12-29-2015

Proactive Decision Support for Intelligent Routing of Unmanned Aerial Systems in Dynamic and Uncertain Mission Environments

bala kishore nadella
university of connecticut, balakishore.nadella@gmail.com

Proactive Decision Support for Intelligent Routing of Unmanned Aerial Systems in Dynamic and Uncertain Mission Environments

Bala Kishore Nadella

B.Tech., Acharya Nagarjuna University, India,

2011

A Thesis

Submitted in Partial Fullfilment of the

Requirements for the Degree of

Masters of Science

at the

University of Connecticut

2016

APPROVAL PAGE

Masters of Science

Proactive Decision Support for Intelligent Routing of Unmanned Aerial Systems in Dynamic and Uncertain Mission Environments

Presented by

Bala Kishore Nadella, B.S.

Major Advisor

Krishna R. Pattipati

Associate Advisor

Yaakov Bar-Shalom

Associate Advisor

Balakumar Balasingam

University of Connecticut

2016

DEDICATION

to

my mom

ACKNOWLEDGEMENTS

Firstly, I would like to express my sincere gratitude to my advisor *Prof. Krishna R. Pattipati* for the continuous support of my study and related research, for his patience, motivation, and immense knowledge. His guidance was invaluable in helping me complete my research and this thesis. I could not have imagined having a better advisor and mentor.

Also I would like to thank the rest of my thesis committee: *Prof. Yaakov Bar-Shalom*, and *Dr. Balakumar Balasingam*, for their insightful comments and encouragement. Special thanks to *Ciara Sibley, Joseph Coyne, Jim Thomas* for their valuable suggestions during my research.

I would also like to thank *Gopi Vinod Avvari, Manisha Mishra*, and *David Sidoti* for their guidance and support over the past two years. Many thanks to other labmates *Ali Abdollahi, Avnish Kumar, Bharath Pattipati, Devaki Pasupuleti, Lingyi Zhang, Niranjan Raghunathan, Pujitha Mannaru, Rajeev Ghimire*, as well, for their timely support.

I would also like to thank my roommates *Goutham Kukkadapu, Kranti Pothapu* and other friends for their moral support and for always being there for me.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

1.1 Proactive Decision Support: What and Why?

Future maritime battlespace environments are expected to be more complex and distributed, and given our ability to collect massive amounts of data using heterogeneous sources, it is imperative to process the collected data and present decision-relevant information to the decision makers (DM) in a timely manner for making effective decisions under dynamic, uncertain and unpredictable mission conditions (e.g., sudden changes in mission goals, environment, assets and mission tasks). In addition to data/information processing, it is also crucial to dynamically allocate the scarce and expensive resources to maximize the amount of decision-relevant information collected, while increasing the probability of mission success. These challenges incur the need to develop Proactive Decision Support (PDS) tools for the acquisition, fusion, and transfer of the *right* data/information/knowledge from the *right* sources in the *right* context to the *right* DM at the *right* time for the *right* purpose, a concept known as 6R [1]. By context-driven, we mean dynamically integrated knowledge that is (i) relevant to

the mission, the environment, assets and tasks (including activities), (ii) informed by up-to-date data sources/INTEL, and (iii) congruent with the workflow and individual DM's role in the mission, workload, time pressure and expertise.

PDS is a systematic decision framework for automated processing, interpretation, and development of intelligent decisions using large volumes of structured, unstructured and semi-structured data, while simultaneously decreasing the time necessary to arrive at a decision. Fig. 1.1 briefly describes the PDS framework, where the environmental changes are given as intelligence updates, chat commands and situational reports. Given the real-time situational reports/updates, it is important to detect and diagnose the cause for the change or event occurrence in order to evaluate and predict the impact and severity on the current plan. Based on the calculated impact or the probability of occurrence of the event, the decision model should anticipate and adapt its plans to the changes in the mission environment. Further, as these plans are conveyed to the DM/user as recommendations, it is crucial to represent the context-relevant courses of action (COAs) in a meaningful and succinct manner. In this thesis, we focus on developing and validating PDS algorithms for a) proactive allocation of Unmanned Aerial Systems (UASs) by extracting, processing, and integrating context relevant information, while balancing operator workloads; b) enhancing dynamic routing and re-routing capabilities; c) unobtrusively conveying COA recommendations to DMs; and d) adapting plans as new targets of opportunity appear or information is updated about a target and/or UAS. The proposed algorithms are embedded in the Supervisory Control Op-

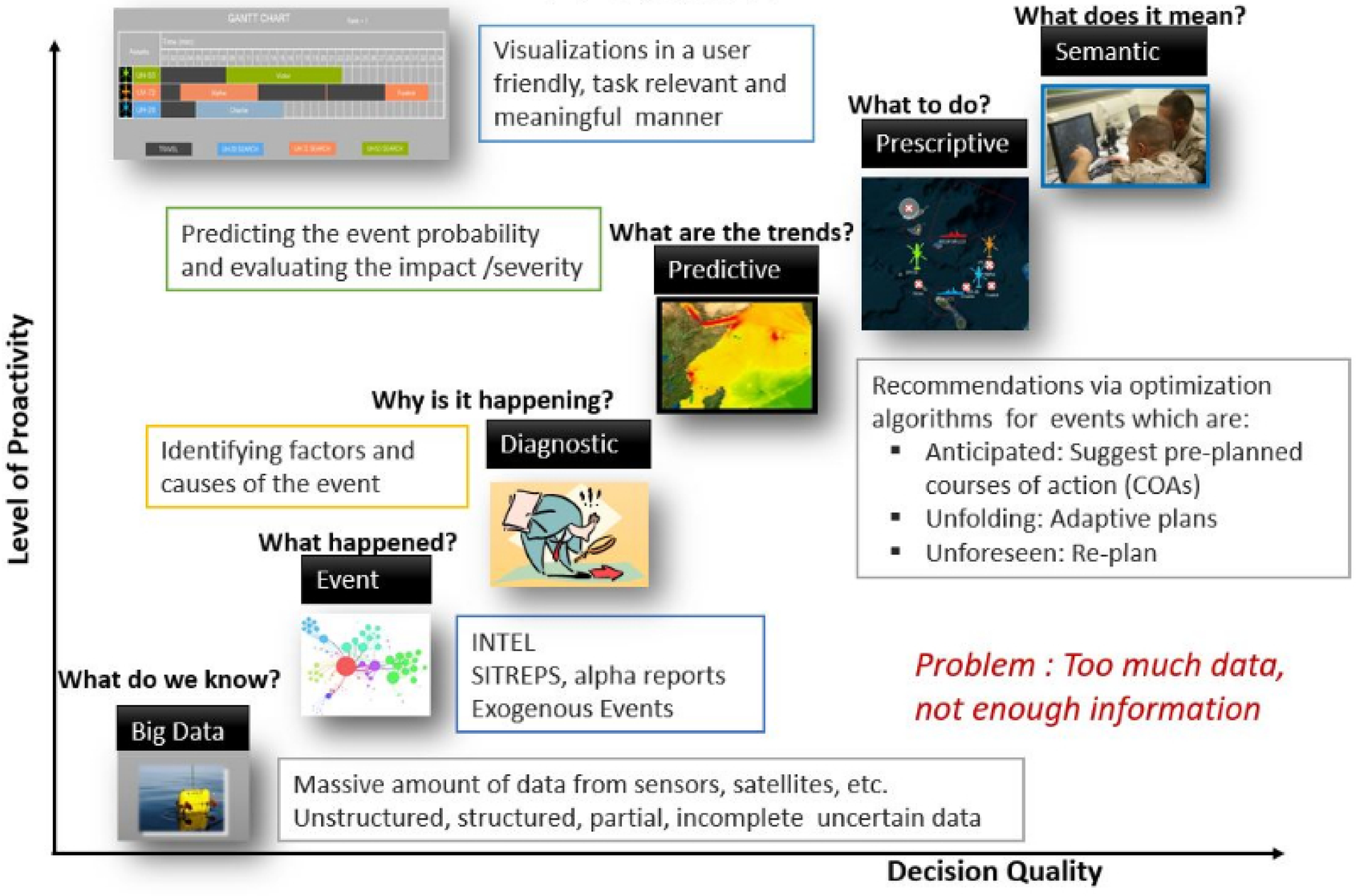

Fig. 1.1: Proactive decision support system in a nut shell.

erations User Testbed (SCOUT^{TM}), an experimental paradigm developed by the Naval Research Laboratory-Washington D.C.

1.1.1 Motivating Application

Unmanned Aerial Systems (UASs) are becoming ubiquitous and have been playing an increasingly important role in military and civilian operations due to their exceptional operational advantages, such as high-risk mission acceptance and ultra-long endurance capabilities, which cannot be reasonably performed by manned aircraft. Current UAS

operations are characterized by teams of operators with highly specialized roles, where the task demands on each operator are independent and highly variable resulting in sub-optimal tasking, and consequently reduced mission performance and occasional mishaps. Channelized attention (or cognitive tunneling) has been implicated in numerous operational mishaps, which could have been preventable. A 2012 U.S. UAS System Report to Congress [2, 3] stated that 68% of UAS accidents are attributable to human error. As an example, operators flying an MQ-1B Predator in Afghanistan in 2009 were so focused on a fierce firefight that they failed to notice that the unmanned aircraft was headed toward a mountain [4]; the aircraft was destroyed on impact, and damage was estimated to be $3.9 million. In effect, eight of the soldiers, who were to be provided air support by the Predator, were killed. If alerts and tasking had been appropriately provided to the right operators on the team, those lives could have been saved.

Additionally, future UAS operations envision multiple ground-based, aerial, surface and subsurface unmanned vehicles to be simultaneously controlled by teams of operators, which requires effective coordination and workload hand-offs among operators. Thus, research is needed to develop methods to quantify the value of information and to proactively present this information to the UAS operators for faster, context-specific anticipatory decision making.

Motivated by the need to assist UAS operators in efficiently managing their workloads, we consider a decision task where an operator must manage multiple UASs and determine the best routes to send their assets to search for targets of varying reward.

Targets have some degree of uncertainty associated with their locations, requiring the UAS to search for targets within the circular regions of varying radii. During mission execution, the user must update UAS parameters and target parameters (e.g., speed, altitude etc.), based upon real-time intelligence provided via chat messages to re-balance workload among human operators, and (re)schedule operator-to-task assignments depending on the operator workload. The dynamic assignment and routing algorithms developed in this thesis provide time-critical decision support to the user on emerging and changing tasks, thereby increasing the time available for human decision making.

We embed our proposed decision support algorithms within the Supervisory Control Operations User Testbed (SCOUT^{TM}) [5], an experimental human subject testbed developed by the Naval Research Laboratory-Washington DC. SCOUT^{TM} was designed for the purpose of exploring UAS operator performance in a single operator-multi UAS environment, but assumes some advances in automation necessary to conduct supervisory control operations involving multiple heterogeneous systems. One such advancement includes updating plans based on mission context in order to provide courses of action (COA) recommendations to the operators in a proactive and unobtrusive manner. Specifically, the decision support algorithms we embed in SCOUT^{TM} software have the following capabilities: a) dynamic allocation of targets to UASs in order to maximize reward within the target deadlines; b) plan updates based on context changes (eg., new target, updated information); and c) plan adaptation to operators' risk propensity (viz., risk seeking, risk neutral, risk averse). The overall goal is to develop a proactive

SCOUT^{TM}, which permits the evaluation of COAs, while assuring that the operators are attending to the right task at the right time and that task demands do not exceed the operator's capabilities in a multi-mission environment.

The real-time update of the UAS target parameters or the mission context changes need to be conveyed to the operator in a transparent and easily understood manner. UAS operators would benefit greatly from the COA recommendation algorithms proposed in this thesis by way of enhanced rapid planning and re-planning capabilities, effective allocation of the UAS to maximize mission success, while handling high workloads. Additionally, SCOUT^{TM} has the ability to capture and synchronize data streams from eye tracking sensors, heart and respiration sensor systems, mouse logging and keystroke logging, all of which are synchronized with simulation events. This allows the investigation of the impact of decision support on both the operator's physiological state and simulation-interaction (e.g., decision support causes the operator's eye gaze patterns to become more varied and keystrokes to be less frequent). Real-time analysis of the operator data (performance, eye tracking, physiological, mouse and keystroke data) can drive alerting and decision support when data indicates that performance is trending negatively.

1.1.2 Related Research

The UAS routing problem is closely related to the vehicle routing problem (VRP). The VRP involves the synthesis of a set of minimum-cost vehicle routes for a fleet

of vehicles that provide service to a set of customers. The VRP has a wide range of applications in the real world, e.g., collection of mail, pickup of children by school buses, inspection tours, etc. In [6], Mandell proposed a method to convert the m-vehicle routing problem to a single-tour traveling salesman problem (TSP) passing through n+m points, thus converting the VRP into a TSP with restrictions. Christofides [7, 8] proposed a branch-and-bound method with a spanning tree and shortest path relaxations of the VRP problem.

The UAS allocation and routing problem considered in this thesis has a number of distinct features that makes it more challenging than the existing VRP problem. These include asset safety, uncertain service times, and dynamic targets. Ter Mors [9] proposed a context aware route planning, where a conflict free shortest-time route plan is developed, while avoiding deadlock situations. However, context aware route planning is computationally expensive in comparison with traditional route planning techniques. From the perspective of computational complexity, VRPs are quite difficult and become even worse when time window constraints are added. Kohl [10] proposed an algorithm for the VRP with time windows using Lagrangian relaxation of the constraint set requiring that all customers must be serviced.

The open vehicle routing problem (OVRP) is a variant of the standard VRP. Sariklis [11] proposed a heuristic method to solve a symmetric OVRP that does not include a maximum route length restriction. Li [12] proposed VRP algorithms with stochastic service times and compared their performance against tabu search, deter-

ministic annealing, and neighborhood search techniques.The OVRP corresponding to the UAS assignment and routing problem has a source node for each UAS (its initial position) and a route (a sequence of targets) that terminates at the last target on its route. The OVRP is known to be *NP*-hard and, consequently, the UAS routing problem discussed in this thesis is *NP*-hard, as well. Unlike the conventional OVRP, here the UASs start from arbitrary locations; they have variable speeds and sensor sweep widths, rendering the search (service) times dynamic. In addition, targets have opportunity windows (deadlines).

Due to the limited success of exact methods, considerable attention and research effort have been devoted to the development of efficient approximate algorithms (or heuristics) which can provide near optimal solutions to large-sized problems. Tabu search implementation is the first metaheuristic implementation of VRP by Taillard [13]. Osman [14] proposed tabu search with simulated annealing which reduces the computational time by more than 50%. Later Toth [15] proposed granular tabu search based on the use of drastically restricted neighborhoods also known as *granular*, and may be seen as an efficient implementation of candidate-list strategies proposed for tabu-search algorithms. Tarantilis [16] proposed an adaptive memory heuristic called BoneRoute - a population-based method where a new solution is produced out of components of routes of previous solutions. Li [17] proposed a deterministic annealing method, a variant of metaheuristic using a variable-length neighbor list to solve large scale VRP's. Mester [18, 19] proposed active guided evolution strategies and also ge-

netic search with large neighborhood search to solve large scale vehicle routing problems with time windows. Pisinger [20] proposed an adaptive large neighborhood search by choosing among a number of insertion and removal heuristics, to intensify and diversify the search where high quality solutions are produced, as the algorithm is self calibrating. Nagata [21] proposed a edge assembly-based memetic algorithm, a hybrid evolutionary algorithm that combines the global and local searches. Prins [22] proposed a heuristic which is a combination of greedy randomized adaptive search procedure and evolutionary local search hybrid. The main features of this algorithm are its simple structure, and alternating between solutions encoded as tours and a fast local search based on a sequential decomposition of moves.

1.2 Organization of Thesis

The rest of the thesis is organized as follows: Section 2.1 introduces the problem and section 2.2 formulates it as an OVRP. Section 2.3 describes the optimal and heuristic algorithms to solve the problem and discusses simulation results. Decision support enhancements to SCOUT™ are discussed in chapter 3. Finally, the thesis concludes with a summary and future work in sections 4.1 and 4.2, respectively.

1.3 Publications

[1] **B. K. Nadella**, G. V. Avvari, D. Sidoti, M. Mishra, L. Zhang, and K. R. Pattipati, C. Sibley, J. Coyne, S. Monfort, "Proactive Decision Support for Dy-

namic Assignment and Routing of Unmanned Aerial Systems," *accepted to IEEE Aerospace Conference*, Big Sky, Montana, 2016.

[2] G. V. Avvari, D. Sidoti, M. Mishra,L. Zhang, **B. K. Nadella**, and K. R. Pattipati, Dynamic Asset Allocation for Counter-SmugglingOperations Under Disconnected, Intermittent and Low-Bandwidth Environment *in Computational intelligence in Security and Defense Applications*, May 2015.

[3] D. Sidoti, X. Han, L. Zhang, D. F. M. Ayala, M. Mishra, **B. K. Nadella**, S. Sankavaram, D. Kellmeyer, J. A. Hansen, and K. R. Pattipati, Context-Aware Dynamic Asset Allocation for Maritime Interdiction Operations Part II: Application and Sensitivity Analysis, *submitted to Systems, Man and Cybernetics: Human-Machine Systems IEEE Transactions* on, 2015

[4] D. Sidoti, X. Han, L. Zhang, D. F. M. Ayala, M. Mishra, **B. K. Nadella**, S. Sankavaram, D. Kellmeyer, J. A. Hansen, and K. R. Pattipati, Context-Aware Dynamic Asset Allocation for Maritime Interdiction Operations Part I: Approximate Dynamic Programming Approaches, *submitted to Systems, Man and Cybernetics: Human-Machine Systems IEEE Transactions* on, 2015

[5] D. Sidoti, G. V. Avvari, M. Mishra, L. Zhang, **B. K. Nadella**, J. A. Hansen, and K. R. Pattipati, A Multi-Objective Shortest Path Algorithm for Ship Routing in a Dynamic Weather-Impacted Environment, *submitted to Systems, Man and Cybernetics: IEEE Transactions* on, 2015.

Chapter 2

Problem Description and Solution Approach

2.1 Problem Description

NRL-DC developed the Supervisory Control Operations User Testbed (SCOUT^{TM}), which is a highly flexible simulation/testbed environment created for the purposes of exploring operator tasking and performance during unmanned systems missions. This testbed will be utilized for data collection and iterative development and assessment of optimization algorithms. Human subject experiments within NRLs SCOUT^{TM} will be used to assess the impact of various alerting and task allocation algorithms within a single operator-multiple UAS environment. The SCOUT^{TM} 1.0 version is a single user platform that allows an operator to control multiple simulated unmanned assets. SCOUT^{TM} was developed to be a complex unmanned systems simulation that users (operators) can learn to interact with quickly, but features many of the core elements of unmanned vehicle control (e.g., route planning, target risk assessment, restricted operating zone management and customer/team-member communication). It also enables the investigation of higher order cognitive processes, such as decision making

under uncertainty and the impact of low or high workload on risk taking behavior in a UAS environment. The key objective of this thesis is to attempt to develop a proactive version of SCOUT™.

SCOUT™ has 1) a mission planning phase where the user must determine the best plan for sending multiple unmanned vehicles to search for multiple targets, with varying priority levels and constraints, and 2) a mission execution phase where the user monitors vehicle statuses, responds to communication requests, and makes updates to the overall plan as new targets or intelligence reports are provided. The design of SCOUT™ and the tasking/functionality was informed by interaction and feedback from actual UAS operators. However, in order to enhance user motivation, SCOUT™ has elements of a game, such as points that are awarded for finding targets and responding to communications, as well as points being deducted for sending a UAS into a restricted area without requesting access.

SCOUT™ has a mission editor graphical user interface (GUI), which allows rapid and intuitive design of new scenarios of interest, such as varying mission management levels or planning difficulty by altering the numbers of targets and their associated constraints (deadlines, priority level, and location uncertainty level). The GUI enables scheduling of events at specific times, such as where and when various targets appear and types of unmanned assets the user is controlling.

SCOUT™ has the ability to capture and synchronize data streams from eye tracking sensors, heart and respiration sensor systems, mouse logging and keystroke logging,

(**a**) Left Screen - shows the position of the UASs, targets in Google Maps, and information about the UASs and targets

(**b**) Right Screen - shows the sensor feed, UAS information (speed, altitude), and two chat windows for INTEL updates and commands

Fig. 2.1: Supervisory Control Operations User Testbed (SCOUT^{TM})

all of which are synchronized with simulation events. This allows the investigation of the impact of decision support on both the user's physiological state and simulation-interaction (e.g., decision support causes the user's eye gaze patterns to become more varied and keystrokes to be less frequent). Real-time analysis of the user data (performance, eye tracking, physiological, mouse and keystroke data) can drive alerting and decision support when data indicates that performance is trending negatively.

2.2 Problem Formulation

The notation used in the rest of this thesis is included in Table 2.1. Consider a scenario where a team of UASs, $U = \{u_i | i = 1, \ldots, m\}$, are to be scheduled to search for a set of geographically distributed targets, $T = \{t_j | j = 1, \ldots, n\}$. Each target t_j is characterized by its geographic location and a region of uncertainty around it, given by the radius r_j, reward R_j and a deadline D_j by which the targets need to be searched to obtain the reward. The opportunity to search for each target disappears after the corresponding deadline has passed. Each UAS u_i, has a velocity v_i, and sensor sweep width sw_i.

During the mission, a UAS u_i can obtain a reward R_j by completely searching the uncertainty region of target t_j before the target's deadline D_j. If the target's uncertainty region is not searched completely, a partial reward is earned (to be formalized below). In addition, targets cannot enter restricted operating zones (ROZs), modeled as n-sided polygons, without obtaining prior permission to do so. If a UAS enters a ROZ, then

a penalty is incurred by the UAS, which reduces the cumulative reward. The mission scenario is dynamic in that new targets may appear at random times and positions, and information on a UAS may be updated at any time. A screen shot of the SCOUT^{TM} user interface is shown in Fig. 2.1. The SCOUT^{TM} user interface displays two windows to the operator: one on the left screen and the other on the right screen. The left screen shows the positions of targets, UASs, and ROZs on the Google map. It has the target information box, which provides the operator with information on the minimum time for each UAS to reach a target and the latest time to leave the target after searching. It also has UAS route builder boxes, where an operator assigns targets to UAS(s). The right screen shows the sensor feed, and speed of each UAS. It also includes Intelligence and Command chat boxes, which provide the updates on target position, uncertainty radius, and UAS speed during mission execution.

Table 2.1: SUMMARY OF NOTATION

T	Set of targets
U	Team of UASs
ROZ	Group of restricted operating zones
$\mathbb{T}$	Set of target indices
$\mathbb{U}$	Set of asset indices
i	Asset index
j	Target index or dummy index

k	Target index
ψ_i	Dummy index (Depot)
r_j	Radius of uncertainty of target t_j
R_j	Reward associated with target t_j
v_i	Velocity of UAS u_i
sw_i	Sweep width of UAS u_i
ξ_{ijk}	Binary decision variable
τ_k	Arrival time of assigned UAS at target t_k
$E_{ik}(\tau_k)$	Utility obtained from target t_k
t_{ijk}	Time taken by UAS u_i to travel from target t_j to target t_k
s_{ik}	Time spent by UAS u_i in searching for target t_k in targets, uncertainty circle
fs_{ik}	Time required by UAS u_i in searching for target t_k in targets, uncertainty circle completely
D_k	Deadline of target t_k
E	Cumulative utility
MT	Mission time
α	Sequence index
$\pi(i)$	Set of all feasible single routes from UAS u_i
$P_{(i,\alpha)}$	Path sequence assigned to UAS u_i
$\Theta_{(i,\alpha)}$	Set of targets explored in path sequence $P_{(i,\alpha)}$

$(n_\alpha)_i$	Number of targets UAS u_i explores in the sequence $P_{(i,\alpha)}$
F_m	Set of unassigned targets
$\Theta_{(r,\kappa[r])}$	Feasible single route assigned to UAS

2.2.1 Modeling Search Time

Mission time is defined as the maximum time taken by the UASs to completely explore the targets in their respective routes (referred to as makespan in the scheduling literature). Traversal times and search times are the constituent components of mission time. While calculating the distance traversed by UAS u_i to reach target t_j, we first evaluate if the line of sight (LOS) between the UAS and the target intersects with any of the active ROZ. If there is an ROZ conflict, and no LOS exists between u_i and t_j, then a graph of nodes and arcs is created, where the nodes are the vertices of each ROZ and the positions of u_i and t_j. The arcs are formed by joining a node to every other node in the graph. Arc cost corresponds to the distance between the nodes if the arc does not intersect any ROZ, and is set to infinity, otherwise. We use Dijkstra's algorithm [23] to obtain the shortest path between u_i and t_j without intersecting the ROZ (and hence avoids incurring any penalty). The intermediate nodes in the path are added as waypoints of the path from u_i to t_j as shown in Fig. 2.2.

Each target has a radius of uncertainty around it. Therefore, once the UAS u_i approaches the vicinity of the target t_j, it performs the search activity in the form of

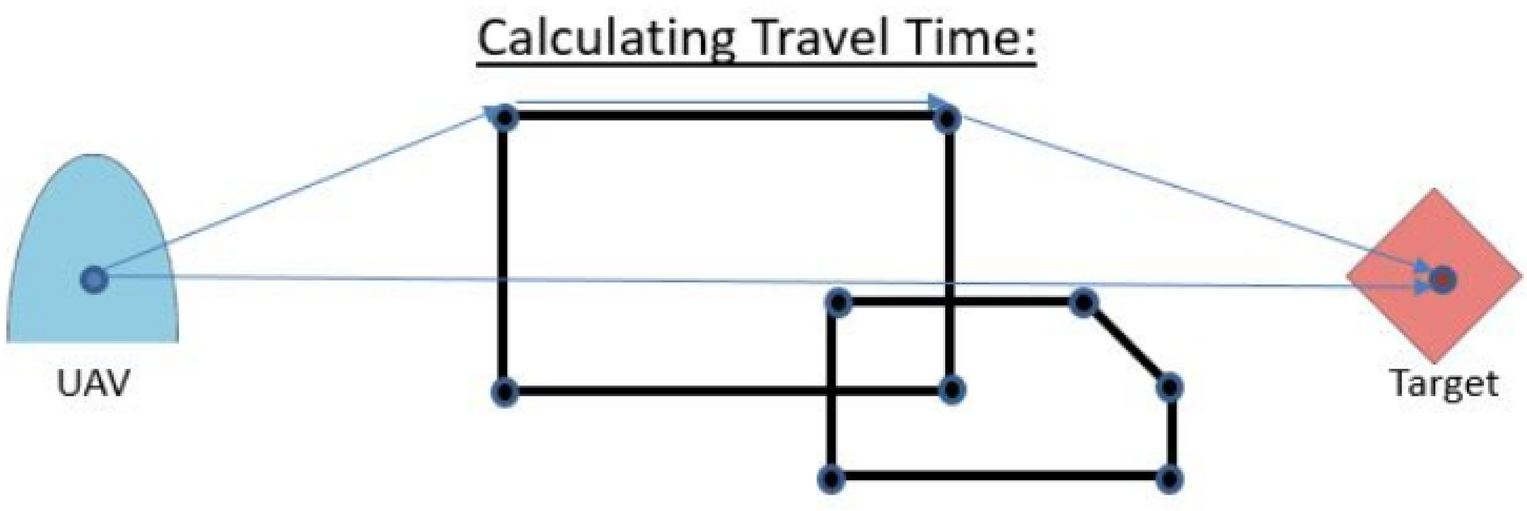

Fig. 2.2: Travel time calculation

concentric circles as shown in Fig. 2.3. Given the sweep width sw_i of the sensor associated with UAS u_i, the sensor scans the uncertainty region from the center of the uncertainty region to its outer edge. The time taken by the UAS u_i to completely search target t_j's uncertainty region is given by

$$fs_{ij} = \sum_{p=1}^{\left\lceil \frac{r_j}{sw_i} \right\rceil} \frac{2\pi \left((p - \frac{1}{2}) \times sw_i \right)}{v_i} \tag{2.1}$$

where r_j is the uncertainty radius of target t_j, sw_i is the sweep width of the sensor, and v_i is the velocity of UAS u_i. The total time spent by UAS u_i on target t_j is the sum of time taken to reach the target t_j and the time taken to search its uncertainty region.

2.2.2 Expected Reward for Partial Searches

Each target t_j has a corresponding deadline, before which the UAS u_i has to search in the target's uncertainty region to obtain the associated reward. There may be allocations

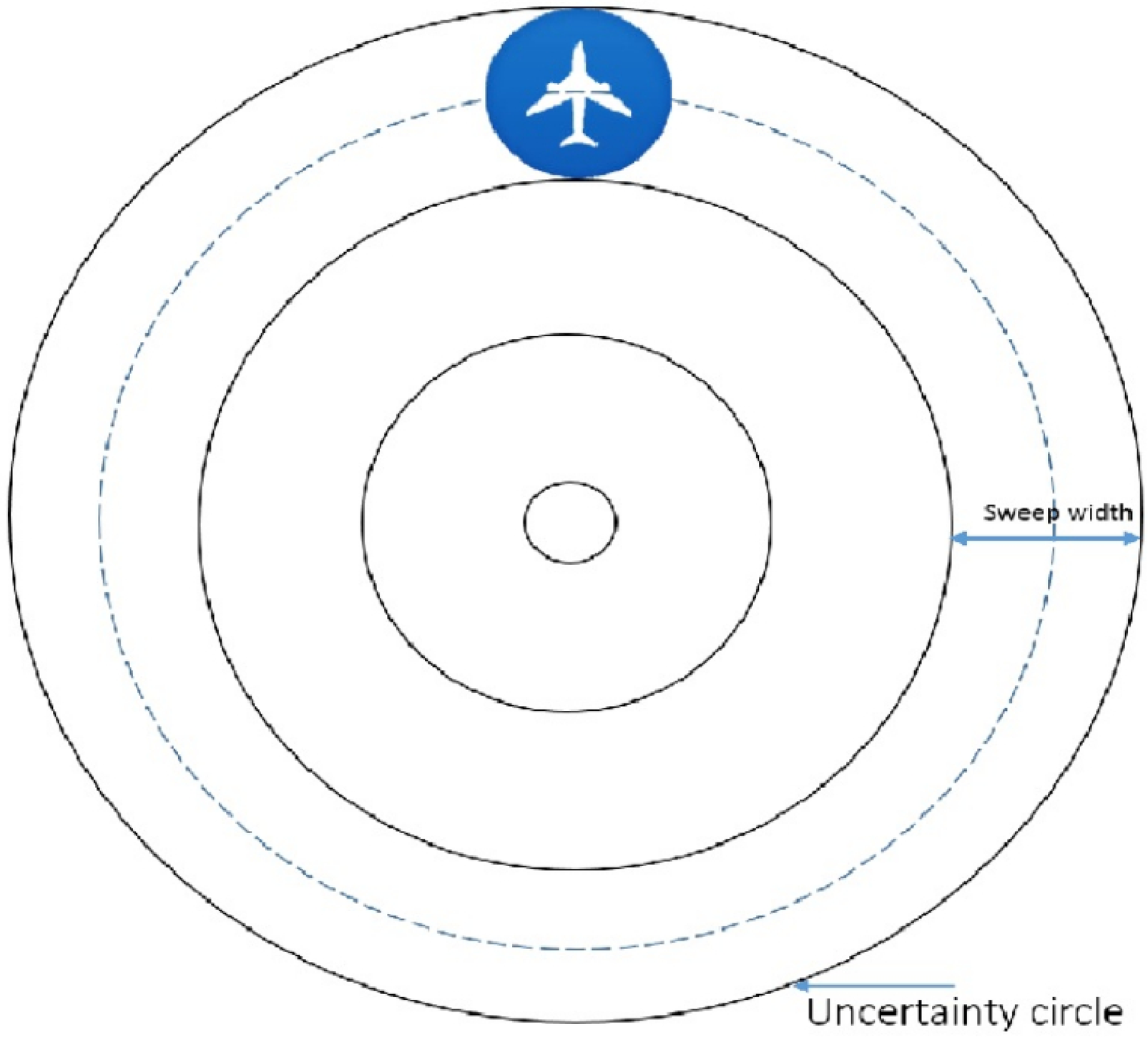

Fig. 2.3: UAS u_i associated with sensor of sweep width sw_i searching in the uncertainty region of target t_j in concentric circles.

where the UAS u_i can only partially search the uncertainty region of target t_j. This depends on the UAS's arrival time τ_j at target t_j and its deadline D_j. The time spent by

the UAS u_i in searching target t_j's uncertainty region before its deadline D_j is given by

$$s_{ij}(\tau_j) = \begin{cases} \max(0,\ D_j - \tau_j), & \text{if } \tau_j + fs_{ij} > D_j \\ \\ fs_{ij}, & \text{otherwise} \end{cases}$$

The estimated reward from partial search is the ratio of the time searched by UAS u_i in the uncertainty region of target t_j to the time required by UAS u_i to search the uncertainty region completely, multiplied by the corresponding reward R_j. Formally,

$$R_{ij}(\tau_j) = \frac{s_{ij}(\tau_j)}{fs_{ij}} \times R_j \tag{2.2}$$

2.2.3 Modeling Risk Propensity of Operators

Expected utility theory has dominated the analysis of decision making under risk. It has been generally accepted as a normative model of rational choice [24]. It is assumed that all reasonable people would wish to obey the axioms of the theory [25], and that most people actually do, most of the time [26].

In order to incorporate risk into our decision support system, we provide the COA recommendations to the human operator based on his/her risk propensity. Risk propensity of an operator can be broadly classified into three categories: a) risk averse, where targets are devalued if they cannot be completely searched; b) risk neutral, where targets are valued based on the percentage of the corresponding uncertainty region that can be searched; and c) risk seeking, wherein targets with high rewards (and low probability of complete search) are prioritized over targets with low rewards (and high probability

of complete search).

When UAS u_i arrives at target t_j at time τ_j, the expected utility (E) based on operator's risk propensity is defined as

$$E_{ij}(\tau_j) = R_j \times \rho_{ij}\left(x, \frac{s_{ij}(\tau_j)}{fs_{ij}}\right) \tag{2.3}$$

where the weighting function $\rho(\bullet)$ is defined as:

$$\rho_{ij}\left(x, \frac{s_{ij}(\tau_j)}{fs_{ij}}\right) = \min\left(\frac{s_{ij}(\tau_j)}{fs_{ij}x}, 1\right) \tag{2.4}$$

Here, x is the risk factor defining the risk propensity of an operator (this is essentially the utility theory-based approach). The plot of this weighting function is shown in Fig. 2.4. The red and green colored regions of the plot depict the risk-seeking and the risk-averse behavior, respectively. The yellow line corresponding to $x = 1$ shows the risk-neutral behavior. As can be seen, for a risk-seeker, the value of the weighing function $\rho(\bullet)$ rises rapidly with increasing $s_{ij}(\tau_j)/fs_{ij}$, whereas this rise is much more gradual for a risk-averse decision maker, implying that a risk-averse operator spends more time than a risk-seeker to obtain the same reward.

In order to illustrate this concept, consider the scenario in Table 2.2. Here, the target Bravo has a maximum achievable reward of 750 and only 88% of its area can be searched within the stipulated deadline. Based on our risk-propensity algorithm, a risk-seeking operator (x = 0.6) would expect the maximum achievable utility of 750, while the risk-neutral operator assigns it a utility of 660 ($= 0.88 \times 750$). On the other hand, a risk-averse operator (x = 1.4) devalues this target to an expected utility of 471

Table 2.2: TARGET EXPECTED UTILITY BASED ON USER RISK PROPENSITY

Target	Reward	Risk Seeking (x<1) (x=0.6)		Risk Neutral(x=1) (x=1)		Risk Averse(x<1) (x=1.4)	
		Area Searched	Expected Utility	Area Searched	Expected Utility	Area Searched	Expected Utility
Kilo	1500	62%	1500	100%	1500	100%	1500
Charlie	1000	100%	1000	89%	890	100%	1000
Bravo	750	88%	750	88%	660	88%	471
Alpha	500	69%	500	29%	148	–	–

$(= 0.88 \times 750/1.4)$.

2.2.4 Optimal UAS Scheduling Problem Formulation

Given $\mathbb{T} = \{1,\ldots,n\}$, the set of target indices, $\mathbb{U} = \{1,\ldots, m\}$, the set of UAS indices, and $\psi = \{\psi_i | i = 1,\ldots,m\}$, a set of initial locations of the UAS, the prioritized bi-objective function is given by

$$\max E = \max \sum_{i\in\mathbb{U}} \sum_{j\in\psi_i\cup\mathbb{T}} \sum_{k\in\mathbb{T}} E_{ik}(\tau_k)\, \xi_{ijk} \tag{2.5}$$

$$\min MT = \min \max_{i\in\mathbb{U},\, k\in\mathbb{T}} (\tau_k + s_{ik}) \tag{2.6}$$

subject to

$$\sum_{i\in\mathbb{U}} \sum_{j\in\psi_i\cup\mathbb{T}} \xi_{ijk} \leq 1 \tag{2.7}$$

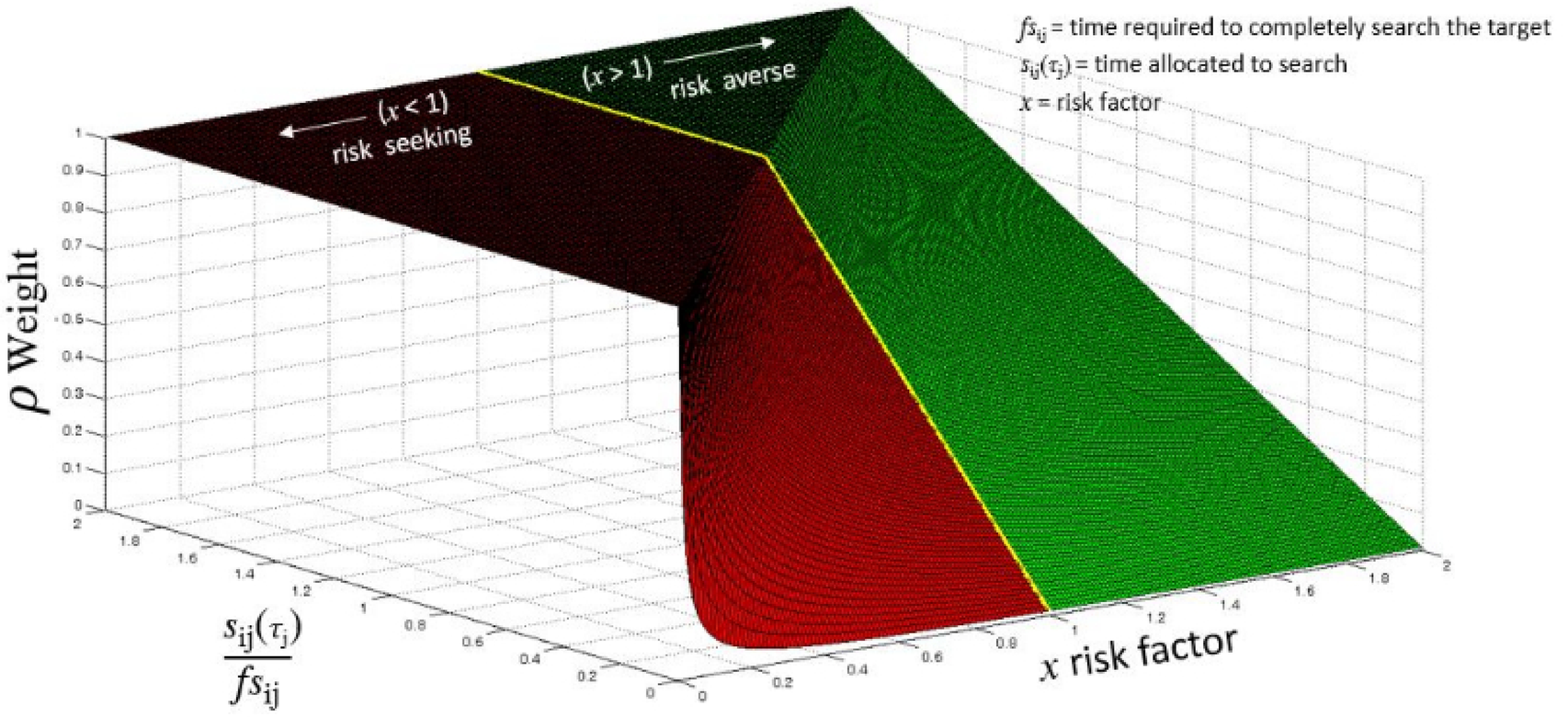

Fig. 2.4: Weight as a function of risk factor (x) and ratio of search times $\left(\frac{s_{ij}(\tau_j)}{fs_{ij}}\right)$.

$$\sum_{k \in \mathbb{T}} \xi_{i\psi_i k} \leq 1 \tag{2.8}$$

$$\sum_{i \in \mathbb{U}} \sum_{k \in \mathbb{T}} \xi_{i\psi_i k} \leq m \tag{2.9}$$

$$\tau_k = \sum_{i \in \mathbb{U}} \sum_{j \in \psi_i \cup \mathbb{T}} (\tau_j + s_{ij}(\tau_j) + t_{ijk}) \times \xi_{ijk} \tag{2.10}$$

$$0 \leq \tau_k \leq D_k \tag{2.11}$$

$$\tau_k - \tau_j > 0 \;\; \forall \xi_{ijk} = 1 \tag{2.12}$$

$$\tau_{\psi_i} = 0, \ \ \forall i \tag{2.13}$$

where

$$\xi_{ijk} = \begin{cases} 1, \text{ if UAS } u_i \text{ is assigned to target } t_k \\ \quad \text{immediately after target } t_j \text{ or UAS'} \\ \quad \text{initial position} \\ 0, \text{ otherwise} \end{cases}$$

The objective is to find a solution with the maximum cumulative utility from the set of feasible solutions. If there are multiple solutions with the same maximum utility, the objective is to select a solution with the minimum mission time (makespan) in (2.6). Thus, the objective function in (2.5) has higher priority over the objective function in (2.6). Constraint (2.7) states that, not every target is assigned to a UAS. Constraints (2.8)–(2.9) ensure that each UAS is assigned only once and the number of UASs assigned must not exceed the available number of UASs. The definition of the arrival time variable and the deadline constraints are given in (2.10)–(2.13).

2.3 Solution Approach

2.3.1 Optimal Solution

Each UAS u_i can visit a subset of the targets in T represented by a path sequence $P_{(i,\alpha)}$ and the concomitant target set $\Theta_{(i,\alpha)}$, given by

$$P_{(i,\alpha)} = \{T_{i_1}, \ldots, T_{i_{n_\alpha}})\}_i; \Theta_{(i,\alpha)} = \{i_1, \ldots, i_{n_\alpha}\} \tag{2.14}$$

where α is the sequence index and $(n_\alpha)_i$ is the number of targets UAS u_i explores in the sequence $P_{(i,\alpha)}$; $\Theta_{(i,\alpha)}$ is the set of targets explored in the sequence $P_{(i,\alpha)}$. A feasible assignment is determined by finding the target sets for the *m* UASs such that each target is searched by only one UAS, i.e., $\Theta_{(i,j)} \cap \Theta_{(l,n)} = \phi \; \forall \; i \neq l$, $i, l \in \{1,2,3,\ldots,m\}$. Let $P(\alpha, \beta, \ldots, \gamma)$ correspond to a feasible set of routes for the m UASs represented by an ordered list

$$P(\alpha, \beta, \ldots, \gamma) = \{P_{(1,\alpha)}, P_{(2,\beta)}, \ldots, P_{(m,\gamma)}\} \tag{2.15}$$

A depth-first tree search algorithm is employed to generate all the feasible solutions. Let, $\kappa = [\alpha, \beta, \ldots, \gamma]$ be the set of sequence indices. Then, $\Theta_{(i,\kappa[i])}$ is a feasible single route, which includes targets that are not explored by the previous UASs. The set $\pi(i)$ is generated by forming all possible sets of targets, so that the UAS u_i reaches the last target in its route before the target's deadline. The state represented by $P(\alpha, \beta, \ldots, \gamma)$ is shown in Fig. 2.5, where

$$F_m = T - \sum_{i=1}^{m} \Theta_{(i,\kappa[i])} \tag{2.16}$$

represents the set of unassigned targets after stage m. Once we reach the bottom of the tree or the end of m stages, we obtain the ordered list $P(\alpha, \beta, \ldots, \gamma)$, which specifies the optimal routes for all m UASs.

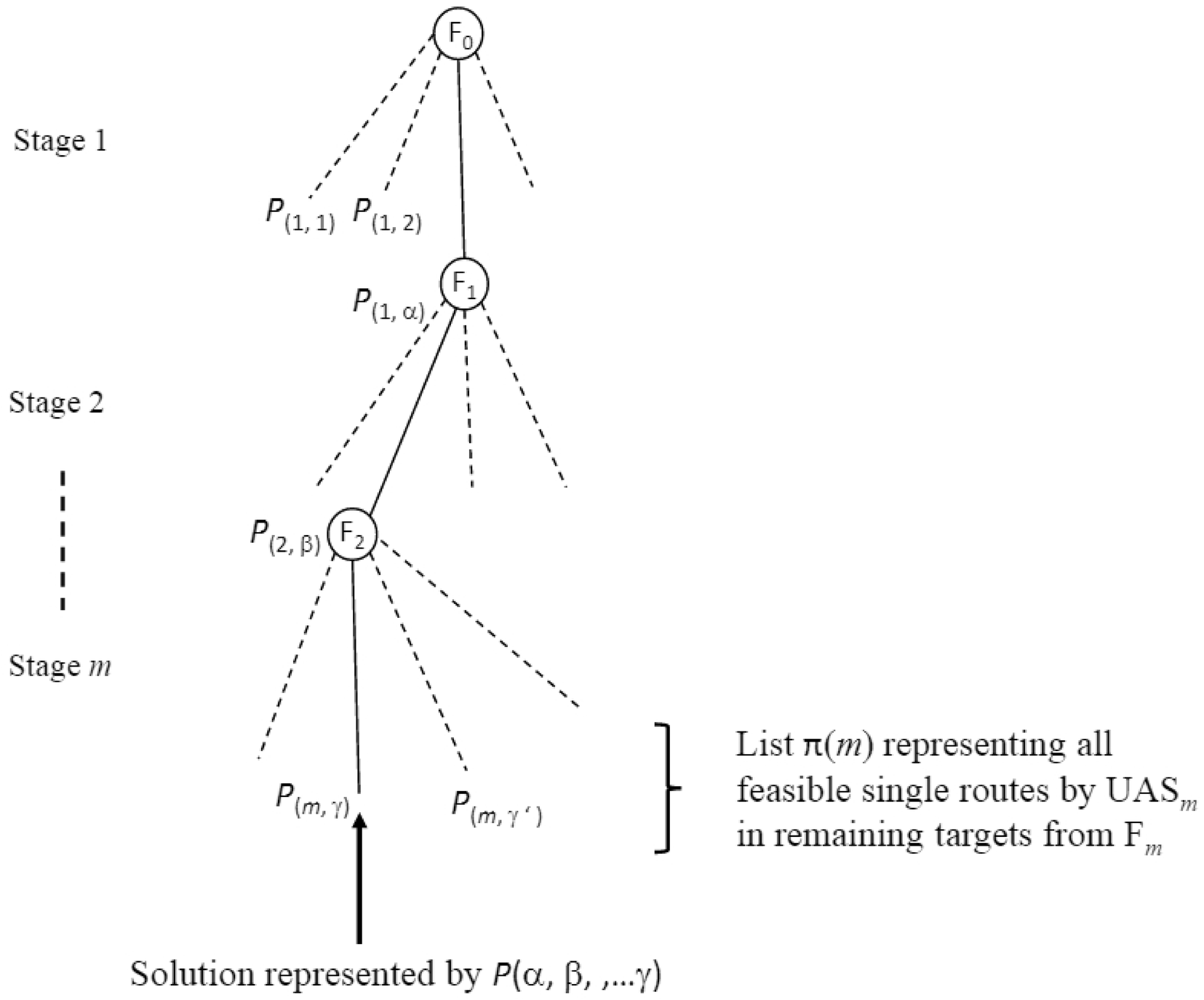

Fig. 2.5: Tree structure for branch on route algorithm.

Let the expected utility corresponding to the ordered list $P(\alpha, \beta, \ldots, \gamma)$ be denoted $E(\alpha, \beta, \ldots, \gamma)$. The target sequence which gives the optimal utility is

$$(\alpha^*, \beta^*, \ldots, \gamma^*) = \arg \max_{\alpha,\beta,\ldots,\gamma} E(\alpha, \beta, \ldots, \gamma) \tag{2.17}$$

and the ordered list $P(\alpha^*, \beta^*, \ldots, \gamma^*)$ is the optimal solution with $E(\alpha^*, \beta^*, \ldots, \gamma^*)$ as the concomitant optimal expected utility. The flow of operations to obtain the optimal route sequence utilizing the above tree search procedure is shown in Table 2.3.

2.3.2 Obtaining q-best Solutions

Sometimes the optimal solution is not the only solution of interest. The operator might be interested in top q solutions ($q > 1$). Murty [27] proposed an algorithm to efficiently compute a set of q-best solutions to an assignment problem, which has proven useful in [28] and [29]. We display these assignments to the operator, where q is specified by the operator. The q-best solutions also provide a means to rank order the performance of UAS operators. Fig. 2.6 shows a case where an operator requested the four best solutions for a scenario described in Table 2.4.

2.3.3 Heuristic Methods

Since OVRP is an *NP*-hard combinatorial optimization problem, the heuristics are generally used in practice [30]. Here, we compare the performance of the optimal UAS-target assignment algorithm with two heuristics: a) target equalization; and b) path time equalization.

Target Equalization

The target equalization method strives to equalize the number of targets assigned to each UAS in a greedy manner. This is done in stages. First, the UASs are prioritized

Table 2.3: PROCEDURE FOR BRANCH ON ROUTE ALGORITHM

1. Read in information pertaining to a team of UASs, $U = \{u_i | i = 1, \ldots, m\}$, to be scheduled to search for a set of geographically distributed targets, $T = \{t_j | j = 1, \ldots, n\}$.

2. For each UAS u_i, find the set of all feasible single routes, denoted by $\pi(i)$, using a depth-first search.

3. Find all the combinations of m feasible single routes, each denoted by $P(\alpha, \beta, \ldots, \gamma)$, and consolidate them into an ordered list.

4. Calculate the cumulative expected utility and the mission time for every $P(\alpha, \beta, \ldots, \gamma)$.

5. Save the top solution, as measured by the cumulative expected utility obtained and, if two solutions have the same cumulative expected utility, then choose the solution which has the shorter mission time.

Table 2.4: SCENARIO DESCRIPTION

Assets	Velocity (km/hr)	Sweep Width (km)	Targets	Reward	Uncertainty Radius (km)	Target Deadline (mm:ss)
UH - 28	203	0.75	Charlie	1000	4.5	14:45
UV - 72	380	1.50	Foxtrot	250	3.4	32:15
UH - 53	203	1.00	Alpha	1000	5.0	15:15
			Victor	1500	3.5	31:30
			Kilo	1500	11	26:45

Fig. 2.6: q-best allocations; 4-best solutions are shown, as per the operator's request (q = 4).

based on their speeds and then, each UAS is assigned to the best available target based on its expected utility. This process is continued until each UAS is assigned to a target in the current stage. This stage-wise assignment is repeated until all the targets are

assigned.

Path Time Equalization

The path time equalization method strives to equalize the mission time of each UAS route. The first target is assigned to each UAS in the same way as was done in the target equalization method. Then, a target is selected and assigned to a UAS so that the cumulative time of all routes is a minimum. This process is repeated until all the targets are assigned. This method differs from the target equalization method in that it chooses the next UAS based on the cumulative time taken by all UASs, rather than a prioritized order. This is an explicit attempt to balance route times.

2.3.4 Modes of Operation

SCOUT^{TM} is developed to operate in two modes: a) parallel mode, where the optimal COA recommendations are generated (independent of the operator's actions); b) coordinated mode, where the optimal COA recommendations are generated after taking the operators' actions into consideration. The latter is akin to a navigation system that adapts to a driver's route.

Consider a scenario of three UASs and five targets with specifications shown in Table 2.4. Suppose the operator assigns target Alpha to UH-28 and target Charlie to UV-72 before the mission execution phase. The suggested allocations for the parallel and the coordinated modes are shown in Fig. 2.7. In the parallel mode, the maximum

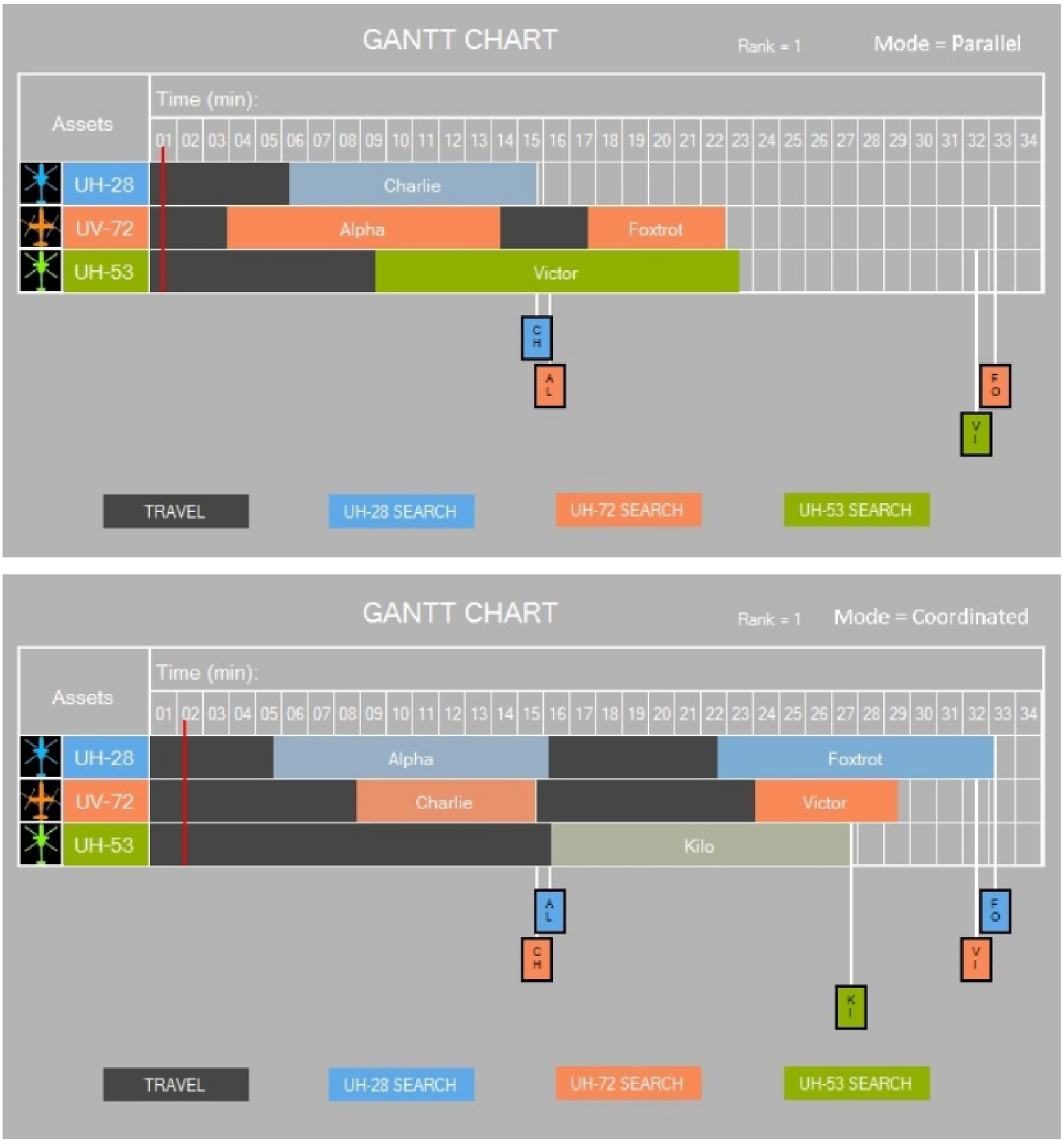

Fig. 2.7: Gantt chart depicting the change in COA when an assignment is made by the operator (Parallel versus Coordinated modes).

utility solution is computed, whereas, in the coordinated mode, it computes the constrained optimal assignment, given the operator's actions. In the coordinated mode, it can be observed that, given target Alpha is assigned to UH-28 and target Charlie is assigned to UV-72 by the operator, the algorithm allocates the remaining targets to UASs in a way which maximizes the remaining achievable utility.

2.3.5 Simulation Results

In order to assess the performance of optimal allocation over heuristic methods, we evaluated these algorithms on different scenarios, each having three UAS and geographically distributed targets. The number of targets is varied from 5 to 13 in increments of 2, where each UAS starts from an arbitrary position. Fig. 2.8 shows the expected utility versus the number of targets. The simulations are performed on a computer with an Intel i7 processor with 16 GB of RAM in C#.

It is evident that the optimal allocation can be substantially better than the heuristic ones, but when run times are considered, the optimal allocation takes multiples orders of magnitude more time as compared to solving by target or path time equalization methods. So, for small numbers of targets, as is the case in SCOUT^{TM}, the optimal allocation is preferred. For large scale scenarios involving hundreds of UASs and targets, heuristic methods are preferred.

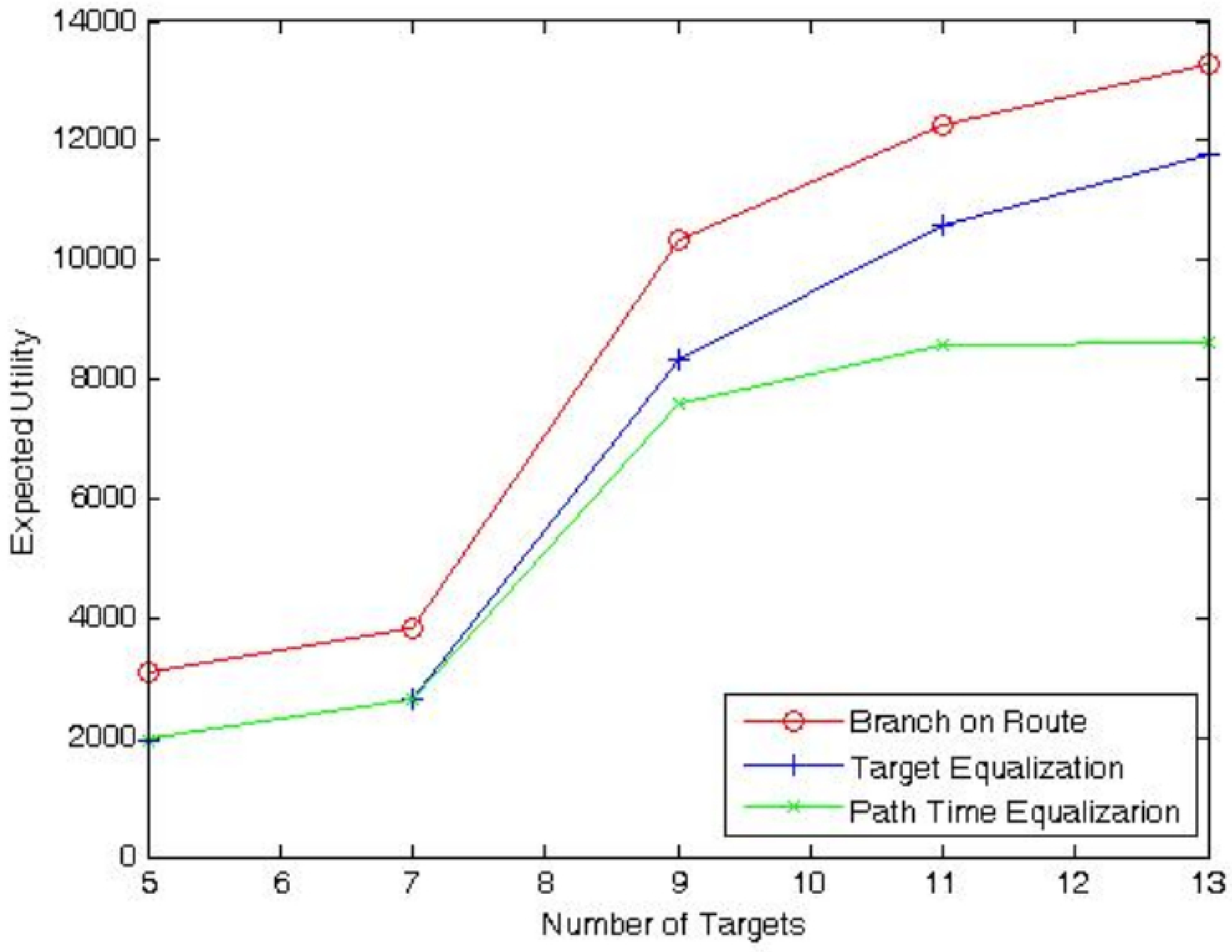

Fig. 2.8: Expected utility as a function of the number of targets.

Chapter 3

Enhancements made to SCOUT^{TM} software

3.1 Decision Support Enhancements to SCOUT^{TM} Software

The decision support algorithms embedded in SCOUT^{TM} provide the following capabilities to reduce the operator workload and to increase the available time for decision making.

3.1.1 Optimal Assignment

The optimal assignment at 6.5 minutes into the mission scenario is shown in Fig. 3.1, where the UASs are shown in the left column. Each row gives information on the assigned route (targets) for each UAS and the expected cumulative utility to be obtained from that route, UAS location, and the utility expected from the UAS-target assignments. The algorithm adapts to context changes, such as a target appearance or information update, that occur within the scenario and provides the updated optimal solution to the operator. All the past interaction history is maintained to allow the operator to analyze and compare how the change in context affected the optimal allocation

and adapt the plans accordingly.

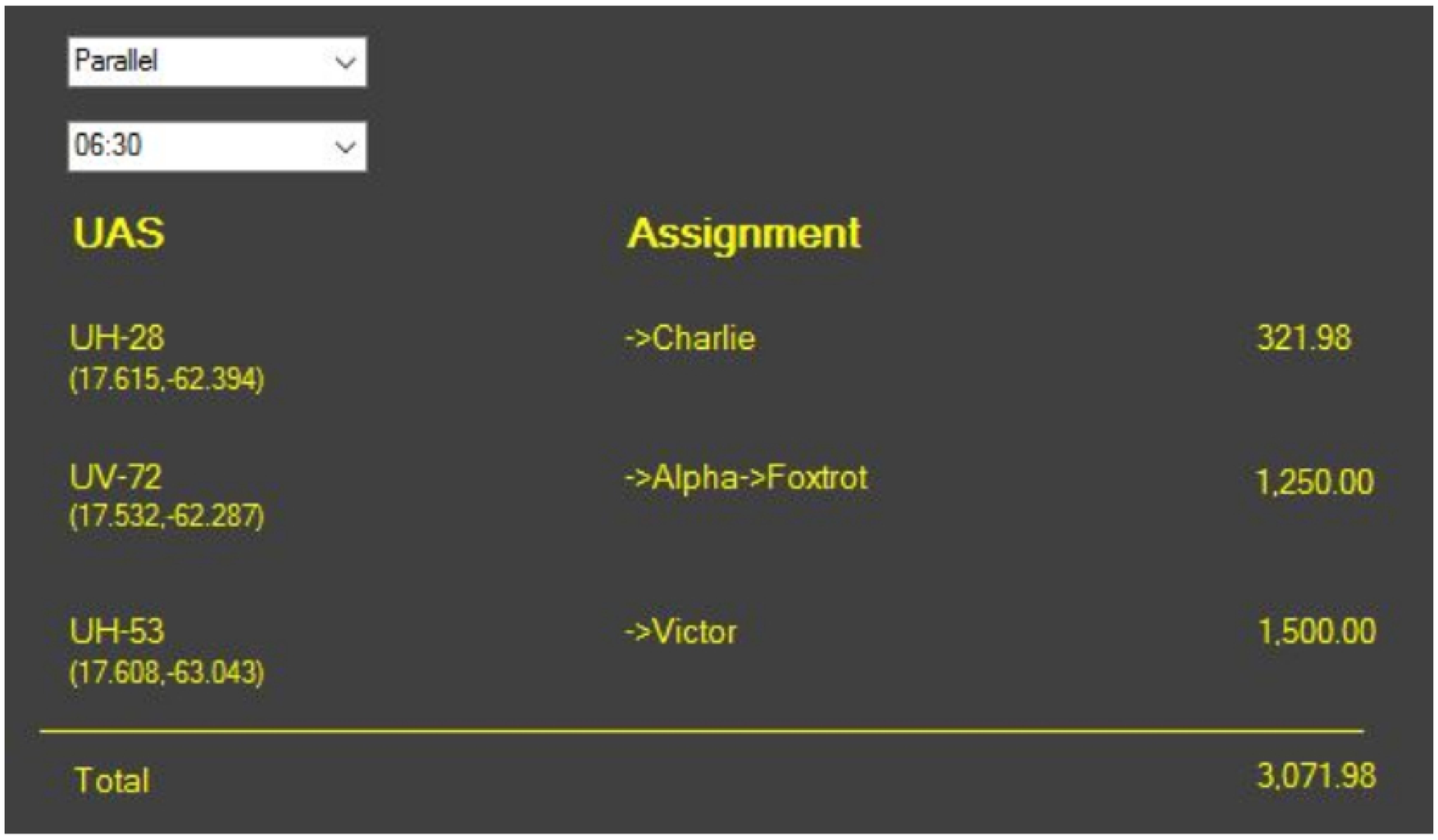

Fig. 3.1: Optimal assignment after re-planning at $t = 6$ minutes and 30 seconds during the mission execution phase.

3.1.2 Gantt Chart

A Gantt chart is a graphical illustration of a schedule to aid the operator in decision making. It is a pictorial representation of UAS-target allocation over time, where UASs are shown along the vertical axis and a time scale is shown along the horizontal axis, as in Fig. 2.7. The vertical markers represent the deadlines of the corresponding targets and the red bar is the time slider which indicates the current time. The Gantt chart is integrated into SCOUT^{TM} and enables the operator to experiment with alternative

plans.

3.1.3 Utility Chart

A utility chart provides the time evolution of cumulative utility in the parallel and the coordinated modes, as shown in Fig. 3.2. For a given mode, the thick line represents the actual utility obtained so far and the dashed line represents the future expected utility.

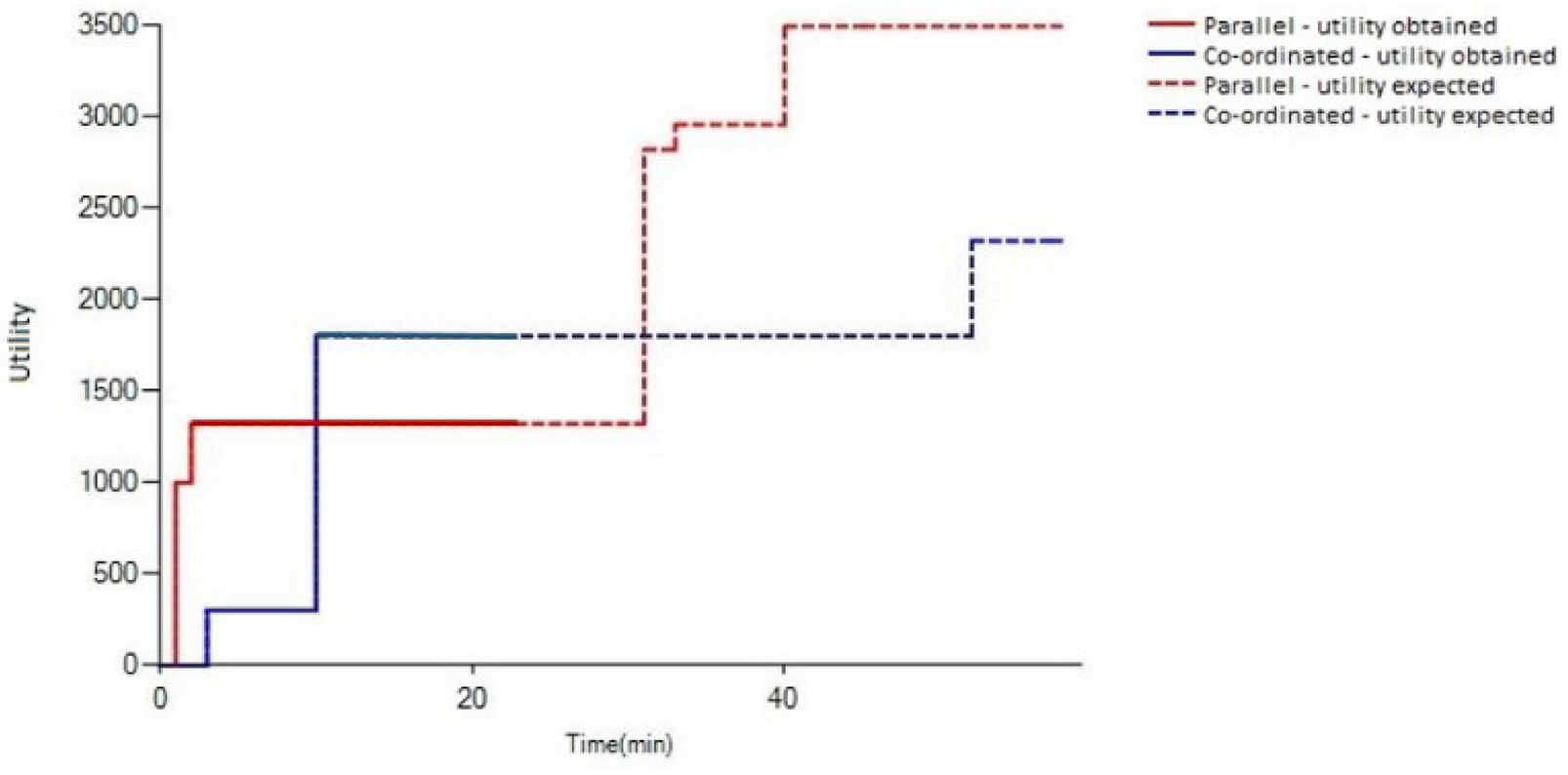

Fig. 3.2: A utility chart showing comparisons between the user obtained utility versus the optimal utility.

3.1.4 Event Chart

An event chart allows the operator to keep track of all the events that occurred in chronological order as shown in Fig. 3.3. An event, may be: a) a new target; b) an Intelligence update via chat update on a target's position or its uncertainty radius; c)

an Mission Command request via chat command to change the speed of a UAS; d) an operator adding a way point manually.

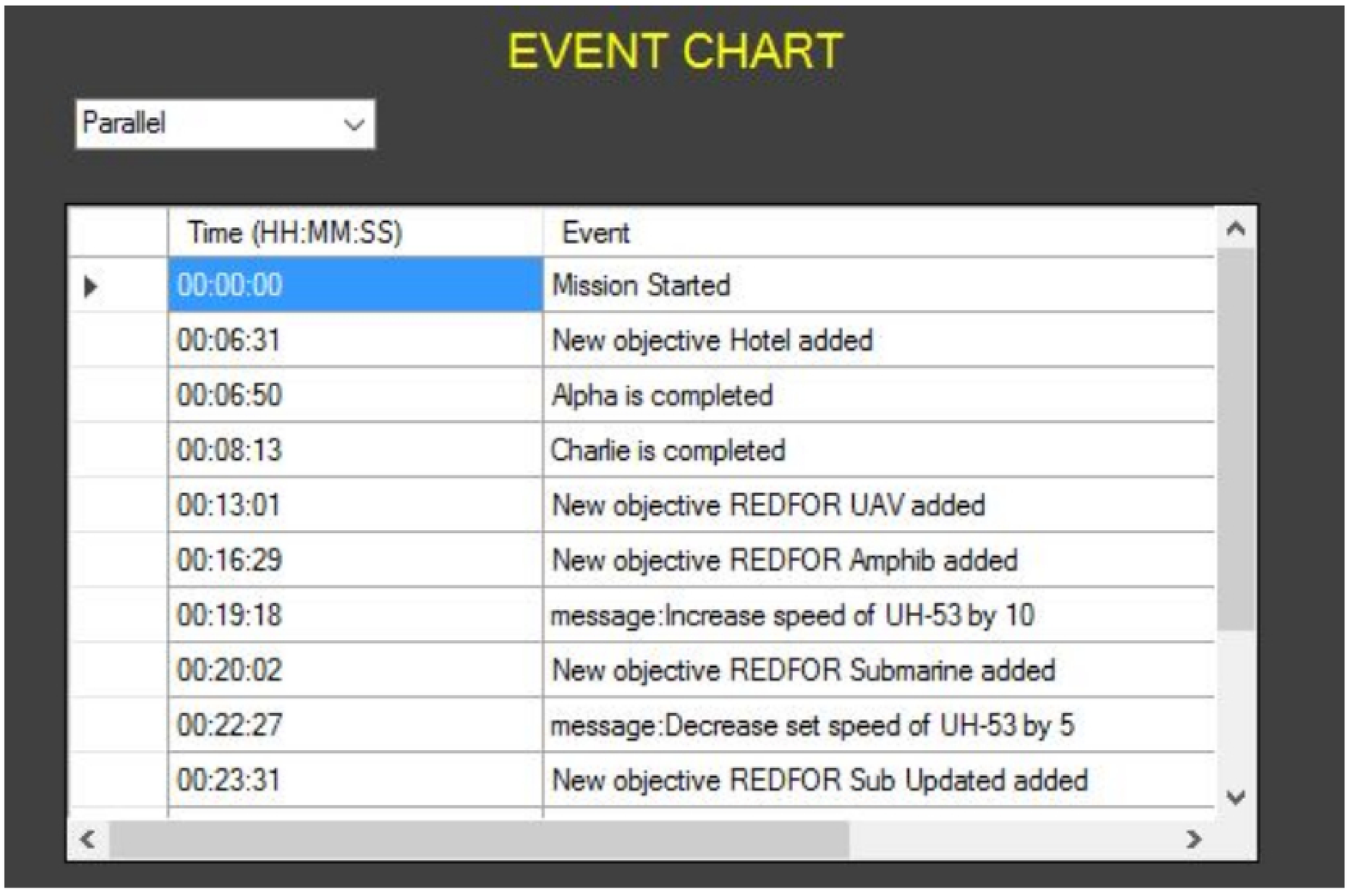

Fig. 3.3: An event chart showing all the mission events in chronological order.

Chapter 4

Conclusions and Future Work

4.1 Conclusion

The existing unmanned air systems (UAS) mission planning processes require an enormous amount of human cognitive processing, and is therefore error-prone, leading to mission failures. The enhanced version of the proactive supervisory control operations user testbed (SCOUT^{TM}) discussed in this thesis provides the user with rank-ordered courses of actions (COA) options, while adaptively updating plans based on user workload under changing mission contexts. It enables the investigation of higher order cognitive processes, such as decision making under uncertainty and the impact of low or high workload on risk taking behavior in a UAS environment. It also provides a timely comparison of the operator's solution in comparison to the optimal one. The q-best solutions provide a means to rank-order operator performance.

4.2 Future Work

Though the exact algorithm that we present in this thesis gives an optimal solution, it works only for relatively small instances and can find solutions for up to 15 targets and 3 assets. To scale the problem to a large number of assets and targets, we present heuristic methods, such as the rollout algorithm [31] (one step lookahead, two step lookahead) to generate near optimal solutions, even for a large number of targets and assets. The quality of these heuristics will be assessed based on their accuracy, speed, simplicity and flexibility.

In the future, we plan to enhance our proactive decision support algorithms to learn the behavior of the operator's risk propensity factor (x) (risk seeker/risk averse/risk neutral) on-line and provide the operator symbiotic recommendations. We want to develop a framework for automatically learning human user risk-model from the action sequence of the human collaborating with the robot (here, the parallel mode). For this, we will first use the operator's action sequences to cluster them into different human-types (risk averse, seekers or neutral) using an unsupervised learning algorithm. A separate reward function will then be learned for each of these operator behaviour clusters.

The newly learned model can then be incorporated into a mixed-observability Markov decision process [32] (MOMDP) formulation, where the operator type would be a partially observable variable. With this framework, we could infer the operator type of a new user that was not included in the training set, and can compute a policy

for the operator that will be aligned to the preferences of this particular operator.

Appendix A

Appendix A

A.1 Vehicle Routing Problem

Given a list of customers, distances between them and a set of vehicles, VRP finds tours that minimize the total length of the tours, such that one vehicle visits each location. It is first formulated in 1959 by G. Dantzig and J. Ramser [33] for the Truck Dispatching Problem. VRP is used to solve Real-life objectives, such as minimize distance, maximize profit, minimize time, maximize customer satisfaction, minimize employee workload etc. Typically, a normal VRP problem consists of a depot, m identical vehicles based at the depot, n customers with respective demands. Its objective is to determine a set of m or atmost m vehicle routes starting and ending at the depot, visiting each customer exactly once, satisfying the capacity constraint of minimal total cost. A typical VRP is shown in fig. A.1.

The current scenario discussed in this thesis differs from common VRP in the following ways: a) multiple depot's or assets have different starting locations; b) non-identical vehicles ie., each asset has different speed and sensor sweep width; c) each

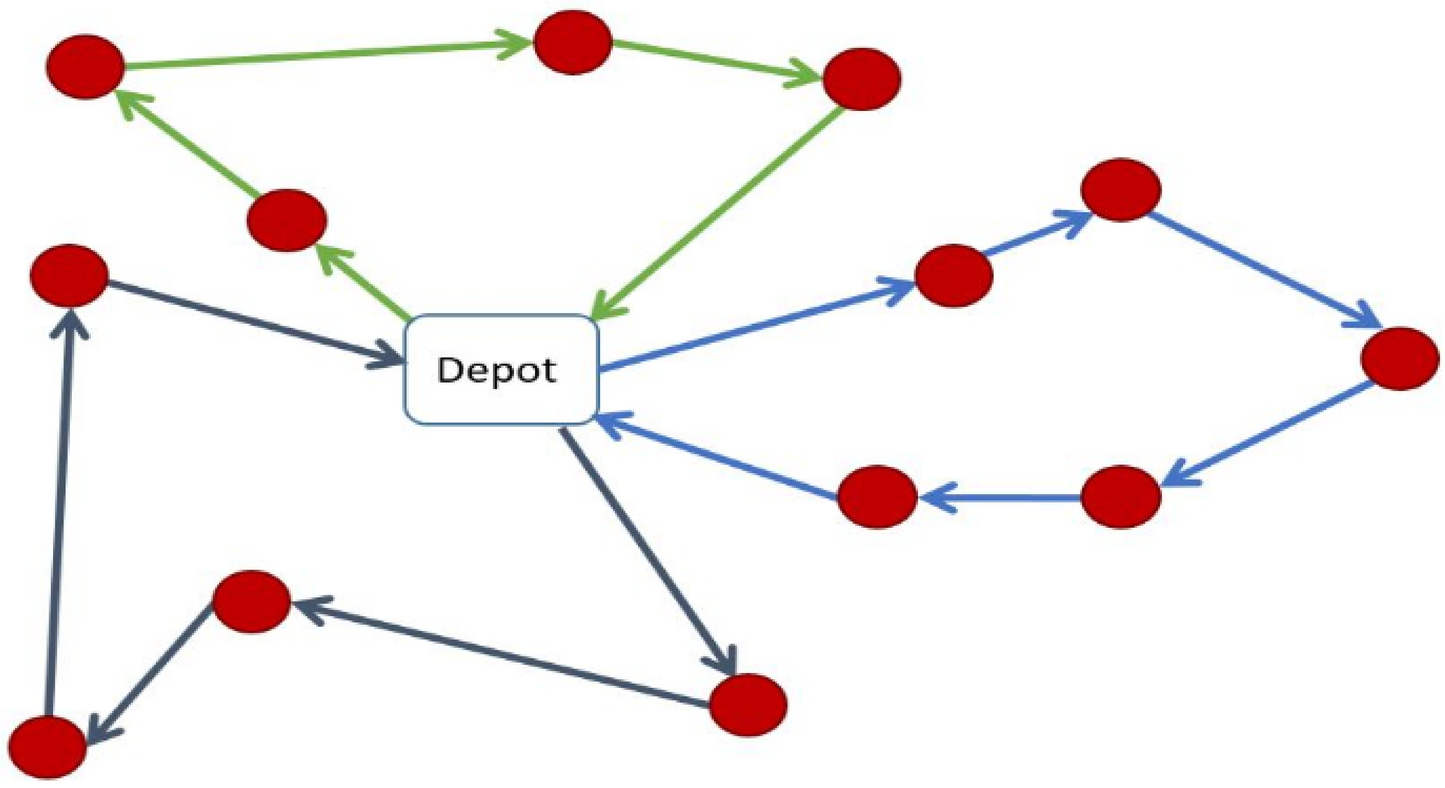

Fig. A.1: Typical vehicle routing problem.

target has specified radius of uncertainty; d) each target has specific deadlines before which it has to be searched; e) time deadlines lead to partial searches making it more uncertain for the asset to obtain reward; f) environment contains restricted operating zones into which assets cannot enter without prior permission. This can be nearly related to one of the variants of VRP called open vehicle routing problem (OVRP) and a sample scenario is shown in fig. A.2.

A.2 Depth First Search Algorithm

Depth first search (DFS) is an algorithm for traversing or searching a tree or graph data structures. Starting at the root node, it explores a path all the way to a leaf node before backtracking and exploring another path. The time and space analysis of DFS differs

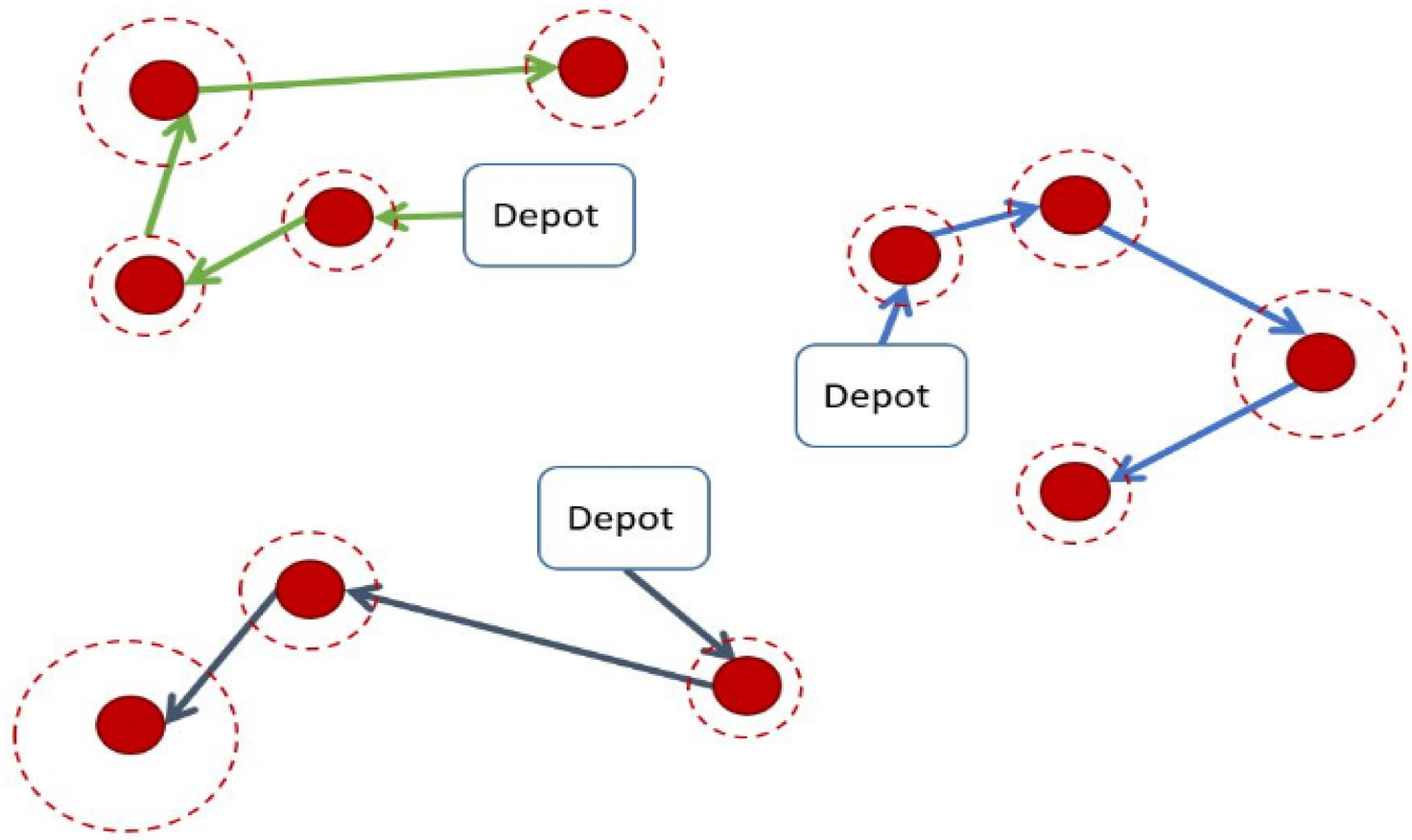

Fig. A.2: Problem scenario defined in thesis related to open vehicle routing problem.

according to its application area. In theoretical computer science, DFS is typically used to traverse an entire graph, and takes time $\Theta(|V| + |E|)$, linear in the size of the graph. In these applications, it also uses space $O(|V|)$ in the worst case to store the stack of vertices on the current path and the already visited vertices. A sample iteration in DFS is shown in Fig. A.3.

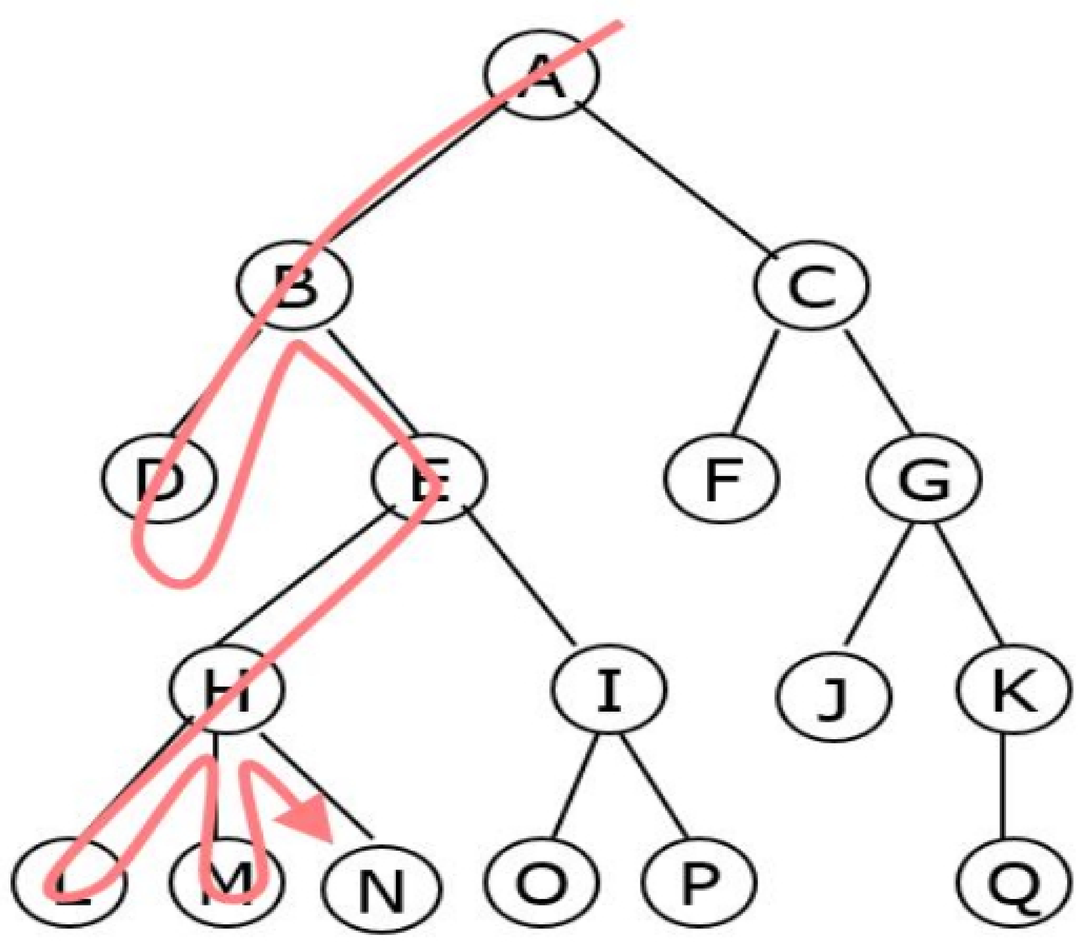

Fig. A.3: Sample path of exploring nodes of a tree by depth first search algorithm.

Bibliography

[1] A. Smirnov, "Context-driven decision making in network-centric operations: Agent-based intelligent support," DTIC Document, Tech. Rep., 2006.

[2] J. Gertler, "US unmanned aerial systems." DTIC Document, 2012.

[3] K. W. Williams, "A summary of unmanned aircraft accident/incident data: Human factors implications," DTIC Document, Tech. Rep., 2004.

[4] USAF. (accessed October 21, 2015) UAV crash blamed on distraction of battle. [Online]. Available: http://archive.airforcetimes.com/article/20100402/NEWS/4020326/UAV-crash-blamed-distraction-battle

[5] C. Sibley, J. Coyne, and J. Morrison, "Research considerations for managing future unmanned systems," in *2015 AAAI Spring Symposium Series*, 2015.

[6] M. Bellmore and S. Hong, "Transformation of multisalesman problem to the standard traveling salesman problem," *Journal of the ACM (JACM)*, vol. 21, no. 3, pp. 500–504, 1974.

[7] N. Christofides, "The vehicle routing problem," *Revue française d'automatique, d'informatique et de recherche opérationnelle. Recherche opérationnelle*, vol. 10, no. 1, pp. 55–70, 1976.

[8] N. Christofides, A. Mingozzi, and P. Toth, "Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations," *Mathematical programming*, vol. 20, no. 1, pp. 255–282, 1981.

[9] A. W. Ter Mors, C. Witteveen, J. Zutt, and F. A. Kuipers, "Context-aware route planning," in *Multiagent System Technologies*. Springer, 2010, pp. 138–149.

[10] N. Kohl and O. B. Madsen, "An optimization algorithm for the vehicle routing problem with time windows based on lagrangian relaxation," *Operations Research*, vol. 45, no. 3, pp. 395–406, 1997.

[11] D. Sariklis and S. Powell, "A heuristic method for the open vehicle routing problem," *Journal of the Operational Research Society*, pp. 564–573, 2000.

[12] F. Li, B. Golden, and E. Wasil, "The open vehicle routing problem: Algorithms, large-scale test problems, and computational results," *Computers & Operations Research*, vol. 34, no. 10, pp. 2918–2930, 2007.

[13] É. Taillard, "Parallel iterative search methods for vehicle routing problems," *Networks*, vol. 23, no. 8, pp. 661–673, 1993.

[14] I. H. Osman, "Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem," *Annals of operations research*, vol. 41, no. 4, pp. 421–451, 1993.

[15] P. Toth and D. Vigo, "The granular tabu search and its application to the vehicle-routing problem," *Informs Journal on computing*, vol. 15, no. 4, pp. 333–346, 2003.

[16] C. D. Tarantilis and C. T. Kiranoudis, "Boneroute: an adaptive memory-based method for effective fleet management," *Annals of operations Research*, vol. 115, no. 1-4, pp. 227–241, 2002.

[17] F. Li, B. Golden, and E. Wasil, "Very large-scale vehicle routing: new test problems, algorithms, and results," *Computers & Operations Research*, vol. 32, no. 5, pp. 1165–1179, 2005.

[18] D. Mester and O. Bräysy, "Active guided evolution strategies for large-scale vehicle routing problems with time windows," *Computers & Operations Research*, vol. 32, no. 6, pp. 1593–1614, 2005.

[19] O. Bräysy, W. Dullaert, and M. Gendreau, "Evolutionary algorithms for the vehicle routing problem with time windows," *Journal of Heuristics*, vol. 10, no. 6, pp. 587–611, 2004.

[20] D. Pisinger and S. Ropke, "A general heuristic for vehicle routing problems," *Computers & operations research*, vol. 34, no. 8, pp. 2403–2435, 2007.

[21] Y. Nagata and O. Bräysy, "Edge assembly-based memetic algorithm for the capacitated vehicle routing problem," *Networks*, vol. 54, no. 4, pp. 205–215, 2009.

[22] C. Prins, "A grasp× evolutionary local search hybrid for the vehicle routing problem," in *Bio-inspired algorithms for the vehicle routing problem*. Springer, 2009, pp. 35–53.

[23] M. Sniedovich, "Dijkstra's algorithm revisited: the dynamic programming connexion," *Control and cybernetics*, vol. 35, no. 3, p. 599, 2006.

[24] R. L. Keeney and H. Raiffa, *Decisions with multiple objectives: preferences and value trade-offs*. Cambridge university press, 1993.

[25] J. Von Neumann and O. Morgenstern, "Games and economic behavior," *Princeton, NJ*, 1944.

[26] D. Kahneman and A. Tversky, "Prospect theory: An analysis of decision under risk," *Econometrica: Journal of the Econometric Society*, pp. 263–291, 1979.

[27] K. Murthy, "An algorithm for ranking all the assignments in order of increasing costs," *Operations Research*, vol. 16, no. 3, pp. 682–687, 1968.

[28] X. Han, H. Bui, S. Mandal, K. R. Pattipati, and D. L. Kleinman, "Optimization-based decision support software for a team-in-the-loop experiment: Asset package selection and planning," *Systems, Man, and Cybernetics: Systems, IEEE Transactions on*, vol. 43, no. 2, pp. 237–251, 2013.

[29] R. L. Popp, K. R. Pattipati, and Y. Bar-Shalom, "m-best SD assignment algorithm with application to multitarget tracking," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 37, no. 1, pp. 22–39, 2001.

[30] T. Maddula, A. A. Minai, and M. M. Polycarpou, "Multi-target assignment and path planning for groups of UAVs," in *Recent Developments in Cooperative Control and Optimization*. Springer, 2004, pp. 261–272.

[31] D. P. Bertsekas, *Dynamic programming and optimal control*. Athena Scientific Belmont, MA, 1995, vol. 1, no. 2.

[32] V. Jääskinen, V. Parkkinen, L. Cheng, and J. Corander, "Bayesian clustering of dna sequences using markov chains and a stochastic partition model," *Statistical applications in genetics and molecular biology*, vol. 13, no. 1, pp. 105–121, 2014.

[33] G. B. Dantzig and J. H. Ramser, "The truck dispatching problem," *Management science*, vol. 6, no. 1, pp. 80–91, 1959.