edge (i, j) leads to an unvisited vertex j, then we go to i and j which becomes our current vertex. It proceeds until it reaches to a point where it has no unvisited node. At this point it starts back tracking. The process terminates when backtracking leads back to the start vertex.

The applications of a depth first algorithm are

1. Find a path from one vertex to another
2. Whether or not graph is connected
3. Computing a spanning tree of a connected graph.

The input to the algorithm is graph G, and vertex v and the outcome are the edges labeled as discovery and back edges which are visited while backtracking in the connected component.

Depth first search (G, *v*)

For all edges e incident on v do
  If edge e is unexplored then
    Return the end point of e distant to v
   If vertex w is unexplained then
     - mark e as a discovery edge
     - Recursively call DSF (G, w)
   else
     - mark e as a back edge

Figure 3:15 Depth-first search

The running time of this algorithm is θ (V + E).

3.3.8 A star algorithm

The A* [23] alled as 'A-star' is like other graph searching algorithms in that it can potentially search a huge area of the map. It is like Dijkstra's algorithm in that it can be used to find a shortest path. It is similar to breadth first search (BFS) in that it can use a heuristic to guide itself. In the simple case, it is as fast as BFS. Since in the worst case breadth-first search has to consider all paths to all possible nodes the time complexity of breadth-first search is increased depending on the number of nodes and edges in the graph. The best-established algorithm for the general searching of optimal paths is A*. The best-established algorithm for the general searching of optimal paths is A*. This heuristic search ranks each node by an estimate of the best route that goes through that node.

```
A* (G,s,w) G-Graph, w-weight, s-source
var closed=the empty set
var q=make_queue (path(start))
while q is not empty
 var p=remove_first(q)
  var x=the last node of p
 if x in closed
 continue
  if x=goal
 return p
 add x to closed
for each y in successors(p)
enqueue (q, y)
return failure
```

Figure 3:16 A star algorithm

The typical heuristic formula is expressed as:

$$f(n) = g(n) + h(n) \quad [3.4]$$

where f (n) is the score assigned to node n, this is the estimate of the best solution that goes through n, g(n) is the actual cheapest cost of arriving at from the start node to n and h(n) is the heuristic estimate of the cost to the goal from n.

The A star algorithm works as follows:

1) Add the starting node to the OPEN list.
2) Repeat

    a. Search for lowest cost node on the OPEN list. This is the current square.
    b. Switch it to the CLOSED list.
    c. For all nodes adjacent to this current node
        - If it is no path to the adjacent node or if it is on the CLOSED list, ignore it else
        - If it is not on the OPEN list, add it to the OPEN list. Make the current node the parent node of the adjacent node. Record the heuristics f (n), g(n) and h(n) of the node.
        - If it is on the OPEN list already, check to see if this path to that node is shorter, using g (n) cost as the measure. A lower g (n) cost means that this is a shorter path. Then, change the parent of the node to the current node, and recalculate the g (n) and f (n) values of the node. Keeping OPEN list sorted by f (n) score, then resort the list to account for the change.

3) Stop when :

    - Target node is added to the CLOSED list, in which case the path has been found or

- If failed to find the target node and the open list is empty; then, there is no path.

4) Save the path. Working backwards from the target node, from each node to its parent node the reverse path is traversed until the starting node is reached.

The time for this algorithm to run is O (n).

3.4 *Conclusion and comparison of shortest path algorithms*

Different path finding algorithms were studied in order to formulate and study a search algorithm in an intelligent environment. The path finding algorithm used for the search depends on the number of nodes. The various algorithms were compared. For example, if the goal is to the south of the starting position, BFS will tend to focus on paths that lead southwards. It shows that BFS can find paths very quickly compared to Dijkstra's algorithm.

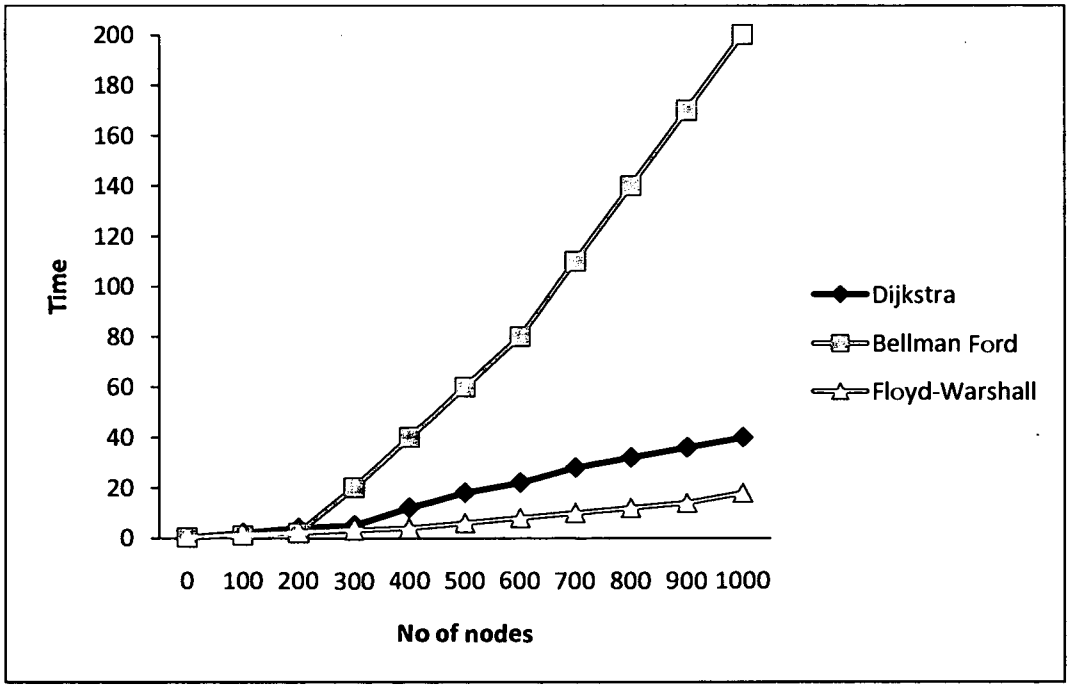


Figure 3:17 Comparison of computation time

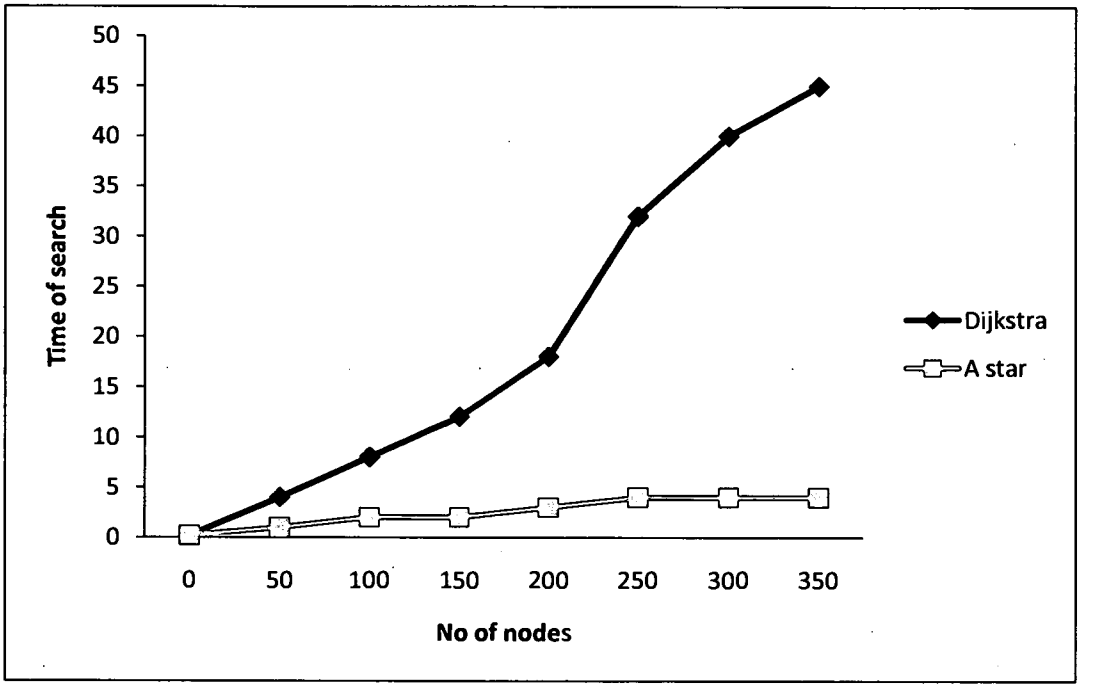
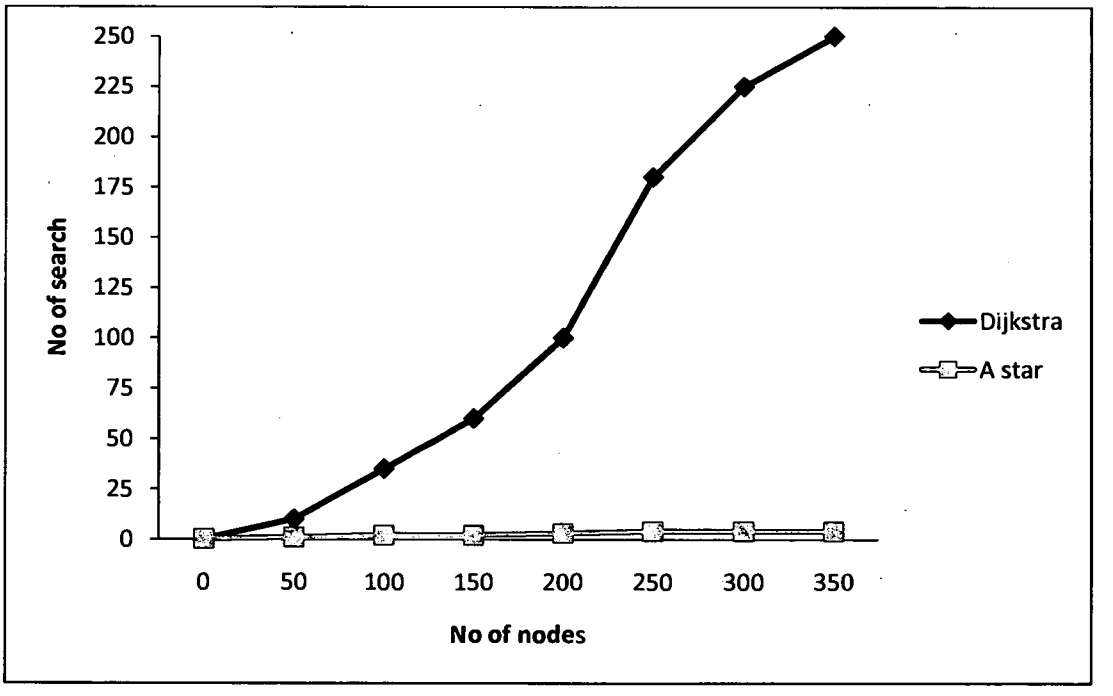


Figure 3:18 The time for search increases as the number of nodes increases



The trouble is that BFS is greedy and tries to move towards the goal even if it is not the right path. Since it only considers the cost to get to the goal and ignores the cost of the path so far, it keeps going even if the path it's on has become really

long. A* was developed to combine the virtues of the heuristic approaches like BFS and formal approaches like Dijkstra's algorithm. However, A* is built on top of the heuristic, and although the heuristic itself does not give a guarantee, A* can guarantee a shortest path. The experiments were done in MATLAB environment.

These search algorithms were studied and compared in order to be implemented in various environments. Once the destination is found using the path finding algorithm the information flow path can be decided.

4 Co-operative target observation and new simulated annealing

Teams of cooperating UAVs have the potential to: accomplish the missions in a shorter period, simultaneously accomplish many goals, cost less, be less detectable and more survivable than a single large vehicle; damage to a single UAV does not necessarily cause the entire mission to fault. The higher degree of coordination [27, 28] helps us to achieve better overall efficiency even though the Unmanned Aerial Vehicles usually have limited sensor capabilities relative to their large counterparts. The accomplishment of goal therefore relies heavily on cooperation between appropriate neighbours. In this chapter, we have used Simulated Annealing to achieve cooperation between the UAVs.

Cooperation [32-35] leads to reliability, robustness and distributed sensing. The cooperative target observation (CTO) is an area where multiple UAVs can work in coordination to monitor the target(s) continuously, or to stay in a position to reach a potential target as quickly as possible. Our paper deals with the optimisation of the position of each UAV so that it can monitor as many targets as possible. The optimised position helps the UAV(s) to move towards any appropriate target as the need arises. There are various methodologies used to detect and monitors the targets [36-39]. To achieve optimised positioning of UAVs, such that they watch as many targets as possible, we use Modified Simulated Annealing Algorithm.

The ultimate goal of moving the UAVs to optimised positions are: To place the onboard sensors such that the targets are in view; and for the UAVs to move to new targets or those observed by other UAVs as quickly as possible when the need arises. There are various strategies for exploring the search space, in order to optimize the position of the autonomous vehicles. The Modified Simulated Annealing Algorithm works in a similar manner to Simulated Annealing algorithm by accepting the worse solution at high temperature, but when both positions observe the same number of targets, checks the quality of the offspring and the current position's position with Euclidean distance as heuristics. We have

combined the Simulated Annealing Algorithm [31] and the improvement for checking the quality of the observing agent position [26]. This combined algorithm has been applied on the tested for cooperative target observation in Multi-Agent Simulation Environment (MASON) [30].

This chapter is organized as follows. Section 1 presents the network team control and the advantages of it. Section 2 focuses on various CTO the basic simulated annealing algorithm and the new simulated annealing algorithm and how it is used to solve various combinatorial problems. We also introduce the concept of Cooling Schedule, and the importance of it to our application. It also concentrates on the actual algorithm we use for our application. Section 3 deals with the model description and the actual implementation block diagram.

4.1 Networked Team Control

There is significant work and research in mobile ad hoc networks and in UAV team control, but little work attempts to combine these into a single framework where both systems are simultaneously designed or analyzed. Network communications between the team is required for performance enhancement of the team [7].Shared knowledge is a necessary condition for coordination [8] but little work has been done with communication as control parameter[9,48-50]. The optimal way of knowledge sharing for team collaboration needs an intelligent communication protocol [12].

In networked control systems used for UAV team control, team control is focused on ignoring the fact that individual vehicle motion would have significant impact on the overall performance of the network. Moreover it is often assumed that the underlying network is available continuously. In order to fulfil the vision of Next Generation Networks for UAV communications maintaining a real-time flow despite frequent topology changes and irregularity in movement and service further research is required [22]. While mobility has been extensively investigated in MANETs, it is generally presented as random mobility instead of controlled mobility. A networked control system[14-20] consists of numerous physical and computational agents that share information to close control loops over network

resources [14]. An "agent" could be an aircraft, an on-board sensor, a control module, the ground station or an ensemble of such devices.

There are numerous technical challenges in developing viable cooperative planning and control methods for teams of UAVs. A significant issue is the level of complexity involved in planning and coordinating a team of vehicles with varying capabilities and competing interests so that a team objective is attained in an optimal manner [8]. This is especially true for applications with unknown and rapidly changing environments, requiring UAVs to continuously react and re-plan. UAV team management and control covers a wide area of research interests including mission planning and re-planning [27], optimal sensor (UAV) placement [28], task allocation [29], formation control [30], obstacle avoidance [3], and fault tolerant distributive control [31-33]. However, only a little amount of work [9] has included network performance and network service guarantee as a team objective.

Each UAV must use onboard sensors and processing to capture, represent, and interpret the relevant environmental cues (eg. location, geometry, spectral content, etc), and then autonomously combine and manipulate this information such that the result is a series of control actions that represent both the individual and collective priorities of the systems, subsystems, and payloads within the cooperative. Some of the considerations [78] for design of the communication algorithm are:

a. The number of UAV's is potentially large
b. Scalability is desirable as UAV's may leave or join the co-operative or teams
c. Humans need to set goals for and interact with the UAV's and their payloads
d. Each individual UAV in a team needs to possess its own complex behaviour
e. Humans can be supervisors and/or controllers of UAV's and/or payloads
f. Supervisors and UAV's are potentially distributed over a wide geographic area
g. Situational awareness events can require quick responses
h. There is a high probability of losing resources

4.1 *Co-operation, co-ordination and collaboration in unmanned vehicles*

Co-operation [32-35] of a system can be defined as the components of the system working together to achieve overall goal of a system. Collaboration can be defined as two or more system components working together by knowledge sharing to achieve a common goal. Coordination is the components of the system, acting willingly in harmonious combination to achieve a goal.

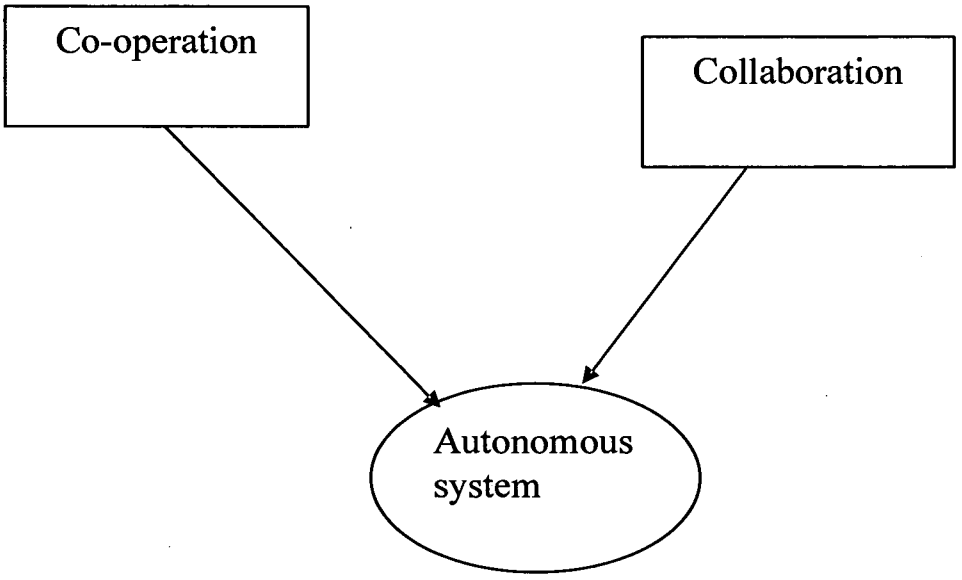


Figure 4:1 Co-operation and collaboration

4.1.1 Co-operation

An autonomous system has many components or groups [32-35]. These groups work together to achieve same goal. When agents form small group with each group member having different levels of knowledge, then the each of the group member act together to all the other members of the group can work on the same goal. In co-operative target observation, the expertise of each member of the

group might be different, like the sensor range, computing strength, etc. When a group is formed, the expertise of each is shared among the group members and the group works co-operatively to watch the targets continuously.

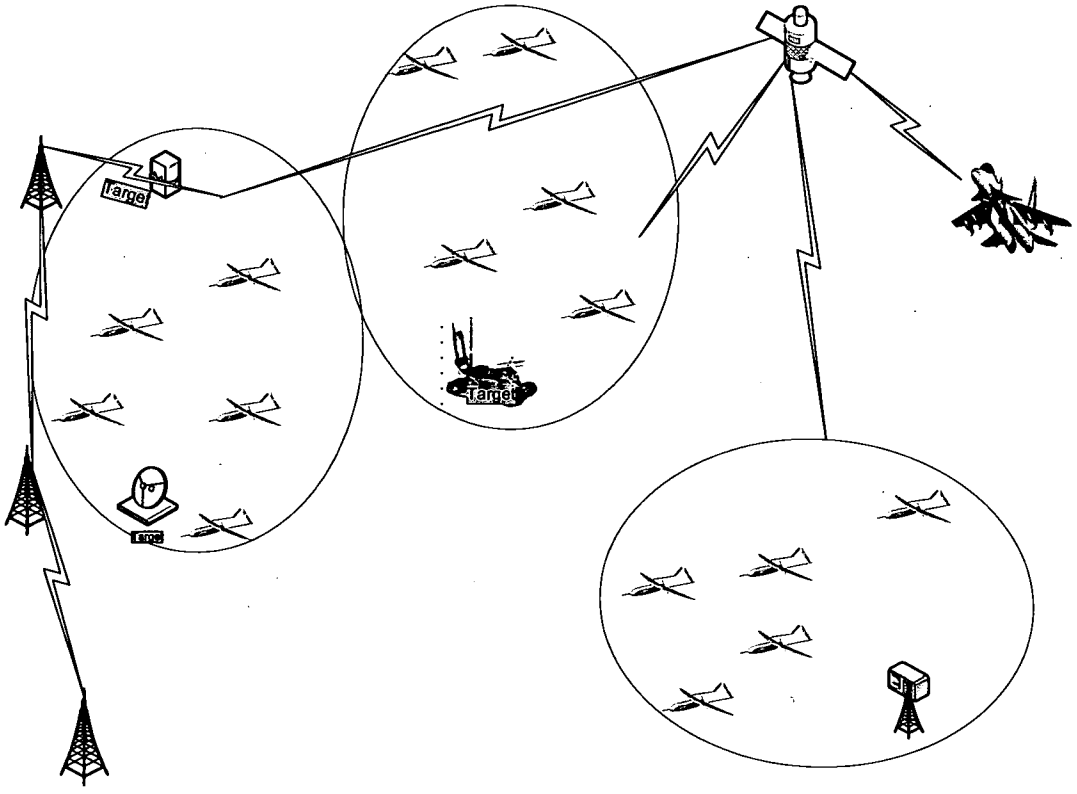


Figure 4:2 Co-operation of UAVs for target observation

The small group members which are the small UAVs have limited sensing range. However as they are acting in a group to watch the movement of the targets, the sensing range is increased.

The advantages of co-operative learning and working in a group are

1. Gain from other member's efforts and expertise
2. Improved overall performance, which is based on the every member's performance and group co-operation
3. If any member is lost in the action, that members work is done by any member or a few members in the group

4. Effort put in is much less by each member of the group, than if it has to be done by a single large counterpart.

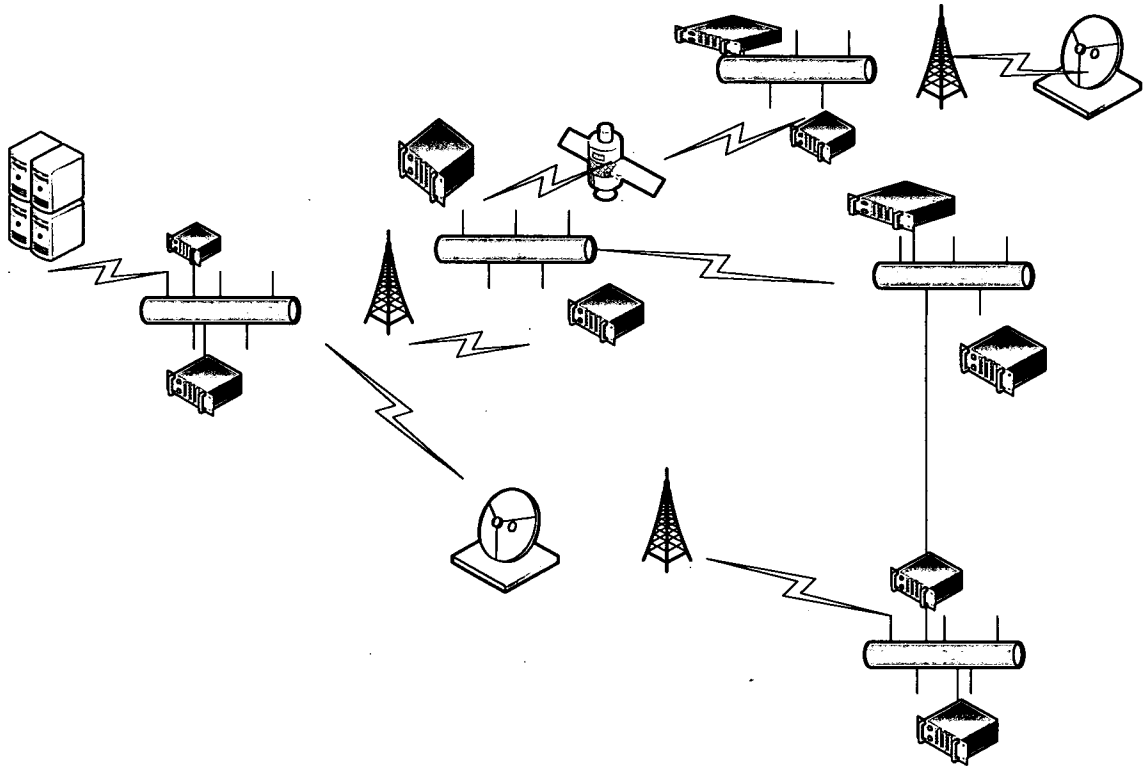


Figure 4:3 Importance of decentralised observation

4.1.2 Collaboration

Collaboration is shared knowledge or learning to achieve better decision making. Collaboration between the widely spread unmanned vehicles or combats, each has its own knowledge about the closest surrounding tasks. However there is a need for this knowledge to be shared to enable better decision making. The knowledge about the neighbours and also the knowledge gained by the neighbours can be obtained by collaboration. This enables to gain knowledge and make right and timely decisions.

Message sharing or knowledge sharing can be done by any agent communication methods. Each member shares its own information like the expertise, sensor

range, routing history, trusted neighbours, etc to its immediate neighbours. These neighbours can pass on this information to its neighbours and subsequently can be passed on to the entire group. Thus, two or more members of the group work together at an intersection to achieve the goal. Therefore collaboration needs a structured communication technique for knowledge sharing.

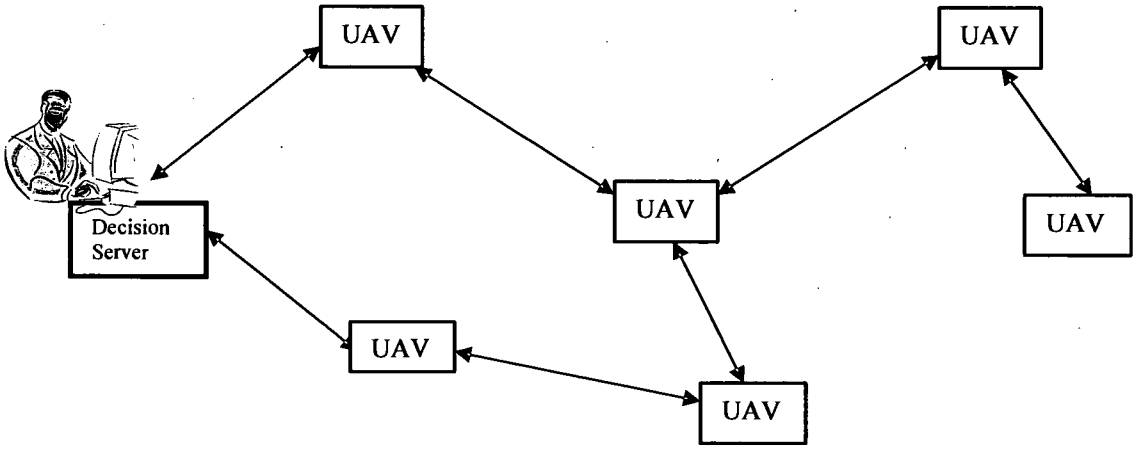


Figure 4:4 Collaboration with human intervention

Collaborative knowledge sharing occurs as a result of interactive message passing to do a common task.

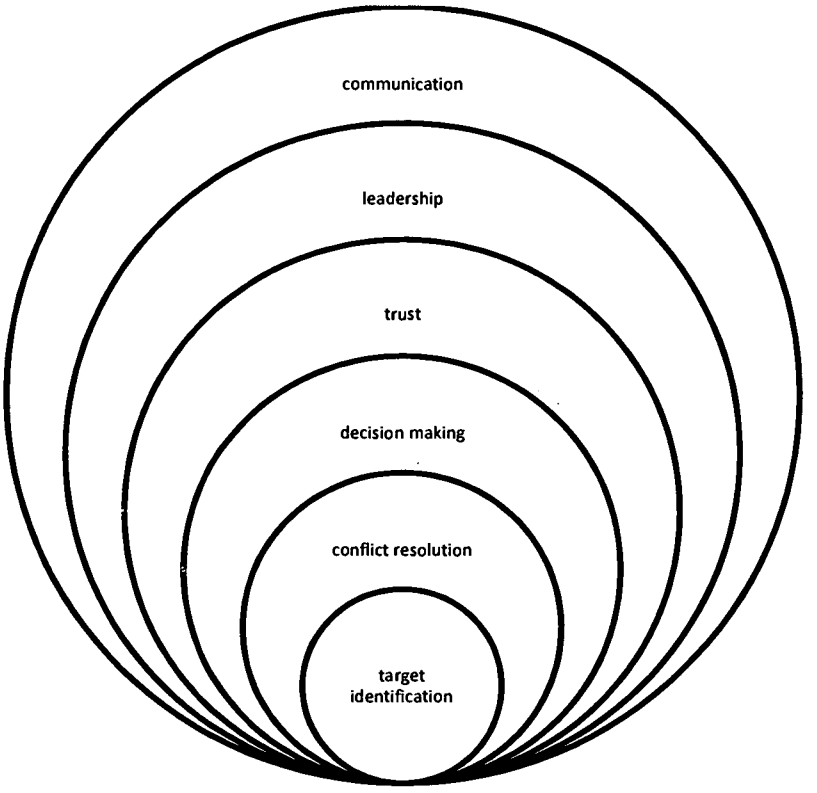


Figure 4:5 Tasks in successful collaboration

The characteristics of successful collaboration in target observation are:

1. Leadership
2. Decision making
3. Trust
4. Communication
5. Conflict resolution
6. Target identification

4.2 *Related work*

Simulated Annealing Algorithm has been used to solve many combinatorial optimization problems and some continuous variable problems. The applications include but are not limited to Travelling Salesman Problem (TSP)[34], Capacited Vehicle Routing Problems (CVRP)[48], Job Scheduling Problems[47], Timetabling Problems[41], and Selection of Communication Protocols/Standards[49].

There is a considerable amount of prior work in the area of multi target observation and cooperation. The Cooperative multi-robot observation of multiple moving targets (CMOMMT) [42] is based on the distributed motion control of the robots that have onboard sensors. The algorithm minimises the time for target to escape the detection region of the robot's sensor. To achieve this it uses a robot motion control mechanism based on distributed architecture and an adaptive action to achieve fault tolerance cooperative control for multiple robot teams. CMOMMT has been simulated and implemented on multi-robot teams.

P-CMOMMT [46] is based on CMOMMT but uses different competitive or cooperative relationship in robots. The P-CMOMMT is said to over come the conflicts between the individual benefit and the overall benefit. The individual benefit is the number of targets observed by the individual agents and the overall benefit is the total number of targets observed by the group. The overall benefit is slightly less than the sum of the individual benefits as there is overlapping

observations of the same targets by different observers. To achieve this they use the cover rate of robots.

An algorithm allowing the cooperation of multiple robots with the centralized human operator [37] has three layers of hierarchy architecture. The tasks are performed by supervisor agent, mission planner module, coordination and task refinement modules. The supervisor divides the goal into tasks; passes on the tasks to a mission planner, which is responsible for communicating with the coordinating agents. This model of hierarchy architecture simplifies interaction and allows easier implementation of coordination strategies.

4.3 *Simulated annealing (SA)*

The term simulated annealing (SA) derives from the analogous physical process, of heating up metal beyond its melting point and then slowly cooling the substance to obtain a strong crystalline structure. In simulation, a minima of the cost function corresponds to the ground state of the substance. The simulated annealing process lowers the temperature by slow stages until the system freezes and no further changes occur. At each temperature, the simulation must proceed long enough for the system to reach a steady state or equilibrium. It is a technique to find a good solution to an optimisation problem by trying random variations of the current solution. A worse variation is accepted as the new solution with a probability that decreases as the computation proceeds. The slower the cooling schedule, or rate of decrease, the more likely the algorithm is to find an optimal or near optimal solution.

There are different types of generalized optimization schemes SA is considered a meta-heuristic. Its generality and applicability stems from its foundation in thermodynamics and statistical mechanics. The SA algorithm searches the set of solutions, referred to as a search space; similar to thermodynamic system changes from one energy state to another. The development of SA therefore established a mathematical analogy between optimization problems and thermodynamic systems thus creating a new foundation from which to analyze and solve such problems. Simulated annealing is a generalization of a Monte Carlo method for

examining the equations of state and frozen states of n-body systems [43]. A simple Monte Carlo simulation randomly selects a point somewhere is the search space and all points are used to find out information about the search space, for minimizing multivariate functions.

The cooling schedule of simulated annealing algorithm depends on the starting temperature, the final temperature, decrement in temperature in each step and the number of iteration at each temperature. There are different cooling schedules being applied for various applications [31] suggested that there can be many cooling schedules used with the simulated annealing optimisation problem

$$S_{cf}(t) = \alpha \exp(-\lambda t) \qquad [4.1]$$

The cooling schedule $S_{cf}(t)$ has been chosen as the exponential plot and depends on the parameters λ which is a constant and usually is very small. In our model we have experimented with different values of λ and found that 0.001 is enhancing the performance of the system, α is the initial temperature.

4.3.1 New simulated annealing algorithm

The New Simulated Annealing Algorithm is a Simulated Annealing Algorithm with the improvement of the UAV position choice especially when it observes same number of targets at both current and offspring positions. This algorithm was developed and applies for the application of target monitoring using group or groups of UAVs. This algorithm is based on the basic simulated annealing algorithm, however the novelty of this algorithm is based on the probabilistic conditions.

The New Simulated Annealing Algorithm for Cooperative Target Observation is performed as follows.

1. Place the UAVs in random positions
2. Evaluate the number of targets detected at the current and offspring position of the UAVs
3. Accept the offspring position if it detects more targets

4. If offspring position observes less targets accept the offspring position with a probability of acceptance P_{accept} as in equation

5. If the offspring and current positions observe the same number of targets then compare the quality of the current and offspring position by evaluating $Q_{watched}$ for current and offspring position

$$Qwatched = \begin{cases} \sum_{w=W} \sum_{\tau=T} dist(w,\tau) if \frac{s}{3} \leq dist(w,\tau) \leq s \\ 0 \qquad\qquad otherwise \end{cases} \quad [4.2]$$

where W and T are observers watching targets and watched targets respectively. where w represents watchers ,τ represents targets and s is the sensor range of the UAV and dist(w, τ) is the distance between the watchers and the targets.

6. If $Q_{watched}$ is less for an offspring position than the current position, accept offspring.

7. If $Q_{watched}$ for offspring position and current position are the same then evaluate $D_{unwatched}$

$$D_{unwatched} = \sum_{w \in W'} \sum_{\tau \in T'} \min dist(w,\tau) \quad [4.3]$$

where W' and T' are observers not watching any targets and unwatched targets respectively. The least distance between w and τ is min dist (w, τ). If $D_{unwatched}$ of offspring position, is less than $D_{unwatched}$ of current position then accept the offspring position otherwise reject it.

4.4 *Model description*

The model in Figure 4:6 consists of monitoring UAVs which are also called as watchers and targets. The watchers are represented by blue concentric circles. The inner circle represents the sensing range and outer circle represents the detecting range of the UAV. The sensing range is 's/3' and detecting range is 's' where s is the range from the centre of the circle to outer concentric circle. The red dots represent the moving targets. The lines from the centre of the circle, represents the direction of movement of the UAVs for the search of targets. The number of targets and the number of watchers can be varied. When the targets or any objects

come inside the detecting range the UAV, can detect the target and can move towards it when needed. The speed of both the watching UAVs and the targets can be varied. The sensing range of the UAVs also can be varied. By varying the group size of the UAVs, the system can be partially decentralized.

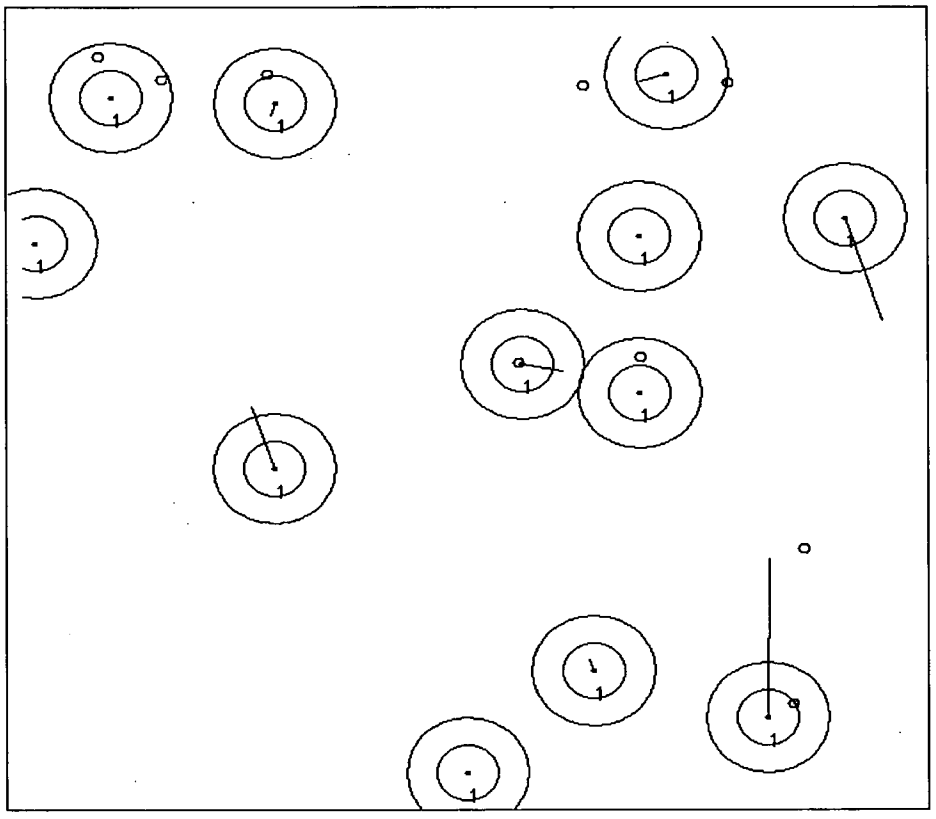


Figure 4:6 Simulation environment for co-operative UAVs

The parameters that can be varied for every simulation are:

1. Number of UAVs,
2. Number of targets,
3. Speed of UAVs,
4. Speed of Targets,
5. Number of groups of UAVs,
6. Sensor Range of UAVs,
7. Update Rate or the rate at which the position of the UAVs can be changed
8. Time Steps or the number of steps after the UAVs can change the position

We evaluate the destination points or the next move points of UAVs. The next move is dependent on whether the offspring position is accepted or not. This is the position where a UAV can view maximum number of targets at that time. The offspring position is chosen based on the algorithm mentioned in section 4.3.1. The next move of the UAV depends on whether the offspring position is accepted. The offspring position choice depends on the algorithm explained in pattern 4.2. This is done every 20-update rate. The lines in the UAVs denote that the UAVs are moving towards the destination positions. This model is developed in the test bed CTO using multi agent simulation toolkit (MASON) [30]. This model is very similar to the CMMOT model. The targets move randomly in the search space, by varying its speed. UAVs compute the destination positions or the offspring positions every 20-update rate. The update rate can be varied if the UAVs have to change the position more often due the increased target speed or the large number of targets.

4.4.1 Principle of operation and block diagram

The Figure 4:7 describes the operation of New Simulated Annealing algorithm for monitoring maximum number of targets as possible. The position of the UAV at any instant is called as the current position p_i (d), this is initially is set as random position inside the search space, and the position of the offspring is the next destination position of the UAV p_j (d) which is set initially at random position.

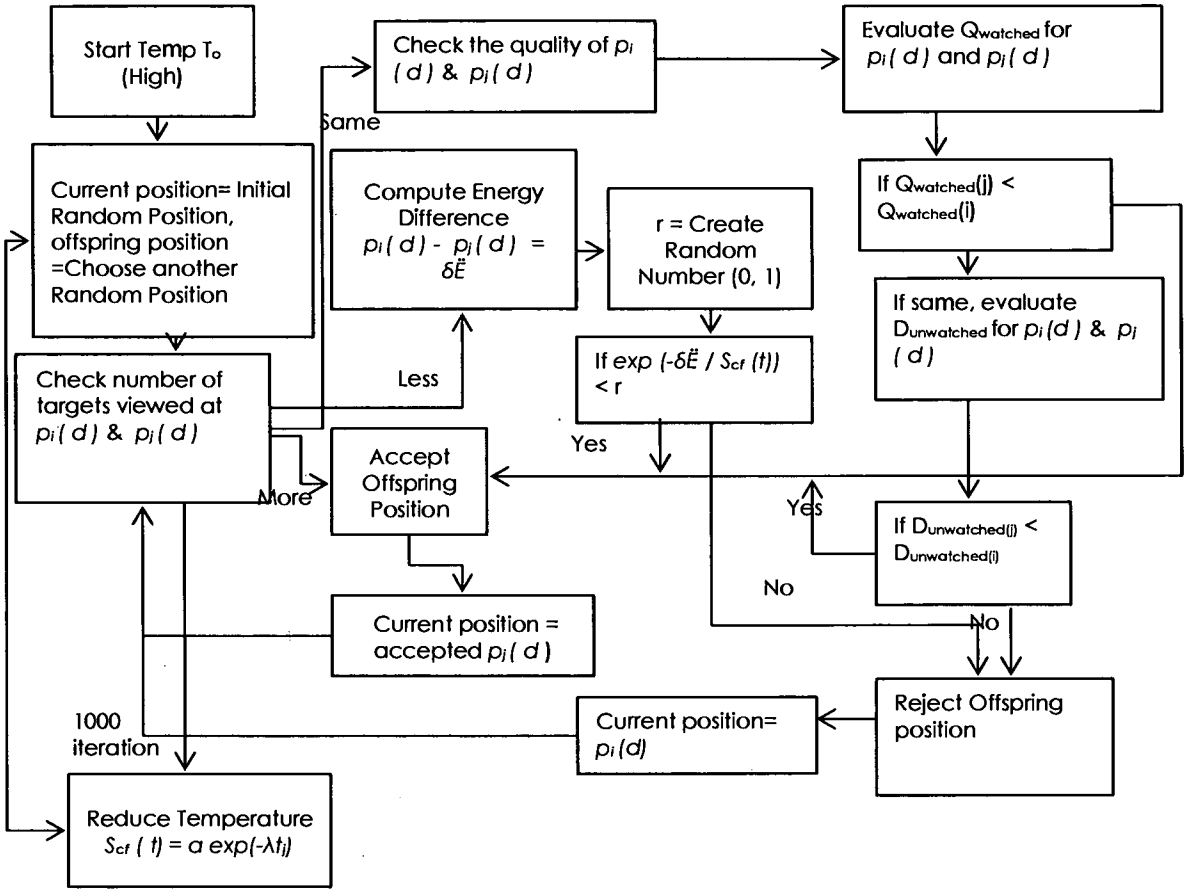


Figure 4:7 Basic building block for new simulated annealing in target observation

$$Paccept = \begin{cases} 1 & pj\,(d) < pi(d) \\ 1 & pj\,(d) = pi(d), Q\ watched\ (j) < Q\ watched(i) \\ 1 & pj\,(d) = pi(d), Q\ watched\ (j) = Q\ watched(i), \\ & \quad D\ unwatched\ (j) < D\ unwatched(i) \\ 1 & pj\,(d) > pi(d), \quad exp\left(\dfrac{pi\,(d) - pj(d)}{Scf(t)}\right) < r \\ 0 & otherwise \end{cases} \quad [4.4]$$

$$Paccept = \begin{cases} 1 & pj\,(d) < pi(d) \\ 1 & pj\,(d) > pi(d), \quad exp\left(\dfrac{pi\,(d) - pj(d)}{Scf(t)}\right) < r \\ 0 & otherwise \end{cases} \quad [4.5]$$

To optimize the position of the UAVs to achieve our goal we make use of the equation 4.4. The number of targets viewed by all the UAVs at least once is maximized. To achieve our goal, we move each UAV to a position so that its sum of the Euclidean distance to maximum number of targets is minimized. The novelty of the new simulated annealing algorithm is clear from the equations 4.4 and 4.5. The probability of acceptance has got two more inclusions when the number of targets monitored is the same in both the current and the offspring positions. The simulated annealing algorithm uses the probability of acceptance when UAV monitors more number of targets and lesser number of targets. Therefore when it comes to a situation when the same number of targets is monitored, from current and offspring positions, either the current or the next position is chosen randomly for the next move. However, this does not lead improvement of to the system performance which is denoted in graphs in chapter 6. Including two other conditions makes the system to perform much better in terms of computation time, number of targets monitored for various sensor ranges, group sizes and target speeds.

The probability of accepting the new destination or the offspring position p_j (d) depends on the number of targets it views at p_j (d). If the UAV at this position can view more targets, then the new destination p_j (d) is accepted and the UAV is moved to this new destination. If the UAV views lesser number of targets at P_j(d) than at p_i (d) even then, p_j (d) is accepted with a probability of exp(-δE/ S_{cf}(t)). However, if UAV at p_j (d) views equal number of targets as p_i (d) then $Q_{watched}$ is evaluated for p_j (d) and p_i (d). $Q_{watched}$ relates to the observing UAVs and watched targets.

If $Q_{watched}(j) < Q_{watched}(i)$, then again the new destination p_j (d) is accepted. However, when $Q_{watched}(j) = Q_{watched}(i)$, we evaluate $D_{unwatched}$ for p_j (d) and p_i (d). $D_{unwatched}$ relates to the unobserving UAVs and unwatched targets. If $D_{unwatched}(j) < D_{unwatched}(i)$, then we accept the new destination otherwise, reject it and stay in the current position to evaluate another offspring position.

The sequence of events taking place in the new simulated annealing in a tunable decentralized environment is as given in the figure below.

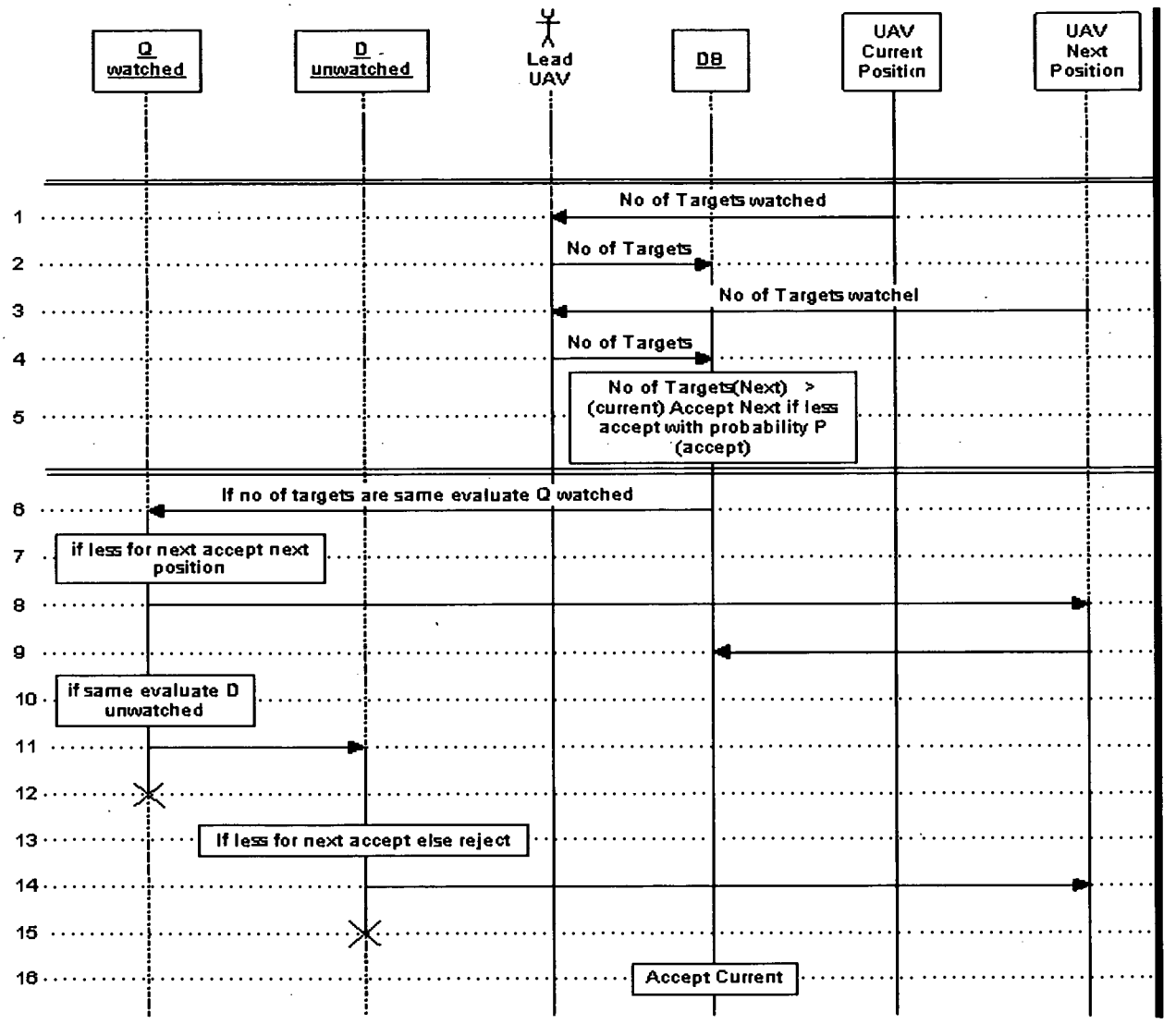


Figure 4:8 Sequence diagram for the sequence of events in co-operative target observation

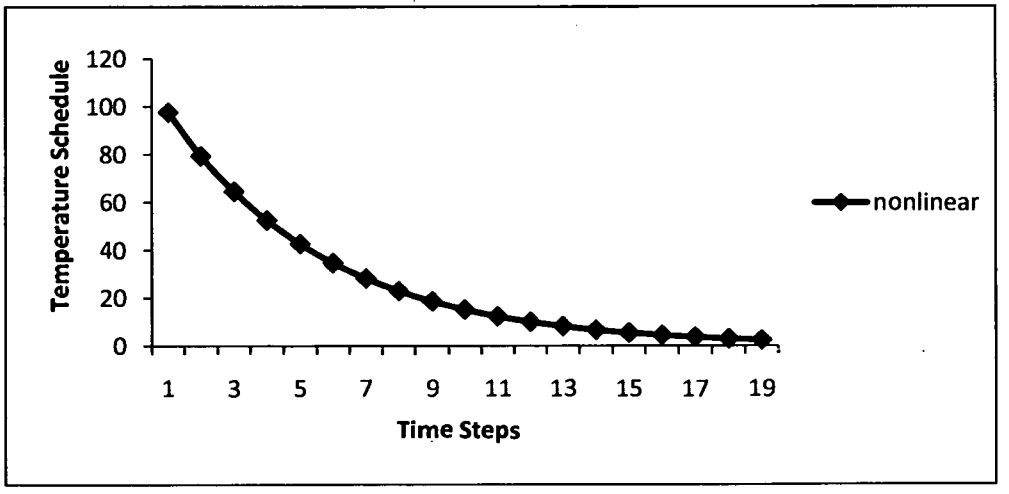


Figure 4:9 Exponential cooling schedule used in simulation

The initial temperature usually is taken to be high so that there is an option for searching the entire search space at an initial temperature and that the system is not stuck up in the local minima rather searches for the global minima and is able to move to all the states. The choice of high initial temperature allows the worst solutions to be accepted; however, when the temperature decreases as per the cooling schedule then the probability of acceptance P (accept) of worse solution decreases.

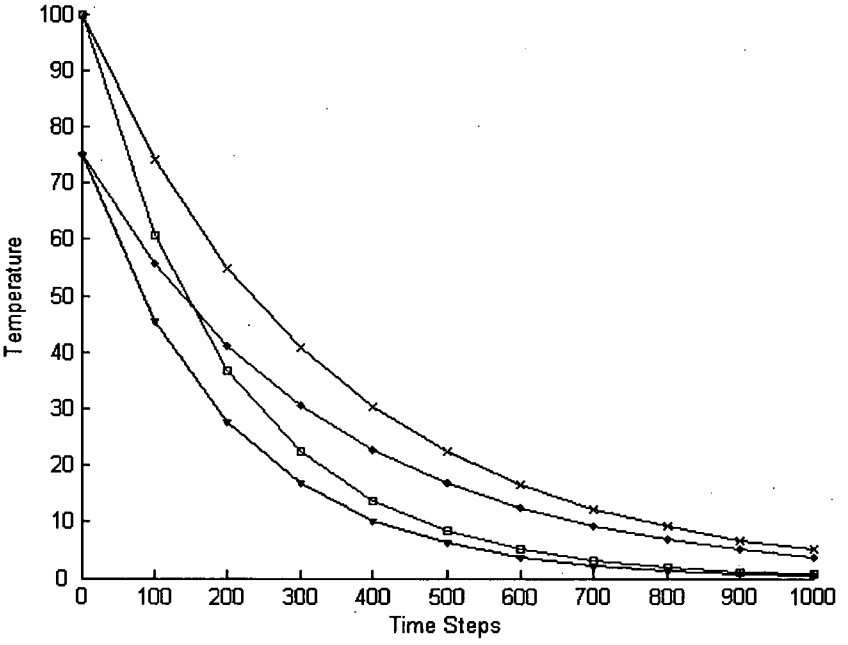


Figure 4:10 Cooling schedule with various values of α

Figure 4:10 , depicts the cooling schedule in equation (1) for α value of 20 and 15, the λ value of 0.001 and 0.01, and t from 0 to 1000. We find the S_{cf} (max) to S_{cf} (min) varies exponentially. The probability of acceptance at higher temperatures accepts worse solutions or the positions which observes lower number of targets, whereas that is not the case at lower temperatures. Simulated Annealing lets the simulation explore the entire search space at higher temperatures and obtain the best possible solution at lower temperatures.

$$P_{accept} = \exp\left(\frac{-(p_i(d) - p_j(d))}{S_{cf}(t)}\right) \qquad [4.5]$$

where $p_i(d)$ - $p_j(d)$ = $\delta\ddot{E}$ which is the difference in cost or the Euclidean distance, $p_i(d)$ is the Euclidean distance of the observing UAV to the targets in its current position and $p_j(d)$ the Euclidean distance of the observing UAV at the offspring position, and $S_q(t)$ is the cooling schedule.

5 Temporal Difference Learning and Simulated Annealing for Optimal Control

The trade-off between exploration and exploitation has an important impact on the performance of temporal difference learning. There are several action selection strategies; however, it is unclear which strategy is better. The impact of action selection strategies may depend on the application domains and human factors. This chapter addresses a modified Sarsa (λ) control algorithm by sampling actions in conjunction with the new simulation annealing technique. A game of soccer is utilized as the simulation environment, which has a large, dynamic and continuous state space. The empirical results demonstrate that the quality of convergence has been significantly improved by using the simulated annealing approach.

An intelligent agent is expected to be capable of adapting and learning in an uncertain environment. The ability of learning can be built up by interacting with the environment in which it is situated. Reinforcement learning is a proper approach for an agent to learn from experience. Reinforcement learning is the learning of a mapping from situations to actions so as to maximize a scalar reward or reinforcement signal [58], which mainly includes Dynamic Programming (DP), Monte Carlo Method, and Temporal Difference Learning technique (TD).

DP [53, 57] has been well investigated under the formalism of Markov Decision Processes (MDPs). TD [67] is a form of asynchronous DP, and the value function is estimated by sample episodes.

TD method addresses the problem of approximating the optimal action strategy through and while interacting with the environment. Trial–and–error, delayed rewards, and trade-off between exploration and exploitation are three important features in TD algorithm. As a kind of unsupervised learning, TD has to balance exploration and exploitation during the learning period. To effectively build up

the reward structure, the strategy dealing with the trade-off between exploration and exploitation has to be considered. Some strategies have been used for balancing exploration and exploitation, e.g., greedy strategies, randomized strategies, annealing-like strategies. It is unclear which policy is better and the performance may depend on the task and on human factors. Only few careful comparative studies are available [68].

The simulated annealing (SA) [64] is a technique for solving combinatorial optimization problems. SA starts with a high temperature and gradually decreases the temperature over time. As a consequence, the agent starts with the high possibility of exploration and turns towards exploitation by incorporating a probability function in accepting or rejecting new solutions. SA approach does not require large computer memory, which can speed up the computation. In this chapter, the simulated annealing technique is utilized for sampling possibility distribution over actions at a given state, so as to improve the quality of performance.

Algorithm Sarsa (λ) is proposed by combining on-policy learning algorithm with SA method. A comparative study is conducted in a real-time, stochastic, and dynamic testbed called SoccerBots [50]. The simulation results are compared by using ϵ greedy policy and ϵ greedy policy with simulated annealing, in conjunction with a linear approximation function called Tile Coding [51]. The experimental results demonstrate that the quality of the performance can be enhanced by using the simulated annealing technique.

5.1 *Temporal Difference Learning*

Temporal difference learning (TD) is a prediction method which is used to solve Reinforcement learning. It makes use of Monte Carlo method and dynamic programming method [67, 68] to solve the Reinforcement learning problems. TD is a form of model-free approach and is a combination of DP and Monte Carlo Method The idea of TD is to learn how to take action through trial-and error interactions in a dynamic environment. The value function is accumulated by an incremental learning model and infinite horizon discount model. The prediction is

adjusted to match the desired results or the more accurate results so this process is called bootstrapping. The incremental learning model accumulates the value function based on temporal difference errors, as given in Equation 5.1

$$V_{t+1} = V_t + \alpha[R_{t+1} - V_t] \qquad [5.1]$$

where the parameter a is learning rate, $0 \leq a \leq 1$, R_{t+1} is the accumulated reward.

Based on the infinite horizon discount model, the one-step update at a given time t and state s is shown in equation 5.2.

$$V_{t+1}(s) = V_t(s) + \alpha\,[\,r_{t+1}(s) + \gamma V_{t+1}(s') - V_t(s)] \qquad [5.2]$$

where the parameter γ is the discount rate, $0 \leq \gamma \leq 1$. The rewards in the future are geometrically discounted by the parameter γ, $rt+1$ is the immediate reward. The use of eligibility traces is another mechanism to improve the convergence of value function [60,67]. The aim of eligibility traces is to assign credit or blame to the eligible states or actions. The ways for evaluating the traces include accumulating traces and replacing traces.

The accumulating traces are given by:

$$et(s, a) = \gamma\lambda et{-}1(s, a), \text{ if } s = st$$
$$\gamma\lambda et{-}1(s, a) + 1, \text{ if } s = st \qquad [5.3]$$

whereas replacing traces use $et(s, a) = 1$ for the second update [68].

The trade-off between exploration and exploitation is a problem the learning agent has to deal with. On one hand, the agent must explore the environment explicitly, in case to fall into local optimum. On the other hand, the agent need to exploit the existing knowledge to make the value function converge quickly. However, how can the agent know that the sequence of actions that has been found is the best? It may be helpful to integrate some parametric optimization techniques with reinforcement learning.

Normally, there are some action selection strategies [68]: e.g., ε greedy policy is to select a random action with probability 1-ε *and* the optimal action with probability 1. The drawback is that unlucky sampling may cause the rewards obtained from optimal action to fall into a local optimum. On the other hand, randomized strategy is to select the action according to probability p, which is sampled by Boltzmann distribution. Yet, another strategy is the annealing-like approach, which is used in this research.

5.2 *Simulated Annealing*

SA [10] is based on the theory of Markov processes and motivated by the physical process of annealing [64] in solids. The solids are heated to a temperature above the melting point and then allowed to cool. When the solid is cooled to a lower temperature, the solid reaches the stable crystalline structure. If the cooling is done slowly then the crystalline structure is stable. However, if the cooling is done fast then, the solid obtained has crystalline imperfection. Therefore, for the system to reach equilibrium, the cooling has to be done slowly. Slower does not mean it can be done forever to obtain stable state. Simulated Annealing is analogous to this process.

The SA algorithm searches the search space similar to the thermodynamic change of energy-to-energy state. First random position in the search space is chosen to be the initial state or current state and another random position is chosen as the next state. The value functions at both the current state and the next state are evaluated. These value functions are compared. If the next state yields a better solution, then that is chosen as the best solution. Even otherwise, the next state is chosen with a probability. This probability of acceptance of worse solution is P(accept) which is expressed as given in Equation 5. 4.

$$P\ (\text{accept}) = \exp\ (-c/T) < r \qquad [5.4]$$

Where $c = \Delta E$ is the change in the evaluation function, T is the current temperature and r is a random number between 0 and 1. It is not required to stick on to a particular solution, which could lead to locking up in local optimization

rather than global optimization. The problem of getting locked to local optimum is overcome in SA by letting the possibility of the choice of accepting the worse solution with a certain probability of P(accept).

The cooling schedule or temperature schedule T of SA is how the temperature is decremented and is the mapping of the temperature to time. It depends on starting temperature, final temperature, decrement in temperature (linear or nonlinear) and number of iterations at each temperature. There are various types of cooling schedules available that are used in practice. SA Algorithm [65] has been used to solve many combinatorial optimization problems and some continuous variable problems. The choice of the temperature schedule or the cooling schedule can be either linear or non-linear.

The algorithm is

Current node = MAKE-NODE (INITIAL-RANDOM
STATE [Problem, Schedule [t=cooling schedule]]).
For t = 1 to ∞ do:
If T = 0 then return Current;
Next node = a randomly selected successor of Current;
ΔE = VALUE [Next] – VALUE [Current]
If ΔE ≥ 0 then Current = Next;
Else Current = Next only with probability exp(ΔE/T)<r (random number);

Figure 5:1 Basic simulation annealing algorithm

5.3 *Details of Simulation Environment and Algorithms*

TeamBots is a Java-based collection of application programs and Java packages for multiagent mobile robotics research [50]. Each soccer team can have no more than 5 players. Two teams are built and the individual and team strategies for each team are defined.

The scenario is defined to learn a ball interception skill for the soccer player in SoccerBots. For the ball interception, the ball is kicked at a certain angle and speed. The player is away from the ball at a certain distance and angle to ball, in order to intercept the ball with the highest speed and the fewest steps. The intercepting ball problem is that a soccer agent is trained to find the optimal interception direction at each step toward the ball in order to catch an approaching ball in the shortest time.

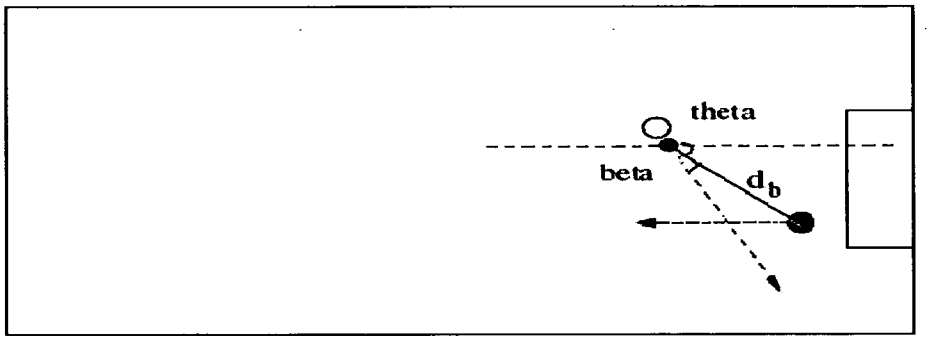


Figure 5:2 Ball Interception problem

In addition, the state space reduction technique is necessary for a large, stochastic, and dynamic environment. In this paper, a linear approximation function known as tile coding [51] is utilised to avoid state space from growing exponentially [61].

5.4 *Combination algorithm*

The details of modified Sarsa (λ) with replacing traces are given in Fig 5. 3. In Fig 5:3, a is a learning rate, γ is a discount rate. The action selection policy is the exploration strategy, which can be ε greedy policy, i.e. the agent takes a random action with probability and takes the best action with probability (1 -ε). Q(s,a) is the state-action value, δ and θ are matrices to represent action, feature and related feature weights, and F is the feature to trace the eligible state process. The parameter T is a temperature, and ψ is the temperature-decay factor. Lower temperatures cause a greater difference in selection probability for actions. In the limit as $T = 0$, the action selection becomes the same as greedy action selection.

SA based exploration works similarly to neighborhood search based exploration by searching the set of all possible actions, but reducing the chance of getting stuck in a poor local optimum by allowing moves to inferior actions to be controlled by a randomized scheme [69]. The parameter T is initially high, allowing many inferior actions to be accepted, and is slowly reduced to a value where inferior actions are nearly always rejected. For the annealing-like approach, the temperature T is set to 120 and temperature-decay factor ψ is set to 0.99. The annealing process is illustrated in Fig. 5.4.

Initialise Q(s,a) arbitrarily and e(s, a) = 0, for all s, a.
Repeat (for each episode):
Initialise s, a;
Initialise T ← ψ T;
Repeat (for each step of episode):
Choose *ap* in s based on action selection policy (e.g., _-greedy);
Choose *ar* in s at random;
Generate an random possibility *Prandom*;
Calculate $Q(s, ar)$ and $Q(s, ap)$;
Calculate the possibility of accept *Paccept* using
$exp((Q(s,ar) - Q(s, ap))/T)$;
If $Q(s, ar) \geq Q(s, ap)$;
a ← *ar*;
Else If *Prandom* ≤ *Paccept*;
a ← *ar*;
Else
a ← *ap*;
Take action a, observe r, s';
Choose a' from s' using action selection policy derived from Q ;
δ ← r + γQ(s', a') - Q(s, a);
e(s, a) ← 1;
For all s, a:
Q(s, a) ← Q(s, a) + $a\delta$e(s, a);
e(s, a) ← $\gamma\lambda$e(s, a);
s ← s';
until s is terminal.

Figure 5:3 Sarsa (λ) with modified simulated annealing algorithm

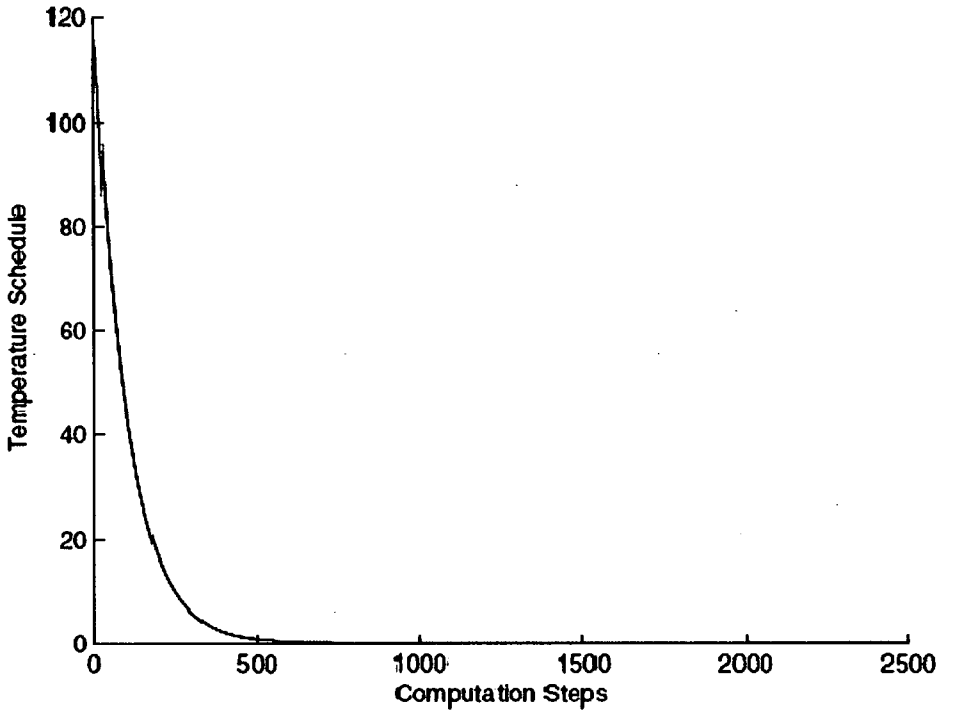


Figure 5:4 Cooling schedule used in this algorithm

The initial start temperature T, the temperature-decay factor ψ, and final temperature T' have to be scaled properly; otherwise, the poor results may occur. In [70], the choice process of their values includes to estimate the mean of the distribution of state values to define a maximum energy scale of the system, its standard deviation to define the maximum temperature scale, and the minimum change in energy to define the minimum-temperature scale.

To compare the realistic performance, the simulation is conducted with different set of parameter values with a number of experiments [62, 63]. By setting the value to 0.1, the set of optimal parameter values can be found: $a = 0.005$, $\gamma = 0.93$, and $\lambda = 0.9$. The performance may be heavily influenced by the action selection strategy. By running the episodes 2500 times, the average rewards can be obtained using algorithm of Fig5. 3 with ε greedy policy. The average rewards can be generated using algorithm in Fig. 3 with annealing-like policy. By comparing the average reward at different episodes, it is clear that the convergence with annealing-like policy is much quicker than that with _-greedy policy.

The curves in Fig.5. 5 illustrate that annealing-like policy can significantly improve the performance, both in speed of convergence and eventually convergence.

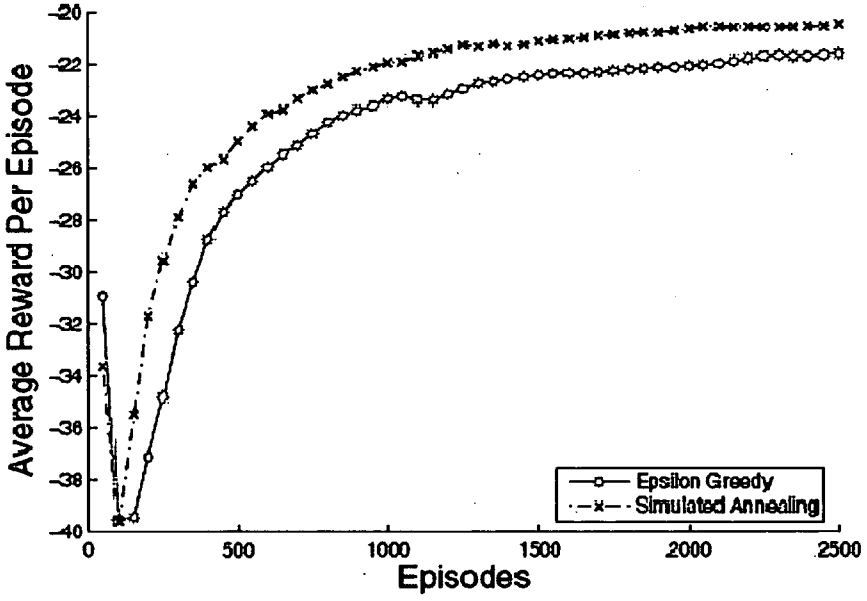


Figure 5:5 Comparison of combined algorithm with epsilon greedy algorithm

Salesman problem (TSP), capacited vehicle routing problems (CVRP), job scheduling problems, timetabling problems, and selection of communication protocols/standards and various other applications [54] makes use of simulated annealing algorithm.

A version of simulated annealing called Adaptive Simulated Annealing is employed with the reinforcement learning algorithm, which shows further improvements in algorithmic convergence properties [52]. A Q-learning algorithm with simulated annealing is introduced to balance exploration and exploitation [56]. In [69], three action selection methods, i.e., neighborhood search based exploration, simulated annealing based exploration, and tabu search based exploration, are evaluated and compared on a discrete reinforcement learning task (robot navigation).

5.5 *Softmax or Boltzmann Distribution Action Selection.*

A combined use of reinforcement learning and simulated annealing is proposed in [66]. A theoretically established approach tailored to reinforcement learning following Softmax action selection policy are discussed. It has been proven that Boltzmann's formula converges to uniform distribution as T goes to infinity and to the greedy distribution as T goes to 0. In addition, an application example of agent-based routing will also be illustrated. Another work [52] on adaptive simulated annealing (ASA) based reinforcement learning method is proposed. Here ASA [58] allows far-reaching access of the state space, and permits much faster annealing and hence faster convergence. The action can be generated according to a Boltzmann probability, which provides some kind of "annealing" that is the spirit of the other annealing performed in value function maximisation.

5.5.1 The ε greedy Action Selection with SA.

SA based exploration is discussed in [69]. The ε greedy algorithm is a method using near-greedy action selection rule. It behaves greedily (exploitation) most of the time, but every once in a while, say with small probability ε (exploration), instead select an action at random. This part of the research evaluates the role of heuristic based exploration in reinforcement learning. Three methods are compared: neighborhood search based exploration, simulated annealing based exploration, and tabu search based exploration. SA based exploration works by searching the set of all possible actions, but reducing the chance of getting stuck in a poor local optimum by allowing moves to inferior actions. When a non-greedy action is selected, this action is evaluated by SA based exploration approach.

A similar work done in [56] is to explore the possibility of improving the simple ε greedy approach by appropriately reducing ε during the learning process. The SA

approach is combined with Q (λ). The task of finding the optimal policy in Q-learning is transformed into search for an optimal solution in a combinatorial optimization problem. Then the Metropolis criterion from SA algorithm is applied to the search procedure in order to control the balance between exploration and exploitation. The improved algorithm is tested in the puzzle simulation domain.

5.6 *Improvements*

The parametric optimization techniques can be combined with TD algorithms to improve the overall performance. This paper provides a comparative study by using the simulated annealing technique to balance between exploration and exploitation. The experimental results demonstrate that the algorithm in Fig.5. 3 with the annealing-like approach converge much quicker. For a large, stochastic, and dynamic system, utilizing the annealing-like technique can reduce computational cost and learn quickly.

6 Findings, conclusion and future work

6.1 *Findings and conclusion*

Simulation of Co-operative target observation based on the algorithm developed as given in chapter 4. The simulation was performed in MASON with three algorithms, new Simulated Annealing Algorithm, Simulated Annealing Algorithm and Hill Climbing Algorithm. The results were compared for various data sets.

- Number of simulations run for each data 100
- Number of UAVs watching 144
- Number of targets 225
- Number of Time Steps 2000
- Update Rate 20
- Speed of target 0.1 to 0.9 in steps of 0.1
- Speed of Watchers 0.1 to 1.0 in steps of 0.1

The starting temperature is taken from 100 units and then reduced exponentially with the λ value as 0.001 and 't' the time steps ranging from 1 to 1000. However, the results showed that the temperature need not be reduced to a very low value where t = 1000 but the optimised solution is obtained when the value of t is around 100. The UAVs are positioned initially at some random position next, the offspring position is chosen at some random position close to the current position, and the qualities of both the positions are compared. The quality of a UAV performing its goal is at which position it can view maximum number of targets. The model description was explained in detail in section 4.4. The various parameters and data that is involved in the simulation are explained in detail.

The parameters that can be varied for every simulation are:

1. Number of UAVs,
2. Number of targets,

3. Speed of UAVs,
4. Speed of Targets,
5. Number of groups of UAVs,
6. Sensor Range of UAVs,
7. Update Rate or the rate at which the position of the UAVs can be changed
8. Time Steps or the number of steps after the UAVs can change the position

The analysis of the modified simulated annealing algorithm was experimented for various target speeds, UAV speeds, sensor ranges, number of UAVs, number of targets, number of subgroups and update rate. We have also compared the Hill Climbing algorithm with the modified simulated annealing algorithm, and found various interesting results.

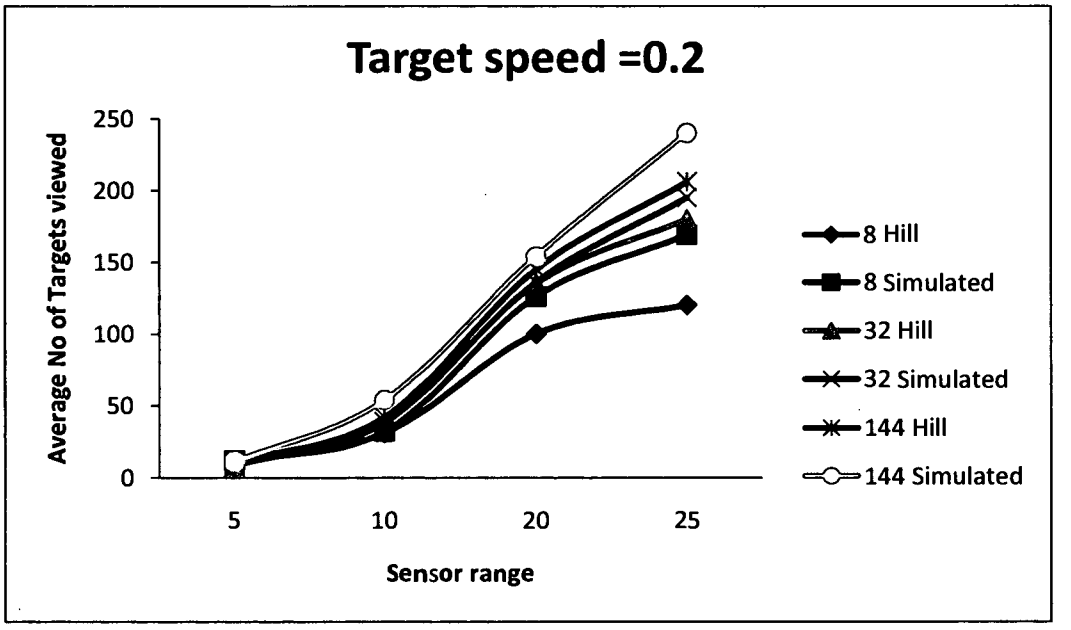


Figure 6:1 Low target speed and varying group size

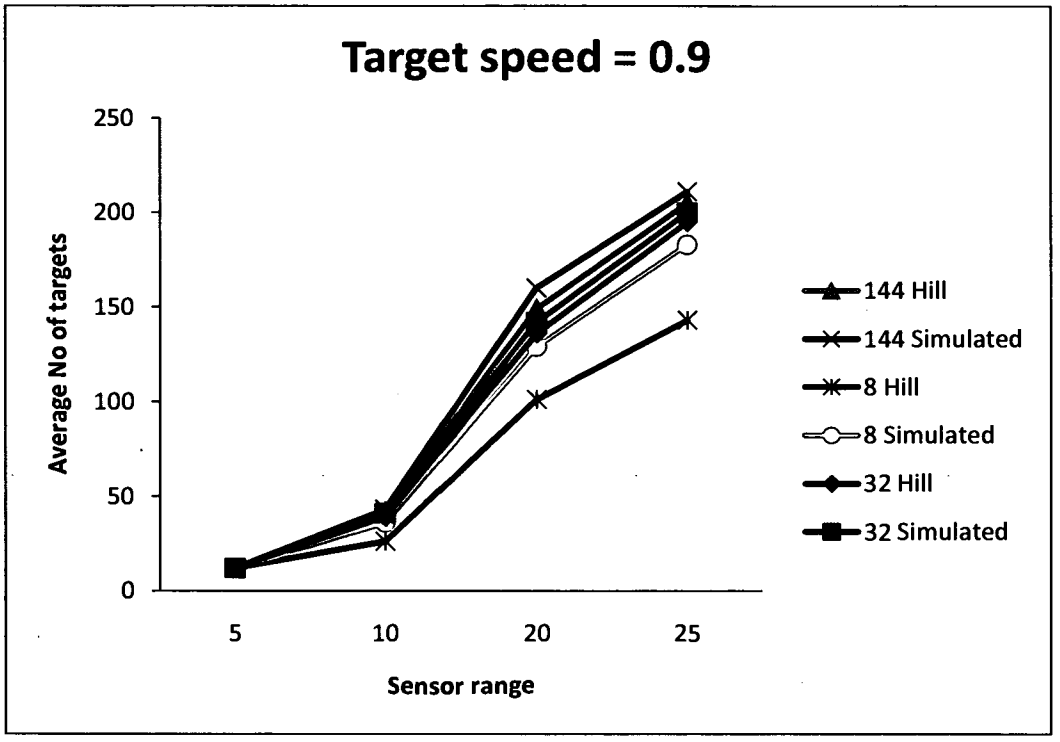


Figure 6:2 High target speed and varying group sizes

Figure 6:1 and Figure 6:2 show the comparison of Modified simulated Annealing Algorithm performance for cooperative target observation at different target speed. We find that at lower target speed performance of Simulated Annealing algorithm is enhanced as the number of targets viewed is more compared to higher target speed.

As the speed of the target increases, the performance of the algorithm reduces. We find that this algorithm is sensitive to target speeds and UAV speeds. Figure 6:3 and 6:4 shows the % of targets viewed for different group sizes and sensor ranges. Here, we again compare the two algorithms and find that the new simulated annealing algorithm performs much better than the Hill Climbing Algorithm. At higher sensor ranges, we find the difference in the percentage of the targets viewed is much more efficient for our algorithm when compared with the Hill Climbing algorithm. Figure 6:5 and Figure 6:6 show that the lower the speed of the target the more number of targets viewed.

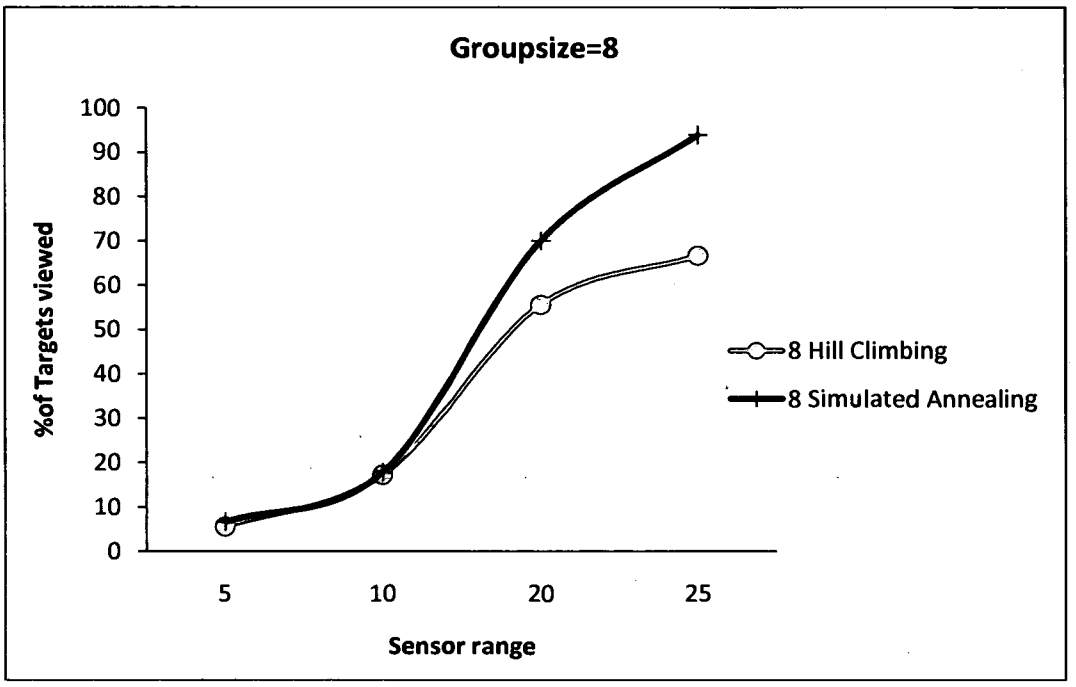


Figure 6:3 Small group sizes

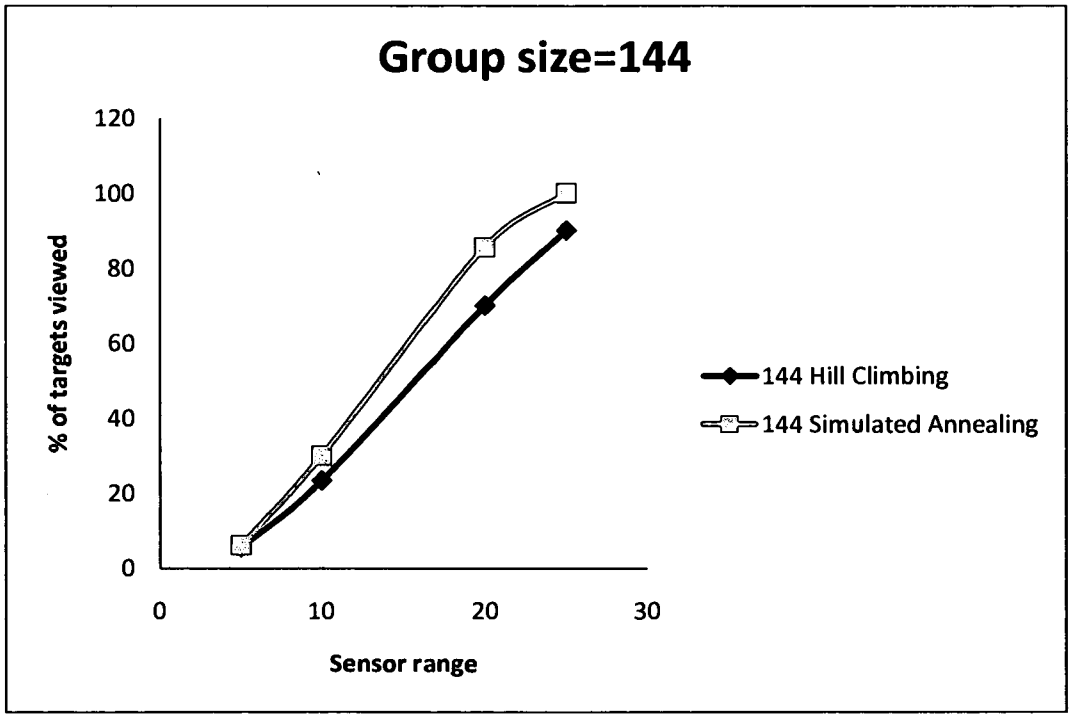


Figure 6:4 Large group sizes

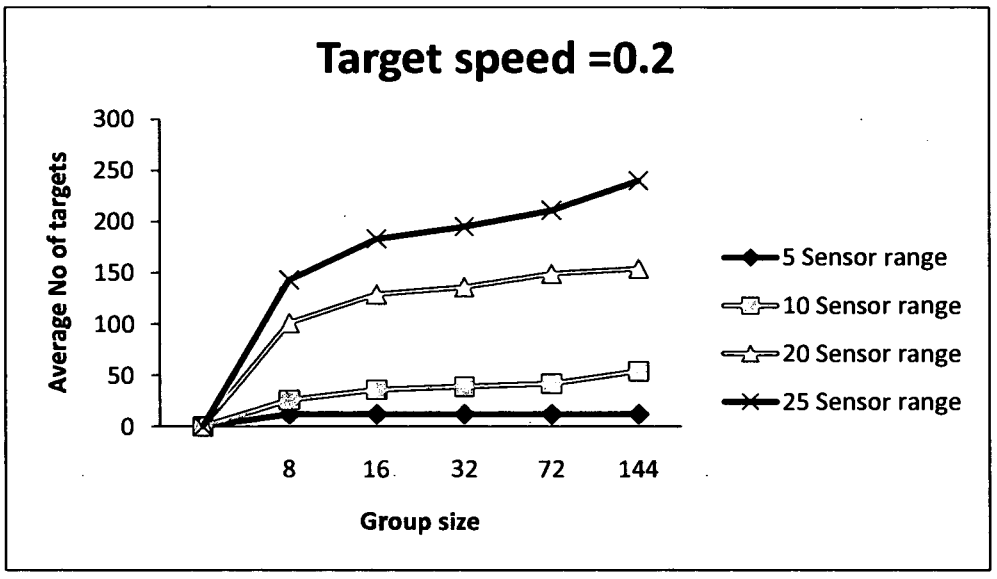


Figure 6:5 Low target speed and varying sensor range

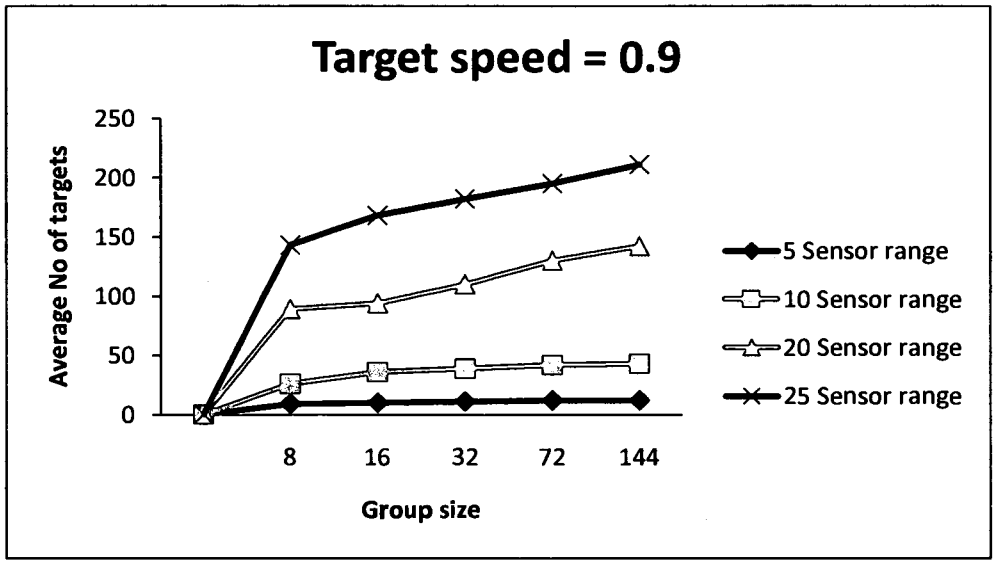


Figure 6:6 High target speed and varying sensor range

The target speed of 0.9 and 0.2 are taken. We find that the number of targets viewed is more for UAVs group with higher sensor range. We find that the performance of the algorithm is better when it is centralized or the group size is 144.

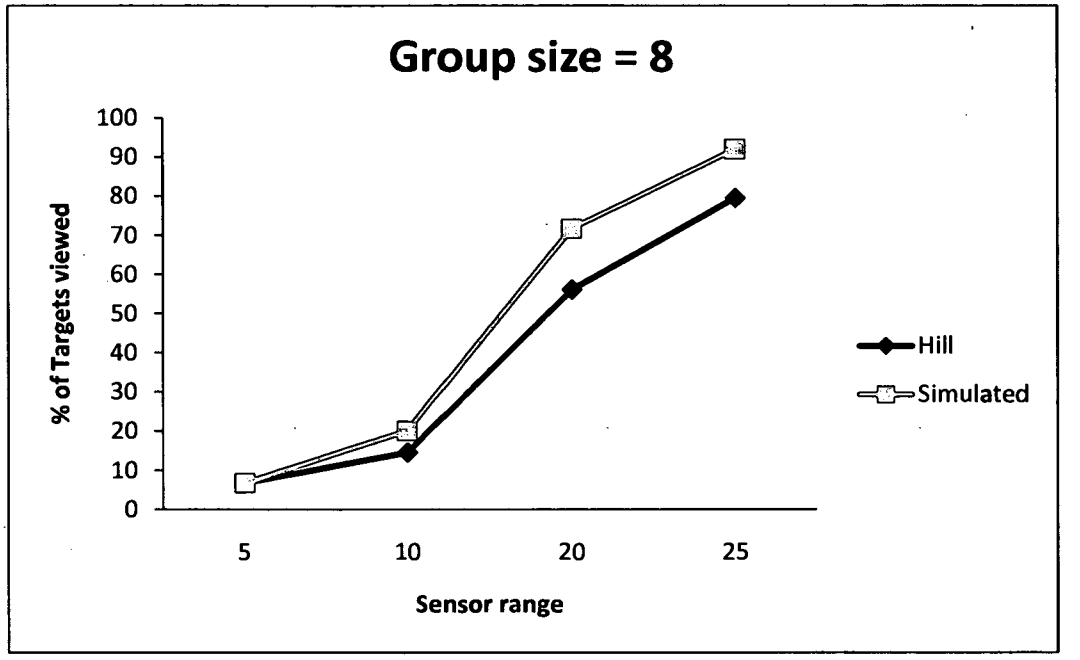


Figure 6:7 Small group size and varying sensor range

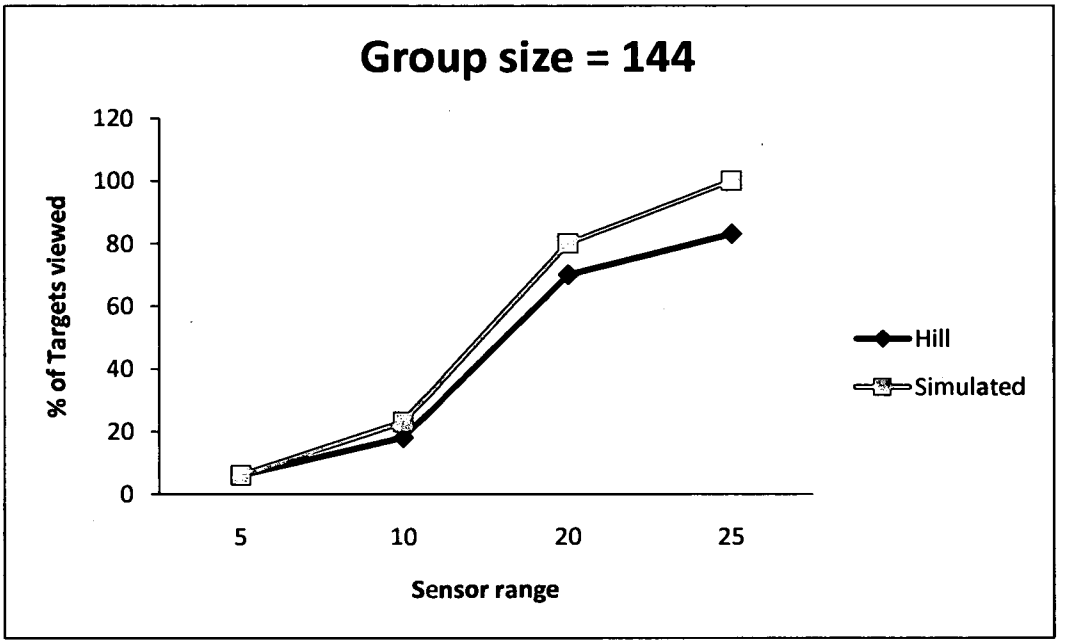


Figure 6:8 Centralised group with varying sensor range

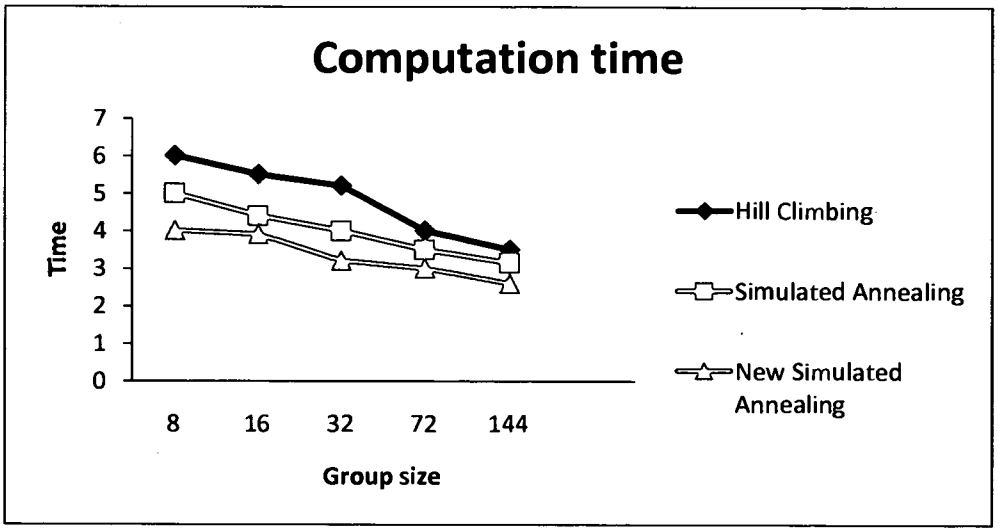


Figure 6:9 Computation complexity

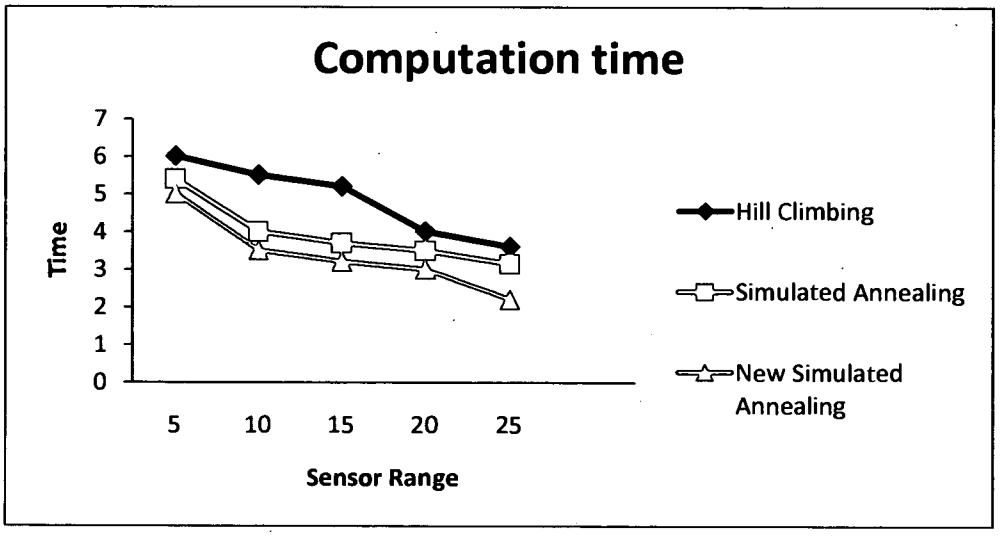


Figure 6:10 Computation complexity

6.2 *Statistical methods of the algorithms*

6.2.1 Mathematical representation

The hill climbing algorithm, simulated annealing algorithm and new simulated annealing algorithm can be expressed using statistical equations as equation 6.1, 6.2 and 6.3

$$Paccept = \begin{cases} 1 & pj\,(d) < pi(d) \\ 0 & otherwise \end{cases} \quad [6.1]$$

$$Paccept = \begin{cases} 1 & pj\,(d) < pi(d) \\ 1 & pj\,(d) > pi(d), \quad exp\left(\dfrac{pi\,(d) - pj(d)}{Scf(t)}\right) < r \\ 0 & otherwise \end{cases} \quad [6.2]$$

$$Paccept = \begin{cases} 1 & pj\,(d) < pi(d) \\ 1 & pj\,(d) = pi(d), Q\ watched\ (j) < Q\ watched(i) \\ 1 & pj\,(d) = pi(d), Q\ watched\ (j) = Q\ watched(i), \\ & D\ unwatched\ (j) < D\ unwatched(i) \\ 1 & pj\,(d) > pi(d), \quad exp\left(\dfrac{pi\,(d) - pj(d)}{Scf(t)}\right) < r \\ 0 & otherwise \end{cases} \quad [6.3]$$

Where p_i (d) is the distance of UAVs current position to the target, p_j (d) is the next position distance of UAV to target. Q watched (i) and Q watched (j) are watched targets from current and next positions respectively. D unwatched (i) and D unwatched (j) are the unwatched targets from current and next positions.Scf(t) is the cooling schedule.

6.2.2 Statistical comparison

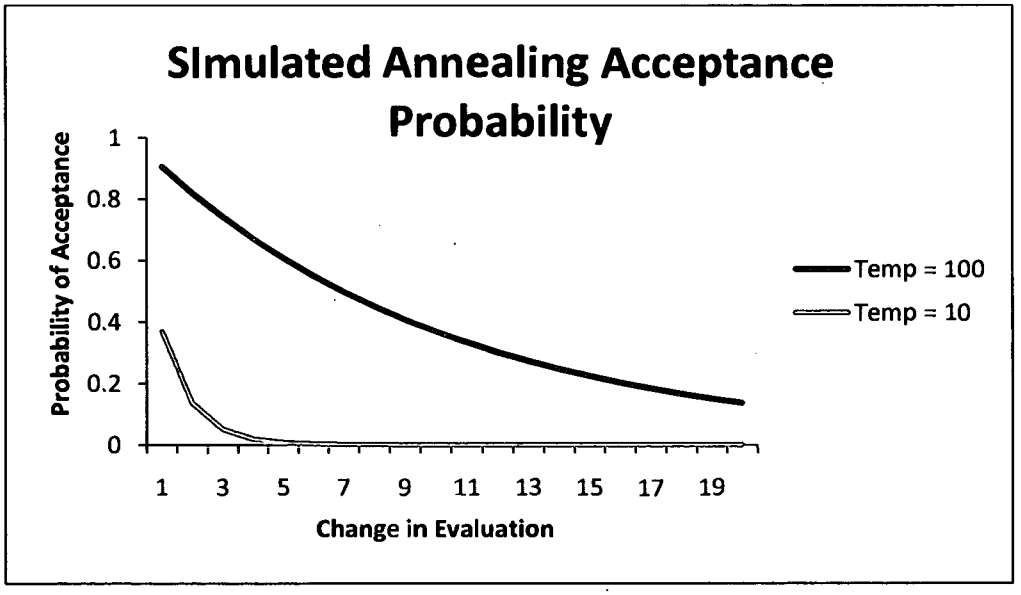


Figure 6:11 Probability acceptance for varying values of Temp= t

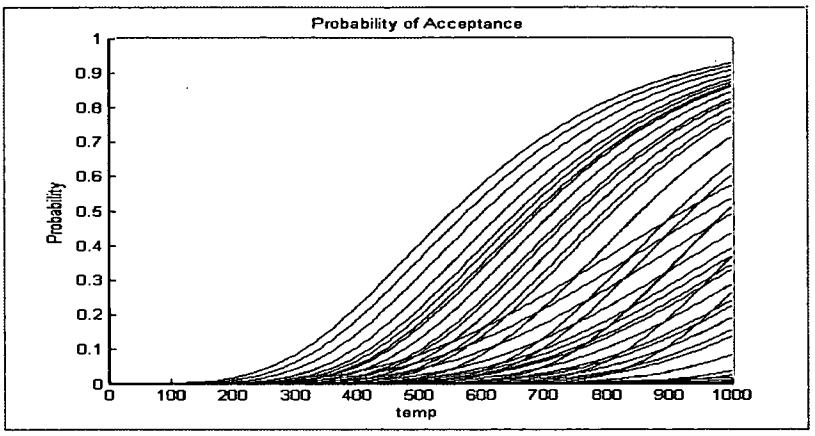


Figure 6:12 Probability of acceptance for New simulated annealing for varying values of t. Graph was generated using MATLAB

Figure 6:11 represent the probability of acceptance for varying evaluation function which is the cooling schedule of the algorithm. Figure 6:12 represent the probability of acceptance of new simulated annealing algorithm. The initial value of t was varied, and for the fall in the value of t the probability of acceptance was evaluated and the graph was drawn.

6.3 *Conclusion*

This research was mainly based on the new simulated annealing algorithm that was developed for applications in target monitoring, avoiding bushfire disaster, security and surveillance, search and rescue operations. The algorithm was simulated and its results were compared with some of the existing algorithms. The path finding algorithms and the search algorithms were studied and compared. This was done to identify the best path finding method that can be used for intelligent co-operation of UAVs to complete their mission. The comparison is conducted with respect to the number of searches required and the time taken for computation.

6.4 *Future work*

In future there is a great potential for the researchers to combine this algorithm with the other major algorithms such as genetic algorithms. We have combined this algorithm with the temporal difference learning algorithm and achieved good

results. The degree of co-operation can be increased with the combined algorithms for monitoring as many targets as possible.

The future work could also include obstacles and to avoid those obstacles we intend to incorporate an obstacle avoidance algorithm. This model also has worked with the assumption of limitation of communication. However, our next step would be to include communication between the UAVs, which will enable us to achieve a more realistic model to achieve our goal. Thus by including communication with co-operation the overall system performance for target monitoring can be enhanced.

References

[1] Hart EP, Nilsson NJ, Raphael B. A formal basis for the heuristic determination of minimum cost paths. IEEE Transaction, System Science and Cybernetics 1968;SSC-4(2):100–7.

[2] Chieh-Chih, W., et al., Simultaneous Localization, Mapping and Moving Object Tracking. Int. J. Rob. Res., 2007. 26(9): p. 889-916.

[3] Nilsson JN. Problem-solving methods in artificial intelligence. NewYork: McGraw-Hill; 1971.

[4] Newell A, Simon HA. Human problem solving. Englewood Cliffs, NJ: Prentice-Hall; 1972.

[5] Pearl J. Heuristics: intelligent search strategies for computer problem solving.Addison-Wesley Publishing Company; 1984.

[6] Gallo G, Pallottino S. Shortest path methods. In: Florian M, editor. Transportation planning models. Amsterdam: Elsevier Science Publishers; 1984. p. 227–56.

[7] Hung SM, Divoky JJ. A computational study of efficient shortest path algorithms. Computers and Operations Research 1988;15(6):567–76.

[8] VurenVT, Jansen GRM. Recent developments in path finding algorithms: a review. Transportation Planning andTechnology 1988;12:57–71.

[9] Cherkassky BV, Goldberg AV, Radzik T. Shortest paths algorithms: theory and experimental evaluation. Mathematical Programming 1996;73(2):129–74.

[10] Zhan FB, Noon CE. Shortest path algorithms: an evaluation using real road networks. Transportation Science 1998;32(1): 65–73.

[11] Jacob R, Marathe M, Nagel K. A computational study of routing algorithms for realistic transportation networks. Journal of Experimental Algorithmics 1999;4:1–19.

[12] Fu L. Real-time vehicle routing and scheduling in dynamic and stochastic traffic networks. Unpublished Ph.D. Dissertation, University of Alberta, Edmonton, Alberta, 1996.

[13] Zhan FB, Noon CE. Shortest path algorithms: an evaluation using real road networks. Transportation Science 1998;32(1): 65–73.

[14] Jacob R, Marathe M, Nagel K. A computational study of routing algorithms for realistic transportation networks. Journal of Experimental Algorithmics 1999;4:1–19.

[15] Fu L. Real-time vehicle routing and scheduling in dynamic and stochastic traffic networks. Unpublished Ph.D. Dissertation, University of Alberta, Edmonton, Alberta, 1996.

[16] Anthony Finn, Kuba Kabacinski, & Samuel Picton Drake Design Challenges for an Autonomous Cooperative of UAV's Defence Science & Technology Organisation Department of Defence, Australia. IDC 2007

[17] John J. Murray, Chadwick J. Cox, George G. Lendaris, Richard Saeks. Adaptive Dynamic Programming. IEEE Transactions on Systems, MAN. and CYBERNETICS 2002.

[18] B. Sathyaraj, R. Chellappa Doss. Route Maintenance using Mobility Prediction for Mobile Ad hoc Networks. IEEE International Conference Mobile Adhoc and Sensor Systems Conference, MASS 2005.

[19] E. M. Royer and C.-K. Toh, A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks, IEEE Personal Communications Magazine, April 1999,pp. 46-55.

[20] Chabini,I and Ganupati,S. Design and implementation of parallel dynamic shortest path algorithms for intelligent transportation systems applications. Transportation Research Record , 2001

[21] Li, Wei-Wu. New trends in route guidance algorithm research of intelligent transportation system. Journal of Zhejiang University, 2005, pp 819-824.

[22] Cavendish, D., Gerla, M. Internet QoS routing using the Bellman-Ford algorithm. High Performance Networking. International Conference on High Performance Networking (HPN'98), 1998.

[23] Tan, Guan-Zheng; He, Huan; Sloman, Aaron. Global optimal path planning for mobile robot based on improved Dijkstra algorithm and ant system algorithm. Journal of Central South University of Technology,Vol 13, pp 80-86.

[24] Cvetanovic, Z.; Nofsinger, C. Parallel Astar search on message-passing architectures. IEEE Comput. Soc, 1990. pp.82-90.

[25] Gayraud, T. Authie, G. A parallel algorithm for the all pairs shortest path problem. Parallel Computing. Proceedings of the International Conference 1992.

[26] Luke, S., et al. Tunably decentralized algorithms for cooperative target observation. 2005. Utrecht, Netherlands: Association for Computing Machinery, New York, NY 10036-5701, United States.

[27] Bowyer, R.S. and R.E. Bogner. Self-organizing team formation for target observation. 2001. Orlando, FL: Society of Photo-Optical Instrumentation Engineers.

[28] Brown, K., R.S. Bowyer, and D. Koks. Target location by self-organizing autonomous air vehicles. 2002. Orlando, FL, USA: SPIE-Int. Soc. Opt. Eng.

[29] Chaharsooghi, S.K. and N. Jafari, A simulated annealing approach for product mix decisions. Scientia Iranica, 2007. 14(3): p. 230-235.

[30] Sean Luke, C.C.-R., Liviu Panait, and Keith Sullivan. MASON: A New Multi-Agent Simulation Toolkit, in SwarmFest Workshop. . 2004.

[31] Kirkpatrick, S , Gelatt, C.D., Vecchi, M.P. 1983. Optimization by Simulated Annealing. Science, vol 220, No. 4598, pp 671-680.

[32] Ding, Y., et al. P-CMOMMT algorithm for the cooperative multi-robot observation of multiple moving targets. 2006. Dalian, China: Institute of Electrical and Electronics Engineers Inc., Piscataway, NJ 08855-1331, United States.

[33] Dufour, J.Y. and V. Martin. Active/passive cooperative image segmentation for automatic target recognition. 1994. San Diego, CA, USA.

[34] Fang, L., P. Chen, and S. Liu, Particle swarm optimization with SA for TSP problem. WSEAS Transactions on Information Science and Applications, 2007. 4(6): p. 1157-1162.

[35] Goradia, A., et al. Cooperative multi-target surveillance using a mutational analysis approach. 2005. Monterey, CA, USA: IEEE.

[36] Mitchell, J.W., S.J. Rasmussen, and A.G. Sparks. Effects of target arrival rate on mission performance of cooperatively controlled UAVs with communication constraints. 2004. Nassau, Bahamas: IEEE.

[37] Ogras, U.Y., O.H. Dagci, and U. Ozguner. Cooperative control of mobile robots for target search. 2004. Istanbul, Turkey: IEEE.

[38] Qu, W., D. Schonfeld, and M. Mohamed. Parallel multiple target tracking using multiple cooperative trackers. 2006. San Jose, CA, United States: International Society for Optical Engineering, Bellingham WA, WA 98227-0010, United States.

[39] Tsourdos, A., et al. Cooperative localisation and map building for swarm of UAVs multi-target registration. 2005. London, United Kingdom: Institution of Electrical Engineers, Stevenage, SG1 2AY, United Kingdom.

[40] Ruan, Q.-M., et al., Using constraint satisfaction to cooperate satellites' activities for the mission of area target observation. Yuhang Xuebao/Journal of Astronautics, 2007. 28(1): p. 238-242.

[41] David Abramson., et al. Simulated Annealing Cooling Schedules for the School Timetabling Problem. Asia-Pacific Journal of Operational Research, Vol. 16, 1-22.

[42] Parker, L.E. and B.A. Emmons. Cooperative multi-robot observation of multiple moving targets. 1997. Albuquerque, NM, USA: IEEE, Piscataway, NJ, USA.

[43] Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E. 1953. Equation of State Calculation by Fast Computing Machines. J. of Chem. Phys., 21, 1087-1091.

[44] Ukita, N., Real-time cooperative multi-target tracking by dense communication among Active Vision Agents. Web Intelligence and Agent Systems, 2007. 5(1): p. 15-29.

[45] Yin, C. and G. Hodges, Simulated annealing for airborne EM inversion. Geophysics, 2007. 72(4): p. 189-195.

[46] Yingying, D., et al. P-CMOMMT algorithm for the cooperative multi-robot observation of multiple moving targets. 2006. Dalian, China: IEEE.

[47] Zhang, C.Y., et al., A very fast TS/SA algorithm for the job shop scheduling problem. Computers and Operations Research, 2008. 35(1): p. 282-294.

[48] S.W.Lin., et al. Applying Simulated Annealing Approach for Capacitated Vehicle Routing Problems Systems, Man and Cybernetics, 2006.IEEE Oct. 2006. Vol: 1 p.639-644. Taiwan.

[49] Manousakis, K. et al. Applying Simulated Annealing for Domain Generation in Ad Hoc Networks Vol 7, June 2004 :p.3864 – 3868.IEEE International conference on Communications.

[50] http://www.cs.cmu.edu/~trb/Teambots/Domains/Soccerbots TeamBots (2000)

[51] Albus, J.S. A Theory of Cerebellar Function. Mathematical Biosciences 10, 25–61 (1971)

[52] Atiya, A.F., Parlos, A.G., Ingber, L. A Reinforcement Learning Method Based on Adaptive Simulated Annealing. In: Proceedings of the 46th IEEE International Midwest Symposium on, pp. 121–124 (2003)

[53] Bellman, R. Dynamic Programming. Princeton University Press, Princeton (1957)

[54] Chaharsooghi, S.K., Jafari, N. A Simulated Annealing Approach for Product Mix Decisions. Scientia Iranica 14(3), 230–235 (2007)

[55] Dowsland, K.A. Simulated Annealing. In: Modern Heuristic Techniques for Combinatorial Problems (1995)

[56] Guo, M., Liu, Y., Malec, J. A New Q-learning Algorithm Based on the Metropolis Criterion. Systems, Man and Cybernetics, Part B, IEEE Transactions on 34(5), 2140–2143 (2004)

[57] Howard, R.A. Dynamic Programming and Markov Processes. MIT Press, Cambridge (1960)

[58] Ingber, L. Very Fast Simulated Re-annealing. Mathematical Computer Modelling 12(8), 967–973 (1989)

[59] Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by Simulated Annealing. Science 220(4598), 671–680 (1983)

[60] Klopf, A.H. Brain Function and Adaptive Systems–A Heterostatic Theory. Technical report, AFCRL–72–0164, Air Force Cambridge Research Laboratories, Bedford, MA (1972)

[61] Leng, J., Fyfe, C., Jain, L. Reinforcement Learning of Competitive Skills with Soccer Agents. In: Apolloni, B., Howlett, R.J., Jain, L. (eds.) KES 2007, Part I. LNCS (LNAI), vol. 4692, Springer, Heidelberg (2007)

[62] Leng, J., Jain, L., Fyfe, C. Simulation and Reinforcement Learning with Soccer Agents. Journal of Multiagent and Grid systems, IOS Press, The Netherlands 4(4) (to be published, 2008)

[63] Leng, J., Jain, L., Fyfe, C. Convergence Analysis on Approximate Reinforcement Learning. In: Apolloni, B., Howlett, R.J., Jain, L. (eds.) KES 2007, Part I. LNCS (LNAI), vol. 4692, pp. 85–91. Springer, Heidelberg (2007)

[64] Leng, J., Sathyaraj, B.M. and Jain, L.C., Temporal difference Learning and Simulated Annealing for Optimum control: A Case study, Lecture Notes in Artificial Intelligence, LNAI 4953, KES-AMSTA 2008, Springer-Verlag, Germany, 2008, pp. 495-504.

[65] Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., Teller, E.: Equation of State Calculations by Fast Computing Machines. J. Chem. Phys. 21, 1087–1092 (1953)

[66] Russel, S., Norwig, P.: Artificial Intelligence: A Modern Approach. Prentice-Hall, Englewood Cliffs (2003)

[67] Stef˙an, P., Monostori, L. On the relationship between learning capability and the boltzmann-formula. In: Monostori, L., V´ancza, J., Ali, M. (eds.) IEA/AIE 2001. LNCS (LNAI), vol. 2070, pp. 227–236. Springer, Heidelberg (2001)

[68] Sutton, R.S.: Learning to Predict by the Method of Temporal Differences. Machine Learning 3, 9–44 (1988)

[69] Sutton, R.S., Barto, A.G. Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)

[70] Vien, N.A., Viet, N.H., Lee, S., Chung, T. Heuristic Search Based Exploration in Reinforcement Learning. In: Sandoval, F., Prieto, A.G., Cabestany, J., Gra˜na, M. (eds.) IWANN 2007. LNCS, vol. 4507, pp. 110–118. Springer, Heidelberg (2007).

[71] White, S.R. Concepts of scale in simulated annealing. In: AIP Conference Proceedings, vol. 122, pp. 261–270 (1984).

[72] J. Schlecht, K. Altenburg, B. M. Ahmed, and K. E. Nygard. Decentralized Search by Unmanned Air Vehicles using Local Communication. presented at International Conference on Artificial Intelligence, Las Vegas, NV. 2003

[73] R. J. Bamberger, D. h. Scheidt, R. C. Hawthorne, et al. Wireless Network Communications Architecture for Swarms of Small UAVs in AIIA 3rd Unmanned Unlimited Technical Conference. Chicago, IL: AIAA, 2004.

[74] R. W. Beard and T. W. McLain. Multiple UAV Cooperative Search under Collision Avoidance and Limited Range Communication Constraints, presented at IEEE 42nd Conference on Decision and Control, Maui, HI, 2003.

[75] P. Gaudiano, B. Shargel, E. Bonabeau, and B. Clough. Swarm Intelligence: A NewC2 Paradigm with an Application to Control of Swarms of UAVs, in 8th ICCRTS Command and Control Research and Technology Symposium. Washington, D.C., 2003.

[76] D. L. Gu, G. Pei, M. Gerla, and X. Hong. UAV aided Intelligent Routing for Ad Hoc Wireless Networks, presented at IEEE WCNC, Chicago, IL, 2000.

[77] Rathinam, M. Zennaro, T. Mak, and R. Sengupta. An Architecture for UAV Team Control, in 5th IFAC Symposium on Intelligent Autonomous Vehicles. Lisbon, Portugal, 2004.

[78] C. Dixon, E. Frew, and B. Argrow. Radio Leashing Unmanned Aircraft: Problem Formulation and Cooperative Control Solution, presented at AIAA's Infotech@Aerospace Conference, Virginia, 2005.

[79] T. W. McLain and R. W. Beard. Coordination Variables, Coordination Functions, and Cooperative Timing Missions. Journal of Guidance, Control & Dynamics, vol. 28, no. 1, pp. 150-161, 2005

[80] D. Goldenberg, J. Lin, A. S. Morse, B. E. Rosen, and Y. R. yang. Towards Mobility as a Network Control Primitive, in Mobihoc '04. Roppongi, Japan: ACM, 2004.

[81] L. Schenato, S. Oh, S. Sastry, and P. Bose. Swarm Coordination for Pursuit Evasion Games using Sensor Networks in IEEE conference on Robotics and Automation, April 2005, pp. 2493 – 2498.

[82] J. E. Hurtado, R. D. Robinett, C. R. Dohrmann, and S. Y. Goldsmith. Distributed Sensing and Cooperating Control for Swarms of Robotic Vehicles, presented at IASTED Conference on Control and Applications, Honolulu, HI, 1998.

[83] D. L. Gu, G. Pei, M. Gerla, and X. Hong. UAV aided Intelligent Routing for Ad Hoc Wireless Networks, presented at IEEE WCNC, Chicago, IL, 2000.

[84] B. Argrow, D. Lawrence, E. Rasmussen. UAV Systems for Sensor Dispersal, Telemetry, and Visualization in Hazardous Environments, presented at 43rd Aerospace Sciences Meeting and Exhibit , Nevada , January 2005.

[85] W. Zhang, Stability Analysis of Networked Control Systems, PhD Dissertation, Electrical Engineering and Computer Science Dept., Case Western Reserve University, 14 May, 2001.

[86] C. Hadjicostis and R. Touri. Feedback Control Utilizing Packet Dropping Network Links, in 41st IEEE Conference on Decision and Control (CDC 2002). Las Vegas, NV, 2002, pp. 1205-1210.

[87] B. Lincoln. A Simple Stability Criterion for Control Systems with Varying Delays, presented at 15th IFAC World Congress, 2002.

[88] M. S. Branicky, S. M. Phillips, and W. Zhang. Stability of networked control systems: Explicit analysis of delay, presented at American Control Conference, Chicago, IL, 2000.

[89] D. P. Spanos and R. M. Murray. Robust Connectivity of Networked Vehicles, in 43rd IEEE Conference on Decision and Control. Paradise Island, Bahamas: IEEE, 2004.

[90] B. Sinopoli, L. Schenato, Massimo Franceschetti, et al. Time Varying Optimal Control with Packet Losses, in IEEE International Conference on Decision and Control. Atlantis, Paradise Island, Bahamas, 2004.

[91] W. Zhang and M. S. Branicky. Stability of networked control systems with time-varying transmission period, presented at Allerton Conf. Communication, Control, and Computing, Urbana, IL, 2001.

[92] Zhiqiang Wu, Kumar.H, Davari.A. Performance evaluation of OFDM transmission in UAV wireless communication, in IEEE International Conference on Symposium on System Theory March 2005.

[93] B. Sathyaraj, R. Chellappa Doss, Route Maintenance using Mobility Prediction for Mobile Ad hoc Networks, IEEE International Conference Mobile Adhoc and Sensor Systems Conference, MASS 2005.

[94] E. M. Royer and C.-K. Toh. A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks, IEEE Personal Communications Magazine, April 1999,pp. 46-55.

[95] R.Chellappa, A. Jennings, N. Shenoy. A Comparative Study of Mobility Prediction in Fixed Wireless Networks and Mobile Ad hoc Networks, Proceedings of IEEE International Conference on Communications ICC 2003, Anchorage, Alaska USA, May 2003.

[96] Zhengming Shen,Thomas. J. Location-triggered routing protocol in mobile ad hoc networks, Proceedings of Third IEEE International Symposium on Network Computing and Applications, 2004.

[97] Xiaojiang Du; Dapeng Wu; Wei Liu; Yuguang. Multiclass routing and medium access control for heterogeneous mobile ad hoc networks, IEEE Transactions on Vehicular Technology, Volume 55, Issue 1, Jan. 2006 Page(s):270 – 277.

[98] http://www.sce.carleton.ca/netmanage/tony/swarm.html.

[99] D. F. Hougen, M. D. Erickson, P. E. Rybski, et al., Autonomous Mobile Robots and Distributed Exploratory Missions, in Distributed Autonomous Robotic Systems 4, L. E. Parker, G. Bekey, and J. Barhen, Eds. New York: Springer-Verlag, 2000, pp. 221-230.

[100] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, Coverage control for mobile sensing networks, in IEEE Transactions on Robotics and Automation. Arlington, VA, 2004, pp. 1327-1332.

[101] M. J. Hennebry, A. Kamel, and K. E. Nygard, An Integer Programmng Model for Asssigning Unmanned Air Vehicles to Tasks, in Recent Developments in Cooperative Control and Optimization, S. Butenko, R. Murphey, and P. Pardalos, Eds. Dordrecht, Netherlands: Kluwer Academic Publishers, 2003.

[102] J. P. Desai, J. Ostrowski, and V. Kumar, Controlling formations of multiple mobile robots, presented at Conference on Robotics and Automation, Leuven, Belgium, 1998.

[103] B. Sinopoli, L. Schenato, M. Franceschetti, et al., Time Varying Optimal Control with Packet Losses, in IEEE International Conference on Decision and Control. Atlantis, Paradise Island, Bahamas, 2004.

[104] A. Winfield, Distributed sensing and data collection via broken ad hoc wireless connected networks of mobile robots, in Distributed Autonomous Robotic Systems 4, L. E. Parker, G. Bekey, and J. Barhen, Eds. New York: Springer-Verlag, 2000, pp. 273-282.

[105] L. Pollini, F. Giulietti, and M. Innocenti, Robustness to Communication Failures within Formation Flight, in American Control Conference. Anchorage, AK, 2002.

[106] Z. J. Haas, M. R. Pearlman, and P. Samar. The Zone Routing Protocol (ZRP) for Ad Hoc Networks, Available: www.ietf.org/proceedings/02nov/I-D/draft-ietf-manet-zone-zrp-04.txt, Jul. 2002.

[107] W. H. Liao, Y. C. Tseng, and J. P. Sheu, GRID: A fully location-aware routing protocol for mobile ad hoc networks, Telecommun. Syst., vol. 18, no. 1–3, pp. 37–60, 2001.

[108] Y. B. Ko and N. H. Vaidya, Location-aided routing (LAR) in mobile ad hoc networks, in Proc. ACM/IEEE MobiCom., Dallas, TX, Aug. 1998, pp. 66–75.

[109] K. Xu, X. Hong, and M. Gerla, An ad hoc network with mobile backbones, in Proc. IEEE ICC 2002, vol. 5, New York, NY, Apr. 2002, pp. 3138–3143.

[110] B. Karp and H. T. Kung, GPSR: Greedy perimeter stateless routing for wireless networks, in Proc. ACM/IEEE MobiCom., Boston, MA, Aug. 2000, pp. 243–254.

[111] P. Sinha and S. Krishnamurthy, Scalable unidirectional routing with zone routing protocol (ZRP) extensions for mobile ad hoc networks, in Proc. WCNC, Chicago, IL, Sep. 2000, pp. 1329–1339.

[112] Z. Ye, S. V. Krishnamurthy, and S. K. Tripathi, A routing framework for providing roubustness to node failures in mobile ad hoc networks, J. Ad Hoc networks, vol. 2, no. 1, pp. 87–107, 2004.

[113] X. Du, QoS routing based on multiclass nodes for mobile ad hoc networks, in J. Ad Hoc Networks, vol. 2, no. 3, Jul. 2004, pp. 241–254.

[114] L. Bao and J. J. Garcia-Luna-Aceves, Link state routing in networks with unidirectional links, in Proc. IEEE ICCCN 1999, Boston, MA, Oct. 1999, pp. 358–363.

[115] S. Basagni et al., A distance routing effect algorithm for mobility (DREAM), in Proc. ACM/IEEE MobiCom, Dallas, TX, Aug. 1998, pp. 76–84.

[116] M. Gerla, X. Hong, and G. Pei, Landmark routing for large ad hoc wireless networks, in Proc. IEEE Globecom 2000, San Francisco, CA, Nov. 2000, pp. 11–18.

[117] W. Liu, Y. Zhang, W. Lou, and Y. Fang, DELAR: Device/energy/load aware relaying in heterogeneous wireless ad hoc networks, in Proc. IEEE Military Commun. Conf., Monterey, CA, Nov. 2004.

[118] W. Liu, Y. Zhang, and Y. Fang, Conserving energy in heterogeneous mobile ad hoc networks, in Proc. IEEE Military Commun. Conf., Atlantic City, NJ, Oct. 2005, pp. 17–20.

[119] A. Avudainayagam, Y. Fang, and W. Lou, DEAR: A device and energy aware routing protocol for heterogeneous ad hoc networks, J. Parallel Distributed Comput., vol. 63, no. 2, pp. 228–236, Feb. 2003.

[120] M. S. Branicky, V. Liberatore, and S. M. Phillips, Networked Control System Co-Simulation for Co-Design, in American Control Conference (ACC'03). Denver, Colorado, 2003.

[121] J. R. Hartman, Networked Control System Co-Simulation for Co-Design: Theory and Experiments, M.S. Thesis, Department of Electrical Engineering and Computer Science, Case Western Reserve University, 04 June, 2004.

[122] J. Haas et al., Guest Editorial, IEEE JSAC, Special Issue on Wireless Networks, vol. 17, no. 8, Aug. 1999, pp. 1329–32.

[123] C. Siva Ram Murthy, B.S. Manoj, Ad Hoc Wireless Networks Architectures and Protocols, Prentice Hall, 2004.

[124] Handbook of Wireless Networks and Mobile Computing Edited by Ivan Stojmenovic Copyright © 2000 Wiley [Imprint], Inc.

[125] Azzedine Boukerche., Performance Evaluation of Routing Protocols for Ad Hoc Wireless Networks, Mobile Networks and Applications 9, 333–342, 2004, Kluwer Academic Publishers.

[126] V. Bharghavan, A.Demers, S.Sheker, L.Zhang, MACAW: A media Access for Wireless LANs, Proceedings of ACM SIGCOMM 1994, pp. 212-225.

[127] P. Karn, A New Channel Access Method for Packet Radio, Proceedings of ARRL/CRRL Amateur Radio Computer Networking Conference 1990, pp. 134-140.

[128] C.L. Fullmer and J.J. Gracia-Luna-Aceves, Floor Acquisition Multiple Access (FAMA) for Packet-Radio Networks, Proceedings of ACM SIGCOMM 1995, pp. 262-273.

[129] F. A Tobagi, L.Klienrock, Packet switching in Radio channels: Part II The Hidden Terminal Problem in Carrier Sense Multiple Access and the Busy Tone Solution, IEEE Transactions on Communications, vol. 23 no. 12 pp. 1417-1433, Dec 1975.

[130] J.Deng and Z.J. Haas, Dual Busy Tone Multiple Access (DBTMA): A New Medium Access Control for Packet Radio Networks, Proceedings of IEEE ICUPC 1998 vol 1, pp. 973-977.

[131] C.S Wu and V.O. K. Li, Receiver-Initiated Busy Tone Multiple Access in Packet Radio Networks, ACM Computer Communication Review, vol. 17, no.5, pp. 336-342, Aug 1987.

[132] Talucci, FGerla, M.; Fratta, L. MACA-BI (MACA By Invitation)-a receiver oriented access protocol for wireless multihop networks' The 8th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications. Technical Program, Proceedings, 1997, pt. 2, p 435-9 vol.2.

[133] Toh, C.-K. ,Vassiliou, V.; Guichal, G.; Shih, MARCH: a medium access control protocol for multihop wireless ad hoc networks, MILCOM 2000 Proceedings. 21st Century Military Communications. Architectures and Technologies for Information Superiority, 2000, pt. 1, p 512-16 vol.1.

[134] Shengming Jiang ,Jianqiang Rao, Dajiang He, Xinhua Ling; Chi Chung Ko A simple distributed PRMA for MANETs, IEEE Transactions on Vehicular Technology, v 51, n 2, March 2002, p 293-305.

[135] Chang Wook Ahn, Chung Gu Kang, You Ze Cho, Soft reservation multiple access with priority assignment (SRMA/PA): a novel MAC protocol for QoS-guaranteed integrated services in mobile ad-hoc networks, Vehicular Technology Conference Fall 2000. IEEE, pt. 2, p 942-7 vol.2

[136] Kanodia, V, Chengzhi Li, Sabharwal, A., Sadeghi, B., Knightly, Distributed priority scheduling and medium access in ad hoc networks, ACM Wireless Networks, v 8, n 5, 2002, p 455-66.

[137] Kanodia, V,Li, C., Sabharwal, A, Sadeghi, B., Knightly, Ordered packet scheduling in wireless ad hoc networks: Mechanisms and performance analysis , Proceedings of the International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), 2002, p 58-70.

[138] Karthigeyan.I., Manoj B.S. Murthy C. Siva Ram, A distributed laxity-based priority scheduling scheme for time-sensitive traffic in mobile ad hoc networks, Ad Hoc Networks Journal , v 3, n 1, January, 2005, p 27-50.

[139] D. Clark, S. Shenker and L. Zhang, Supporting real-time applications in an integrated services packet network: Architecture and mechanism, Proceedings of ACM SIGCOMM'92, 1992 pp. 14–26.

[140] Ad-hoc On-Demand Distance Vector Routing, Charles E. Perkins, Elizabeth M. Royer, Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications, New Orleans, February 1999.

[141] Ad Hoc Wireless Mobile Wireless Networks, C-K Toh, Prentice Hall PTR, 2002.

[142] Dynamic Source Routing in Ad Hoc Wireless Networks, D.B. Johnson and D.A. Maltz, Mobile Computing, chapter 5, pp.153-181, Kluwer Academic Publishers, 1996.

[143] Jacquet, P., Muhlethaler, P., Clausen, T. Laouiti, A., Qayyum, A., Viennot, L., Optimized link state routing protocol for ad hoc networks, IEEE Proceedings IEEE International Multi Topic Conference 2001. IEEE INMIC 2001, pp. 62-68.

[144] Z.J. Haas, A new routing protocol for the reconfigurable wireless networks, Z.J. IEEE 6th International Conference on Universal Person Communications Record. ICUPC '97. 1997, pt. 2, pp. 562-6 vol.2

[145] John J. Murray, Chadwick J. Cox, George G. Lendaris, Richard Saeks, Adaptive Dynamic Programming, IEEE Transactions on Systems, MAN. and CYBERNETICS 2002.

[146] Sung-Ju Lee, W. Su, J. Hsu, M. Gerla, R. Bagrodia, A performance comparison study of ad hoc wireless multicast protocols, IEEE INFOCOM 2000, vol. 2, 565 -574.

[147] I El Kabary, S Ghoniemy, A Elnahas, Critical path routing (CPR) protocol for mobile ad hoc networks, Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Networks 2006, p 285-91

[148] J. J. Garcia-Luna-Aceves, Loop-free routing using diffusing computations, IEEE/ACM Trans. on Networking, 1993, pp. 130-141.

[149] Anthony Finn, Kuba Kabacinski, & Samuel Picton Drake Design Challenges for an Autonomous Cooperative of UAV's Defence Science & Technology Organisation Department of Defence, Australia.

[150] P. Sinha, R. Sivakumar, and V. Bharghavan, CEDAR: A Core Extraction Distributed Ad Hoc Routing Algorithm, IEEE Journal on Selected Areas in Communications, vol. 17, no. 8, pp. 1454 – 1466, August 1999.

[151] M. Joa-Ng and I. T. Lu, A Peer-to-Peer Zone based Two-Level Link state Routing for Mobile Ad Hoc Networks, IEEE Journal on Selected Areas in Communications, vol. 17, no. 8, pp. 1415-1425, August 1999.

[152] V. Ramasubramanian, Z. J. Haas, E. G. Sirer, SHARP: A Hybrid Adaptive Routing Protocol for Mobile Ad Hoc Networks, Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing (Mobihoc), Annapolis, Maryland, June 2003.

[153] Lundy, M., Mees, A. 1986. Convergence of an Annealing Algorithm. Math. Prog., 34, 111-124 Lundy, M., Mees, A. 1986. Convergence of an Annealing Algorithm. Math. Prog., 34, 111-124

[154] Johnson, D.S., Aragon, C.R., McGeoch, L.A.M. and Schevon, C. 1991. Optimization by Simulated Annealing: An Experimental Evaluation; Part II, Graph Coloring and Number Partitioning. Operations Research, 39, 378-406

[155] Kirkpatrick, S, Gelatt, C.D., Vecchi, M.P. 1983. Optimization by Simulated Annealing. Science, vol 220, No. 4598, pp671-680

[156] Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E. 1953. Equation of State Calculation by Fast Computing Machines. J. of Chem. Phys., 21, 1087-1091.

[157] Russell, S., Norvig, P. 1995. Artificial Intelligence A Modern Approach. Prentice-Hall

[158] Rutenbar, R.A. 1989. Simulated Annealing Algorithms: An Overview. IEEE Circuits and Devices Magazine, Vol. 5, No. 1, pp 19-26.

[159] Van Laarhoven, P.J.M, Aarts, E.H.L. 1987. Simulated Annealing: Theory and Applications. D. Reidel Publishing.

[160] White, S.R. 1984. Concepts of Scale in Simulated Annealing. Proceedings International Conference on Computers in Design, pp 646-665.

[161] Aarts, E.H.L., Korst, J.H.M. 1989. Simulated Annealing and Boltzmann Machines. Wiley, Chichester.

[162] Hajek, B. 1988. Cooling Schedules for Optimal Annealing. Mathematics of Operations Research, vol 13, No. 2, pp311-329.

[163] Fickes, Michael (October 1, 2004). "Automated Eye In The Sky". *GovernmentSecurity.com*. http://govtsecurity.com/mag/automated_eye_sky/. Retrieved 2009-03-13.

[164] http://www.jervisbwebb.com/Brochures/Smart%20Warehouse%20White%20Paper.pdf

Appendix 1 Abbreviations

ABR	Associativity Based Routing
AP	Access Point
ADV	Adaptive Distance Vector Routing
AODV	Ad Hoc On Demand Distance Vector Routing Protocol
CBRP	Cluster Based Routing Protocol
CGSR	Cluster head Gateway Switch Routing
CEDAR	Core-Extraction Distributed Ad Hoc Routing
CSMA/CA	Carrier Sense Multiple Access/Collision Avoidance
CSMA/CD	Carrier Sense Multiple Access/Collision Detection
DDR	Distributed Dynamic Routing Algorithm
DREAM	Distance Routing E_ect Algorithm for Mobility
DSDV	Destination Sequenced Distance Vector Routing Protocol
DSR	Dynamic Source Routing
DSSS	Direct-sequencing spread-spectrum
DST	Distributed Spanning Tree Protocol
FDM	Frequency Division Multiplexing
FHSS	Frequency-hopping spread-spectrum
FORP	Flow Oriented Routing Protocol
FSLS	Fuzzy Sighted Link State Algorithms
FSR	Fisheye State Routing
Gateway	A network point that acts as an entrance to another network.
GEDIR	Geographic Distance Routing
GPSR	Greedy Perimeter Stateless Routing
GSR	Global State Routing
HSR	Hierarchical State Routing
LANMAR	Landmark Routing Protocol
LAR	Location Aided Routing
LMR	Lightweight Mobile Routing
LRR	Link Reversal Routing
MAC	Medium Access Control
Router	A router is a networking device that forwards data-packets across an internet work, onto their destinations.
OLSR	Optimized Link State Routing
QOS	Quality of Service
SSA	Signal Stability-Based Adaptive Routing
STAR	Source Tree Adaptive Routing
TBRPF	Topology Broadcast Based on Reverse Path Forwarding
TDMA	Time Division Multiple Access. TDMA allows several users to share the same frequency using different timeslots
TORA	Temporally Ordered Routing Algorithm
UAV	Unmanned aerial vehicle
UGV	Unmanned ground vehicle
UUV	Unmanned underwater vehicle
WAR	Witness Aided Routing
WRP	Wireless Routing Protocol
ZRP	Zone Routing Protocol