LNCIS

LECTURE NOTES IN CONTROL AND INFORMATION SCIENCES

381

Michael J. Hirsch
Clayton W. Commander
Panos M. Pardalos
Robert Murphey (Eds.)

Optimization and Cooperative Control Strategies

Proceedings of the 8th International Conference on Cooperative Control and Optimization

Springer

Lecture Notes
in Control and Information Sciences 381

Editors: M. Thoma, M. Morari

Michael J. Hirsch, Clayton W. Commander,
Panos M. Pardalos, Robert Murphey (Eds.)

Optimization and Cooperative Control Strategies

Proceedings of the 8th International Conference on Cooperative Control and Optimization

Springer

Editors

Dr. Michael J. Hirsch
Raytheon, Inc.
P.O. Box 12248
St. Petersburg, FL 33733
USA
E-Mail: Michael_J_Hirsch@Raytheon.com

Dr. Panos M. Pardalos
Department of Industrial and Systems
Engineering
University of Florida
303 Weil Hall
P.O. Box 116596
Gainesville, FL 32611
USA
E-Mail: Pardalos@ufl.edu

Dr. Clayton W. Commander
GNC Branch
AFRL, Munitions Directorate
101 W. Eglin Blvd., Ste. 331
Eglin AFB, FL 32542
USA
E-Mail: Clayton.commander@gmail.com

Dr. Robert Murphey
GNC Branch
AFRL, Munitions Directorate
101 W. Eglin Blvd., Ste. 331
Eglin AFB, FL 32542
USA
E-Mail: Murphey@eglin.af.mil

ISBN 978-3-540-88062-2 e-ISBN 978-3-540-88063-9

DOI 10.1007/978-3-540-88063-9

Lecture Notes in Control and Information Sciences ISSN 0170-8643

Library of Congress Control Number: 2008935490

Typeset & Cover Design: Scientific Publishing Services Pvt. Ltd., Chennai, India.

Printed in acid-free paper

5 4 3 2 1 0

springer.com

Preface

The optimal control of cooperating and collaborating systems has continued to see increased research across the military domain. Much of autonomous unmanned vehicle research is presently geared toward cooperative control problems. Other domains (e.g., medical, homeland security, manufacturing, etc.) are also devising strategies for solving cooperative control problems. There have been some novel solutions proposed over the years for solving problems in cooperative control. However, in light of decentralized systems, and the sharing of meaningful information to the participants in the system, cooperative control continues to be one of the most difficult in the applied sciences. Only the continued dedicated research in this area will allow for novel solutions applicable to the current and future problems of cooperative control. This present volume, as well as volumes from previous years, clearly illustrate innovative solutions from some of the best and brightest cooperative control and optimization researchers.

This volume represents the most recent in a series of publications dealing with recent research and challenges in the field of optimal cooperative control. Most of the chapters in this book were presented at the Eighth International Conference on Cooperative Control and Optimization, which took place in Gainesville, Florida, January 30 – February 1, 2008. It is our belief that this book will be an invaluable resource to faculty, researchers, and students in the fields of optimization, control theory, electrical engineering, computer science, applied mathematics, and robotics.

We gratefully acknowledge the financial support of the Air Force Research Laboratory, The Center for Applied Optimization at The University of Florida, and Raytheon, Inc. We thank the contributing authors, the anonymous referees, the conference participants, and Springer Publishing for making the conference so successful and the publication of this book possible.

July 2008

Michael J. Hirsch
Clayton W. Commander
Panos M. Pardalos
Robert Murphey

Table of Contents

Effective Algorithms for a Class of Discrete Valued Optimal Control Problems

Louis Caccetta, Ian van Loosen, and Volker Rehbock

Western Australian Centre of Excellence in Industrial Optimisation (WACEIO)
Department of Mathematics and Statistics
Curtin University of Technology
Kent Street, Bentley WA 6102, Australia
{L.Caccetta@exchange,ian.loosen@postgrad,rehbock@maths}.curtin.edu.au

Abstract. We consider a general class of optimal control problems where the control functions are assumed piecewise constant and only take on values from a finite discrete set. The aim in such a problem is to find a sequence of discrete control values and a corresponding set of exact switching times (i.e., times where control should switch between the discrete values) such that a given functional representing cost or risk is minimized. Such problems arise in a range of applications. One application is the 'Transit Path Problem' where an object such as a robot or vehicle (air, naval, space or land) needs to traverse a specified region (discrete or continuous) between two points in a prescribed time so as to avoid detection. The objective is to find a path for the object which satisfies the time constraints and which minimizes the total risk of detection. The risk function is not simple and depends on a range of factors such as the environment, the types of sensors, the speed, direction and position of the vehicle. The main difficulty with these problems is that the range of some of the controls is discrete and hence not convex. Since the gradients with respect to the switching time parameters are discontinuous, ordinary gradients based solution methods perform poorly. An additional difficulty is to determine exactly how many switching times are involved in an optimal solution. We address the first difficulty by using the Control Parameterization Enhancing Transform (CPET) and the second difficulty by solving a sequence of problems which are transformed via CPET. With respect to the transit path problem our strategy involves a two stage approach. The first stage involves a discretization of the problem and the solution of a constrained path problem in a network. The second stage involves the use of an optimal control model and a solution procedure that utilizes the solution obtained from the first stage.

1 Introduction

We consider a class of discrete valued optimal control problems where a cost functional is to be minimized over a class of control functions, some or all of which take on values from a discrete set, and subject to a dynamical system together with appropriate constraints on the states and control variables. Examples of such problems include: the Transit Path Problem which arises when an object needs to traverse between two points through a specified region; optimal driving strategies for the control trains [25], where diesel locomotives have discrete throttle positions; and the design of operating procedures for process start-up, shut down and changeovers [35].

Another example of a discrete valued optimal control problem is the optimal battery recharge problem [53]. For example, a conventionally powered submarine needs to recharge its battery banks periodically. The aim of a battery recharge plan is to identify a sequence of recharge intervals in a given time period. Recharging the batteries requires the submarine to surface, doing so will mean an increase in the noise level of the submarine and consequently this results in an increased danger of being detected. A number of generators operating in standard or supercharged mode are available to recharge the batteries. Therefore, given a time period and a specified distance to be covered, we need to determine when recharging should occur, how many engines are to be used, and in what mode they should run. Also the running of different numbers of engines in different modes results in different levels of exposure. In addition the battery needs to maintain a charge level between an upper and lower bound and should also avoid the lower bound as much as possible. The primary objective is to minimize the total recharge time. On its own this would result in the solution that requires all engines to run in supercharge mode for as short a time as possible. However, the objective must consider different levels of exposure with different operating modes and the need to maintain a reasonable charge level.

To solve discrete valued optimal control problems, it is necessary to search over many possible sequences of control values to find the optimal sequence. In addition, the switching times between changing control actions need to be optimized. These problems are hard to solve numerically since the control functions come from a discrete set and hence are non-convex, and as the search for the optimal sequence has to be carried out over a discrete set, the problem has a combinatorial nature to it. Also, numerical difficulties occur in the integration of the dynamics of the problem due to possible discontinuities at the switching points of the discrete control actions.

We focus our attention on the Transit Path Problem. The objective is to determine an optimal path, in terms of minimizing risk or cost or maximizing reliability, for an object, such as a robot or vehicle, which needs to traverse a specified region, discrete or continuous, between two points. In many instances these transit paths are also required to satisfy more constraints. This problem arises in many areas of real life. For example, the routing of both manned and unmanned military vehicles through a detection field. Examples here include unmanned aerial vehicles (UAVs), strike aircraft and cruise missiles where constraints on the path could include such factors as flight time, fuel consumption or total risk. A similar problem arises in motion planning for robot manipulators through a region that may contain a number of obstacles. If there is any uncertainty associated with the location of the obstacles, the aim would be to plan a path which minimizes the probability of a collision, whilst still reaching the terminal point within a specified time. Another application is the routing of a new highway through an undulating terrain. Costs arise from the cut or fill required for the highway to pass through a certain point. The intention would be to minimise the cost of constructing the road while still adhering to a limit on the total length of the highway. Additional constraints on the curvature and slope of the highway may also be imposed. Other transit path problems occur within the generation of optimal trajectories for air, space, naval and land vehicles [46]. The problem we specifically look at is that of determining an optimal transit path for a submarine moving through a field of sonar sensors, subject to a total time constraint. For a thorough overview of methods relating to the design of optimal routes under the risk from a threat environment in the literature, refer to [41].

The designing of optimal routes under the risk from a threat environment has been studied by a number of authors [8,9,10,13,17,19,22,23,27,32,33,34,36,37, 40,43,44,45,46,47,48,49,50,59,61,62], however only a limited number of these include an additional constraint on the route. The two approaches developed and employed in the literature use either a continuous or a discrete model. The continuous approach is typically based on the technique of calculus of variations whereas the discrete optimization approach, through a network, approximates the original problem by a constrained shortest path problem.

The strategy to be presented in this chapter involves a two stage approach for identifying optimal and near-optimal routes. The first stage involves a discretization of the problem, which results in a Constrained Shortest Path Problem (CSPP), and the development of a network heuristic method, based upon parameterisation. The second stage involves the development of an optimal control model, and the application of the Control Parameterization Enhancing Transform (CPET) technique, and a solution procedure that utilizes the solution obtained in the first stage as a starting point to determine a continuous solution of the problem. In this phase of our procedure we make use of the optimal control software package MISER3 [29].

In the proposed model, each of the sensors can detect the presence of the submarine with a probability which is a given function depending on the distance and speed. This function is not a simple analytical expression, but depends upon a range of factors, including the characteristics of the ocean floor and ocean surface, depths of the sensor and the submarine, and the temperature and salinity of the water [18]. Here we use probability of detection functions reported in Hallam [18]. These were constructed under the assumptions that the geographic location and environmental conditions are known and that the submarine remains at a constant depth. Furthermore, each of the given functions is constructed for a particular constant vessel speed. While there are still further factors influencing the probability of detection (such as machinery states, frequency of the sensor, alertness of sensor operators or quality of the automatic detection, the relative aspect of the submarine and the sensor, the effect of sudden changes in travel direction or speed), the functions from Hallam [18] display sufficient detail to test the feasibility of the proposed method.

The overall probability of detection at any point in time can then be calculated as an appropriate combination of these individual probabilities of detection. Here, we make the assumption that the probability of detection for any one sensor is independent of the probabilities of detection for the other sensors. The objective then is to find a transit path between two fixed positions in the sensor field which will minimize the overall probability of detection while still satisfying a maximum travel time constraint. The difficulty is due to the fact that the transit time must satisfy an upper bound constraint.

The chapter is organised as follows. In Section 2 we describe how to formulate the transit path problem by an integer programming formulation through discretization. The optimal control formulation is described in Section 3 followed by the CPET technique in Section 4. In Section 5 we demonstrate how the optimal control model is highly sensitive to the starting solution. Section 6 presents the computational strategy as well as our heuristic used to solve the problem. This heuristic is tested against the integer programming software package CPLEX in Section 7. In Section 8 we give numerical results for our hybrid method. Finally we summarize our conclusions in Section 9.

2 Discrete Formulation

The first phase of our method is to model and solve the submarine transit path problem as a CSPP. We construct a grid-like network over the region that needs to be traversed, as shown in Fig. 1.

From Fig. 1 the START and FINISH points are assumed to be knot points of the grid. Ignoring the possibility of different speeds for now, we may regard the knot points of the grid as nodes and the grid lines as edges, and thus think of the grid as a graph. Clearly, an equally spaced rectangular grid as shown in Fig. 1 can be generalized to be unequally spaced or even nonrectangular (for example, one may want to place the nodes in low danger areas only) while still maintaining the graph structure. Furthermore, other edges (for example, those joining vertices in a diagonal manner in the diagram) can also be added. We then allow movement along edges only and with each edge we can associate a cost value. The cost depends on the location of the edge in the sensor field and on the speed at which the vessel travels along the edge. It is calculated in the same manner as for the optimal control model presented in the following section. Furthermore, since the distance along each edge and the travel speed are known, we can calculate the time it takes the vessel to traverse each edge simply by dividing the distance by the speed. Both the cost and the travel time for each edge in a path can then be simply added to obtain the total cost and total travel time associated with that path.

Let us assume that the speed s of the submarine comes from a finite discrete set $\{s_1, s_2, s_3, \ldots\}$. Physically, a discrete valued vessel speed is obviously not realistic. However, at the planning level, it is natural for human operators to

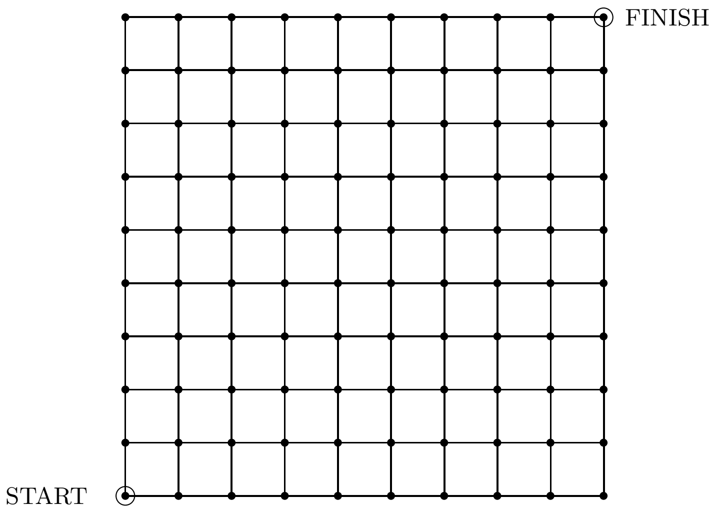


Fig. 1. Structure of the grid to be placed over the sensor field

think of the journey in terms of discrete stages with each stage corresponding to a constant speed section of the journey.

With respect to a simple Graph $G = (N, A)$, where $N = \{1, 2, \ldots, n\}$ is the set of nodes, $|N| = n$, and A is the set of edges, each edge (i, j) has a corresponding cost c_{ij} and a transit time t_{ij}. c_{ij} is the cost of traversing the edge in terms of the exposure to the sensors, while t_{ij} is simply the time required to traverse the edge. For convenience, we denote the start (origin) node by O and the destination node by D. Also let T be the time limit. A directed path (or simply path) is a finite sequence of nodes $P = (i_0, i_1, \ldots, i_m)$, such that $(i_{k-1}, i_k) \in A$ for $k = 1, 2, \ldots, m$. The path cost (or simply cost) of P is defined to be $c(P) = c_{i_0 i_1} + c_{i_1 i_2} + \ldots + c_{i\ \ _{-1} i\ } = \sum_{k=1}^{m} c_{i\ \ _{-1} i\ }$. Similarly the time of P is given by $t(P) = \sum_{k=1}^{m} t_{i\ \ _{-1} i\ }$. The path is time feasible if and only if $t(P) \leq T$. Therefore our problem is to find the minimum cost path $c(P)$ in G from O to D such that it also satisfies $t(P) \leq T$. This problem is referred to as the CSPP. The solution to this problem forms our initial solution for the optimal control phase of our two phase procedure.

The CSPP is closely related to both the Shortest Path Problem with Time Windows (SPPTW) [11] and the Resource Constrained Shortest Path Problem (RCSPP) [3]. The SPPTW consists of finding the optimal route in a network while respecting specified time windows at each node visited. RCSPPs have vectors of weights, or resources, associated with the arcs rather than just a scalar. The SPPTW and RCSPP generally appear as sub-problems to Vehicle Routing Problems (VRP), and are solved using column generation methods to construct optimal routes. Another closely related problem to the CSPP is the so called Multi-Objective Shortest Path Problem (MOSPP), which seeks all nondominated (Pareto) solutions. A special case of the MOSPP is the Bicriterion Shortest Path Problem, whereby there are only two objective functions that are to be minimized, for example both cost and time. It is possible to generate the whole set of Pareto optimal paths and from this one can select the optimal solution to the CSPP. However, this would not be efficient because the number of paths in this set can be very large.

The CSPP can be also be given an Integer Programming Formulation. Here we consider the grid based graph to be a directed network $G = (N, A)$, where A is the set of directed edges, known as arcs. Let T be the time requirement. Then the problem can be specified as

$$\text{Minimize} \sum_{(i,j)\in A} c_{ij} x_{ij} \tag{1}$$

subject to

$$\sum_j x_{ij} - \sum_j x_{ji} = \begin{cases} 1, & \text{if } i = O, \\ -1, & \text{if } i = D, \\ 0, & \text{otherwise}, \end{cases} \tag{2}$$

$$\sum_{i,j} t_{ij} x_{ij} \leq T, \tag{3}$$

$$x_{ij} = 0 \text{ or } 1, \ \forall i, j. \tag{4}$$

The objective function (1) minimizes the total cost of the path. Constraint (2) ensures that a path from O to D is obtained, whilst constraint (3) ensures that the path satisfies the specified time constraint. The problem (1), (2) and (4) is just the standard Minimum Cost Flow Problem which can be solved efficiently [1]. A similar formulation can be constructed for the undirected case.

The decision version of the CSPP is NP-complete [15], therefore no exact polynomial algorithms exist for solving them. However, the CSPP has been studied by a number of authors and both exact algorithms and heuristics have been proposed. Methods used to solve CSPP include: k-shortest path ideas [60]; approximation and exact methods using cost scaling and rounding [14,21,42]; and Dynamic Programming formulations [11,28,31]. A few authors [2,3,4,14] introduce preprocessing techniques to reduce graph sizes which results in improvements in computational run times. All these schemes use some form of a label setting approach.

By using the Integer Programming Formulation, the available solution strategies are: to dualize the time constraint (3) and apply the Lagrangian relaxation method [3,7,8,57]; Couple the Lagrangian method with a k-shortest path routine [20]; Use Lagrangian relaxation together with branch and bound algorithms [54]; Apply a linear integer programming package such as CPLEX [26]; or Develop specialized Branch and Cut methods, using CPLEX for solving the relaxed subproblems [5]. A number of techniques are also available for preprocessing and probing [5,55].

All of the strategies mentioned above are computationally feasible for moderately sized networks. However even if a moderately sized grid were to be placed over the sensor field, the resulting number of edges and arcs in the network would make most of the aforementioned methods computationally time consuming, and in defense situations were computation time may be a critical factor, these methods become somewhat infeasible.

3 Continuous Formulation

As shown in [52] the optimal submarine path problem can be formulated as a discrete valued optimal control problem. To do this we first begin with a description of the sonar field. The field is positioned in the Cartesian plane with coordinates (x, y) indicating the latitudinal and longitudinal distance from the origin in kilometres, where the origin is assumed to be the starting point of the journey. We let $(x(t), y(t))$ represent the (point) location of the submarine at time t. This is governed by the following simple dynamical system

$$\begin{aligned} \dot{x}(t) &= s(t)\cos(\theta(t)),\ x(0) = 0, \\ \dot{y}(t) &= s(t)\sin(\theta(t)),\ y(0) = 0, \end{aligned} \tag{5}$$

where $\theta(t)$ represents the heading angle of the vessel in radians and $s(t)$ is the speed of the vessel in km/h. Note that θ and s are control functions in this problem which need to be chosen so that the resulting path of the vessel minimizes its probability of being detected. We have the control constraints

$$0 \le \theta(t) \le 2\pi,\ \ \forall t \in [0, T),\ \text{and} \tag{6}$$

$$s(t) \in \{s_1, s_2, s_3, ...\},\ \forall t \in [0, T). \tag{7}$$

Note that $s(t)$ is a discrete valued control function as it can only take on values from a finite discrete set.

Suppose that a total of n_s sensors are located at positions (x_i, y_i), $i = 1, 2, ..., n_s$, in the field. We assume for simplicity that these positions remain fixed during the journey and that the sonars are all of the same type with the same detection capabilities. Either of these assumptions can easily be relaxed in the model and in our solution method described. At any instant of time, the Euclidean distance of the submarine from each sensor is given by $r_i(t) = \sqrt{(x(t) - x_i)^2 + (y(t) - y_i)^2}$, $i = 1, ..., n_s$. For each given vessel speed s, a probability of detection profile, $p(r, s)$ can be constructed as a function of the physical distance $r(t)$. Assuming that the sensors operate independently, the instantaneous probability of the vessel being detected is then given by

$$P((x(t), y(t), s(t)) = 1 - \prod_{i=1}^{n} (1 - p_i\,(r_i(t), s(t))). \tag{8}$$

Our aim is to minimize the cumulative probability of being detected over the entire journey. This is equivalent to minimizing the objective functional

$$g(\theta, s, T) = \int_0^T P(x(t), y(t), s(t))dt. \tag{9}$$

Note that the submarine must arrive at its intended destination, so we have the terminal state constraints

$$\begin{aligned} x(T) &= x_T, \\ y(T) &= y_T. \end{aligned} \tag{10}$$

Finally, there is also an inequality constraint on the total time taken to travel from the initial to the terminal point

$$T \leq T_{\mathrm{MAX}}. \tag{11}$$

Note, of course, that the terminal time, T, is variable in this problem. In summary, then, the optimal control model of the submarine transit path problem can be stated as: *Find a terminal time T (satisfying (11)), and control functions $\theta(t)$ (satisfying (6)) and $s(t)$ (satisfying (7)) such that the objective functional (9) is minimized subject to the vessel dynamics (5) and the constraints (10).*

The fact that the control s is restricted to a discrete set puts this problem into a general class of discrete valued control problems. Examples of these problems are studied in [24,39]. The main difficulty in these problems is to determine the exact time points where the discrete valued control should switch between its allowed values. Since the gradients with respect to these switching time parameters are discontinuous [58], ordinary gradient based solution methods perform poorly. An additional difficulty is to determine exactly how many such switching times are involved in an optimal solution. The first of these difficulties has been successfully overcome by CPET, which was initially applied to a similar class of time optimal control problems [38] and later directly to discrete valued optimal control problems [39]. The second difficulty can be partially addressed by solving a sequence of problems which are transformed via CPET, but this remains an active area of research. Essentially, CPET involves a scaling of the time horizon, $[0, T]$, via an auxiliary control function known as the enhancing control. This transforms the original problem into an equivalent canonical form which can then be solved by ordinary gradient based methods such as control parameterization [58] and incorporated into the optimal control software MISER3 [29]. In Section 4 we explain CPET through its application to the example at hand. For a more thorough review and discussion of these techniques, see [51].

4 Control Parametization Enhancing Transform

Before we apply CPET to the Submarine Transit Path Problem, we need to specify the set of allowable speeds $s(t)$, and set a limit on the maximum number of course/speed switchings to be allowed. Note that the heading angle control function, $\theta(t)$, is modeled as a piecewise constant function, which is natural, given that the heading angle ought to remain constant between course changes. Furthermore, for the sake of simplicity, we assume that the switching times for the course changes coincide with switching times for the speed changes. This may appear to be restrictive, but note that this formulation does allow for only one of the controls to change values at a particular switching time, so full generality of the control structure is actually preserved. As discussed in the introduction, for the purposes of our model, we use the probability of detection profiles, $p(r,s)$, that are provided in [18]. Here the submarine is restricted to two speeds, 8km/h and 14km/h. The control constraint (7) therefore becomes

$$s(t) \in \{8, 14\}, \ \forall t. \tag{12}$$

The profiles are given in the form of a set of data points. We use cubic splines to interpolate this data to generate smooth $p(r,s)$ curves, as illustrated in Fig. 2.

The requirement for a smooth interpolation arises because we solve the problem using a gradient based optimization technique. The gradient calculations involve the integration of a set of costate equations, which is carried out more conveniently when the objective integrand is a smooth function of the state variables (although this is not a strict requirement in theory). For the purposes of testing the proposed model and solution method, the qualitative features (in terms of the frequency and magnitude of variation) of these interpolated curves adequately reflect a practical situation.

In terms of deciding on the maximum number of switchings that are to be allowed, in some applications, it is quite important to determine the optimal number of switchings (see [24] for an example), and in such cases it may be necessary to solve a sequence of problems each with a different maximum number of allowed switches [39]. However, in this application, we have a practical limitation on the number of course/speed changes during the time horizon, because course and speed changes physically require a minimum period of time to be implemented. Furthermore, a submarine commander is unlikely to implement a solution which involves an excessive number of course/speed changes. Hence, we

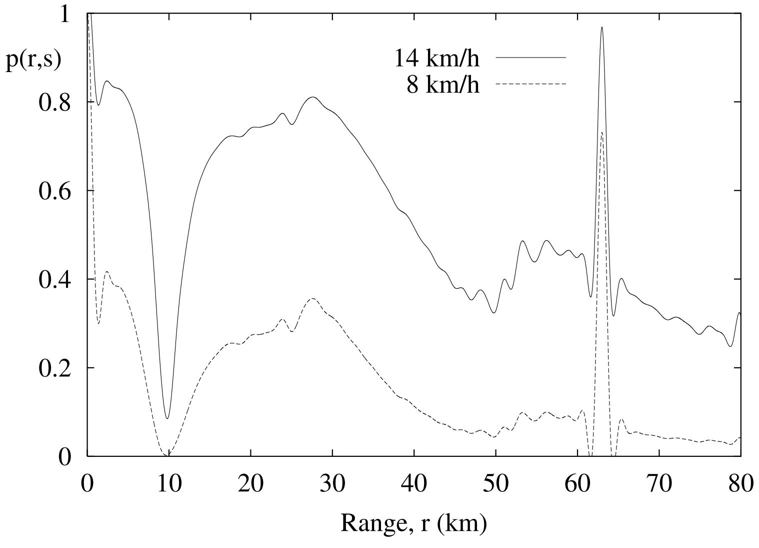


Fig. 2. Probability of detection profiles used in the numerical studies

assume that the maximum number of switchings allowed is $N-1$. The CPET technique may then be applied as follows.

We define a new time horizon $[0,N]$ and partition it into the subintervals $I_1=[0,1)$, $I_2=[1,2)$, $I_3=[2,3)$, ..., $I_N=[N-1,N)$. We then allow $u_1(\tau), \tau \in [0,N)$ to be a piecewise constant function on $[0,N)$ which is consistent with this partition. u_1 is essentially the heading angle of the submarine in the transformed time scale and we still require the control constraints

$$0 \leq u_1(\tau) \leq 2\pi, \ \forall \tau \in [0,N). \tag{13}$$

Furthermore, we define

$$u_2(\tau) = \begin{cases} 14, & \text{if } \tau \in I_k, \ k \text{ odd}, \\ \ 8, & \text{if } \tau \in I_k, \ k \text{ even}. \end{cases} \tag{14}$$

This (fixed) control function takes on the role of $s(t)$ in the transformed problem. Note that it is consistent with the constraint (12). Furthermore, we define the *enhancing control*, $u_3(\tau)$, to be a piecewise constant function consistent with the above partition and subject to the following constraints

$$0 \leq u_3(\tau) \ \leq \ T_{MAX}. \tag{15}$$

The constraint (15) arises due to the total time constraint (11), but, by itself, will not be sufficient to replace (11) entirely.

The main feature of the CPET method is the scaling, via the enhancing control, which relates the original time horizon $[0,T]$ to the new time horizon $[0,N]$. This is done through the following differential equation

$$\frac{dt}{d\tau} = u_3(\tau), \ \tau \in [0,N), \ t(0)=0. \tag{16}$$

Note that integration of (16) over $[0,N)$ will allow us to recover the original time horizon $[0,T]$, where $T=t(N)$. To standardize notation, we set, $x_1=x$, $x_2=y$ and $x_3=t$. The transformed problem may then be stated as follows. Find piecewise constatnt control functions $u_1(\tau)$ and $u_3(\tau)$ (consistent with the above mentioned partition) such that the objective functional

$$\int_0^N P(x_1(\tau), x_2(\tau), u_2(\tau)) u_3(\tau) d\tau \tag{17}$$

is minimized subject to the dynamics

$$\begin{aligned} \dot{x_1}(\tau) &= u_3(\tau)u_2(\tau)\cos(u_1(\tau)), & x_1(0)&=0, \\ \dot{x_2}(\tau) &= u_3(\tau)u_2(\tau)\sin(u_1(\tau)), & x_2(0)&=0, \\ \dot{x_3}(\tau) &= u_3(\tau), & x_3(0)&=0, \end{aligned} \tag{18}$$

and subject to the constraints

$$\begin{aligned} x_1(N) &= x_T, \\ x_2(N) &= y_T, \\ x_3(N) &\leq T_{MAX}. \end{aligned} \tag{19}$$

Note that the third constraint in (19) arises directly from (11).

The transformed problem now simply involves piecewise constant control functions defined on a regular fixed partition of the fixed time horizon $[0, N]$. As such, it can be solved directly by the optimal control software MISER3 [29]. Note that the optimal solution of the original problem can be recovered easily from the solution of the transformed problem, as the original time scale is given by $x_3(\tau)$.

5 Localized Solutions

Past numerical experiences [52] suggests the solution generated by the optimal control model depends very much on the initial starting solution provided by the user. To test whether this is the case with the submarine transit problem, we tested and compared five different initial solutions. We begin by first describing how our test problems were generated.

We have restricted our test problems to sonar fields with $n_s = 4$ sensors, though any number can be easily incorporated into the model. Furthermore, we consider a region of 80km by 80km. In addition we set $x_0 = 0$, $y_0 = 0$, $x_T = 80$ and $y_T = 80$, that is the starting point is located at $(0, 0)$km and the destination is at the point $(80, 80)$km. We constructed 30 problems each with different sensor locations. The locations of the sensors were determined by randomly generating 240 integer values, $\{x_1, x_2, \ldots, x_{240}\}$, between 0 and 80. We then paired these together, (x_1, x_2), $(x_3, x_4), \ldots$, (x_{239}, x_{240}), to give the coordinates, in kilometres, of the sensors in relation to the starting point of the journey. The first four pairs give the sensor locations for the first problem, the next four pairs represent the sensor locations for the second problem, and so on. For our model assume that the sensors remain fixed in these positions for the duration of the submarines journey.

For each of the 30 problems we imposed four different time constraints T, ranging from a low or 'tight' time constraint to a high or 'loose' time constraint. To generate these constraints, we have imposed a grid size over the region, these being either of dimension 20×20 or 80×80, all being equally spaced rectangular grids. We then restrict our attention to this grid. As outlined in Section 2, the result of imposing a grid over the specified region is a graph were the weights of the edges are the cost and travel time of traversing that particular edge. The detection curves of Hallam [18] are used to generate the cost of edges of our graph.

For each problem we determined its minimum time path, denoted by $T_{\min}$. This can easily be determined by simply dividing the shortest possible distance that the submarine must travel, 160km, by the maximum speed that the submarine can travel at, 14km/h. Therefore for each problem, this minimum time value will be the same, $T_{\min} = 11.42857$hrs. We also determine a maximum time value for each of the problems, denoted as $T_{\max}$. This is done by applying Dijkstra's algorithm [12] to the graph, using the arc costs, which will give us the minimum unconstrained shortest path. The time corresponding to the unconstrained shortest path becomes our $T_{\max}$. The four time constraints are then found by the formulae $T_\alpha = (1-\alpha)T_{\min} + \alpha T_{\max}$ for $\alpha = 0.2, 0.4, 0.6$ and 0.8. $T_{0.2}$ represents the 'tightest' constraint and $T_{0.8}$ the 'loosest' constraint.

For our first set of test problems, we generated 30 random paths which served as our initial starting solution for the optimal control phase. We set the amount of switches, $N-1$, to 100 that is we allow for a maximum of 100 course/speed changes. The time constraints for these problems were generated using the 20×20 grid as described above. These random paths used were not necessarily feasible as they were not required to meet the time constraint T nor did they have to finish at the destination point $(80, 80)$km. Also, in most instances, these paths did not stay within the boundaries of the 80km by 80km region.

Our next four sets of test problems were generated in a similar fashion to one another. As an alternative starting solution we also used the Euclidean path which joined the starting point $(0, 0)$km to the destination $(80, 80)$km. This initial diagonal path was either traversed entirely using the slow speed 8 km/h or the fast speed 14 km/h, and for both of these paths we tested the case where we allowed either 100 or 325 switches. The time constraints for the problems with 100 switches were generated using the 20×20 grid and the time constraints for the 325 switch problems were generated via the 80×80 grid. It should be noted that all the diagonal paths using the fast speed are time feasible, whereas when using the slow speed diagonal path only those with the time constraint $T_{0.6}$ and $T_{0.8}$ are always feasible. That is there are some instances within the $T_{0.2}$ and $T_{0.4}$ cases where the initial paths are not feasible in terms of meeting the time constraint.

These five sets of paths were then used as initial guesses for the optimal control model. The MISER3.2 [30] software was then used in conjunction with two of the nonlinear programming solvers, NLPQL [56] and NPSOL [16], to refine these solutions.

All the computational tests discussed in this section were carried out on a Sun Netra X1 with a 500MHz 64-bit Ultra SPARC Processor and 256MB of RAM.

For the random paths, we tested these starting solutions using the NLPQL routine. The results are presented in Table 1.

As can be seen in Table 1 we have broken down the 30 problems into each of their four different time constraints cases T_α, based on α, and averaged each of these 'categories' out as well as given the average over all the 120 problems in

Table 1. Random NLPQL results

α	CPU Time (minutes)	Euclidean Distance (meters)
0.2	4.04127	13.55701
0.4	5.38671	2.80282
0.6	5.94014	1.596362
0.8	6.13041	415.26988
All	5.37463	108.30652

the row All. We can see that as the α value increases, that is the time constraint becomes more loose, the Computational (CPU) Time increases from around 4 minutes to just over 6 minutes. Overall, the random starting solutions resulted in a solution being found by the optimal control method in 5.37 minutes. We also present the average distance each of the solutions determined by NLPQL were from meeting the terminal state condition that $x_T = 80$ and $y_T = 80$, that is how far the submarine was from arriving at the destination point. These results are presented in Table 1 under the column Euclidean Distance. We see here that for the $\alpha = 0.2, 0.4$ and 0.6 cases the solutions were on average 13.55, 2.80 and 1.59 meters out from the destination point $(80, 80)$km respectively. However in our situation, when we consider that a typical (non-nuclear) submarine, such as the Australian Collins Class submarine, is in the vicinity of 70-80m in length all these distances become insignificant and can be ignored. However for the $\alpha = 0.8$ case, the solution was 415.27 meters out from the destination point, and on average over all the problems it was 108.31 meters out. Since 108.31 meters exceeds the length of the submarine the results generated from using a random initial solution would have to be declared, on average, infeasible, as they did not meet the terminal state constraint.

In Fig. 3 we present the solution to one of the random starting solution problems. This problem corresponds to the sensor pattern

$$\{(15, 61), (0, 25), (41, 44), (0, 78)\},$$

with a time constraint of $T_{0.6} = 16.31428$. The final optimal control solution had a cost of 6.05637, a Euclidean distance from the destination point of 0.44064 meters and a computational time of 8 minutes.

In Fig. 3 we display both the initial random starting solution, given as the black line, and the resulting optimal control solution, represented by the white

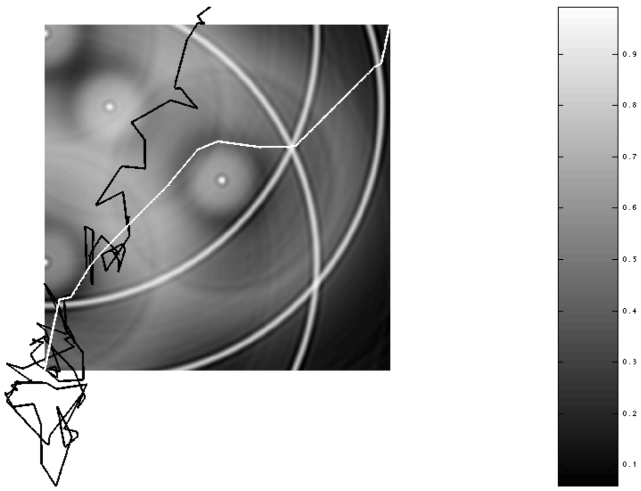


Fig. 3. Random starting solution and optimal transit path

line. As pointed out, and as we can see with this problem, the random path did not have to stay within the prescribed region nor did it have to finish at the destination point. We have chosen to use the probability map associated with the 8km/h speed to plot the paths on, however the path is still made up of both speeds 8km/h and 14km/h. A feature not immediately obvious from these figures is that the speed of the vessel changes according to the level of danger present in the various sections of the path. It can be seen that the transit path stays close to low danger areas as long as it is not forced into a major detour. However at times it is necessary for the submarine to cross high probability of detection regions on its way to the destination. There are 4 such regions along the path in this case, a narrow one initially, a broad one about a third the way along the path, a narrow one about two thirds down the path and another narrow one just before reaching the destination. Contrary to what one might expect, when these high danger areas are traversed the optimal solution is to do so at the high speed. In choosing a high speed through these high danger regions the total amount of time spent there will be reduced, and thus the cumulative effect of exposure is minimized.

From the optimization point of view, it is wiser to put up with a high danger region for a short period of time rather than to take a long period of time detouring through low danger regions. This is clearly a direct reflection of the objective function used in our model.

When testing the Euclidean Path starting solutions the NPSOL routine was employed. The results for both the slow and fast diagonal paths using 100 switching points and the 20×20 grid time constraints are given in Table 2.

Table 2 shows us that the optimal control phase takes, on average, 67.22 minutes and 47.04 minutes to determine a solution using the slow diagonal and fast diagonal path respectively as a starting solution. The only case where the slow diagonal results in a faster CPU time is for $\alpha = 0.2$ where it is around 30 minutes faster than the equivalent fast diagonal. All the initial starting paths resulted in optimal control solutions that came very close to finishing right at the destination. The slow diagonal optimal control path was 2.21 meters out while the fast diagonal optimal control paths were slightly more ‘accurate’ being

Table 2. Euclidean path NPSOL results using 100 switches

	Slow		Fast		
α	CPU Time	Euclidean Distance	CPU Time	Euclidean Distance	Fast Cost/ Slow Cost
0.2	36.09530	1.65446	66.06981	0.70830	1.02714
0.4	128.06026	2.47124	37.76912	2.73971	1.09184
0.6	54.96907	1.06329	40.29762	1.16792	1.09986
0.8	49.76313	3.65839	44.03036	1.19142	1.05613
All	67.22194	2.21185	47.04173	1.45184	1.06874

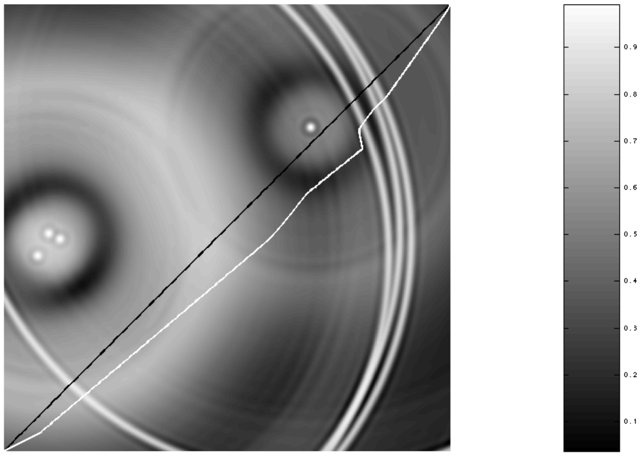


Fig. 4. Fast diagonal path with 100 switches and optimal transit path

only 1.45 meters from meeting the terminals state constraints. It should be noted that all solutions resulting from the optimal control phase meet the time constraint T even those problems that began with a time infeasible path. We have also compared the final objective function value of the path determined by the optimal control phase by taking the ratio of the cost between the two solutions. It shows that the results corresponding to the initial fast diagonal path are, on average, 6.87% greater than that obtained from the slow diagonal path. So by employing a fast diagonal over a slow diagonal starting point we can produce optimal control results in less CPU time that are also more accurate than if we were to use a slow diagonal path. However these solutions have a higher objective function value. Therefore the starting solution will determine the quality of solution obtained, and thus there will be a trade-off between the cost of the path generated and the accuracy and time it takes to compute it.

In Fig. 4 we present the solution to one of the fast diagonal, 100 switches, starting solution problems. This problem has sensors located at

$$\{(6, 35), (55, 58), (10, 38), (8, 39)\},$$

with a time constraint of $T_{0.2} = 13.05714$. Again the starting solution is the black line and the final optimal control solution is represented by the white line. After 65.55 minutes of CPU Time, the final solution has a cost of 6.50491 and a Euclidean distance of 1.23162 meters.

The results for both the slow and fast diagonal paths using 325 switching points are given in Table 3. Again the NPSOL routine was used.

From Table 3 we can see that on average the objective function value of the path determined by NPSOL using the fast diagonal as a starting solution is 6.73% greater than the corresponding slow diagonal. In terms of the computational time, over all the α cases, using the slow diagonal (56.48 minutes) always results

Table 3. Euclidean path NPSOL results using 325 switches

	Slow		Fast		
α	CPU Time	Euclidean Distance	CPU Time	Euclidean Distance	Fast Cost/ Slow Cost
0.2	61.99554	12.34010	85.27216	20.52914	1.07538
0.4	51.04162	79.99248	85.58536	4.77574	1.05912
0.6	60.68678	4.47546	71.62280	31.73964	1.06484
0.8	52.21530	54.44737	70.83199	71.81457	1.06971
All	56.48481	37.79692	78.32808	32.21477	1.06726

in less computation time than employing than the fast diagonal (78.33 minutes). The Euclidean distances the final solutions are from the destination point are, on average, 37.80 meters for the slow diagonal and 32.21 meters for the fast diagonal. These distances might seem a bit excessive in comparison to the cases involving 100 switches, however, since they are still within the prescribed 70–80 meter length of a typical diesel class submarine we can safely declare them as being feasible. In fact any solution with a distance within 100 meters of the terminal constraint we will consider being feasible. Again all solutions resulting from the optimal control phase meet the time constraint T. So in this instance, unlike the 100 switches case, by employing a slow diagonal starting point we can produce optimal control results with a smaller objective function value in less CPU time than if we were to just use a fast Euclidean starting solution. Again however, the fast diagonal path produces a more accurate solution.

What we notice in the examples shown in Figs. 3 and 4, is that the optimal control path stays within the general vicinity of the starting solution path. From observing the sonar fields of these examples it would seem to be more prudent to construct a path along the very bottom and far right hand side of the enclosed region where there are substantial areas of low probability. But for the random path, since the initial path generated lies within the top portion of the area the optimal control model is 'encouraged' to find a solution within this location. Similarly for the fast diagonal path, the optimal control model only tweaks the initial path slightly downward into a safer region rather than dragging it all the way to the bottom of enclosed region.

Even though it is not possible to make a direct comparison between the results given in Tables 2 and 3, since the time constraints on the problems are different, we still see that there is a substantial difference between the Euclidean distances the solutions are from the destination point not to take notice. On average over the α cases, we see that for 100 switches the solutions are accurate to within 0.71 to 3.66 meters whereas the accuracy for 325 switches is within the vicinity of 4.48 to 85.59 meters. So we can see that the lesser the amount of switches used in the initial solution the more accurate the final solution will be. Therefore just as in the previous studies we mentioned earlier, we believe the number of switches used in submarine transit path problem will have a bearing on the quality of the solution obtained.

Finally, comparing the results from Table 1 to those in Table 2, where in both cases we used 100 switches, we see that using the random path is substantially quicker in computational time than using either a slow or fast Euclidean path. But one should not employ a random starting solution over the Euclidean paths as the solutions were infeasible. However, these observations may be the result of the nonlinear solver used within MISER3.2 and not the quality of the starting solution employed as our initial starting solution.

6 Hybrid Method

We have shown that the final Optimal Control solution is highly sensitive to the starting solution employed. The Random and Euclidean paths we used were shown to be ineffective initial solutions for our Submarine Transit Path problem. It may well be worth some extra effort in developing and implementing a procedure to calculate a 'superior' starting solution as it could produce results in the optimal control phase which are superior to those obtained from using some trivial starting solution such as the random and Euclidean paths. What we need is a method to produce initial paths that are within the locality of the global minimum solution since, as we have shown, the Optimal Control will only move the path around slightly keeping it within the same vicinity. In response to this what we propose to do is to place a grid, such as the one displayed in Fig. 1, over the sensor field and to solve the ensuing CSPP on a network. The solution to this network problem will be our starting solution for the optimal control phase.

To solve the CSPP we introduce a heuristic that generates a feasible solution that is near optimal or optimal with minimal computational time. It should be noted that in defense situations, such as this, computational time can be a critical factor thus we need to find a satisfactory trade-off between computation time and optimality.

Each edge (i, j) in our graph has two weights associated with it, a cost c_{ij} and transit time t_{ij}. The idea is to parameterize these two weights into one label on the edge using the equation $w_{ij}(\beta) = \beta c_{ij} + (1 - \beta)t_{ij}$, where $\beta \in [0, 1]$, then solve the resulting SPP while incrementing β until a time feasible path, $t(P) \leq T$, is found. The heuristic begins by first setting $\beta = 1$ and then applying Dijkstra's algorithm [12] to the resulting graph to find the shortest path P from the source O to the destination D. When $\beta = 1$, each edge weight is just c_{ij} and we are therefore solving the unconstrained SPP. If the corresponding path P is time feasible we can stop. In this trivial case where the time constraint T is meaningless, in terms of the CSPP, the solution is also guaranteed to be the optimal solution to the problem. If $\beta = 1$ does not produce a feasible solution we need to reduce the value of β. By reducing β we are effectively giving the time a higher weighting, applying Dijkstra's algorithm [12] to $w_{ij}(0)$ is the same as determining the minimum time path of the graph. We start decrementing β by

0.1 in each iteration and compute the minimum $w_{ij}(\beta)$ path P until $t(P) \leq T$. Once we have found the appropriate β value that results in a time feasible path we need to refine β to a precision of two decimal places by searching the region $\beta \in [\beta, \beta + 0.09]$. Starting at $\beta = \beta + 0.09$ we keep reducing β by 0.01 until the $w_{ij}(\beta)$ path is feasible.

This path forms our initial guess for the optimal control phase of our approach. It is envisaged that this path is reasonably close to the global solution for the continuous Optimal Control problem, therefore, should be sufficient for our purpose. So our proposed method for solving the Optimal Submarine Transit Path problem involves two phases. In the first stage we discretize the problem, which results in a CSPP, and we then solve the resulting network by the network heuristic described above. The second stage involves the use of the optimal control model that utilizes the solution obtained in the first stage as a starting point. In this phase of our procedure we make use optimal control software package MISER3.2 [30].

7 Comparison of Heuristic and CPLEX

As we demonstrated in Section 2, after discretizing the sensor field with a network, it is possible to formulate the resulting CSPP by an integer program. As such it should be possible to apply the software package CPLEX to the integer programming formulation to determine optimal solutions.

We tested our proposed method on problems generated in the exact same manner as was described in Section 4, using the same number of sensors that are located in the same 30 positions as before. To test our heuristic we imposed three different grid sizes over the region for the network phase, these being of dimension 20×20, 40×40 and 80×80. However, we imposed the time constraint associated with the 20×20 network to the other two grid dimensions.

We formulated all these problems as integer programs. We solved these formulations by using ILOG CPLEX 9.0 on an Intel Pentium 4 PC with a 2.40GHz Processor and 1GB of RAM using Microsoft Visual Studio .NET 2003 as the interface. As pointed out in Section 2 if the order and size of the network is somewhat large CPLEX can become computationally time consuming. Consequently we have introduced two additional stopping criterions, apart from the optimality stopping condition that already exists within CPLEX, to ensure a reasonable CPU time. The first of these stipulates that once the duality gap, or difference, between the objective function value of the current integer solution, which is our upper bound, and lower bound becomes less than 0.1% of the integer solution cost then we stop, and use the current best integer solution as our final solution. However if the optimal solution has not been determined or the duality gap has not been reduced to less than 0.1% after 5 minutes of CPU time we terminate and again use the current integer solution as our final answer.

In Table 4 we present the CPLEX results for the 20×20, 40×40 and 80×80 grids respectively. In the table we display the average CPU time, in seconds, to determine a result and the duality gap associated with this solution. Also shown

Table 4. CPLEX results

	20 × 20 Grid			40 × 40 Grid			80 × 80 Grid		
α	Cost Ratio	CPU Time	Duality Gap %	Cost Ratio	CPU Time	Duality Gap %	Cost Ratio	CPU Time	Duality Gap %
0.2	1.034	31.19	0.07643	1.039	21.48	0.04717	1.043	52.27	0.04830
0.4	1.024	12.77	0.03927	1.028	11.60	0.03255	1.032	65.67	0.04780
0.6	1.007	1.78	0.00828	1.017	5.18	0.02370	1.025	33.66	0.02743
0.8	1.010	0.34	0.01587	1.004	25.24	0.04867	1.003	50.56	0.05670
All	1.019	11.52	0.03496	1.022	15.88	0.03802	1.026	50.54	0.04506

in the table is the ratio, Heuristic/CPLEX, between the objective function values of the corresponding heuristic and CPLEX solutions allowing us to compare the two solution procedures. What we notice is that as the grid dimension increases so to does the average CPU time and Duality Gap. This is a direct result of the number of nodes and edges incorporated in each of these grids. As the order and size of the grid increases so to will the amount of variables and constraints needed to model the problem as an integer programming formulation. This enlarges the solution space and therefore the amount of time needed to find an optimal solution. We can also see that when we compare the CPLEX results against our heuristic results, the heuristic results are on average 1.9%, 2.2% and 2.6% greater than the objective function values obtained via CPLEX for the 20×20, 40×40 and 80×80 grid problems respectively.

In Table 5 we display the duality gap distribution of the solutions. We observe that the number of optimal solutions that are found, that is those problems with a 0% duality gap, decreases as the grid dimension increases. The number of final solutions that are sufficiently close to being optimal, that is the problems with a duality gap between 0 and 0.1%, increases as the grid size increases. The same occurrence can be seen with the amount of problems that have not closed the duality gap to within 0.1% inside of the 5 minute CPU time constraint.

As we mentioned above, even when the network is just of moderate size the CPU time taken to solve the problem to optimality can become prohibitively large. In fact, there is nothing to guarantee that CPLEX would eventually determine the optimal solution even if we were to let it run without any time constraint. To illustrate this point we reran the four 20×20 grid problems that

Table 5. Distribution of CPLEX results

Grid Size	Optimal Solution	Duality Gap $0 - 0.1\%$	Duality Gap $> 0.1\%$
20 × 20	99	17	4
40 × 40	78	37	5
80 × 80	55	54	11

had not resolved to a solution of within 0.1% of the duality gap in the 5 minutes prescribed. However this time, while keeping the duality gap stoping criteria intact, we increased the 5 minute CPU time limit to 1 hour. Similarly we reran the five 40×40 grid problems that had not established a solution to our satisfaction within the 5 minute time limit but we enlarged the CPU time constraint to 4 hours in this case. In both these cases, even with the increase in CPU time, it didn't have any effect whatsoever on the quality of the solution determined by CPLEX, the solutions remained identical to the ones found after only 5 minutes and their duality gaps had not tightened up at all.

The eleven 80×80 grid problems that did not solve to within a duality gap of 0.1% were also rerun using CPLEX but with a 5 hour time limit on this occasion. The results of this are presented in Table 6. With this new 5 hour time constraint only the problem associated with sensor pattern 29 managed to both close its duality gap to less than 0.1% and improve the objective function value of the solution, it required 44.22 minutes of CPU time to do so. Of the ten remaining problems that still had a duality gap of over 0.1% after the 5 hour time limit had expired, just two had improved their objective function value, and they only did so very slightly. For the remaining problems there was no real benefit in letting CPLEX run for the additional CPU time since the only result of doing so was a small tightening up of some of their duality gaps. In fact we can't be sure that duality gaps would eventually close up to within 0.1% even if we were to let CPLEX work on these ten problems over an extended period. For example, when we take the sensor pattern 21 problem shown in Table 6, and let CPLEX run for 4 days the objective function still remains unchanged from that value obtained after only 5 minutes and the duality gap is no different from the result that was obtained after 5 hours.

Table 6. Rerunning of CPLEX on selected 80×80 problems

		5 minute CPU Time Limit		5 hour CPU Time Limit	
Sensor Pattern	Time Constraint α	Obj Fn Value	Duality Gap %	Obj Fn Value	Duality Gap %
1	0.2	8.784	0.227538	8.781	0.167702
1	0.4	7.89	0.276273	7.89	0.276273
2	0.4	7.708	0.146773	7.708	0.137215
5	0.8	4.628	0.430061	4.628	0.407408
7	0.4	8.259	0.170637	8.259	0.141423
9	0.8	7.202	0.162646	7.202	0.159338
13	0.2	8.285	0.115284	8.285	0.115284
21	0.8	6.605	0.176625	6.605	0.13409
26	0.2	9.182	0.379952	9.177	0.302471
27	0.6	7.171	0.127628	7.171	0.106674
29	0.4	8.013	0.138233	8.011	0.097239

It should be noted that when we used our heuristic on these 20×20, 40×40 and 80×80 grid problems it required on average around 0.1, 0.8 and 6.5 seconds, respectively, using the same Intel Pentium 4 PC with a 2.40GHz Processor and 1GB of RAM, to determine the solutions as compared to 11.52, 15.88 and 50.54 seconds using CPLEX. So even when we restrict CPLEX to a 5 minute CPU time limit, our heuristic is still significantly quicker in determining a solution. Therefore by using our heuristic we can achieve optimal or very near optimal solutions in substantially less time than if we were to employ CPLEX.

8 Numerical Results

As mentioned our Two Phase approach uses the solutions obtained from our heuristic as initial starting points for the corresponding optimal control model. We tested our proposed method on 480 problems that were generated in the exact same manner as was described in Section 2, with the addition of a 60×60 grid. Again we imposed the time constraint associated with the 20×20 network across the other grid dimensions. By preserving the same time constraint for all the grid sizes we are able to directly compare the effect that the initial starting solutions generated via the different sizes has on the final path obtained in the optimal control phase. The results were obtained via an Intel Pentium 4 PC with a 2.40GHz Processor and 1GB of RAM.

We need to decide on how many switches to use within the optimal control phase, these amounts are presented in Table 7.

We see from Table 7 that the number of switches increases as the grid size increases so as to incorporate the heuristic solution into the MISER software package. That is, the number of line segments allowed in the optimal control model is chosen sufficiently large so that the heuristic solutions can be fully parameterized. Indeed, a certain percentage of additional segments are allowed in case they are needed for an optimal solution. For example for the 60×60 case, at minimum one would have to travel 120 edges to get from the source to the destination, 60 edges to the right and 60 edges up. Each edge can be traversed in one of two possible speeds, so we double 120 to give us 240. Then as mentioned we allow for some additional switches as well, in case of any backward or downward movements made by the submarine. We include 20 extra switches for the 20×20 grids, 17 for the 40×40 grids, 3 for the 60×60 grids and a further 5 for the 80×80 networks.

In the optimal control phase we employed the NPSOL routine, as previous research [6] suggests it to be the most effective nonlinear solver for our problem.

Table 7. Number of switching points

	20×20	40×40	60×60	80×80
# Switches	100	175	243	325

Table 8. Comparison of different grid sizes

	20×20			40×40		
α	% Improvement	CPU Time	Euclidean Distance	% Improvement	CPU Time	Euclidean Distance
0.2	33.68	6.13	1.63	33.19	14.30	2.07
0.4	26.51	6.98	0.80	26.32	13.64	0.36
0.6	18.58	8.46	1.81	20.27	20.17	0.91
0.8	14.93	9.60	0.80	13.99	15.26	0.95
All	23.43	7.79	1.26	23.44	15.84	1.07

Table 9. Comparison of different grid sizes (cont.)

	60×60			80×80		
α	% Improvement	CPU Time	Euclidean Distance	% Improvement	CPU Time	Euclidean Distance
0.2	32.03	28.88	5.08	28.69	52.57	18.11
0.4	26.95	35.37	3.59	21.33	47.93	14.17
0.6	19.61	38.16	5.03	15.65	55.23	1.56
0.8	12.10	26.97	8.63	8.48	58.66	31.24
All	22.67	32.34	5.58	18.54	53.60	16.27

Table 8 and Table 9 shows us that the % Improvement that is made over the initial solution decreases as the grid size increases, apart from the 20×20 and 40×40 cases where the average improvement is virtually the same. We also observe that the CPU time, in minutes, increases as the grid dimension becomes larger.

For both the heuristic and optimal control solution we measured the average percentage of how far the results obtained were from the corresponding best solution of that approach. For example, for the heuristic phase, after determining the solution for each grid size using the same time constraint, we compared the costs of the paths $C(P)$ against the best $C(P^*)$ of the four solutions using the equation $\left[\frac{C(P)-C(P^*)}{C(P^*)}\right] \times 100$. This procedure was also followed for the optimal control solutions. This will tell us how far, in terms of percentage above the best solution, the path was from the best solution obtained.

From Table 10 we see that as the grid size increases from 20×20 to 80×80 the quality of the heuristic solution generated improves from 3.14% to 1.15% on average above the best solution. This is what we would expect since from a larger grid size we can determine a more refined path that has a lower cost than if we were to use a smaller dimension grid. However, when we look at Table 11 we note that the best solution obtained via the optimal control phase comes from the starting solution generated by the 40×40 grid, it is on average 3.57%

Table 10. Heuristic: Average % above best solution

α	20×20	40×40	60×60	80×80
0.2	2.83	2.25	1.87	1.57
0.4	3.02	2.39	1.43	1.40
0.6	2.68	2.33	1.92	1.42
0.8	4.04	2.09	0.83	0.20
All	3.14	2.27	1.51	1.15

Table 11. Optimal Control: Average % above best solution

α	20×20	40×40	60×60	80×80
0.2	3.14	3.35	4.67	10.10
0.4	5.27	4.74	2.89	11.47
0.6	5.55	3.05	3.47	8.24
0.8	3.87	3.16	4.19	7.68
All	4.45	3.57	3.81	9.37

above the best solution. The worst solution, on average, was found from using the 80×80 path which was 9.37% greater than the best solution.

This shows us that despite using a 'better' path from the 80×80 grid dimension as an initial guess the resulting optimal control path is not as good as the path resulting from using any other of the grid dimensions. This contradicts our belief that the best optimal control solution would be found by using a feasible path which has the lowest cost. These results and observations are a direct consequence of the different number of switching points used within the optimal control phase. We still believe that the final optimal control solution is dependent on a good initial solution, however, the number of switching points also have a large bearing on the quality of result obtained. This would appear to be due to the nature of the CPET transformation which tends to introduce a large number of local minima in the transformed optimal control formulation of the problem. Consequently the likelihood of getting stuck in a local minima increases with the number of switching points.

To reduce the possibility of our optimal control solution becoming caught in a local minimum we need to reduce the number of switches used within the second phase of our procedure. For conformity sake, we have been using the 'maximal' amount of switches, across the board, that was necessary for a one-to-one mapping of any possible network path into the required MISER input format regardless of whether they were required or not. To keep the amount of switches to a minimum we only used a sufficient amount that allowed us to incorporate the heuristic solution into the requisite input format. In Table 12 we display the average minimum number of switching points used for each of the four grid dimensions.

Table 12. Average minimum number of switching points used

	20 × 20	40 × 40	60 × 60	80 × 80
# Switches	17.64167	27.80833	39.375	49.425

Table 13. Comparison of different grid sizes using minimal switching points

	20 × 20			40 × 40		
α	% Improvement	CPU Time	Euclidean Distance	% Improvement	CPU Time	Euclidean Distance
0.2	31.24	1.80	0.45	30.71	2.89	0.50
0.4	26.13	2.55	0.39	25.32	3.42	0.51
0.6	17.59	3.01	0.48	16.76	3.27	0.49
0.8	11.77	3.31	0.66	12.57	3.55	0.50
All	21.68	2.67	0.50	21.34	3.28	0.50

To test the effect that using a minimal amount of switches has on the final solution we use the same initial network solutions as before, with the same time constraints, and again we employ the NPSOL routine. However in this case we use a minimal amount of switches in the optimal control phase.

As we observed with our earlier results, Tables 13 and 14 shows us that the percentage improvement that is made over the initial solution decreases as the grid size increases. Again we also note that as the grid dimension becomes larger the CPU time, in minutes, increases. What we can also discern from these tables as compared to Tables 8 and 9, is that the Euclidean distances have all 'tightened up', especially in the 80 × 80 problems where it has reduced from 16.27 meters out from the destination down to 0.63 meters.

Again, we measured the average percentage of how far the optimal control results obtained were from the corresponding best solution. These results are

Table 14. Comparison of different grid sizes using minimal switching points (cont.)

	60 × 60			80 × 80		
α	% Improvement	CPU Time	Euclidean Distance	% Improvement	CPU Time	Euclidean Distance
0.2	30.94	3.38	0.48	31.49	3.21	0.41
0.4	24.47	3.31	0.42	23.01	4.30	1.31
0.6	17.59	4.66	0.39	16.67	4.66	0.40
0.8	10.97	4.70	0.41	11.26	8.14	0.42
All	20.99	4.01	0.43	20.61	5.08	0.63

Table 15. Optimal control minimal switches: Average % above best solution

α	20×20	40×40	60×60	80×80
0.2	6.03	6.26	5.15	4.32
0.4	4.35	4.88	5.05	7.07
0.6	5.74	6.53	4.89	5.65
0.8	5.48	2.55	3.16	2.25
All	5.40	5.05	4.56	4.82

presented in Table 15. In this instance, when we use minimal switches, the best solution which is obtained from the optimal control phase comes from using the grid not the 60×60 grid not the 40×40 grid as was the case. What stands out most from Table 15, as compared to those in Table 11, is the improvement in the 80×80 solutions. From being the worst at 9.37% above the best solution it is now only 4.82%, the second best of our optimal control solutions. This shows us that, as we believed, using the maximum 325 switches introduces too many local minima into the solution space thus increasing the odds that our initial path will tend towards one of these instead of the global minimum solution. The worst solution, on average, was found from using the 20×20 path which was 5.40% greater than the best solution. The explanation for this being that the 20×20 grid is less likely to produce a path that falls within the vicinity of the global continuous minimum solution then using a higher grid dimension, also an average of 17.64 switches in the optimal control phase is perhaps too small a number to allow for a sufficient resolution of the problem.

Finally in Table 16 we summarise the difference in the results from using both the maximum and minimum amount of switches. Along with % Improvement and CPU Time numbers that we have already presented, we also include the Cost and Time ratios between the two solutions. That is for the Cost Ratio, we divide the cost of the optimal control solution where we used the maximum switches by the objective function value of the optimal control solution using the minimum number of switches. Similarly we do the same for the CPU Times between the two. Apart from the 80×80 case, we can see from the Cost Ratio column that by reducing the number of switches to a minimum the cost of the path found is roughly 2% worse off than if we were to use the maximum amount of switches in the optimal control phase. However when we look at the Time Ratio between the two we notice that there is an order of magnitude speed up in the CPU times by between 3 to 8 times. So by reducing the number of switches to a minimum we can significantly speed up the CPU time with the only side effect being a tiny decrease in the quality of the path determined. However, when we consider the 80×80 problems not only does reducing the switches to a minimum speed up the CPU time by a magnitude of over 10.5 times but it also improved the solution path by 3.22%.

So in summary, there needs to be a balance struck with regard to the size of the grid used and the number of switches used in the optimal control phase.

Table 16. Comparison of the results for maximum and minimum switches

	% Improvement			CPU Time		
Grid Size	Max Switches	Min Switches	Cost Ratio Max/Min	Max Switches	Min Switches	Time Ratio Max/Min
20×20	23.43	21.68	0.9816	7.79	2.67	2.92
40×40	23.44	21.34	0.9760	15.84	3.28	4.82
60×60	22.67	20.99	0.9819	32.34	4.01	8.06
80×80	18.54	20.61	1.0322	53.60	5.08	10.56

We need to reduce the amount of switches to a point where we relegate the number of, and therefore chances of becoming caught in a, local minima in the transformed optimal control problem, and also to decrease the computational time of finding a solution path, but not so much that it reduces the quality of the path determined. Reducing the amount of switches means there are less course and/or speed changes that can be made in the continuous problem. We need to allow for an adequate amount so that the submarine can effectively navigate the sensor region without putting itself at unnecessary risk.

9 Conclusions

This chapter has established and demonstrated a computational procedure for the resolution of a discrete valued optimal control path design problem where a transit path needs to be determined for a submarine passing through a sensor field. The objective is to minimise the probability that the submarine will be detected while still satisfying a total time constraint on the route.

Our approach to solving the submarine transit path problem was to use two phases. In the first phase we generate a solution for the discretized network problem, using a simple efficient heuristic. In the second phase of our method, we refine the network solution by employing it as an initial starting point for optimal control approach. Extensive numerical testing successfully demonstrates the use of our methodology. Improvements can be made in the range of 16 to 24% by applying the optimal control method over the discretized network model.

Through our research we also established that the determination of a good solution, via the use of our two phase method, is very much dependent on a number of factors. These factors include:

1. The initial route used as the starting solution for the optimal control phase. We have shown that the final optimal control solution will stay within the vicinity of the primary solution. Using some trivial initial solution will be totally inefficient as it will more than likely not be in the area of the global minimum solution. Therefore there is a need to discretize the original problem and to determine a more intuitive initial solution.
2. The dimension of the grid used to discretize the sensor field. The greater the dimension of the grid used the closer the solution found from our heuristic

will be to a continuous solution. However an increase in the dimension results in a network with a larger size and order and consequently more computational time is required to determine a solution.

3. The number of switches used within the optimal control phase. Not only does a larger number of switches result in an increase in computational time but it also introduces more local minima's into the continuous solution space. This therefore increases the likelihood that an initial solution will become trapped in a local minimum and not gravitate towards the global minimum solution. It was demonstrated that by reducing the amount of switches used to a minimum, one could substantially diminish the amount of computational time needed to determine a solution with only a very small reduction in the quality of the solution obtained.

References

1. Ahuja, R.L., Magnanti, J.L., Orlin, J.B.: Network Flows: Theory, algorithms and applications. Prentice-Hall, Englewood Cliffs (1993)
2. Aneja, Y.P., Aggarwal, V., Nair, K.P.K.: Shortest chain subject to side constraints. Networks 13(2), 295–302 (1983)
3. Beasley, J.E., Christofides, N.: An algorithm for the resource constrained shortest path Problem. Networks 19(4), 379–394 (1989)
4. Boland, N., Dethridge, J., Dumitrescu, I.: Accelerated label setting algorithms for the elementary resource constrained shortest path problem. Operations Research Letters 34, 379–394 (1989)
5. Caccetta, L.: Branch and cut methods for mixed integer programming problems. Progess in Optimization II. Applied Optimization Series, vol. 39, pp. 21–44. Kluwer Publishers, Dordrecht (2000)
6. Caccetta, L., Loosen, I., Rehbock, V.: Optimal transit path problem for submarines. Modelling Supplementary Volume of the DCDIS, Series B, Applications & Algorithms, pp. 874–880 (2005)
7. Carlyle, W.M., Wood, R.K.: Lagrangian relaxation and enumeration for solving constrained shortest-path problems. In: Proceedings of the 38th Annual ORSNZ Conference, University of Waikato, pp. 3–12 (2003)
8. Carlyle, W.M., Royset, J.O., Wood, R.K.: Routing military aircraft with a constrained shortest-path algorithm (in review, 2007)
9. Chan, Y.K., Foddy, M.: Real time optimal flight path generation by storage of massive data bases. In: Proceedings of the IEEE NEACON 1985, Institute of Electrical and Electronics Engineerings, New York, pp. 516–521 (1985)
10. Chaudhry, A., Misovec, K., D'Andrea, R.: Low observability path planning for an unmanned air vehicle using mixed integer linear programming. In: 43rd IEEE Conference on Decision and Control, Paradise Island, Bahamas, December 14–17, pp. 14–17 (2004)
11. Desrochers, M., Soumis, F.: A generalized permanent labelling algorithm for the shortest path problem with time windows. INFOR 26, 191–212 (1988)
12. Dijkstra, E.W.: A note of two problems in connection with graphs. Numerische Mathematik 1, 269–271 (1959)
13. Dogan, A.: Probabilistic approach in path planning for UAVs. In: Proceedings of the 18th IEEE International Symposium on Intelligent Control, Houston, Texas, October 5–8, pp. 608–613 (2003)

14. Dumitrescu, I., Boland, N.: Improved preprocessing, labeling and scaling algorithms for the weight constrained shortest path problem. Networks 42(3), 135–153 (2003)
15. Garey, M.R., Johnson, D.S.: Computers and intractability: A guide to the theory of NP-completeness. W.H. Freeman and Co., San Francisco (1979)
16. Gill, P.E., Murray, W., Saunders, M.A., Wright, M.H.: User's guide for NPSOL (version 4.0): A fortran package for nonlinear programming. Report SOL 86–2, Department of Operations Research, Stanford University (1998)
17. Grignon, L., Prasetio, Y., Toktas, B., Yen, J., Zabinsky, Z.: A basis space-time network formulation for aircraft rerouting with airspace closures. Technical Report, Department of Industrial Engineering, University of Washington, Seattle, Washington (2002)
18. Hallam, C.L.: Hierarchical path generation: An application to submarine transit paths. Honours Dissertation. Murdoch University, Western Australia (1997)
19. Hallam, C.L., Harrison, K.J., Ward, J.A.: A multiobjective optimal path algorithm. Digital Signal Processing 11(2), 133–143 (2001)
20. Handler, G., Zang, I.: A dual algorithm for the constrained shortest path problem. Networks 10(4), 293–310 (1980)
21. Hassin, R.: Approximation schemes for the restricted shortest path problem. Mathematics of Operations Research 17, 36–42 (1992)
22. Hebert, J., Jacques, D., Novy, M., Pachter, M.: Cooperative control of UAVs. In: AIAA Guidance, Navigation and Control Conference and Exhibit, Montreal, Canada (2001)
23. Hespanha, J.P., Kizilocak, H.H., Ateskan, Y.S.: Probalistic map building for aircraft-tracking radars. In: Proceedings of the American Control Conference, Arlington, Virginia, vol. 6, pp. 4381–4386 (2001)
24. Howlett, P., Pudney, P., Benjamin, B.: Determination of optimal driving strategies for the control of a train. In: Computational Techniques and Applications: CTAC 1991, Computational Mathematics Group, Australian Mathematical Society, pp. 241–248 (1991)
25. Howlett, P.: Optimal strategies for the control of a train. Automatica 32, 519–532 (1996)
26. ILOG CPLEX 9.0: User's Manual, ILOG, S.A (2003), `http://www.ilog.com/`
27. Inanc, T., Misovec, K., Murray, R.M.: Nonlinear trajectory generation for unmanned air vehicles with multiple radars. In: Proceedings of the 43$^{\text{rd}}$ IEEE Conference on Decision and Control, Paradise Island, Bahamas, December 14–17, pp. 3817–3822 (2004)
28. Jaumard, B., Semet, F., Vovor, T.: A two-phase resource constrained shortest path algorithm for acyclic graphs. Les Cahier du GERAD, G-96–48 (1996)
29. Jennings, L.S., Fisher, M.E., Teo, K.L., Goh, C.J.: MISER3: Solving optimal control problems - an update. Advanced Engineering Software 13, 190–196 (1991)
30. Jennings, L.S., Teo, K.L., Fisher, M.E., Goh, C.J.: MISER3 version 2, optimal control software, theory and user manual Department of Mathematics, University of Western Australia (1997), `http://cado.maths.uwa.edu.au/miser/manual.html`
31. Joksch, H.C.: The shortest route problem with constraints. Journal of Mathematical Analysis and Application 14, 191–197 (1966)
32. Jun, M., D'Andrea, R.: Path planning for unmanned aerial vehicles in uncertain and adversarial environments. Cooperative Control: Models, Applications and Algorithms, 99–110 (2003)

33. Kim, J., Hespanha, J.P.: Discrete approximation to continuous shortest-path: Application to minimum-risk path planning for groups of UAV's. In: 42nd IEEE Conference on Decision and Control, Maui, Hawaii, December 9–12 (2003)
34. Latourell, J.L., Wallet, B.C., Copeland, B.: Genetic algorithm to solve constrained routing problems with applications for cruise missile routing. In: Proceedings of SPIE, Applications and Science of Computational Intelligence, vol. 3390, pp. 490–500 (1998)
35. Lakshmanan, R., Stephanopoulos, G.: Synthesis of operating procedures for complete chemical plants-II: A nonlinear planning methodology. Computer and Chemical Engineering 12, 1003–1021 (1988)
36. Leary, J.: Search for a stealthy flight path through a hostile radar defence network. Master's Thesis, Naval Postgraduate School, Monterey, California (1995)
37. Lee, S.H.K.: Route optimization model for strike aircraft. Master's Thesis, Naval Postgraduate School, Monterey, California (1995)
38. Lee, H.W.J., Teo, K.L., Jennings, L.S., Rehbock, V.: Control parametrization enhancing technique for time optimal control problems. Dynamic Systems and Applications 6(2), 243–262 (1997)
39. Lee, H.W.J., Teo, K.L., Rehbock, V., Jennings, L.S.: Control parametrization enhancing technique for discrete-valued control problems. Automatica 35(8), 1401–1407 (1999)
40. Lewis, D.H.: Optimal three-dimensional path planning using visibility constaints. Master's Thesis, Naval Postgraduate School, Monterey, California (1988)
41. Loosen, I.: Optimization techniques for constrained path problems. Ph.D Thesis, Department of Mathematics and Statistiics, Curtin University of Technology (2007)
42. Lorenz, D.H., Raz, D.: A simple efficient approximation scheme for the restricted shortest path problem. Operations Research Letters 28, 213–219 (2001)
43. McFarland, M.B., Acjeru, R.A., Taylor, B.L.: Motion planning for reduced observability of autonomous aerial vehicles. In: IEEE International Conference on Control Applications (1999)
44. Milam, M.: Optimal trajectory generation for constrained dynamical systems. Ph.D Thesis, Control and Dynamical Systems, Caltech (2003)
45. Misovec, K., Inanc, T., Wohletz, J., Murray, R.M.: Low-observable nonlinear trajectory generation for unmanned air vehicles. In: 42nd IEEE Conference on Decision and Control, Maui, Hawaii, December 9–12 (2003)
46. Murphey, R., Zabarankin, M., Uryasev, S.: Trajectory optimization in a threat environment. Research Report 2003–9, Department of Industrial and System Engineering, University of Florida, Gainesville, Florida (2003)
47. Norsell, M.: Radar cross section constraints in flight path optimization. In: 41st Aerospace Sciences Meeting and Exhibit, Reno, Nevada, January 6–9 (2003)
48. Novy, M.C., Jacques, D.R., Pachter, M.: Air vehicle optimal trajectories between two radars. In: Proceedings of the American Control Conference, Anchorage, Alaska, May 8–10 (2002)
49. Pachter, M., Hebert, F.: Optimal aircraft trajectories for radar exposure minimization. In: Proceedings of the American Control Conference, Arlington, Virginia, June 25–27 (2001)
50. Pfeiffer, B., Batta, R., Klamroth, K., Nagi, R.: Path planning for UAVs in the presence of threat zones using probabilistic modeling (in review, 2005)
51. Rehbock, V., Teo, K.L., Jennings, L.S., Lee, H.W.J.: A survey of the control parametrization and control parametrization enhancing methods for constrained optimal control problems. In: Progress in Optimization: Contributions from Australia, pp. 247–275. Kluwer Academic Press, Dordrecht (1999)

52. Rehbock, V., Caccetta, L., Hallam, C.L., O'Dowd, R.: Optimal submarine transit paths through sonar fields. Research Report, Department of Mathematics and Statistics, Curtin University of Technology, Western Australia (2000)
53. Rehbock, V., Caccetta, L.: Two defence applications involving discrete valued optimal control. Australian & New Zealand Industrial and Applied Mathematics Journal 44, 33–54 (2002)
54. Ribeiro, C., Minoux, M.: Solving hard constrained shortest path problems by lagrangian relaxation and branch-and-bound algorithms. Methods of Operations Research 53, 303–316 (1986)
55. Savelsbergh, M.W.P.: Preprocessing and probing techniques for mixed integer programming problems. ORSA Journal on Computing 6(4), 445–454 (1994)
56. Schittkowski, K.: NLPQL: A Fortran subroutine for solving constrained nonlinear programming problems. Annals of Operations Research 5, 485–500 (1985/1986)
57. Shapiro, J.F.: A survey of lagrangian techniques for discrete optimization. Annals of Discrete Mathematics 5, 113–138 (1979)
58. Teo, K.L., Goh, C.J., Wong, K.H.: A unified computational approach to optimal control problems. Longman Scientific and Technical, London (1991)
59. Vian, J.L., Moore, J.R.: Trajectory optimization with risk minimization for military aircraft. AIAA Journal of Guidance, Control and Dynamics 12(3), 311–317 (1989)
60. Yen, J.Y.: Finding the k-shortest, loopless paths in a network. Management Science 17, 711–715 (1971)
61. Zabarankin, M., Uryasev, S., Pardalos, P.: Optimal risk path algorithms. In: Murphey, R., Pardalos, P. (eds.) Cooperative Control and Optimization, vol. 66, pp. 271–303. Kluwer Academic, Dordrecht (2002)
62. Zabarankin, M., Uryasev, S., Murphey, R.: Aircraft routing under the risk of detection. Naval Research Logistics 53, 728–747 (2006)

Minimum Time Multi-UGV Surveillance[*]

David Anisi[1] and Petter Ögren[2]

[1] Optimization and Systems Theory,
Royal Institute of Technology (KTH),
SE-100 44 Stockholm, Sweden
`anisi@kth.se`

[2] Department of Autonomous Systems,
Swedish Defence Research Institute (FOI),
SE-164 90 Stockholm, Sweden
`petter.ogren@foi.se`

Abstract. This chapter addresses the problem of concurrent task and path planning for a number of surveillance Unmanned Ground Vehicles (UGVs) such that a user defined area of interest is covered by the UGVs' sensors in minimum time.

We first formulate the problem, and show that it is in fact a generalization of the Multiple Traveling Salesmen Problem (MTSP), which is known to be $\mathcal{NP}$-hard. We then propose a solution that decomposes the problem into three subproblems. The first is to find a maximal convex covering of the search area. Most results on static coverage use disjoint partitions of the search area, *e.g.*, triangulation, to convert the continuous sensor positioning problem into a discrete one. However, by a simple example, we show that a highly overlapping set of maximal convex sets is better suited for minimum time coverage.

The second subproblem is a combinatorial assignment and ordering of the sets in the cover. Since the Tabu search algorithm is known to perform well on various routing problems, we use it as a part of our proposed solution.

Finally, the third subproblem utilizes a particular shortest path sub-routine in order to find the vehicle paths, and calculate the overall objective function used in the Tabu search. The proposed algorithm is illustrated by a number of simulation examples.

1 Introduction

Surveillance is an application area that has received an increasing amount of attention over the last decades. In civilian as well as military applications, automated solutions ranging from security cameras to surveillance UGVs are used in increasing numbers. It is therefore not surprising that the research area of automated positioning and control of surveillance sensors is also active and growing.

[*] Funded by the Swedish defence materiel administration (FMV) and the Swedish armed forces through the Technologies for Autonomous and Intelligent Systems (TAIS) project. 297316-LB704859.

In this chapter we investigate how small scale UGVs, such as the one depicted in Figure 1, can be used in surveillance and security applications.

For the purpose of this chapter, we divide the rich set of work in this field into the following three categories: First moving sensor platforms where the main limitation on field of view is the physical sensor range. Applications where such a formulation is reasonable include demining, vacuum cleaning, UAV-search and outdoor pursuit-evasion games. The second category consists of problems dealing with positioning of static sensors where occluding objects and walls present the main limitation on field of view. Such formulations are found in the so called *Art Gallery Problems*, where the number of guards required to monitor a building is to be minimized. The third category consists of problems where moving sensors are to cover an area where again, occluding objects and walls present the main limitation of field of view. This last category includes applications such as pursuit-evasion games or exploration and mapping, in urban or indoor environments. We will now discuss each of these categories in more detail.

In the first category, where sensor range is the main limitation on the extension of the visible area we find problems such as vacuum cleaning and demining, [2, 13], general coverage [6, 14, 18], multi robot coverage [11] and some robotic security applications [5]. Furthermore, a number of UAV surveillance papers, such as [1, 19, 22] fall into this category. The last set of papers also consider the combined problems of ordering a set of surveillance areas, and planning the search sweep of each individual area. The pursuit-evasion problem, where a number of pursuers try to find an evader is sometimes also formulated in this way [15].

In the second category, the field of view of stationary sensors is limited by occluding objects instead of physical sensor range. This corresponds to indoor or urban environments, where the distance between *e.g.*, walls, is in general smaller than the range of the sensor, *e.g.*, a camera or a laser scanner. The first group of results in this category comes from combinatorial geometry, and addresses Art

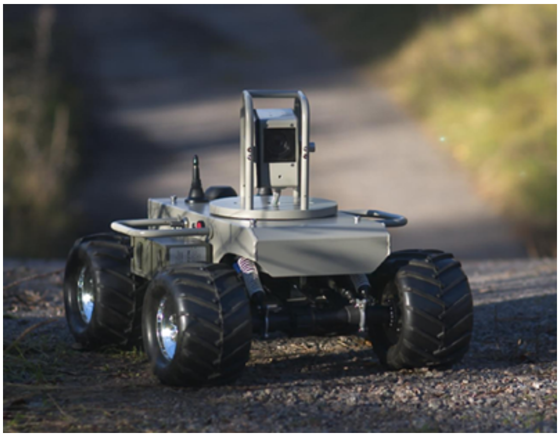


Fig. 1. This Surveillance UGV testbed, developed by SAAB Aerotech, will be used in real-world experiments

Gallery Problems, see *e.g.*, [3, 16] and the excellent survey in [26]. This work has then been built upon in [9] where a feedback solution to the guard positioning was proposed.

In the third category, where the field of view of moving guards is mainly limited by occluding walls and other objects, we also find results building on the Art Gallery work. In an indoor environment, the pursuit-evasion problem can be solved with a guarantee that the evader will be caught. Such results are found in [8, 12]. Some of this work also deals with the situation where the area is unknown. These problems are sometimes referred to as exploration and mapping, and examples include [4, 27].

Some papers address coverage problems that do not fall into one of the above categories. Examples include [7], where the mean squared distance from a sensor to a random event is minimized, and [24], where both sensor range and occlusions are incorporated into a combined planning of both UAV and sensor movements.

In the first category, many papers study the problem of covering an area in minimum time. However, when occlusions and not sensor range are the main limitations to field of view, as in the third category, we have found no paper addressing the minimum time coverage problem. In this chapter we formulate such a problem and propose an algorithm to solve it.

The organization of this chapter is as follows. In Section 2 some concepts and results from combinatorial geometry and multi vehicle routing problems is given. Then, in Section 3, we state our problem and propose a solution in Section 4. Simulations illustrating the approach are presented in Section 5. Finally, the chapter is concluded in Section 6.

2 Theoretical Background

In this section we review some tools from combinatorial geometry and combinatorial optimization that will be used in the rest of the chapter.

2.1 Art Gallery Problems

We start by reviewing some terminology from combinatorial geometry. In this section we closely follow the approach of Urrutia [26], but add the sensor range R into some of the definitions.

A *polygon* Q in the plane is an ordered sequence of points $q_1, \dots, q_n \in \mathbb{R}^2$, $n \geq 3$, called vertices of Q together with the line segments q_i to q_{i+1} for $i = 1, \dots, n-1$ and q_n to q_1, called edges. In the following we assume that none of these edges intersect. A polygon is called *orthogonal* if adjacent edges are orthogonal.

Given a polygon Q and and a set of m disjoint polygons $Q_1, \dots, Q_m$ contained in Q we call the set $A = Q \setminus \{Q_1 \cup ... \cup Q_m\}$ a *polygon with m holes*. A polygon with holes is called *orthogonal* if every pair of edges are either orthogonal or parallel. The areas to be searched in this chapter, denoted A, are all going to be of this form. An orthogonal polygon with holes is depicted in Figure 2(a).

Given two points p and q in A we say that p is *visible* from q if the line segment joining p and q is totally contained in A, and $||p - q|| \leq R$, where R is

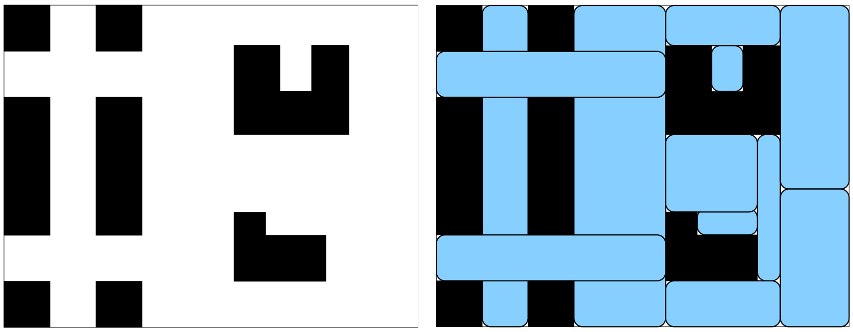

(a) A polygon with holes (b) Convex Cover of the polygon.

Fig. 2.

the maximal sensor range. A set of points $H = \{h_1, \ldots, h_k\} \subset A$ *guards* A if for all $p \in A$ there exists $h_i \in H$ such that p is visible from h_i.

In order to define our minimum time coverage problem we make the following definitions. A *path* P is an ordered set of points $P = p_1, \ldots, p_n$. A *convex cover* C of A is a set of convex sets $C = \{c_i\}$, such that $A \subseteq \cup_i c_i$. A convex cover C is *visited* by the path $P = \{p_1, \ldots, p_n\}$ if $\forall c_i \in C \ \exists p_j \in P : \ p_j \in c_i$. We define a *maximal* convex cover of A to be a convex cover $C = \{c_i\}$ of A, such that[1] $|c_i| \leq R$, and for all i, there is no convex set $s \subseteq A$, $|s| \leq R$ such that $s \supset c_i$.

An example of a convex cover can be found in Figure 2(b). Note that the four leftmost sets are maximal and overlap, while the sets to the right are disjoint and not maximal. Hence the depicted cover is not maximal. Below we will argue that the overlapping in a maximal cover is preferable in minimum time coverage applications.

Given the above definitions we can state the following lemma before we define the main problem addressed.

Lemma 1. *If there exists a convex cover C of A such that the path P visits C, then P guards A*

Proof. Since P visits C, and every set c_i in C is convex, P guards every set c_i. Furthermore, since $A \subseteq \cup_i c_i$, P guards A. ■

3 Problem Formulation

In this section we first informally state the Minimum Time UGV Surveillance Problem (MTUSP) and then show that it is $\mathcal{NP}$-hard.

Informally, the problem we are studying is the following: Given a set of surveillance UGVs and a user defined area to be covered, find paths such that every point of the area can be seen from a point on a path and such that the time for executing the search in parallel is minimized.

[1] Here, $|s| = \sup_{a,b \in s} \text{dist}(a, b)$ denotes the diameter of the set s.

Problem 1 (Minimum Time UGV Surveillance Problem). *Given N vehicles and an area A, find a set of waypoint paths $P = \{P_1, \ldots, P_N\}$ that solve the following optimization problem*

$$\min_P \max_i \sum_{k=1}^{n_i-1} ||p_{ik} - p_{i(k+1)}||$$
$$s.t. \quad \cup_i P_i \text{ guards } A$$

Here $P_i = \{p_{i1}, \ldots, p_{in_i}\}$ and the start and finish depots, denoted by $p_{i1}, p_{in_i}, i \in \mathcal{Z}_N^+ \triangleq \{1, \ldots, N\}$ may be given.

An example solution to a MTUSP can be found in Figure 3.

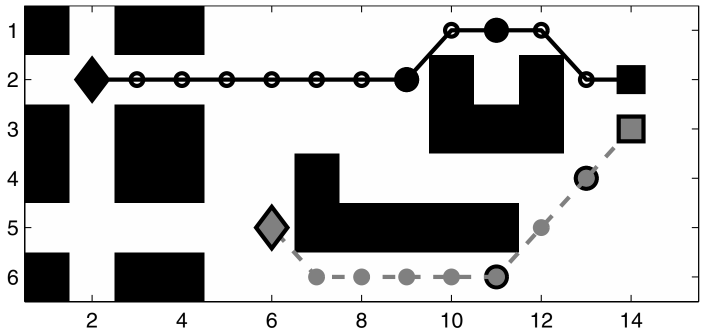


Fig. 3. An approximate solution to the Minimum Time UGV Surveillance Problem (MTUSP) involving two UGVs on the area in Figure 2. Note that all the obstacle free area can be seen from some point on the UGV paths. Details on this and other simulations can be found in Section 5.

Remark 1 (Sensor field of view). In the problem statement above we demand that each point in A is visible from some point in P. This is reasonable in the case of omni-directional sensors. It is however also relevant in the case of cameras mounted on pan-tilt units. In these cases the time right before and after passing p_{ik} must be used to cover the areas visible from p_{ik}. If necessary, the UGVs will have to slow down to facilitate the sensor coverage. A similar argument can be made for the case when the sensor is one or more laser scanners.

Remark 2 (Variations). Throughout this chapter we are focusing on the minimum time coverage problem. However, other closely related problems can also be addressed using the same approach. For instance when surveillance of some particular region has higher priority, or when battery power is a scarce resource and the user wishes to minimize the overall distance traveled by the UGVs. A third option is to make the UGVs avoid high threat areas. All these variations can be incorporated into the solution algorithm presented below by simply varying the considered objective function and edge costs.

We end this section with showing that the problem defined above is $\mathcal{NP}$-hard.

Proposition 1. *Problem 1 (MTUSP) is $\mathcal{NP}$-hard.*

Proof. The proof will built upon a polynomial reduction from an *arbitrary* instance of a well-known $\mathcal{NP}$-hard problem, namely the Euclidean-TSP (ETSP) to a *special* instance of MTUSP[2].

Given an ETSP instance, $(n, [d_{ij}])$, where n is the number of cities to be visited and $[d_{ij}]$ denotes the inter-city distances, we are free to choose the following parts of Problem 1 (MTUSP) such that the achieved optimal solution corresponds to that of the given ETSP.

1. The number of UGVs, N
2. The start and finish depots for all UGVs, $p_1^i, p_{n+1}^i, i \in \mathcal{Z}_N^+$
3. The obstacle configuration
4. The area to be surveyed, A
5. The maximal sensor range, R

Regarding the number of UGVs, $N = 1$ is a natural selection. In order to achieve a tour for this single UGV, we may locate the start and finish depot, p_1^1, p_{n+1}^1 at an arbitrary city cite, as long as they are set equal. Further, an obstacle-free environment is chosen and the area A is taken as the union of isolated points located at the city cites. Finally, we set $R = 0$.

Due to these choices, the area A is fully guarded if and only if the UGV visits all the city locations and hence the optimal solution of this specially designed instance of the MTUSP will coincide with the optimal solution of the given ETSP. This completes the proof. ∎

Knowing that MTUSP is $\mathcal{NP}$-hard, we can not hope to solve all problem instances to optimality in reasonable time but must adopt heuristic solution methods. In fact, the comparative study [25] shows that for similar class of problems, finding globally optimal solutions (by commercial software packages like CPLEX) is not a viable approach.

4 Proposed Solution

In this section we will propose a solution to the MTUSP described above. The solution encompasses three subproblems, as illustrated in Figure 4. In the first subproblem, the computationally intractable problem of finding the minimum time paths that enable *complete* regional surveillance, is turned into a finite dimensional combinatorial optimization problem. This is achieved by finding a *maximal convex cover* of A, as defined in Section 2. In the second subproblem, the order in which to visit the sets in the cover is determined using Tabu search. The third subproblem, which is called as a subroutine of the second one to evaluate the objective function in the Tabu search, involves a shortest path problem on a graph, constructed from the given visitation order.

[2] Consult [10, 17, 23] to read more about showing $\mathcal{NP}$-hardness through reduction.

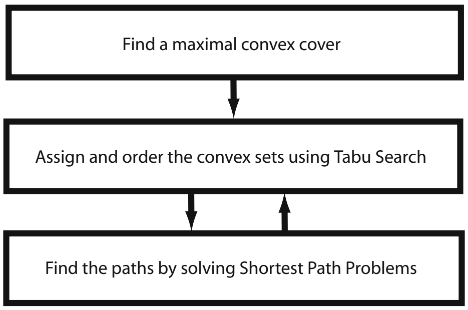

Find a maximal convex cover
Assign and order the convex sets using Tabu Search
Find the paths by solving Shortest Path Problems

Fig. 4. The proposed solution relies on decomposing the problem into three subproblems

Formally we state the algorithm below.

Algorithm 1 (Proposed solution). *The algorithm consists of the following two steps:*

1. *Create a maximal convex cover $C = \{c_1, \ldots, c_M\}$ of A in accordance with Algorithm 2.*
2. *Solve the following combined assignment and ordering problem using Tabu search:*
$$\min_{\pi} F(\pi) = \alpha \max_{i} f_i(\pi) + (1 - \alpha)\Sigma_i f_i(\pi) \tag{1}$$
where π is a permutation of $\mathcal{Z}^+_{M+N}$, representing the assignment/ordering of the M convex sets to the N vehicles, $\alpha \in [0\ 1]$, and $f_i(\pi)$ is the optimal path length of UGV i given the constraints in π. The value of $f_i(\pi)$ is found in a sub-routine by using a shortest path formulation to solve the following optimization problem:

$$\begin{aligned} f_i(\pi) = \min\nolimits_{P_i} \ & \Sigma_k ||p_{ik} - p_{i(k+1)}|| \\ s.t. \ & P_i \ \textit{guards} \ \cup_{I_i^\pi} c_j \\ & P_i \ \textit{visits} \ c_{I_i^\pi(j)} \ \textit{before} \ c_{I_i^\pi(j+1)}, j \in \mathcal{Z}^+_{|I_i^\pi|-1} \end{aligned} \tag{2}$$

Here, I_i^π is the index of the sets in C that are assigned to UGV i in the minimization of F. In (1), $\alpha = 1$ corresponds to the minimum time problem and $\alpha = 0$ corresponds to the minimum distance problem. Examples of both these options are found in Figures 10 and 11 in Section 5.

Having stated Algorithm 1 we first note that it does indeed result in a complete covering of the surveillance area A, *i.e.*, it produces a feasible solution to Problem 1. This is clear from the following three observations in conjunction with Lemma 1:

1. A convex cover is created.
2. All sets are assigned to different UGVs in (1).
3. Paths visiting all assigned sets are created in (2).

It is now time to describe and motivate the different subproblems in detail. This will be done in Section 4.1 through 4.3. But first we make the following assumption.

Assumption 1. *Throughout the rest of this chapter we assume that the area A is orthogonal, see Section 2. This assumption is due to the nature of Algorithm 2. Finding a maximal convex cover for a non-orthogonal environment is however not a hard problem and it should be noted that the rest of the solution in Algorithm 1 can handle any general polygon-with-holes type of environment.*

4.1 Finding a Maximal Convex Cover

Since the polygons are all orthogonal, one can see that the maximal convex sets c_i must be rectangles aligned with the polygon. With this fact in mind we can apply the following procedure to find a maximal convex cover.

Algorithm 2 (Maximal convex cover)

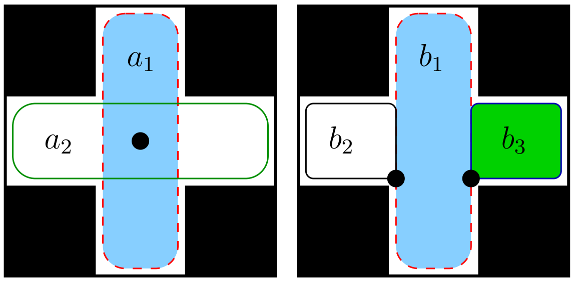
```
1. Make a discretization of the area $A$ and construct the corresponding graph
   representation, $\mathcal{G}(A)$. Since $A$ is orthogonal, a variable sized grid can be
   created with grid boundaries intersecting all points in the polygon $Q$ and
   holes $Q_1, \ldots, Q_m$.
2. Find a yet uncovered cell, $p$.
3. Start growing a rectangle $c_i$ from $p$ until it is bounded by $|c_i| \leq R$, or the
   holes on all four sides.
4. While uncovered cells exist, goto 2.
```

When no more uncovered grid cells can be found the process terminates and A is covered, $A \subseteq \cup_i c_i$. Having described how to find a maximal convex cover in detail, we now discuss a number of related issues.

We first note that decomposing the problem into subproblems, first creating a maximal cover and then finding paths visiting that cover, might remove the optimal solution from the new set of feasible solutions. Since Problem 1 is $\mathcal{NP}$-hard however, our aim is not to solve the problem to optimality, but rather to produce high-quality solutions in reasonable time.

The second thing to notice is that one straight forward solution to the problem would be to first solve a so-called Art Gallery Problem[3] to find a small set of points guarding A, and then solve an ETSP visiting these points. This would however not be efficient, since the points are chosen to be as few as possible, not to permit short vehicle paths. For the same reason, we do not choose a cover with as few sets as possible. That option furthermore happens to be an instance of the optimal *set cover problem* which is one of Karp's 21 $\mathcal{NP}$-complete problems [20].

The benefit of a maximal convex cover is illustrated in Figure 5. If the area A is cross–shaped, as depicted in the figure, then the entire area can be *instantly* surveyed from any point in $a_1 \cap a_2$. Using disjoint orthogonal sets however, the minimum time path for visiting all the orthogonal polygons (b_1, b_2, b_3), and thereby be sure to have surveyed the entire area, is strictly larger than zero.



(a) Maximal convex cover. (b) Rectangular disjoint convex cover.

Fig. 5.

4.2 Assignment and Ordering of the Convex Sets Using Tabu Search

In this section we describe how we propose to solve the optimization problem in (1), *i.e.*,

$$\min_{\pi} F(\pi) = \alpha \max_{i} f_i(\pi) + (1-\alpha)\Sigma_i f_i(\pi).$$

Above, π is a permutation representing the assignment/ordering of the sets c_i to the UGVs, and $f_i(\pi)$ is evaluated by solving another optimization problem, as explained in Section 4.3 below.

In order to solve the assignment and ordering problems simultaneously we first give the sets and UGVs id-numbers. Assign the id numbers $1,\dots,M$ to the convex sets $c_i, i \in \mathcal{Z}_M^+$. Let furthermore the N vehicles have id numbers $M+1,\dots,M+N$. The search space for the Tabu search then consists of all permutations of the id numbers, *i.e.*, $\mathcal{Z}_{M+N}^+$. The interpretation of a sequence of id numbers is then best explained by means of the following example: Let $M=14$, $N=3$, and the final sequence be

$$\pi = (\underline{15},\ 1,\ 4,\ \underline{17},\ 10,\ 14,\ 9,\ 3,\ 8,\ \underline{16},\ 2,\ 13,\ 12,\ 7,\ 6,\ 5).$$

This corresponds to the following assignments:

Set with id numbers	assigned to UGV with id number
1 4	15
2 13 12 7 6 5	16
10 14 9 3 8	17

The details of the implementations are, apart from the evaluation of $f_i(\pi)$, identical to those presented in [21]. Hence we refer the interested reader to that paper for a detailed description. We just note that the neighborhood search is performed by pairwise interchanging components in π and the Tabu condition corresponds to requiring a minimum number of iterations before switching a particular pair again.

We now turn to see how the function $f_i(\pi)$ is evaluated in each Tabu step, and how the individual UGV paths are found.

4.3 Path Planning and Functional Evaluation by Solving Shortest Path Problems

In the Tabu step above, each UGV is assigned a number of sets c_i and an order of visitation. The problem is now to decide what part of each set to pass through, in order to make the resulting UGV path as short as possible while visiting the assigned sets in the correct order. Formally, we need to solve the optimization problem in (2) for a given assignment and ordering π, *i.e.*,

$$\begin{aligned} f_i(\pi) = \min_{P} \;& \Sigma_k ||p_{ik} - p_{i(k+1)}|| \\ \text{s.t.} \;& P_i \text{ guards } \cup_{I} \; c_j \\ & P_i \text{ visits } c_{I \;\; (j)} \text{ before } c_{I \;\; (j+1)}, j \in \mathcal{Z}^+_{|I \;\; |-1} \end{aligned}$$

Given a pair of starting and finishing positions for each vehicle[3], we construct a particular graph for each vehicle. This graph, which is termed a *Route Graph*, has the starting and finishing positions as its first and last node. As depicted in Figure 6, the intermediate nodes are extracted from the ordering π and correspond to the nodes of $\mathcal{G}(A)$ inside the convex sets c_j, $j \in I_i^\pi$. To obtain the edge costs for the Route Graph, an "all pairs shortest path problem" is solved in the graph representation of A, $\mathcal{G}(A)$. This can be done by running Dijkstra's algorithm,[23], once with each node of $\mathcal{G}(A)$ as source node.

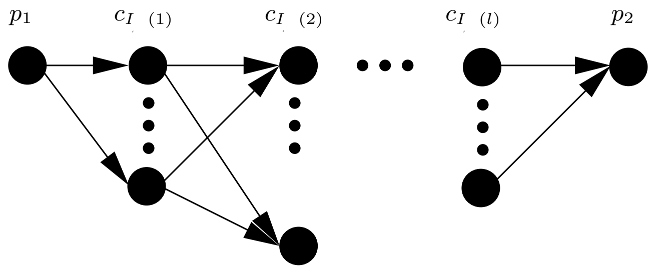


Fig. 6. A graph representation of the route of one UGV

We illustrate the Route Graph with the following example. Assume as in the table above that the UGV with id number 15 starts from some point p_1 and is assigned to visit first c_1 and then c_4 on its way to p_2. These sets and positions can be found in Figure 7.

The shortest path starting from p_1, visiting at least one node in c_1 and then visiting at least one node in c_4 and finally ending up at p_2, is plotted in the figure as well as in the Route Graph. As can be seen, the fact that c_1 and c_4 overlap makes $q_{1,4}$ coincide with $q_{4,3}$, enabling a very short path.

The evaluation of $f_i(\pi)$ corresponds to solving a shortest path problem in the Route Graph; a task for which polynomial time algorithms such as Dijkstra or

[3] For applications indifferent to finishing point, it is possible to let the optimization routine choose it freely.

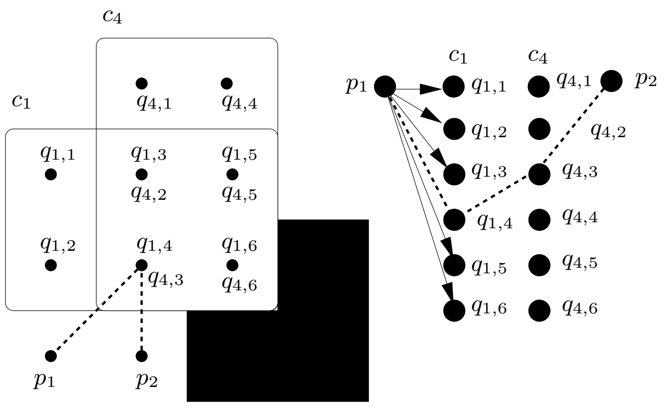


Fig. 7. Example scenario with corresponding Route Graph and optimal path (dashed). Note how the fourth node in c_1, $q_{1,4}$, coincides with the third one in c_4, $q_{4,3}$, hence the cost of the edge between them equals zero in the Route Graph.

A* exist. Note that the solution of this optimization problem yields both the UGV path P_i and its length $f_i(\pi)$.

5 Simulations

In this section, a small selection of examples are presented. The objective is to highlight some of the key characteristics of the proposed solution method. Throughout this section, the search area, A, is chosen to be all of the obstacle free space. Implemented in MATLAB, the area representation is normally taken as a random matrix with obstacle density $\rho \in \{0.3 \quad 0.7\}$. It is assumed that

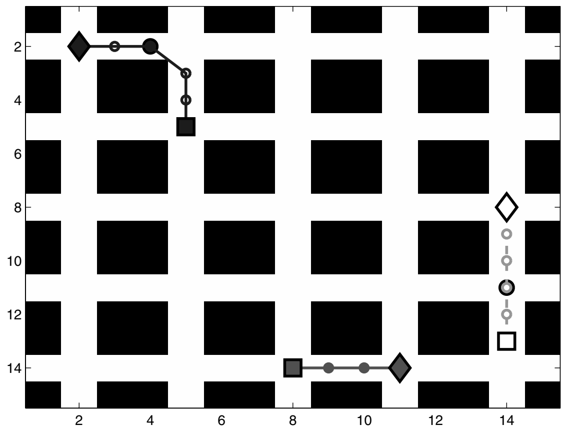


Fig. 8. The Manhattan grid is surveyed cooperatively in minimum time. The starting points of the UGVs are marked with ■.

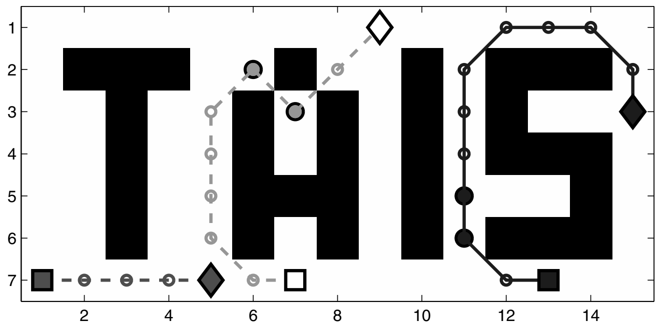
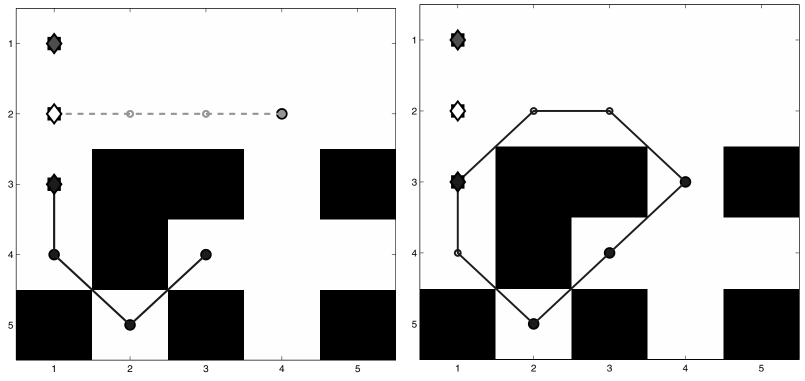


Fig. 9. Minimum time surveillance. Note how the UGVs collectively guard A.

the obstacles have been enlarged with the diameter of the vehicle so that paths passing between two obstacles do not imply collision. Furthermore, the initial position of the vehicles are marked with a square (■), while the final positions are marked with a diamond (♦). These two, together with the filled larger circles represent the surveillance points for guarding A.

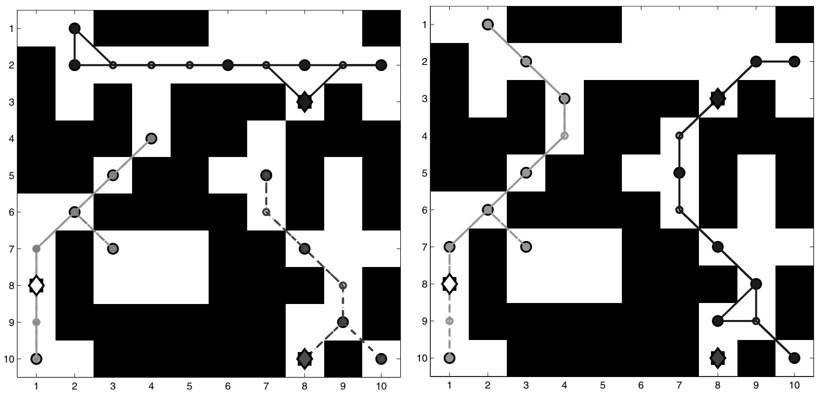
The first two simulations, found in Figures 3 and 8, illustrate the cooperative nature of the MTUSP. The final positions of the vehicles are here *free* variables to be chosen by the optimization routine. As can be seen, this extra degree of freedom is used constructively so that the vehicles survey the horizontally and vertically aligned "streets" in a cooperative manner with the common objective of minimizing the search time, *i.e.*, we have chosen $\alpha = 1$ in (1). These simulations are also a testimony of the advantage of using a highly overlapping cover.

Figure 9 illustrates a possible drawback of choosing the objective function as pure minimum time. Here, the route of the vehicle to the left, (dash-dotted), is unnecessarily long since a complete coverage would also have been achieved if the vehicle did not move at all. However, the minimum time objective has no



(a) Minimum time objective. (b) Minimum distance objective.

Fig. 10.



(a) Minimum time objective. (b) Minimum distance objective.

Fig. 11.

way of distinguishing between these two solutions and regards them as equally good since the time for executing them in parallel is indeed equal.

Figures 10 and 11 further illuminate the interplay between the choice of the objective function and the obtained solutions. In Figures 10(a) and 11(a), the solutions are once again obtained by minimizing the total search time. It can be noted that these solutions distribute the work load quite evenly over the vehicle fleet. In Figures 10(b) and 11(b) however, the objective has been set to minimize the total distance traveled by the vehicles, *i.e.*, $\alpha = 0$ in (1). Since this option does not take into consideration the division of the work load between the different vehicles, the resulting solutions often do not utilize some of the vehicles at all. This may be of interest when *e.g.*, battery power must be saved, or when unemployed vehicles can be used to perform other tasks.

6 Concluding Remarks

The Minimum Time UGV Surveillance Problem (MTUSP), where it is *occlusion*, and not sensor range, that is the main limitation to the sensors' field of view, is at the focal point of this chapter. We initially show that this problem is in fact $\mathcal{NP}$-hard, hence we cannot hope to solve all instances to optimality in reasonable time. We then proceed by proposing a decomposed solution method that encompasses finding a maximal convex cover, performing Tabu search on the assignment and ordering of the convex cover and finally, solving shortest path problems in the so called Route Graphs. The simulations demonstrate the advantage of using a maximal and highly overlapping convex cover, the cooperative nature of the MTUSP and the interplay between minimum time and minimum distance solutions.

Future research involves various interesting extensions of the current problem formulation, for instance, imposing path-wise constraints that require the induced information graph to be kept recurrently connected.

References

1. Ablavsky, V., Stouch, D., Snorrason, M.: Search Path Optimization for UAVs Using Stochastic Sampling With Abstract Pattern Descriptors. In: Proceedings of the AIAA Guidance Navigation and Control Conference, Austin, TX (August 2003)
2. Acar, E.U., Choset, H., Zhang, Y., Schervish, M.: Path Planning for Robotic Demining: Robust Sensor-based Coverage of Unstructured Environments and Probabilistic Methods. International Journal of Robotics Research 22(7), 441–466 (2003)
3. Bjorling-Sachs, I., Souvaine, D.L.: A Tight Bound for Guarding Polygons With Holes. Technical report, Report LCSR-TR-165, Lab. Comput. Sci. Res., Rutgers Univ., New Brunswick, NJ (1991)
4. Burgard, W., Moors, M., Fox, D., Simmons, R., Thrun, S.: Collaborative Multi-Robot Exploration. In: IEEE International Conference on Robotics and Automation (ICRA), vol. 1 (2000)
5. Carroll, D., Everett, H.R., Gilbreath, G., Mullens, K.: Extending Mobile Security Robots to Force Protection Missions. In: AUVSI Unmanned Systems, pp. 9–11 (2002)
6. Choset, H.: Coverage for Robotics–A Survey of Recent Results. Annals of Mathematics and Artificial Intelligence 31(1), 113–126 (2001)
7. Cortes, J., Martinez, S., Karatas, T., Bullo, F.: Coverage Control for Mobile Sensing Networks. IEEE Transactions on Robotics and Automation 20(2), 243–255 (2004)
8. Efrat, A., Guibas, L.J., Har-Peled, S., Lin, D.C., Mitchell, J.S.B., Murali, T.M.: Sweeping Simple Polygons With a Chain of Guards. In: Proceedings of the 11th ACM-SIAM Symposium on Discrete Algorithms, pp. 927–936 (2000)
9. Ganguli, A., Cortes, J., Bullo, F.: Distributed Deployment of Asynchronous Guards in Art Galleries. In: Proceedings of the 2006 American Control Conference (2006)
10. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. WH Freeman & Co., New York (1979)
11. Ge, S.S., Fua, C.: Complete Multi-Robot Coverage of Unknown Environments with Minimum Repeated Coverage. In: Proceedings of the IEEE International Conference on Robotics and Automation (2005)
12. Gerkey, B.P., Thrun, S., Gordon, G.: Visibility-Based Pursuit-Evasion with Limited Field of View. The International Journal of Robotics Research 25(4), 299–315 (2006)
13. Guo, Y., Parker, L.E., Madhavan, R.: Towards Collaborative Robots for Infrastructure Security Applications. In: Proceedings of The 2004 International Symposium on Collaborative Technologies and Systems, pp. 235–240 (2004)
14. Hazon, N., Kaminka, G.A.: Redundancy, Efficiency and Robustness in Multi-Robot Coverage. In: Proceedings of the 2005 IEEE International Conference on Robotics and Automation, pp. 747–753 (2005)
15. Hespanha, J.P., Kim, H.J., Sastry, S.: Multiple-Agent Probabilistic Pursuit-Evasion Games. In: Proceedings of the 38th IEEE Conference on Decision and Control, 1999, Phoenix, AZ, vol. 3, pp. 2432–2437 (1999)

16. Hoffmann, F., Kaufmann, M., Kriegel, K.: The art gallery theorem for polygons with holes. In: 32nd Annual Symposium on Foundations of Computer Science (San Juan, PR, 1991), pp. 39–48. IEEE Comput. Soc. Press, Los Alamitos (1991)
17. Horst, R., Pardalos, P.M., Thoai, N.V.: Introduction to Global Optimization. Kluwer Academic Pub., Dordrecht (2000)
18. Huang, W.H.: Optimal Line-Sweep-Based Decompositions for Coverage Algorithms. In: Proceedings of ICRA 2001 IEEE International Conference on Robotics and Automation, vol. 1 (2001)
19. John, M., Panton, D., White, K.: Mission planning for regional surveillance. Annals of Operations Research 108, 157–173 (2001)
20. Karp, R.M.: Reducibility among combinatorial problems. In: Complexity of computer computations, Proc. Sympos., IBM Thomas J. Watson Res. Center, Yorktown Heights, N.Y., pp. 85–103. Plenum, New York (1972)
21. Ögren, P., Wirkander, S.-L., Stefansson, A., Pelo, J.: Formulation and Solution of the UAV Paparazzi Problem. In: AIAA Guidance, Navigation, and Control Conference and Exhibit; Keystone, CO, USA, August 21–24 (2006)
22. Panton, D.M., Elbers, A.W.: Mission planning for synthetic aperture radar surveillance. Interfaces 29(2), 73–88 (1999)
23. Papadimitriou, C.H., Steiglitz, K.: Combinatorial optimization: algorithms and complexity. In: Combinatorial optimization: algorithms and complexity. Dover Publications Inc., Mineola (1998)
24. Skoglar, Nygards, Ulvklo: Concurrent Path and Sensor Planning for a UAV - Towards an Information Based Approach Incorporating Models of Environment and Sensor. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2006, vol. 1 (2006)
25. Thunberg, J., Anisi, D.A., Ögren, P.: A comparative study of task assignment and path planning methods for multi-UGV missions. In: Hirsch, M.J., Commander, C.W., Pardalos, P.M., Murphey, R. (eds.) Optimization and Cooperative Control Strategies. LNCIS. Springer, Heidelberg (2008)
26. Urrutia, J.: Art gallery and illumination problems. In: Sack, J.-R., Urrutia, J. (eds.) Handbook of computational geometry, pp. 973–1027. North-Holland Publishing Co., Amsterdam (2000)
27. Yamauchi, B.: Frontier-Based Exploration Using Multiple Robots. In: Proceedings of the second international conference on Autonomous agents, pp. 47–53 (1998)

A Distributed Network Enabled Weapon-Target Assignment for Combat Formations

N. Léchevin[1], C.A. Rabbath[2], and M. Lauzon[2]

[1] Numerica Technologies, Inc.
2459 Pie-XI N., Québec, Qc, Canada G3J 1X5
nicolas.lechevin.numerica@drdc-rddc.gc.ca
[2] Defence Research and Development Canada - Valcartier
2459 Pie-XI N., Québec, Qc, Canada G3J 1X5
{camille-alain.rabbath,marc.lauzon}@drdc-rddc.gc.ca

Abstract. A distributed decision-making capability resulting in near-optimal weapon-target assignments for formations of unmanned combat vehicles is proposed. The decision-making is based on a modified version of the cross-entropy method distributed over the formations. For the formations to agree about a single consistent target assignment, a new consensus algorithm is proposed so that exact agreement can be reached in finite time through a communications graph that is at least weakly connected. The decision-making algorithm enables the blue-team combat vehicles to engage, or visit, an ordered sequence of targets while grouping into formations of varying dimensions on their way to the sites. The additional degree of freedom in the formulation of the optimization problem allows mitigating the risks of destruction of the combat vehicles when facing hostile red units. The weapon-target assignment aims at maximizing a global utility function that expresses the overall weapons effects. Constraints on the autonomy of each formation is taken into account in the formulation of the optimization problem. The engagement dynamics is represented by means of an attrition model. It is shown through numerical simulations that the proposed weapon-target assignment outperforms, in specific cases, the solution to a travel salesman problem, where all the vehicles are grouped into a single formation. The numerical simulations also suggest that the performance of the proposed algorithm is dependent on the number of formations per blue team, and on the number of vehicles per formation.

1 Introduction

A weapon-target assignment strategy generally involves the solution to a combinatorial optimization problem. In such case, one seeks optimal weapon-target pairing ensuring destruction of all targets while minimizing damage incurred from battle and satisfying constraints on energy expenditure. Weapon-target assignment problems can usually be tackled by means of appropriate relaxation techniques or heuristics. Techniques that have been investigated include stochastic programming [1], genetic algorithms [2], and mixed-integer linear programming (MILP) [3]. Several methods and tools aim at distributing optimal control of fleets of unmanned aerial vehicles. Techniques include MILP [4], dynamic programming [5], and game-theoretic algorithms such as fictitious play [6], to name a few. A vast body of research has been devoted to the domain of air operations, which is a field closely related to weapon-target assignment. The decision making is naturally addressed within a game-theoretic framework. For instance, probabilistic attrition-type, discrete-time modeling of two opposing forces engaged in a military air operation is utilized in [7,8,9] to derive one-step and two-step lookahead Nash policies. In [10], spatial distribution of air campaign resources in the battlefield is addressed by means of minimax strategies. Also related to target assignment is the vehicle routing problem, where a multi-vehicle dynamic traveling repair-person problem is solved by means of decentralized algorithms [11,12]. The decentralized target assignment strategy proposed in [12] is locally optimal when the time intensity of the Poisson process, which models target occurrence, approaches either zero or infinity.

A Network Enabled Weapon-Target Assignment Scheme (NEWTAS) is proposed to control, in a distributed manner, multiple formations of Unmanned Combat Vehicles (UCVs). Assuming the vehicles share information through wireless communication links, NEWTAS is aimed at aiding ground crew in optimally assigning targets to weapons for arbitrarily large teams of UCVs. NEWTAS assigns a sequence of targets to every blue-team formation. The sequence is obtained by minimizing a global utility function that incorporates the attrition entailed by hostile red forces and constrains the total energy expenditure of the UCVs. The set of tactical targets is arranged as an ordered sequence of targets to be engaged or visited, depending on context, by any given formation. The pairing of a formation with a sequence of tactical targets constitutes the solution to the so-called multirouting problem. NEWTAS is tailored to the following scenario. A combat theater consists of multiple formations of UCVs, denoted as the blue team, facing tactical targets and adversarial ground units (red team) located randomly. Blue-team hierarchical decision-making architecture comprises several modules, as depicted in Figure 1. NEWTAS sits at the highest level of the decision-making. At the lower level of the decision-making, the Path and Munitions Planning (PMP) functions are obtained by solving a two-player, stochastic game [13]. Such scheme enables deployment of the blue team formations from one target to the next in a near-optimal manner, despite the presence of ground units. Allowing the formations to merge into large groups and to divide into small teams is the key capability enabling maximal survivability of the combat vehicles, which evolve in a partially known, adversarial environment [13,15,16]. The decision-making system composed of NEWTAS and PMP is to be contrasted with the approach in [7,8] where limited horizon stochastic games are solved sequentially to decrease the computational load. With the proposed NEWTAS, the weapon-target assignment problem is solved over the entire horizon. Furthermore, as opposed to techniques applied to attrition-model-based air operations [7,8,14], NEWTAS is distributed. Hence, the networked formations are capable of executing NEWTAS online such as, for instance, when significant events render a weapon-target assignment obsolete and require re-assigning targets to UCVs.

This chapter focuses on the highest level of the decision-making hierarchy by proposing a multirouting solution to the weapon-target assignment problem. Assigning an ordered sequence of targets to be engaged, or visited, by the UCV formations constitutes the so-called multirouting solution provided by NEWTAS. The states of the formations are governed by an attrition model [7,8]. This model is a function of the number of healthy blue-team UCVs and of the number of ground units faced during the mission. The proposed NEWTAS is characterized by a Cross-Entropy-Method-Based [17] Algorithm (CEMBA) distributed over the networked formations. CEMBA offers several attractive features when applied to the weapon-target assignment problem considered in this chapter. First, stochasticity of CEMBA allows reaching optimality in probability with a trade-off between the maximum reachable value of the global utility and the size of the sample sets. Fixing the size of the sample set allows one to constrain the computation time of the near-optimal assignment. Second, the Markov-chain-based algorithm complies with the objective of distributing NEWTAS over a network of formations. To ensure consistency of the state variables that are communicated among distributed computing units, CEMBA is coupled to a novel consensus algorithm reaching an exact agreement in finite time. Case studies and numerical simulations illustrate the effectiveness of the proposed NEWTAS. In particular, it is shown that the weapon-target assignments obtained with NEWTAS can lead to improved performances in terms of reduced energy expenditure and minimization of the number of UCVs lost, due to red force engagements, when compared to the sequencing and routing of a single, large formation of blue-team UCVs obtained with the solution to a classical Traveling Salesman Problem (TSP). However, performance of NEWTAS is highly dependent upon the attrition model and on the value of the following parameters: the number of formations per blue team, and the number of vehicles per formation. Indeed, numerical simulations suggest that NEWTAS applied to a small number of large formations, for example 4 formations of 16 vehicles, results in a performance superior to that obtained with NEWTAS on a relatively large number of small formations, such as 8 formations of 8 vehicles.

2 Nomenclature

B	Blue team
B^-	$\{1, ..., \varphi\}$
$d_{i,k'}$	Matrix exchanged among neighboring formations (for consensus)
E_p	Expectation computed from P
F_i	ith formation of B, $(i = 1, ..., \varphi)$
$f(X, P)$	Probability of generating X
$\mathcal{G}_c$	Communication graph of B
h_s, h_f	Step sizes (slow and fast rates)
I	Indicator function
$M_{1,i,k'} M_{2,i,k'}$	Internal memory of F_i (consensus algorithm)

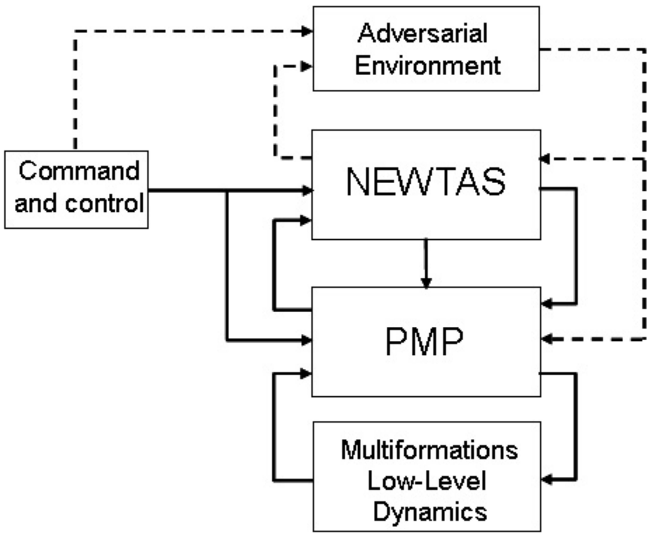


Fig. 1. Overview of the decision-making architecture

m_φ	Number of munitions per vehicle
N_i	Set of neighbors of F_i
$\widehat{n}_{i,k'}$	Matrix exchanged among neighboring formations (for consensus)
n	Number of targets plus B's base
n_φ	Number of vehicles per formation
P	$[P_1, ..., P_\varphi]$
P_i	Transition matrix of the Markov chain associated with F_i
Pr_P	Probability computed from P
$P_{i,j}$	Engagement factors
$p^i_{jj'}$	$(j, j\prime)$-th entry of P_i
$Q_{i,j}$	Attrition factors
$q_{i,j}$	Probability of destruction of a vehicle of F_i
R	Red team
$r_j(l+1)$	Number of ground units located between $X_{i,l}$ and $X_{i,l+1}$
S	Set of targets augmented with B's base
S^-	$\{1, ..., n\}$
S^*	Set of targets
$S_{d,i}$	Length of the path followed by F_i
S_d	Sum of $S_{d,i}$ over B
T_j	Targets if $j > 1$, blue team's base if $j = 1$
$t_{s,k}, t_{f,k}$	Discrete time instants (slow and fast rates)
U	Blue team's global utility
u^i_j	Lagrangian multipliers
$u^j_{1,i,k'}$, $u^j_{2,i,k'}$	Input variables of F_i (consensus algorithm)
V_j	Value of T_j

X	Set of paths followed by all the formations in B
X_i	Path followed by F_i
$X_{i,g}$	Waypoint in X_i
X_k^s	sth entry of the sample set of paths obtained at $t_{s,k}$
$\mathcal{X}$	Set of allowed paths for B
$\widetilde{\mathcal{X}}$	Unconstrained $\mathcal{X}$ with respect to the number of target visits
$\mathcal{X}_i$	Set of allowed paths for F_i ($\mathcal{X}$)
$\widetilde{\mathcal{X}}_i$	Set of allowed paths for F_i ($\widetilde{\mathcal{X}}$)
$\widetilde{\mathcal{X}}^i_{jj'}(r)$	Set of routes in $\widetilde{\mathcal{X}}_i$ whose rth transition occurs along $(T_j, T_{j'})$
$\widetilde{\mathcal{X}}^i_{j'}(r)$	Set of all routes in $\widetilde{\mathcal{X}}_i$ whose rth transition starts from T_j
$y_{1,j,k'}, y_{2,j,k'}$	Output variables of F_i (consensus algorithm)
$z_{i,j}(l)$	Number of F_i's healthy vehicles between $X_{i,l}$ and $X_{i,l+1}$
$\beta_{i,j}$	Probability acquisition of a vehicle of F_i by a ground unit
$\mu_{i,j}$	Normalizing factor in the attrition model
φ	Number of formations in blue team

3 Weapon-Target Assignment Formulated as a Vehicle Multirouting Problem

Assigning targets to weapons and planning the routes of UCVs may be solved prior to mission. Knowing that limited communication capacities are available prior to and during mission, it becomes clear that the global optimization problem may become infeasible in finite time. Thus, we propose the two-level hierarchical decision-making system shown in Figure 1. First, information on the tactical target locations, and on the ground unit types and locations is used to derive an ordered sequence of targets to be engaged, or visited, by the UCV formations, as illustrated in Figures 1 and 2. This is the so-called vehicle multirouting problem, which is solved with the proposed NEWTAS. Second, the formations path planning and the deployment of munitions, labeled as the PMP function in Figure 1 and illustrated in Figure 3, is achieved by means of a one-step lookahead rollout policy [13,16].

Several definitions and assumptions are presented before stating the NEWTAS objective.

Definition 1 (*Red team*). Team R is made up of $n-1$ tactical targets, labeled $T_2, ..., T_n$, which are protected by static ground units. Note that we let T_1 represent the base of the blue team, the starting point of the mission. When a blue-team formation engages a ground unit, the latter is allowed to shoot at most one munition. Blue-team potential losses caused by R are detailed in the attrition model of *Definition 3*. For the purpose of simulations, in this chapter, tactical targets and ground units are scattered randomly throughout the urban area. The ground units may be located close to tactical targets, for protection purposes, and in-between tactical targets along pathways that may be followed by the blue-team formations, as depicted in Figure 2. Let $S^* = \{T_2, ..., T_n\}$ denote the set of $n-1$ tactical targets. Let V_j and p_j be the value of target T_j, and the probability that T_j is destroyed by a single weapon, respectively.

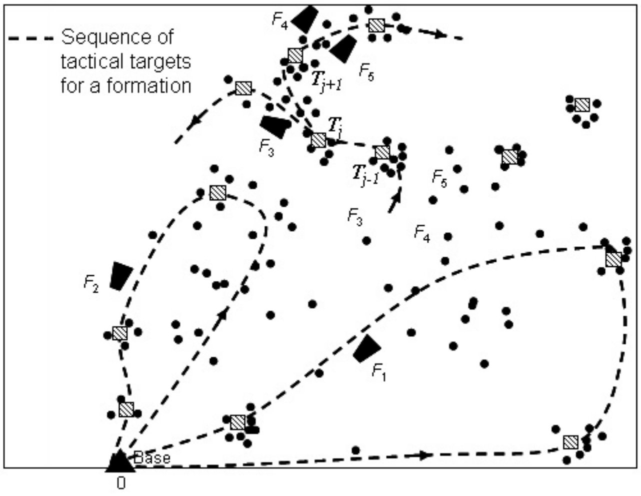


Fig. 2. Urban theater composed of $F_i$blue-team formations (black trapezoid), red-team ground units (black circles), and tactical targets (black striped squares). The sequences of tactical targets obtained with the weapon-target assignment NEWTAS are sketched for formations F_1 to F_5.

Definition 2 (Blue team). At the onset of the mission, team $B = \{F_1, ..., F_\varphi\}$ is composed of $\varphi \in \mathbb{N}$ formations. Each formation $F_i \in B$ is comprised of n_φ UCVs. A UCV carries $m_\varphi \in \mathbb{N}$ munitions. Every vehicle of a formation F_i is assumed to move with speed $v_i \in [v_m, v_M]$ along straight-line paths whose extremities are tactical targets that have to be engaged or visited. B is able to split into φ distinct formations. Conversely, grouping into large formations, and into the entire set B, is possible. Each UCV is allowed to fire at the ground units of the red team. Losses caused by B is detailed in the attrition model of *Definition 3*. B has perfect knowledge of S^*. To each formation F_i is associated an n_i-tuple $X_i = (X_{i,1}, ..., X_{i,n_i})$ of $n_i > 1$ sites that F_i has to visit. All of $X_{i,g} \in S^*$, where S^* and $g \in \{2, ..., n_i - 1\}$, are distinct. Furthermore, $X_{i,1} = X_{i,n_i} = T_1$. The path length associated with X_i is defined as $S_{d,i} = \sum_{k=1}^{n_i-1} dist(X_{i,k}, X_{i,k+1})$, where $dist(a, b)$ denotes the distance between a and b. Similarly, the total length of the φ routes that are associated with $X = \{X_1, ..., X_\varphi\}$ is defined as $S_d(X) = \sum_{i=1}^{\varphi} S_{d,i}$. Let d_i represent the maximum path length that F_i can follow. The set $\mathcal{X}$ of allowed n_i-tuple X_i is defined as $\mathcal{X} = \{X_i \in \mathcal{X}_i, i = 1, ..., \varphi \text{ s.t. } S^* \subset \cup_{i=1}^{\varphi} X_i\}$, where $\mathcal{X}_i = \{X_i = (X_{i,g})_{g=1,...,n_i} \text{ s.t. } S_{d,i} \leq d_i, X_{i,1} = T_1, X_{i,n_i} = T_1; \forall i' \neq i, \forall j \in S_i, \forall j' < j : X_{ij} \neq X_{ij'}, X_{ij} \neq X_{i'j'}\}$, where $S_i = \{1, ..., n_i - 1\}$. The following sets, $B^- = \{1, ..., \varphi\}$, $S^- = \{1, ..., n\}$, and $S = \{T_1, S^*\}$, where T_1 is the common deployment and gathering base of B, are used in Section 4.

The dynamics of engagements between B and R are modeled by the following attrition model.

Definition 3 (Attrition model). When travelling from target $X_{i,l}$ to target $X_{i,l+1}$, a formation F_i may be faced with hostile ground units. Let $r_j(l + 1)$ denote the

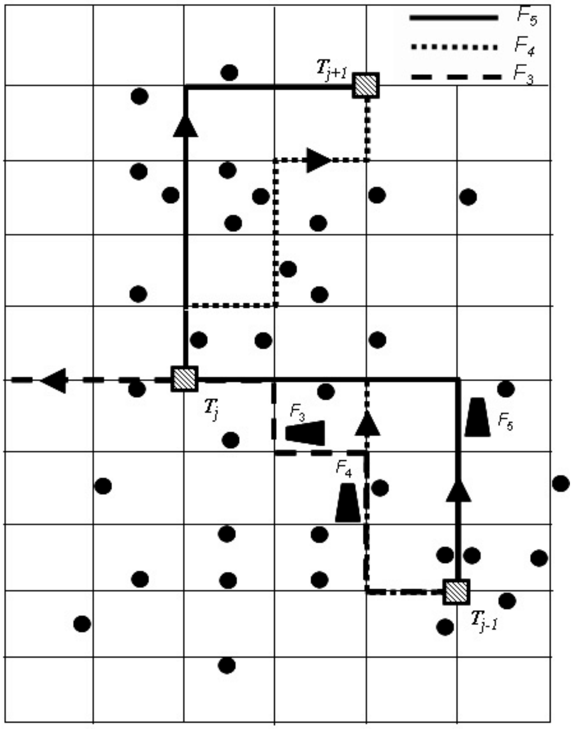


Fig. 3. Path planning obtained with PMP for formations F_3, F_4, and F_5. Formations F_4 and F_5 move from tactical target T_{j-1}, to T_j, and then to T_{j+1}. Formation F_3 goes from T_{j-1} to T_j.

number of ground units located between $X_{i,l} = T_j$ and the next target $X_{i,l+1}$. The Markov chain defined in Section 4 determines how $X_{i,l+1}$ is realized from $X_{i,l}$. Let $z_{i,j}(l)$ and $z_{i,j}(l+1)$ represent the number of F_i's healthy vehicles when F_i begins to move toward $X_{i,l+1}$ from $X_{i,l}$, and when F_i reaches $X_{i,l+1}$, respectively. A healthy vehicle is understood as being undamaged and with some level of lethality. The transition of the blue team's state, $z_{i,j}(l)$, is governed by

$$\begin{aligned} z_{i,j}(l+1) &= (1 - P_{i,j}(l+1)Q_{i,j}(l+1))z_{i,j}(l), \\ P_{i,j}(l+1) &= \beta_{i,j}(1 - \exp(-\mu_{i,j}r_j(l+1)/z_{i,j}(l))), \\ Q_{i,j}(l+1) &= 1 - (1 - q_{i,j})^{r\ (l+1)}, \end{aligned} \tag{1}$$

where $\beta_{i,j}$ is the probability that a ground unit acquires a vehicle of F_i, and $\mu_{i,j}$ is a normalizing factor [7,8]. $q_{i,j}$ is the probability that a vehicle in formation F_i is destroyed by one of the $r_j(l+1)$ ground units. $P_{i,j}$ and $Q_{i,j}$ correspond to the engagement and attrition factors, respectively. Transition from $r_j(l)$ to $r_j(l+1)$ evolves similarly to (1), where r_j is substituted for $z_{i,j}$ and vice versa. Parameters related to the dynamics in $r_j(l)$ may take different values from those related to $z_{i,j}(l)$.

Assumption 1. Let $\mathcal{E}_B$ represent the set of all pairs of formations F_i and F_j that are linked by wireless communication. The communication graph $\mathcal{G}_c = (B, \mathcal{E}_B)$ is assumed undirected and time invariant when solving the weapon-target assignment problem. During a time interval $[k'h_f, (k'+1)h_f)$, each formation F_i communicates to its neighbor a pair of matrices $\widehat{n}_{i,k'}, \widehat{d}_{i,k'} \in R^{\varphi \times \varphi}$ along with an identity number $M_{2,i,k'}$. Variables $\widehat{n}_{i,k'}, \widehat{d}_{i,k'}$, $M_{2,i,k'}$, k', and h_f are defined in Section 5.

Remarks 1. (i) Straight-line paths and attrition model, which are utilized to derive NEWTAS, constitute a coarse modeling of the actual trajectories and the low-level path-following dynamics, which are commanded by PMP. The rollout-policy-based PMP proposed in [13,16] is obtained by using a low granularity modeling of the theatre, as shown in Figure 3. (ii) Although not demonstrated in this chapter, the proposed NEWTAS can be re-executed online, during the course of the mission, whenever significant events, such as pop-up threats, render the original weapon-target assignment obsolete. In this chapter, it is assumed that NEWTAS is carried out prior to mission. (iii) The number of targets that F_i has to visit may not be equal to n. However, blue team B, as a whole, is required to visit all the tactical targets of S^* at least once during the course of the mission. (iv) While not theoretically proven, it is shown in Section 5, that the proposed algorithm displays a certain level of robustness with respect to communication loss. When communication links are temporarily lost, the communication graph becomes a time-varying, possibly intermittently unconnected, graph.

NEWTAS objective. From *Definition 1,* the value of T_j being engaged by a_j UCVs can be expressed by

$$\begin{aligned} U_j(X) &= V_j(1-(1-p_j)^a\ ), \\ a_j &= \textstyle\sum_{i=1}^{\varphi}\sum_{l=1}^{n} I(X_{i,l}=T_j) z_{i,j}(l), \end{aligned} \tag{2}$$

where I stands for the indicator function. U_j in (2) leads to the blue team's global utility

$$U(X) = \sum_{j=1}^{n} U_j(X). \tag{3}$$

Optimization Problem. From *Definitions 1* and *2*, and from *Assumption 1*, we aim to obtain a weapon-target assignment expressed by means of the set of φ routes, X^*, verifying

$$X^* = \arg\max_{X \in \mathcal{X}} U(X). \tag{4}$$

The global utility maximizer X^*, which is constrained by the autonomy d_i of F_i, $i \in B^-$, is expected to be reached by a parallelized version of a CEMBA algorithm [17] distributed over a weakly connected graph $\mathcal{G}_c$ of computing nodes.

4 Application of Cross-Entropy Method in NEWTAS

4.1 Preliminaries

Let time be discretized as $t_{s,k} = kh_s$, $k \in \mathbb{N}^+$,where h_s is the step size at which CEMBA is updated. To simplify the presentation of the algorithm it is first assumed that the communication graph $\mathcal{G}_c$ in *Assumption 1* is complete, which implies that the global information state of the graph is available to all $F_i \in B$ at any $k \in \mathbb{N}^+$. A consensus algorithm is introduced in the next section so that NEWTAS may be implemented in a distributed manner even when the communication graph is only weakly connected.

CEMBA is inspired from the algorithm dedicated to the TSP, pp. 51-53 in [17], where the probability mass function of X, which is central to the cross-entropy method applied to TSP, is modified to solve the weapon-target assignment expressed as a vehicle multirouting problem. Recall that, unlike the problem defined in (4), TSP involves either a single vehicle or a single formation of vehicles. The latter being relevant to our combat scenario. For the sake of clarity, parts of the algorithm proposed in [17] are recalled. The algorithm requires finding a Markov chain on the graph $\mathcal{G}_S = (S, \mathcal{E}_S)$, where $\mathcal{E}_S$ represents the set of edges that relate any pair of nodes in S. Such a Markov chain aims at seeking a near-optimal solution X^+ in the sense that $\{X^+ \in \mathcal{X}, U(X^+) > \gamma\}$ is a rare event for some level $\gamma > 0$; that is, $\Pr_P(U(X^+) > \gamma) = E_p[I_{\{U(X^+)>\gamma\}}]$ is very small, typically under 10^{-5}. $P = [P_1, ..., P_\varphi]$ is the concatenation of the transition matrix applied to F_i and defined in the sequel. $\Pr_P$ and E_p stand for probabilities and expectations that are computed from P, respectively. The Markov chains with one-step transition matrix

$$\begin{aligned} &P_i = (p^i_{jj'})_{j,j' \in S^-}, \forall i \in B^-, \\ &p^i_{jj} = 0, \forall j \in S^-, \end{aligned} \tag{5}$$

are first derived from an extended set $\widetilde{\mathcal{X}}$ where routes of any type are allowed. By any type we mean routes that start and end at T_1, and can visit $n-1$ sites of S^*, which are not necessarily distinct. A simple heuristic is then employed to constrain routes to the set $\mathcal{X}$ of allowed routes.

4.2 Proposed Algorithm

Let $\widetilde{\mathcal{X}} = \{\widetilde{X}_i \in \widetilde{\mathcal{X}}_i, i = 1, ..., \varphi\}$, where $\widetilde{\mathcal{X}}_i = \{\widetilde{X}_i = (X_{i,g})_{g=1,...,n}$ s.t. $X_{i,1} = T_1$, $X_{i,g} \in S^*, g \in \{2, ..., n\}\}$. To $\widetilde{X}_i \in \widetilde{\mathcal{X}}_i$ is associated the path $(\widetilde{X}_i, T_1)$ that F_i has to follow. Extension of $\widetilde{\mathcal{X}}$ to $\mathcal{X}$ comes from the fact that constraints $X_{ij} \neq X_{ij'}$ and $X_{ij} \neq X_{i'j'}$ introduced in $\mathcal{X}$ are not included in the definition of $\widetilde{\mathcal{X}}$. Let $\widetilde{U}$ be the utility defined over $\widetilde{\mathcal{X}}$ as follows

$$\widetilde{U}(X) = \begin{cases} U(X) & \text{if } X \in \mathcal{X}, \\ -\infty & \text{otherwise.} \end{cases} \tag{6}$$

The problem defined in (4) is equivalent to finding the maximizer of $\widetilde{U}(X)$ with $X \in \widetilde{\mathcal{X}}$. The probability of generating $X \in \widetilde{\mathcal{X}}$ is given by

$$\begin{aligned} f(X,P) &= \prod_{i=1}^{\varphi} f(X_i, P_i), \\ f(X_i,P_i) &= \prod_{r=1}^{n} \prod_{j=1}^{n} \prod_{j'=1}^{n} (p_{jj'}^{i})^{I(X_{\ } \in \widetilde{\mathcal{X}}_{\ \ ,}(r))}, \end{aligned} \tag{7}$$

where $\widetilde{\mathcal{X}}_{jj'}^{i}(r)$ stands for the set of routes in $\widetilde{\mathcal{X}}_i$ whose rth transition occurs along the edge $(T_j, T_{j'})$.

The cross-entropy method relies on an iterative procedure, which is designed to yield at each iteration step k a set of transition matrices $P_{1,k}, ..., P_{\varphi,k}$, concatenated in P_k, such that the Kullback-Leibler divergence between some function of P_k, denoted $f(P_k)$, and the function $f(P^+)$ gives the best likelihood ratio estimator of $\Pr_{P^+}(U(X^+) > \gamma)$ [17]. Minimization of the Kullback-Leibler divergence applied to (7) at k along with the use of Lagrangian multipliers u_j^i, which ensure that the jth row of $P_{i,k}$ sums to one [17], leads to the following maximization

$$\max_{P} \min_{u\ ,j\in S^-,i\in B^-} (E_P\ [I(U(X_k) > \gamma)) \ln f(X_k, P_k)] + \sum_{i=1}^{\varphi} \sum_{j=1}^{n} \sum_{j'=1}^{n} u_j^i (p_{k,jj'}^i - 1)), \tag{8}$$

where $X_k = \{X_{1,k}, ..., X_{1,k}\} \in \widetilde{\mathcal{X}}$ is the set of routes generated by P_k at k, $p_{k,jj'}^i$ is the (j,j')-th entry of $P_{i,k}$, and

$$\ln f(X_k, P_k) = \sum_{i=1}^{\varphi} \sum_{r=1}^{n} \sum_{j=1}^{n} \sum_{j'=1}^{n} I(X_{i,k} \in \widetilde{\mathcal{X}}_{jj'}^{i}(r)) \ln p_{k,jj'}^{i}. \tag{9}$$

Differentiating (8) with respect to $p_{k,jj'}^i$ and noting that $\sum_{j'=1}^{n} p_{k,jj'}^i = 1$ gives

$$u_j^i = -E_P\ [I(\widetilde{U}(X_k) \sum_{j'=1}^{n} \sum_{r=1}^{n} I(X_{i,k} \in \widetilde{\mathcal{X}}_{jj'}^{i}(r))]. \tag{10}$$

Substituting (10) for u_j^i in the derivative of (8) with respect to $p_{k,jj'}^i$ yields

$$p_{k,jj'}^{i} = \frac{E_P\ [I(\widetilde{U}(X_k) > \gamma) \sum_{r=1}^{n} I(X_{i,k} \in \widetilde{\mathcal{X}}_{jj'}^{i}(r))]}{E_P\ [I(\widetilde{U}(X_k) > \gamma) \sum_{r=1}^{n} I(X_{i,k} \in \widetilde{\mathcal{X}}_{j'}^{i}(r))]}, \tag{11}$$

where $\widetilde{\mathcal{X}}_{j'}^{i}(r)$ stands for the set of all routes in $\widetilde{\mathcal{X}}_i$ whose rth transition starts from node T_j. The estimator of (11) where expectations are replaced by averaging over the sample $(X_{i,k-1}^1, ..., X_{i,k-1}^N)$ is expressed as

$$p_{k,jj'}^{i} = \frac{\sum_{s=1}^{N} I(\widetilde{U}(X_{k-1}^s) > \gamma) \sum_{r=1}^{n} I(X_{i,k-1}^s \in \widetilde{\mathcal{X}}_{jj'}^{i}(r))}{\sum_{s=1}^{N} I(\widetilde{U}(X_{k-1}^s) > \gamma) \sum_{r=1}^{n} I(X_{i,k-1}^s \in \widetilde{\mathcal{X}}_{j'}^{i}(r))}, \tag{12}$$

where $X_{k-1}^s = (X_{1,k-1}^s, ..., X_{\varphi,k-1}^s)$ is obtained from P_{k-1}. P_{k-1} is computed at $k-1$ from formula in (12).

As noticed in [17], P_k defined in (12) is likely to yield few n_i-tuples, $X_{k,i}$, that belong to $\mathcal{X}_i$. Thus, relatively large sample sets are necessary to efficiently utilize the estimator in (12). To avoid using sample sets that are prohibitively large, a heuristic procedure, labeled HP(k,s), is proposed so as to increase the probability that $X^s_{i,k}$ belongs to $\mathcal{X}_i$. It consists in setting to zero appropriate columns of $P_{i,k}$ so that $X^s_{i,k}$ belongs to the set that corresponds to $\mathcal{X}_i$ without the constraint on F_i's autonomy. Omitting $S_{d,i} \leq d_i$ during the route generation process is inevitable since it is not possible to assess the length of a route until it has been fully generated.

Let $X^s_{i,g,k}$ denote the gth element of route $X^s_{i,k}$. Consider a matrix's row $Ro = [\alpha_{r1}, ..., \alpha_{rn}]$, where $\sum_{i=1}^{n} \alpha_{ri} = 1$, from which a new row, Ro', is obtained by setting some entries, $\alpha_{rj_1}, ..., \alpha_{rj}$, of Ro to zero. Then normalization of Ro' corresponds to $\overline{Ro'} = Ro' / \sum_{\alpha \notin \{\alpha_1, ..., \alpha\}} \alpha_{ri}$. HP$(k,s)$ generates the sample $X^s_k = (X^s_{1,k}, ..., X^s_{\varphi,k})$ from P_k as follows.

Step 1. (*Initialization*) Set $X^s_{1,1,k} = T_1, ..., X^s_{\varphi,1,k} = T_1$, and $Q_1 = [Q_{1,1}, ..., Q_{i,1}, ..., Q_{\varphi,1}]$, where $Q_{i,1}$ corresponds to $P_{i,k}$ whose $(1,1)$ entry is set to zero. Q_1 is normalized to $\overline{Q}_1$.
Step 2. (*lth iteration*) Let $X^s_{1,l-1,k} = T_{j_{1,\ -1}}, ..., X^s_{i,l-1,k} = T_{j_{\ ,\ -1}}, ..., X^s_{\varphi,l-1,k} = T_{j_{\varphi,\ -1}}$ be the realization of $\overline{Q}_{l-1}$ obtained at the $(l-1)$th step of the recursion. Proceed as follows.
Step.2.1. $Q_{i,l}$ is obtained from $\overline{Q}_{i,l-1}$ by setting its $(j_{1,l-1}, ..., j_{i,l-1}, ..., j_{\varphi,l-1})$-th columns to zero. $Q_{i,l}$, for all $i = 1, ...\varphi$, is then normalized to give $\overline{Q}_l$.
Step.2.2. Generate successively $X^s_{1,l,k}, ..., X^s_{\varphi,l,k}$ from $\overline{Q}_{1,l}, ..., \overline{Q}_{\varphi,l}$, respectively.
Step.2.3. If $X^s_{i,l,k} = T_1$ for $i = 1, ..., \varphi - 1$, then the first column of $\overline{Q}_{\varphi,l}$ is set to zero. The new matrix is normalized to $\overline{\overline{Q}}_{\varphi,l}$.
Step.2.4. Generate $X^s_{\varphi,l,k}$ from $\overline{\overline{Q}}_{\varphi,l}$.
Step 3. Set l to $l+1$. Reiterate from *Step 2* as long as $B \cap \{X^s_{1,l,k}, ..., X^s_{\varphi,l,k}\} = B \backslash \{T_1\}$.
Step 4. (Stopping condition) Let l' be the iteration step such that $B \cap \{X^s_{1,l',k'}, ..., X^s_{\varphi,l',k'}\} = B \backslash \{T_1\}$. Set $X^s_{i,l'+1,k}$ to T_1, whenever $X^s_{i,l',k} \neq T_1$.

Remarks 2. (i) *T.2.3* ensures that the set of φ routes will cover all of the targets of R; otherwise, there is a positive probability that at least one target will not be visited. (ii) The result of this procedure is to be contrasted to that of [17] applied to the TSP, where each generated route, involving only one vehicle or formation, belongs to the desired set $\mathcal{X}_k$. The reason why $X^s_{i,k}$ is sometimes out of $\mathcal{X}_i$ comes from the fact that the proposed procedure HP(k,s) cannot eliminate route $X^s_{i,k}$ whose length is greater than d_i.

Let $g \in \mathbb{N}$, $\alpha \in (0,1)$, $\rho \in (0,1)$ and $P^{(i)}_k$ be the matrix P_k in (12) computed by F_i. Furthermore, $\widehat{n}_{i,k}$, and $\widehat{d}_{i,k}$ denote the matrices constituted of all of the numerators and denominators of $P^{(i)}_k$, respectively.

CEMBA distributed over the complete graph $\mathcal{G}_c$ is defined as follows.

Step 1. (Initialization) Set $k = 1$ and $P_{i,1} = (\mathbf{1}_n - \mathbf{I}_n)/(n-1)$, for all $i \in B^-$, where $\mathbf{1}_n$ and $\mathbf{I}_n$ stand for the $n \times n$ unitary and identity matrices, respectively. Set $\widehat{n}_{i,1,}$ and $\widehat{d}_{i,1}$ to $(\mathbf{1}_n - \mathbf{I}_n)$ and to $\mathbf{I}_n/(n-1)$, respectively.
Step 2. (*kth iteration*) Each formation F_i of B execute the following tasks.
Step.2.1. Compute $R^{(i)}_{k-1}$ by dividing each (θ,θ')-th entry of $\sum_{i=1}^{\varphi} \widehat{n}_{i,k-1}$ by the (θ,θ')-th entry of $\sum_{i=1}^{\varphi} \widehat{d}_{i,k-1}$.
Step.2.2. Compute $\widetilde{P}^{(i)}_{k-1} = \alpha R^{(i)}_{k-1} + (1-\alpha)\widetilde{P}^{(i)}_{k-2}$.
Step.2.3. Generate X^s_k, for $s = 1, ..., N$, by means of HP$(k-1, s)$ applied to $\widetilde{P}^{(i)}_{k-1}$.
Step.2.4. Compute $U(X^s_k)$ and order them as $U_{(1)} < ... < U_{(N)}$.
Step.2.5. Defined $\gamma_k = U_{\lceil(1-\rho)N\rceil}$.
Step.2.6. Compute $P^{(i)}_{i,k}$ by using (12), and communicate $\widehat{n}_{i,k,}$ and $\widehat{d}_{i,k}$ to F_j, where $j \neq i$.
Step 3. (*Stopping condition*) Set k to $k+1$. Reiterate from *Step 2* as long as γ_{k+1} does not change for a number of g subsequent iterations.

As noted at the beginning of this section, $\mathcal{G}_c$ is not, by assumption, complete. Therefore, each formation needs to reach a consensus in finite time so that *Step.2.1* can be carried out. By consensus, it is meant the mean value of $\widehat{n}_{i,k-1}$ and of $\widehat{d}_{i,k-1}$ over all $i \in B^-$, which is computed despite the weak connectedness of $\mathcal{G}_c$.

5 Consensus Algorithm

Information consensus algorithms proposed in the last few years ensure asymptotic convergence in time of variables that are shared through a weakly connected (see Figure 4), possibly time-varying, graph $\mathcal{G}_c$ [18,19]. Information state $I_{i,k}$ of a node i at iteration step k is usually updated by means of rules that depend on neighboring nodes, for example, $I_{i,k} = I_{i,k-1} + \varepsilon \sum_{l \in N} (I_{l,k-1} - I_{i,k-1})$, where N_i and ε denote the set of neighbors of i, and the simulation step size, respectively. Such rules enable each node's information state to approach to $(1/\varphi)\sum_{i=1}^{\varphi} I_{i,1}$ as time goes to infinity. In the sequel a simple communication protocol is proposed so that the information state of every node approaches $(1/\varphi)\sum_{i=1}^{\varphi} I_{i,1}$ as $t \to \infty$.

Since the consensus must be reached within each CEMBA's iteration, the proposed consensus algorithm is implemented with a time that is discretized at a frequency $1/h_f$ which is higher than that of CEMBA; that is, the discretized time instant is defined as $t_{f,k'} = k'h_f$, $k' \in \mathbb{N}^+$,where $h_f = h_s/\nu$, $h_f \in \mathbb{N}^+$, $\nu \in \mathbb{N}^+$. Iteration step k' is set to one each time the consensus algorithm is triggered; that is, whenever CEMBA provides a new set of output matrices $\widehat{n}_{i,k,}$ and $\widehat{d}_{i,k}$, which satisfy

$$\begin{aligned} \widehat{n}_{i,1} &= \widehat{n}_{i,k,}, \\ \widehat{d}_{i,1} &= \widehat{d}_{i,k}. \end{aligned} \tag{13}$$

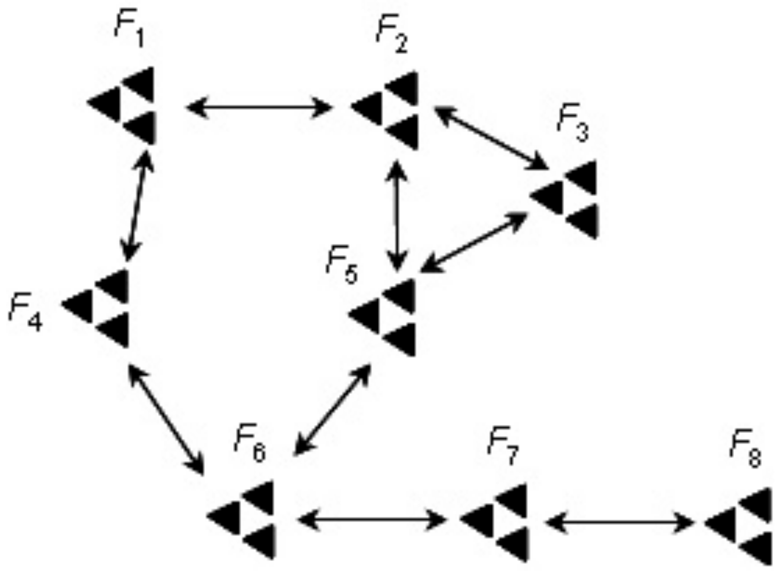


Fig. 4. Consensus reaching among the networked formations

The proposed consensus algorithm is based on the following observation. Assume that the size of data communicated among each pair of adjacent nodes in $\mathcal{G}_c$ is allowed to vary, as opposed to *Assumption 1*. An agent stores, at each discrete time instant $t_{f,k}$, the data communicated from its neighbor and publishes, at the next time instant $t_{f,k+1}$, the content of its internal memory. It is straightforward to notice that the weak connectedness of the communication graph implies that a consensus will be reached after a few iterations since all agents are in possession of the same content. To see this fact, consider, for instance, the graph of Figure 4. The graph comprises 8 interconnected agents, with 3 UCVs per formation.

Assume that one aims at agreeing about the mean value of a variable I_i indexed by $i = \arg(F_i)$, which is the identity of F_i. At iteration step $k = 0$, F_i's internal memory, M_i, is set to i. At $k > 0$, F_i communicates to its neighbors the content of M_i and the set of all I_*, whose index $*$ belongs to M_i. Table 1 presents the content of M_i as k increases, with $B = \{1, ..., 8\}$. The table shows that the consensus is reached after three steps since $M_i = B$ is obtained at $k = 3$ for all $i \in \{1, ..., 8\}$. Each agent is thus able to compute the consensus value $(1/8)\sum_{k=1}^{8} I_k$.

To satisfy *Assumption 1*, we propose to constrain the size of the amount of data that is communicated by F_i to its neighbors, and to randomly select such information from M_i, as explained in the sequel. Let $u^j_{1,i,k'}$ and $u^j_{2,i,k'}$ be the input variables of F_i communicated by $F_j \in N_i$ at $k' \in \mathbb{N}^+$, where N_i stands for the set of neighbors of F_i. The input variable $u^j_{1,i,k'}$ carries the identity of only one of the φ formations selected randomly, whereas $u^j_{2,i,k'}$ carries the state that

Table 1. List of identity variables stored in $M_i$by F_i, $i = 1, ...8$, at each k'

	M_1	M_2	M_3	M_4
$k' = 1$	$\{1\}$	$\{2\}$	$\{3\}$	$\{4\}$
$k' = 2$	$\{1, 2, 4\}$	$\{1, 2, 3, 5\}$	$\{2, 3, 5\}$	$\{1, 4, 6\}$
$k' = 3$	$B^-\backslash\{7, 8\}$	$B^-\backslash\{7, 8\}$	$\{1, 2, 3, 5, 6\}$	$B^-\backslash\{3, 8\}$
$k' = 4$	B^-	B^-	B^-	B^-
	M_5	M_6	M_7	M_8
$k' = 1$	$\{5\}$	$\{6\}$	$\{7\}$	$\{8\}$
$k' = 2$	$\{2, 3, 5, 6\}$	$\{4, 5, 6, 7\}$	$\{6, 7, 8\}$	$\{7, 8\}$
$k' = 3$	$B^-\backslash\{8\}$	B^-	$\{4, 5, 6, 7, 8\}$	$\{6, 7, 8\}$
$k' = 4$	B^-	B^-	B^-	B^-

is eventually used to compute the agreement. The proposed algorithm, labeled CA(k''), can be described as a 3-step algorithm as follows.
Step 1. (*Initialization*) For each formation F_i set, at $t_{f,1}$,

$$\begin{aligned} M_{1,i,1} &= \arg F_i \\ &= i, \\ M_{2,i,1} &= [\widehat{n}_{i,1}, \widehat{d}_{i,1}]. \end{aligned} \tag{14}$$

Step 2. (*Iteration of the consensus algorithm*)
Step.2.1. The interconnection of F_i's inputs with the output of neighbor $F_j \in N_j$ is described, at $t_{f,k'} > 1$, by

$$\begin{aligned} u^j_{1,i,k'} &= y_{1,j,k'-1}, \\ u^j_{2,i,k'} &= y_{2,j,k'-1}, \\ U_{1,i,k'} &= \{u^j_{1,i,k'}, j \in N_i\} \\ U_{2,i,k'} &= \{u^j_{2,i,k'}, j \in N_i\}. \end{aligned} \tag{15}$$

Step.2.2. The internal memory $M_{1,i,k'}$ and $M_{2,i,k'}$ of F_i are updated as follows

$$\begin{aligned} M_{1,i,k'} &= \begin{cases} \{M_{1,i,k'-1}, U_{1,i,k'}\} & \text{if } U_{1,i,k'} \notin M_{1,i,k'-1}, \\ M_{1,i,k'-1} & \text{otherwise,} \end{cases} \\ M_{2,i,k'} &= \begin{cases} \{M_{2,i,k'-1}, U_{2,i,k'}\} & \text{if } U_{2,i,k'} \notin M_{2,i,k'-1}, \\ M_{2,i,k'-1} & \text{otherwise,} \end{cases} \end{aligned} \tag{16}$$

where $\widehat{n}_{i,1}$ and $\widehat{d}_{i,1}$ are defined in (13).
Step.2.3. The fixed size in the amount of data communicated between vehicle F_i and the set of its adjacent vehicles is guaranteed by determining the signals sent by F_i as follows

$$\begin{aligned} y_{1,i,k'} &= \mathtt{rand}(M_{1,i,k'} \backslash Y_{1,i,k'-1}) = i^*, \\ y_{2,i,k'} &= [\widehat{n}_{i^*,k'}, \widehat{d}_{i^*,k'}], \\ \widehat{n}_{i^*,k'}&, \widehat{d}_{i^*,k'} \in M_{2,i,k'}, \end{aligned} \tag{17}$$

where $Y_{1,i,k'-1} = \{y_{1,i,1}, ..., y_{1,i,k'-1}\}$.

Step 3. (*Stopping condition*) The consensus is reached at $k''h_f$, when $k'' \in \mathbb{N}^+$ is such that the equality

$$M_{1,i,k''} = B^- \tag{18}$$

is satisfied for all $i \leq \varphi$. Hence, all of the formations agree on the same information state computed as

$$M_{2,i,k} = 1/\varphi \sum_{i=1}^{\varphi} [\widehat{n}_{i,k''}, \widehat{d}_{i,k''}], \tag{19}$$

where $\widehat{n}_{i,k''}, \widehat{d}_{i,k''} \in M_{2,i,k''}$.

Remark 3. (i) As opposed to consensus algorithms in [18,19], the algorithm in (15)-(19) reaches an agreement among the formations in finite time, although at the expense of a requirement on data storage. (ii) For the stopping condition in *Step 3* to be implementable, $\{M_{1,i,k'},\ i = 1, ..., \varphi\}$ should be available to formation F_i. This is not the case usually since only $M_{1,i,k}$ is available to F_i. It is seen in the sequel that setting $k^{''}$ introduced in *Step 3* to some prescribed value allows reaching an agreement with some level of confidence.

The convergence of the algorithm is illustrated by implementing (15)-(19) in each formation F_i of B. Graph $\mathcal{G}_c$ is shown in Figure 4. At $k' = 1$, $M_{2,i,1}$ is initialized to $i\mathbf{1}_{\varphi}$. The consensus, $M_{2,i,1} = 4.5\mathbf{1}_{\varphi}$, for all $i \in B$, is reached at $k''h_f$. The empirical frequency of the number of time steps k'' necessary to reach the consensus is computed after 1000 simulation runs and is shown in

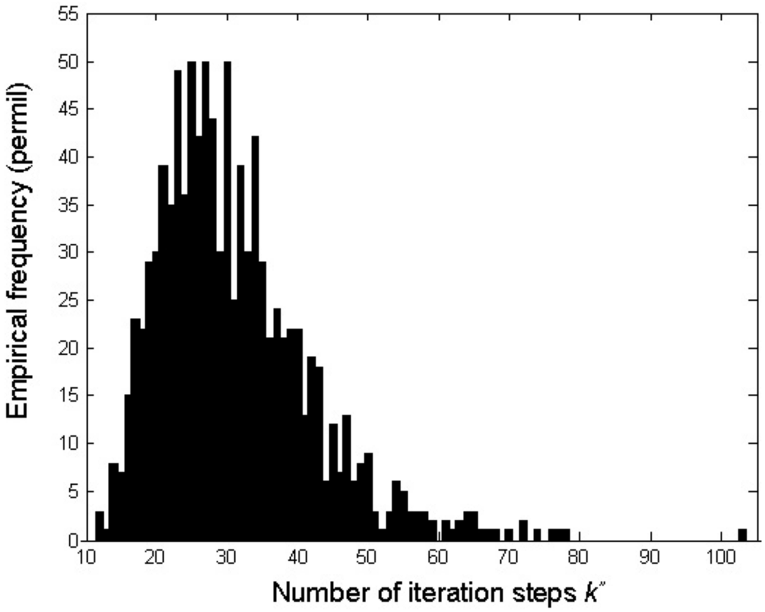


Fig. 5. Empirical frequency of number of time steps $k^{''}$ necessary to reach the consensus

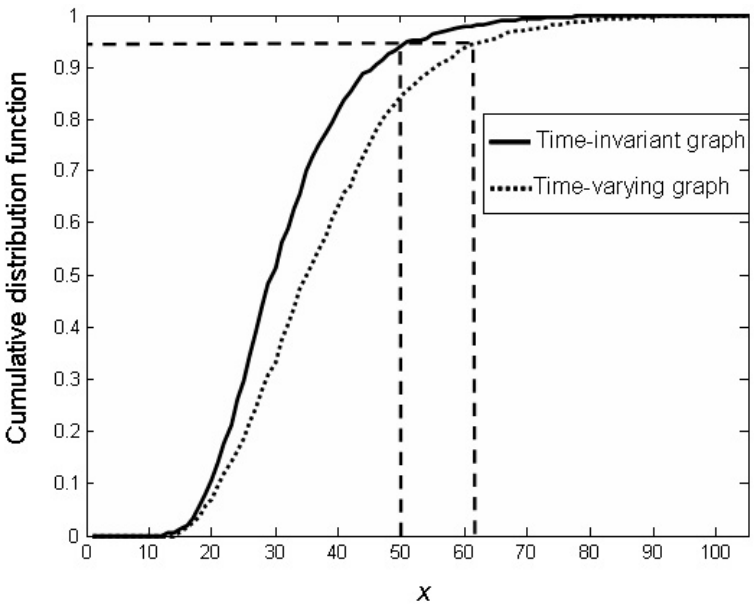
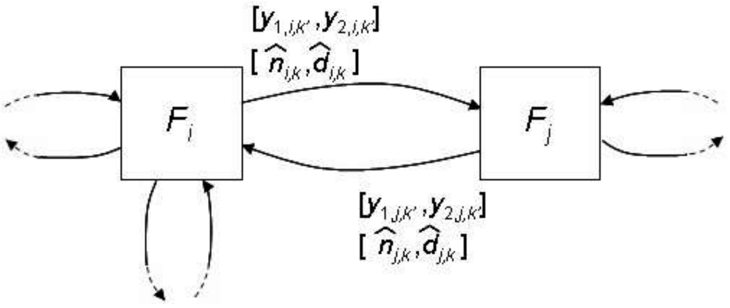


Fig. 6. Cumulative distribution functions, $\Pr(k'' < x)$, of number of time steps necessary to reach the consensus for time-invariant and time-varying graphs

Figure 5. The cumulative distribution function (solid line) in Figure 6 shows that a 95% confidence level is achieved with $k'' = 50$ iterations. Robustness of the algorithm is tested with an intermittent loss of communication links. The scenario is as follows. Edges (F_2, F_5) and (F_3, F_5) are disabled when k' is even, whereas edges (F_1, F_4), and(F_6, F_7) are disabled when k' is odd. Despite the fact that the deactivation of (F_6, F_7) makes $\mathcal{G}_c$ temporarily unconnected, a consensus is reached for every simulation run. As expected, the cumulative distribution function (dashed line) in Figure 6 shows that the 95% confidence level is achieved with a greater number of iterations ($k'' = 63$) when compared to the case of healthy communication links. Such result is important in practice, where wireless communication links among cooperating vehicles are prone to failures in complex environments.

6 Implementation of NEWTAS

NEWTAS can now be presented. Figure 7 shows the distributed architecture of NEWTAS, which is obtained by duplicating in every $F_i \in B$ the process depicted in Figure 8. Every formation F_i executes CA(k'') and CEMBA sequentially. More precisely, CA(k'') becomes the first sub-step, labeled *Step.2.0*, of CEMBA's *Step 2* presented in Section *3.2*. Furthermore $\widehat{n}_{i,k}$, and $\widehat{d}_{i,k}$ in CEMBA's *Step.2.6* are communicated to F_j, where j now belongs to N_i. NEWTAS implemented in F_i can be represented, as shown in Figure 8, as a two-rate algorithm, where



$[y_{1,i,k'}, y_{2,i,k'}]$
$[\widehat{n}_{i,k}, \widehat{d}_{i,k}]$
F_i
F_j
$[y_{1,j,k'}, y_{2,j,k'}]$
$[\widehat{n}_{j,k}, \widehat{d}_{j,k}]$

Fig. 7. Distributed implementation of NEWTAS. Formations exchange data through communication graph G_c.

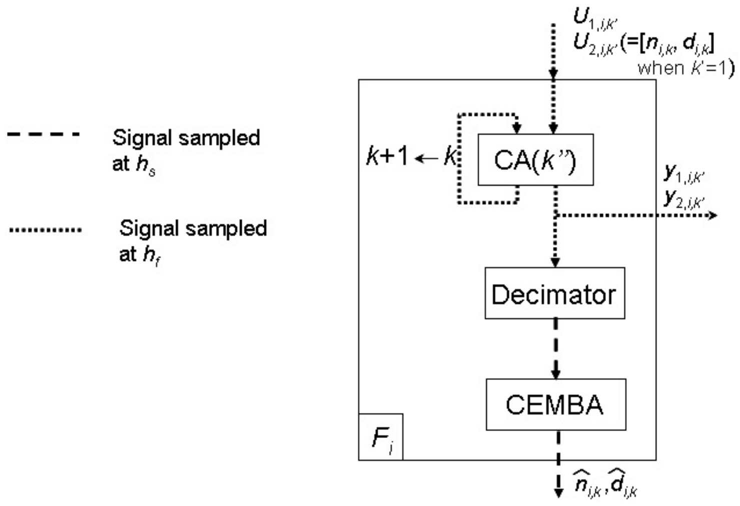


Fig. 8. NEWTAS instantiated in the computing node $F_i \in B$

the decimator enables transitions from the fast-rate subsystem to the slow-rate subsystems. The fast-rate subsystem is composed of CA(k'') operating at h_f. The slow-rate subsystem corresponds to CEMBA whose period is h_s. The formation communicates at the fast rate, which means that the output $\widehat{n}_{i,k}$, and $\widehat{d}_{i,k}$ of CEMBA are communicated every νh_f. Once CA(k'') has reached a consensus, CEMBA proceeds from *Step.2.1* to *Step.2.6.* Sub-steps *Step.2.0-Step.2.6* are iterated as long as the stopping condition of CEMBA (*Step 3*) is not satisfied.

7 Numerical Experiment

NEWTAS is simulated with the Simulink ® software [20]. NEWTAS is applied to the subgraph of $\mathcal{G}_c$ defined as $\{\{F_1, F_2, F_3, F_4\}, \mathcal{E}_B\}$, where $\mathcal{E}_B = \{(F_4, F_1),$ (F_1, F_2), $(F_2, F_3)\}$. Each blue-team formation comprises 16 vehicles. Red team R is composed of 15 tactical targets randomly positioned over a square urban

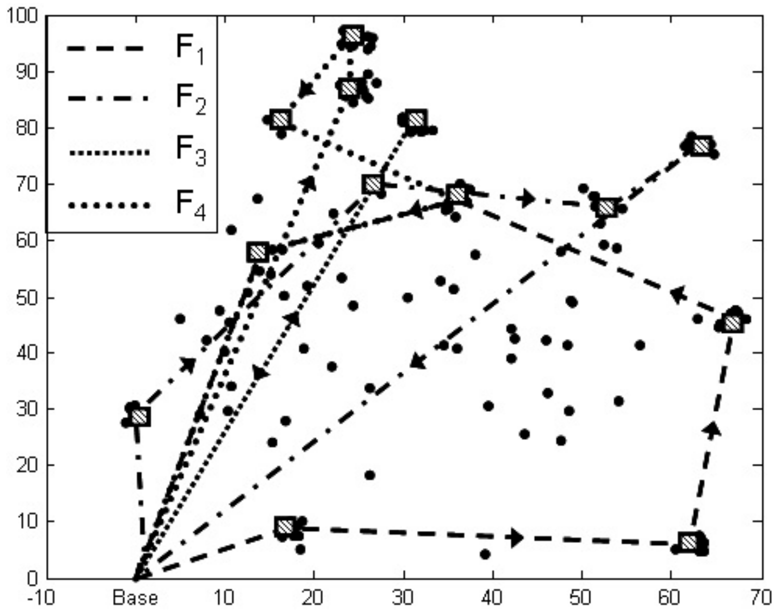


Fig. 9. Weapon-target assignment with 64 UCVs and 15 tactical targets. Routes of formations F_1, F_2, F_3, and F_4 are obtained with NEWTAS.

theater whose edge length is 100 units. The autonomy d_i of each formation is fixed to 240 units. The utility assigned to each target is uniformly randomized over $[0, 1]$. The peak of the global utility is equal to 7.86. The ground units are located randomly although their number is an increasing function of a target's utility and inter-target distance. The attrition model's parameters are as follows: $p_j = 0.7, \mu_{i,j} = 1, \beta_{i,j} = 0.6$, and $q_{i,j} = 0.3$ in (1). The following parameters of CEMBA are used in the simulation: $g = 7$, $\alpha = 5{\cdot}10^{-1}$, $\rho = 5{\cdot}10^{-2}$, and $k'' = 50$.

NEWTAS is implemented on a Pentium 4 processor, with 3.2 GHz clock rate and 2 GB RAM. The execution time is 70 seconds. 31 iterations are needed for the NEWTAS algorithm to converge. The length of the routes followed by F_1, F_2, F_3, and F_4, shown in Figure 9, are 226, 225, 174, and 219, respectively. The average number of healthy vehicles within F_1, F_2, F_3, and F_4 are, at the end of the mission, 3.9, 4.6, 11.5, and 6.5, respectively, which gives a total average number of 26.5 healthy vehicles. Note that the extremum of the global utility is reached. The proposed NEWTAS is compared numerically with the solution to TSP. The results obtained with a solution to TSP given in [17] are shown in Figure 10. With the solution to TSP, the 64 vehicles of the blue team constitute a single formation yielding a total route length of 309 units. This means each vehicle has to travel over 309 units of distance. The number of healthy vehicles that reach the last target is equal to 22 with the solution to TSP. Results obtained with the TSP solution represent a 26% increase in route length with respect to the worst route length (226) obtained with NEWTAS, and a loss of 17% more

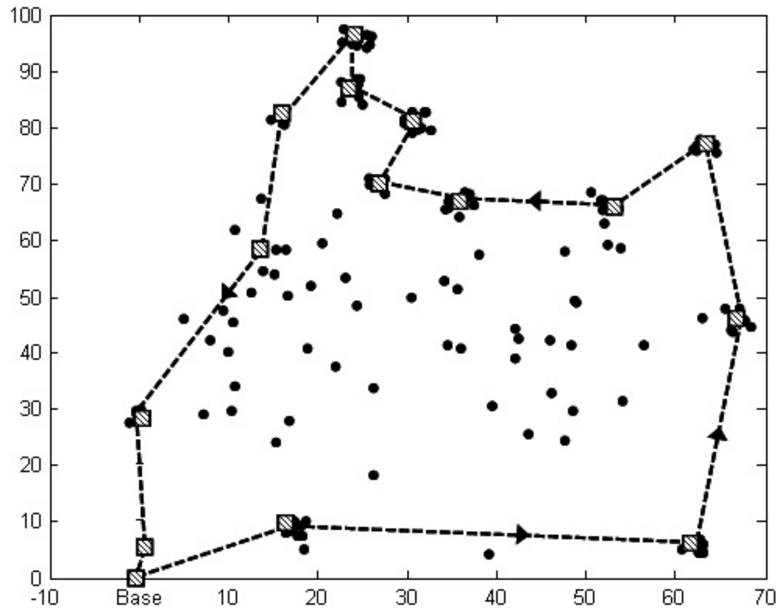


Fig. 10. Weapon-target assignment with 64 UCVs and 15 tactical targets. Routes are obtained by solving the TSP applied to a single formation of 64 vehicles.

vehicles with respect to NEWTAS's total average number of vehicles that remain healthy until the end of the mission. This conclusion corroborates that obtained by carrying out simulations of the low-level decision-making, shown in Figure 1, where PMP solved as a stochastic game advocates formation splitting and merging, when needed, to reduce the risks of losing vehicles while maximizing global weapons effect [13,15,16].

NEWTAS is now applied to 8 formations, each of which is composed of 8 vehicles, thus maintaining the number of vehicles to 64. The route length of $F_1, ..., F_8$ are 228, 237, 196, 194, 233, 198, 204, 224, respectively, leading to an average route length per vehicle of 213 compared to 211 when four formations of 16 vehicles are considered. The main difference comes from the average number of healthy vehicles available at the end of the mission, which is of 12, thus, representing a loss of 54% with respect to NEWTAS applied to 4 formations $\{F_1, F_2, F_3, F_4\}$ of 16 vehicles. This example shows that, given a fixed number of blue-team vehicles, the selection of the size and of the number of formations are critical parameters affecting performance.

8 Conclusions

We proposed a Network Enabled Weapon-Target Assignment Scheme (NEWTAS) aimed at aiding ground crews to assign targets to weapons. Such high-level decision-making algorithm complements low-level path and munitions planning that is used to calculate local paths between two successive tactical targets. The weapon-target assignment problem was formulated in such a way that a group of weapons, or unmanned combat vehicles, are able to join into groups of varying dimensions with the objective of maximizing a global utility function. NEWTAS builds upon a distributed version of the cross-entropy method and a consensus algorithm that allows reaching an agreement in finite time. Each formation first executes a local instance of a cross-entropy-based algorithm. The consensus algorithm is then utilized so that every formation agrees on a common set of variables that are instrumental to the computation of Markov chain transition matrices utilized to generate the route followed by each formation. Once the consensus is reached the cross-entropy-based algorithm iterates again until a stopping condition is satisfied. Simulation results show the advantage of multirouting formation-target assignment, as provided by NEWTAS, over single routing formation-target assignment by maximizing team destruction capability while satisfying autonomy constraints.

Future areas of research include demonstrating that the proposed distributed cross-entropy method can efficiently be utilized to accelerate the global-utility-seeking process when integrated to cooperative health management. A theoretical demonstration that the proposed consensus algorithm is effective with time-varying graphs of large dimensions forming a jointly-connected collection of simple graphs.

References

1. Krokhmal, P., Murphey, R., Pardalos, P., Uryasev, S.: Use of Conditional Value-at-Risk in Stochastic Programs with Poorly Defined Distribustions. In: Butenko, S., Murphey, R., Pardalos, P. (eds.) Recent Developments in Cooperative Control and Optimization, pp. 225–241. Kluwer Academic Publisher, Boston (2004)
2. Shima, T., Ramussen, S.J., Sparks, A.G.: UAV Cooperative Multiple Task Assignments Using Genetic Algorithm. In: Proceedings of the American Control Conference, Portland, OR (2005)
3. Kim, Y., Gu, D., Postlethwaite, I.: Real-Time Optimal Time-Critical Target Assignment of UAVs. In: Hirsh, M.J., Pardalos, P.M., Murphey, R., Grundel, D. (eds.) Advances in Cooperative Control and Optimization. Lecture Notes in Control and Information Sciences, vol. 369, pp. 263–277. Springer, Heidelberg (2007)
4. Richards, A., How, J.P.: Aircraft Trajectory Planning with Collision Avoidance Using Mixed Integer Linear Programming. In: Proceedings of the American Control Conference, Anchorage, Alaska (2002)
5. Alighanbari, M., How, J.P.: Cooperative Task Assignment of Unmanned Aerial Vehicles in Adversarial Environments. In: Proceedings of the American Control Conference, Portland, OR (2005)
6. Arslan, G., Marden, J.R., Shamma, J.S.: Autonomous vehicle-target assignment: a game theoretical formulation. ASME Journal of Dynamic Systems, Measurement and Control 129(5), 584–596 (2007)
7. Cruz, J.B., Simaan Jr., M.A., Gacic, A., Jiang, H., Letellier, B., Li, M., Liu, Y.: Game-Theoretic Modeling and Control of a Military Air Operation. IEEE Transactions on Aerospace an Electronic Systems 37(4), 1393–1403 (2001)

8. Cruz, J.B., Simaan Jr., M.A., Gacic, A., Liu, Y.: Moving Horizon Nash Strategies for a Military Air Operation. IEEE Transactions on Aerospace an Electronic Systems 38(3), 989–997 (2002)
9. Galati, D.G., Simaan, M.A.: Effectiveness of the Nash Strategies in Competitive Multi-Team Target Assignment Problems. In: Proceedings of the 43rd IEEE Conference on Decision and Control, Atlantis, Paradise Island, Bahamas, December 14-17 (2003)
10. Ghose, D., Krichman, M., Speyer, J.L., Shamma, J.: Modeling and Analysis of Air Campaign Resource Allocation: A Spatio-Temporal Decomposition Approach. IEEE Transactions on Systems, Man, and Cybernatics - Part A: Systems and Humans 32(3), 403–418 (2002)
11. Bertsimas, D.J., Van Ryzin, G.J.: Stochastic and Dynamic Vehicle Routing in the Euclidean Plane with Multiple Capacited Vehicles. Operations Research 41(1), 60–76 (1993)
12. Frazzoli, E., Bullo, F.: Decentralized Algorithms for Vehicle Routing in a Stochastic Time-Varying Environment. In: Proceedings of the 43rd IEEE Conference on Decision and Control, Atlantis, Paradise Island, Bahamas, December 14-17 (2003)
13. Léchevin, N., Rabbath, C.A., Lauzon, M.: A Decision Policy for the Routing and Munitions Management of Multiformations of Unmanned Combat Vehicles in Adversarial Urban Environments. IEEE Transactions on Control Systems Technology (to appear, 2008)
14. McEneaney, W.M., Fitzpatrick, B.G., Lauko, I.G.: Stochastic Game Approach to Air Operations. IEEE Transactions on Aerospace an Electronic Systems 40(4), 1191–1216 (2004)
15. Léchevin, N., Rabbath, C.A., Lauzon, M.: Cooperative and Deceptive Planning of Multiformations of Networked UCAVs in Adversarial Urban Environments. In: AIAA Guidance, Navigation and Control Conference and Exhibit, Paper AIAA-2007-6410, Hilton Head, South Carolina, August 20-23 (2007)
16. Léchevin, N., Rabbath, C.A., Lauzon, M.: A Networked Decision and Information System for Increased Agility in Teaming Unmanned Combat Vehicles. In: Proceedings of the 46th IEEE Conference on Decision and Control, New Orleans, LA, December 12-14 (2007)
17. Rubinstein, R.Y., Kroese, D.P.: The Cross-Entropy Method - A unified Approach to Combinatorial Optimization, Monte-Carlo Simulation and Machine. Information Science and Statistics. Springer, Heidelberg (2004)
18. Olfati-Saber, R., Alex Fax, J., Murray, R.: Consensus and Cooperation in Networked Multi-Agent Systems. Proceedings of the IEEE 95(1) (2007)
19. Ren, W., Beard, R.W., Atkins, E.M.: Information Consensus in Multivehicle Cooperative Control. IEEE Control Systems Magazine 27(2), 71–82 (2007)
20. Simulink 7 User's Guide. The MathWorks, Inc., Natick, MA, USA (2007)

Simultaneous Localization and Planning for Cooperative Air Munitions Via Dynamic Programming

Emily A. Doucette[1], Andrew J. Sinclair[1], and David E. Jeffcoat[2]

[1] Auburn University, Auburn AL 36849, USA
[2] Air Force Research Laboratory, Eglin Air Force Base FL, USA

Abstract. This work centers on the real-time trajectory planning for the cooperative control of two aerial munitions that are attacking a ground target in a planar setting. Sensor information from each munition is assumed available, and the individual target-location estimates are fused in a weighted least squares solution. The variance of this combined estimate is used to define a cost function. The problem is posed to design munition trajectories that minimize this cost function. This chapter describes the solution of the problem by a dynamic-programming method. The dynamic-programming method establishes a set of grid points for each munition to traverse based on the initial position of the munition relative to the target. The method determines the optimal path along those points to minimize the value of the cost function and consequently decrease the value of uncertainty in the estimate of the target location. The method is validated by comparison to known solutions computed by a variational method for sample solutions. Numerical solutions are presented along with computational run times to indicate that this method proves effective in trajectory design and target location estimation.

1 Introduction

Research is in progress on the cooperative control of air armaments designed to detect, identify, and attack ground targets with minimal operator oversight. One class of this type of armament is wide-area search munitions, which can be deployed in an area of unknown targets. Current development is focused on the possibilities of enhancing munition capabilities through cooperative control. Important work exists in the literature on the two related problems of cooperative search [1,2,3] and the design of optimal trajectories for single observers [4,5,6,7,8,9,10]. The problem of planning optimal trajectories for cooperative observers has been studied using collocation [11,12]. The problem of cooperative attack was previously investigated using variational methods [13]. This chapter presents a method to drastically reduce the computational expense of the previously implemented variational methods by use of the dynamic-programming (DP) method. This problem of designing an attack trajectory that enhances the ability to estimate the target location will be referred to as simultaneous localization and planning (SLAP).

The methods presented in this chapter will be illustrated for a planar problem with two munitions and one stationary target. In the following section, models for the munition motion and sensor performance are presented. Next, the SLAP trajectory design is posed as a DP problem. The cost-sensitivity to grid resolution is then investigated. Finally, the performance of a target-location estimation algorithm is evaluated along the SLAP trajectories and compared to alternative trajectories.

2 Problem Definition

A scenario will be considered with the two-dimensional plane populated by two munitions and a single fixed target. The state of each munition is given by its position in two dimensional space, $\boldsymbol{x}_1 = [x_1\ y_1]^\mathsf{T}$ and $\boldsymbol{x}_2 = [x_2\ y_2]^\mathsf{T}$. A constant-speed kinematic model is used to describe the motion of the munitions. The heading angles of the munitions are ψ_1 and ψ_2, and the speed of each munition is v. Here, the heading angles are treated as control variables.

$$\begin{aligned} \dot{x}_1 &= v\cos\psi_1 \qquad ; \qquad \dot{x}_2 = v\cos\psi_2 \\ \dot{y}_1 &= v\sin\psi_1 \qquad ; \qquad \dot{y}_2 = v\sin\psi_2 \end{aligned} \tag{1}$$

$$\dot{\boldsymbol{x}}_i = \boldsymbol{f}_i\left(\psi_i\right), \quad i \in \{1,2\} \tag{2}$$

A variable-speed model could be used; however the additional control variables would increase the complexity of the problem and was not considered here. The two velocities were chosen to be equal for the sake of simplicity. Additionally, each munition is considered to carry a sensor that is capable of measuring the target location in the $x-y$ plane. To design trajectories that improve the estimation of the target location, a model of the sensor measurements and their uncertainties is needed. The target has a position described by $\boldsymbol{x}_T = [x_T\ y_T]^\mathsf{T}$. The measurement of the target location by each munition, $\tilde{\boldsymbol{z}}_1 = [\tilde{x}_{T,1}\ \tilde{y}_{T,1}]^\mathsf{T}$ and $\tilde{\boldsymbol{z}}_2 = [\tilde{x}_{T,2}\ \tilde{y}_{T,2}]^\mathsf{T}$, is modeled as shown in Equation (3).

$$\begin{aligned} \tilde{x}_{T,1} &= x_T + w_{x,1}(0,\sigma_{x,1}) \qquad ; \qquad \tilde{x}_{T,2} = x_T + w_{x,2}(0,\sigma_{x,2}) \\ \tilde{y}_{T,1} &= y_T + w_{y,1}(0,\sigma_{y,1}) \qquad ; \qquad \tilde{y}_{T,2} = y_T + w_{y,2}(0,\sigma_{y,2}) \end{aligned} \tag{3}$$

The measurement errors from each munition are assumed to be independent of the errors from the other munition. The x and y measurement errors from each individual munition, $w_{x,i}$ and $w_{y,i}$, however, are treated as correlated Gaussian random variables with zero mean and standard deviations of $\sigma_{x,i}$ and $\sigma_{y,i}$, where $i \in \{1,2\}$. These uncertainties will drive the trajectory design, and they can be selected to model particular sensors.

The error in the target-location measurements from an individual munition is treated as following a zero-mean jointly-Gaussian distribution that is uncorrelated in the down-range and cross-range directions, relative to the true target and munition locations. The errors in these directions, $w_{d,i}(0,\sigma_{d,i})$ and $w_{c,i}(0,\sigma_{c,i})$,

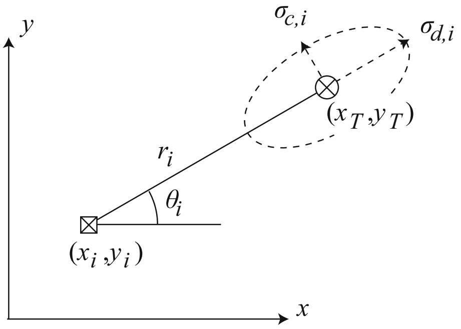


Fig. 1. Measurement of the target by the ith munition and the associated error probability ellipse

can therefore be treated as independent Gaussian random variables. The standard deviations in the down-range and cross-range directions are modeled as functions of the range from the munition to the target.

$$\sigma_{d,i} = 0.1 r_i \qquad ; \qquad \sigma_{c,i} = 0.01 r_i \tag{4}$$

These coefficients do not correspond to specifications of any particular sensor, but model a sensor that is more accurate when close to the target and more accurate in the transverse direction than in the radial direction. The uncertainty in the measurement of the target location by the ith munition is illustrated in Figure 1.

From the down-range and cross-range variables, the errors and the covariance matrix in the x and y coordinates can be found.

$$\begin{bmatrix} w_{x,i} \\ w_{y,i} \end{bmatrix} = \begin{bmatrix} \cos\theta_i & \sin\theta_i \\ -\sin\theta_i & \cos\theta_i \end{bmatrix} \begin{bmatrix} w_{d,i} \\ w_{c,i} \end{bmatrix} \tag{5}$$

$$\boldsymbol{P}_i = \begin{bmatrix} \sigma_{x,i}^2 & \sigma_{xy,i} \\ \sigma_{xy,i} & \sigma_{y,i}^2 \end{bmatrix} = \begin{bmatrix} \cos\theta_i & \sin\theta_i \\ -\sin\theta_i & \cos\theta_i \end{bmatrix} \begin{bmatrix} \sigma_{d,i}^2 & 0 \\ 0 & \sigma_{c,i}^2 \end{bmatrix} \begin{bmatrix} \cos\theta_i & -\sin\theta_i \\ \sin\theta_i & \cos\theta_i \end{bmatrix} \tag{6}$$

Here, θ_i is the bearing angle of the target relative to the ith munition. The range and bearing angle for each target-munition pair are computed as shown below.

$$r_i = \sqrt{(x_T - x_i)^2 + (y_T - y_i)^2} \tag{7}$$

$$\theta_i = \tan^{-1}\left(\frac{y_T - y_i}{x_T - x_i}\right) \tag{8}$$

The measurements provided by both munitions can be fused into a single instantaneous estimate of the target location. This is done using a weighted least-squares estimator (WLSE) [14,15]. The measurements of the target location from each munition are grouped into a measurement vector $\tilde{\boldsymbol{z}} = [\tilde{x}_{T,1}\ \tilde{y}_{T,1}\ \tilde{x}_{T,2}\ \tilde{y}_{T,2}]^\mathsf{T}$. This produces a linear measurement model in terms of the target location.

$$\boldsymbol{z} = \boldsymbol{H}\boldsymbol{x}_T + \boldsymbol{w} \tag{9}$$

$$\boldsymbol{H} = \begin{bmatrix} 1\,0\,1\,0 \\ 0\,1\,0\,1 \end{bmatrix}^\mathsf{T}; \qquad \boldsymbol{w} = \begin{bmatrix} w_{x,1}\ w_{y,1}\ w_{x,2}\ w_{y,2} \end{bmatrix}^\mathsf{T} \tag{10}$$

Here, $\boldsymbol{w}$ is the vector of measurement errors. The covariance of this error vector is given by arranging the covariances from each munition.

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{P}_1 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{P}_2 \end{bmatrix} \tag{11}$$

The instantaneous WLSE of the target location and the associated covariance reduce to the following.

$$\boldsymbol{P} = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{bmatrix} = \left(\boldsymbol{P}_1^{-1} + \boldsymbol{P}_2^{-1}\right)^{-1} \tag{12}$$

The covariance $\boldsymbol{P}$ models the uncertainty in the combined target-location estimate based on the positioning of the two munitions relative to the target. The task of designing trajectories for the munitions in order to enhance the estimation performance can now be posed as the following optimal control problem. Consider the state vector $\boldsymbol{x} = [x_1\ y_1\ x_2\ y_2]^\mathsf{T}$. The heading angles of the munitions can be organized into a control vector $\boldsymbol{u} = [\psi_1\ \psi_2]^\mathsf{T}$. The state vector evolves according to the state equation found by grouping Equation (2), $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{u}) = [\boldsymbol{f}_1^\mathsf{T}\ \boldsymbol{f}_2^\mathsf{T}]^\mathsf{T}$. For boundary conditions, the initial positions of the munitions will be considered a given, and the final position of munition 1 is required to be the target location, $x_1(t_F) = x_T$ and $y_1(t_F) = y_T$. The final position of munition 2 is free.

The goal will be to find the trajectories that minimize the following cost function, which is based on the WLSE covariance.

$$J = \int_0^t \left(\sigma_x^2 + \sigma_y^2\right) \mathrm{d}t \tag{13}$$

The variances of each target location are functions of the states describing the munition configuration. Clearly, this cost function emphasizes the uncertainty over the entire trajectory. Alternative cost functions could be defined using other metrics of the covariance matrix, but these were not investigated here.

3 Dynamic-Programming Approach

The above nonconvex problem has previously been solved using a variational method [13]. Although these solutions demonstrated that significant improvements in the target-location estimate could be achieved, the method was too

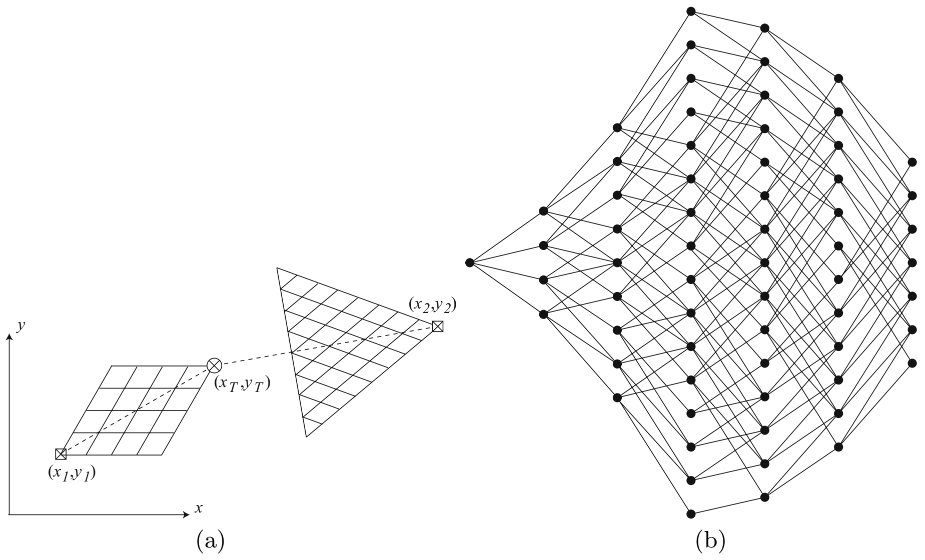


Fig. 2. Example of (a) path grids and (b) DAG where $n = 64$ and $m = 168$ and all edges are directed from left to right

computationally expensive for implementation. This section describes a solution of the problem by dynamic programming (DP), with the goal of reducing computational expense.

Whereas variational methods consider a continuous range of heading angles at any instant in time, the approach considered here only allows a discrete number of possible headings at discrete instants in time. Admissible trajectories were selected considering Equation (2). The trajectories were limited to two possible heading angles at each decision instant. In between decision points, the trajectories follow constant headings. This discretization generates a grid of possible trajectories, as illustrated in Figure 2 (a). This grid of physical points through which the munitions may travel is referred to as the path grid. It is noteworthy that the candidate paths are based on the simple vehicle model in Equation (2); however, the path planning generated from this model could be applied to a higher-order system.

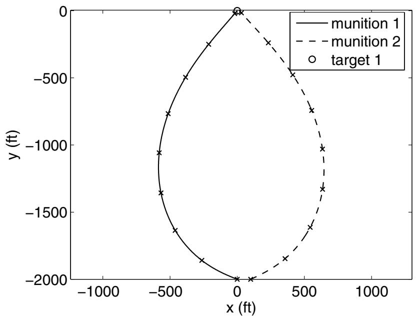
The path grids were laid out for each munition and were structured such that they were symmetric about a reference line from the initial state to the target location. The expansion of this grid is variable about the reference line by an angle, α. Because the results of the variational method showed that the munitions tended to approach the target at orthogonal headings, the path grid was of variable width to allow for outward sweeps. The degree of expansion was determined based on the initial positions of the munitions relative to the target.

The nodes were organized into subsets of nodes that could be reached in a given amount of time. Consequently, the time increment between layers was also assumed constant between each layer. This time increment is calculated from the initial range of the munition that will strike the target, which is assumed to always be the closer of the two munitions. The same time step is used for both munitions in order to preserve synchronized motion of the munitions.

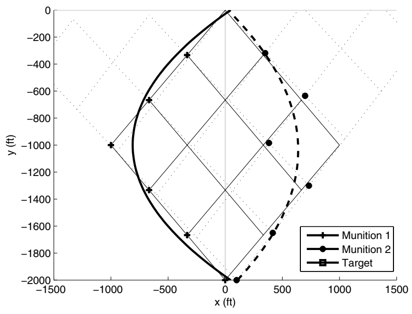
This algorithm implements the Bellman-Ford model for trajectory optimization through dynamic programming in Fortran 77 [16]. Once a cost function is defined and a cost corresponding to each possible set of paths for each munition is calculated according to Equation (13), the combinatorial possibilities of physical node locations for the two munitions were used to form the vertices of a directed acyclic graph (DAG) with n vertices and m edges. Each vertex represents a particular location for each munition at a particular instant in time, and each edge corresponds to the cost value associated with the munitions traveling to those particular locations. The graph is directed and acyclic because the paths must follow the flow of time.

The vertices are arranged into subsets, where each subset represents a particular instant in time. Once the cost along each edge of the DAG is computed, the algorithm marches backward in time from the last layer of vertices to the first vertex to determine the lowest possible cost and the path that produces it. At each subset, the optimal path is computed by comparing the costs to proceed forward. That path and cost are then stored and the algorithm works backwards to the preceding subset, continuing this process until it reaches the initial vertex. The DAG associated with the path grid in Figure 2 (a) is shown in Figure 2 (b). Solutions for this type of graph can be computed in $\mathcal{O}(m)$ time.

Example trajectories produced by the DP method are shown in Figures 3 and 4 using $n = 64$ and $m = 168$. The DP trajectories are also compared to trajectories computed from the variational method. A third order curve fit was used to obtain the smooth trajectories from the grid points of the path grids. The trends of the DP trajectories capture those of the variational method. This is also evidenced by a minor increase in the final cost of approximately 7.75% in each problem, as shown in Table 1. When paired with a computational run-time of 0.1 sec on a 2GHz PC for the DP method, and the advantage of deterministic

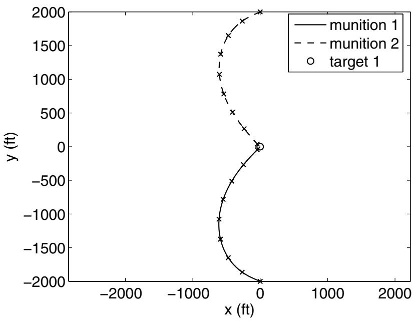


(a)

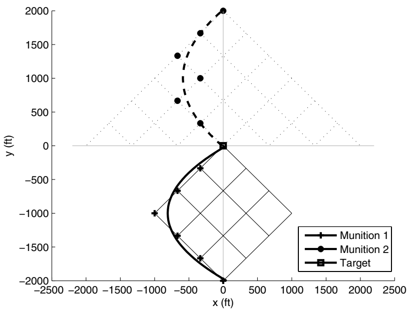


(b)

Fig. 3. Problem 1 SLAP trajectories from variational (a) and DP (b) methods



(a)



(b)

Fig. 4. Problem 2 SLAP trajectories from variational (a) and DP (b) methods

Table 1. Cost for sample SLAP trajectories

Problem	DP Method Cost $n = 64$	DP Method Cost $n = 27$	Variational Method Cost
1	$1.7181 * 10^4$	$1.7209 * 10^4$	$1.59 * 10^4$
2	$2.0317 * 10^4$	$2.0318 * 10^4$	$1.89 * 10^4$

run time, these cost values confirm the effectiveness of this method to the SLAP problem.

4 Cost Sensitivity to Grid Resolution

In implementing the DP method, the resolution of the path grids and the resulting DAG must be selected. In order to determine an appropriate resolution, the sensitivity of the cost to grid resolution was investigated. A lower resolution DAG with $n = 27$ and $m = 70$ was created. The resulting increase in performance cost was less than 1%, as shown in Table 1. This low sensitivity to grid resolution did not motivate the investigation of resolutions greater than $n = 64$.

5 Estimation Performance

The impact of the trajectories on the target-location estimation can now be evaluated. Although the trajectories were designed using a cost function based on the variances from a continuous WLSE algorithm, the estimation performance will be evaluated using a recursive weighted least squares estimation (RWLSE) algorithm with discrete measurement updates. The estimates computed using the DP trajectories are compared to estimates using the variational method and following trajectories from the initial conditions straight to the target location (STT trajectory). In each case, noisy measurements were simulated using the measurement model in Equation (4). The measurements were generated by use of the RANDN command in MATLAB to generate a normal distribution of random numbers.

The munition sensors were assumed to collect measurements of the target location at a rate of 10 Hz. The RWLSE algorithm operated as follows to determine the estimate and the uncertainty at the kth time step [14,15]. The current estimate is computed as follows.

$$\boldsymbol{K}_k = \boldsymbol{P}_{k-1}\boldsymbol{H}^\top \left(\boldsymbol{H}\boldsymbol{P}_{k-1}\boldsymbol{H}^\top + \boldsymbol{R}\right)^{-1} \tag{14}$$

$$\hat{\boldsymbol{x}}_k^{(T)} = \hat{\boldsymbol{x}}_{k-1}^{(T)} + \boldsymbol{K}_k \left(\tilde{\boldsymbol{z}}_k - \boldsymbol{H}\hat{\boldsymbol{x}}_{k-1}^{(T)}\right) \tag{15}$$

The current covariance matrix is computed as shown.

$$\boldsymbol{P}_k = \begin{bmatrix} \sigma_{x,k}^2 & \sigma_{xy,k} \\ \sigma_{xy,k} & \sigma_{y,k}^2 \end{bmatrix} = \left(\boldsymbol{P}_{k-1}^{-1} + \boldsymbol{H}_k^\top \boldsymbol{R}_k^{-1} \boldsymbol{H}_k\right)^{-1} \tag{16}$$

To compare the estimation performance along the different trajectories, the size of the one-sigma uncertainty ellipsoid in the target-location estimate can be used as a metric. At the kth time step, this is given by the product of π with the square root of the product of the eigenvalues of $\boldsymbol{P}_k$. In particular, the ellipsoid size at two seconds prior to impact ($(t_F - 2)$ seconds) will be highlighted. Although t_F is different for each trajectory, at this point in time munition 1 is roughly 600 ft from the target.

Using the initial condition of $x_1(0) = 0$ ft, $y_1(0) = -2000$ ft, $x_2(0) = 100$ ft, and $y_2(0) = -2000$ ft, two munitions on STT trajectories correspond to a one-sigma uncertainty ellipse with an area of 39.7 ft^2 at $(t_F - 2)$ sec, with $t_F = 6.67$, sec. When the two munitions follow the SLAP trajectories obtained through the variational and DP methods shown in Figure 3 however, the area is reduced as shown in Table 2. The error histories for a sample simulation with noisy measurements and three-sigma error bounds ($\pm 3\sigma_{x,k}$ and $\pm 3\sigma_{y,k}$) generated by the RWLSE algorithm are shown in Figure 5. Figure 5(a) shows the errors in the x and y estimates of the target location using the variational method trajectories. Figure 5(b) show the errors using the DP trajectories. Both trajectories give similar, relatively good performance in estimating the target location, however the DP trajectory has slightly slower convergence. This highlights the difference between the uncertainty ellipse area as a performance metric and the cost function used in generating the paths.

Table 2. Area of one-sigma uncertainty ellipse

Problem	STT	Variational Method	DP Method
1	39.7 ft^2	9.1 ft^2	24.5 ft^2
2	40.8 ft^2	9.3 ft^2	23.5 ft^2

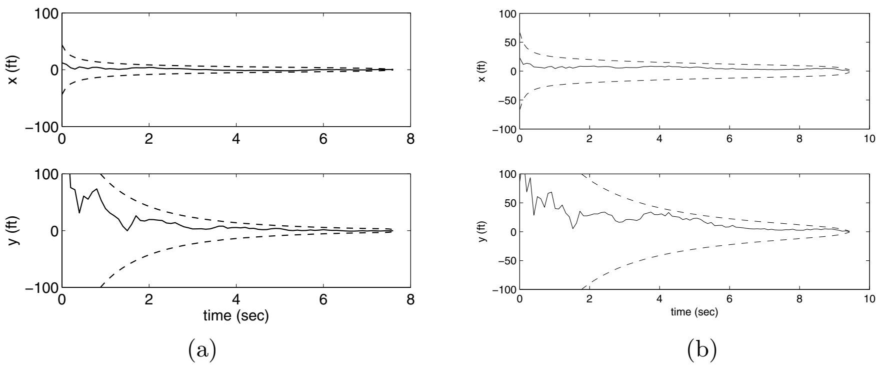


Fig. 5. Estimation errors using (a) Variational Method and (b) DP trajectories with $x_2(0) = 100\,\mathrm{ft}$, and $y_2(0) = -2000\,\mathrm{ft}$

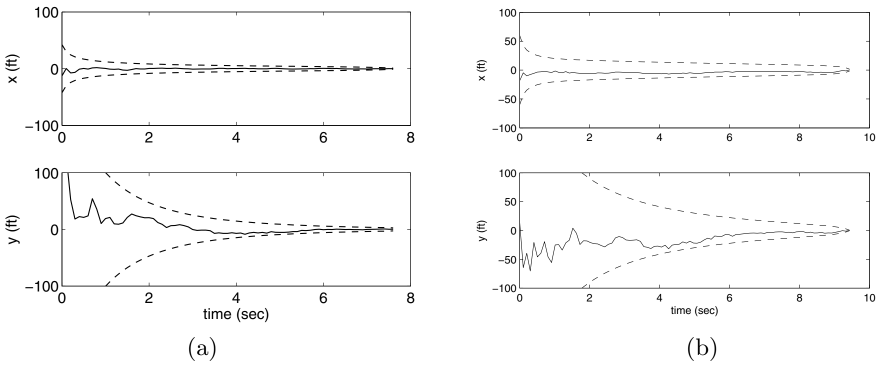


Fig. 6. Estimation errors using (a) Variational Method and (b) DP trajectories with $x_2(0) = 0\,\mathrm{ft}$, and $y_2(0) = 2000\,\mathrm{ft}$

Moving munition 2 to the initial condition $x_2(0) = 0\,\mathrm{ft}$, and $y_2(0) = 2000\,\mathrm{ft}$ corresponds to an uncertainty ellipse with an area of $9.3\,\mathrm{ft}^2$ for the variational method trajectories. With a mesh expansion angle of $\pi/4$, the resulting uncertainty ellipse area was $23.5\,\mathrm{ft}^2$ for the DP trajectories. For these initial conditions, the error histories for a sample simulation with noisy measurements and three-sigma error bounds generated by the RWLSE algorithm are shown in Figure 6.

6 Conclusions

Careful trajectory design can have a significant impact on target-location estimation. In this work, the DP approach was used to demonstrate that SLAP trajectories are practical for real-time implementation. The advantage of this approach is that discretization in both time and spatial coordinates results in a DAG on which the corresponding problem can be solved in a deterministic amount of computation. This allows grid resolution to be selected based on the available computational resources and desired performance.

More accurate target-location estimation could allow more accurate strike capability or the ability to attack the targets that are difficult to detect. Further work is needed to demonstrate the impact of these estimation enhancements on guidance and control performance. In future implementations, heuristic methods may be developed based on insight gained from solutions of the optimal control problem. Additionally, an algorithm could be developed to allow for an adaptive mesh expansion angle. The DP approach will still be a useful development tool to cheaply investigate various solutions.

Acknowledgment

The authors are grateful to Clayton W. Commander, Air Force Research Lab, for his helpful discussion of shortest-path algorithms.

References

1. Chandler, P.R., Pachter, M., Nygard, K.E., Swaroop, D.: Cooperative control for target classification. In: Murphey, R., Pardalos, P.M. (eds.) Cooperative Control and Optimization, pp. 1–19. Kluwer, Netherlands (2002)
2. Jeffcoat, D.E.: Coupled detection rates: An introduction. In: Grundel, D., Murphey, R., Pardalos, P.M. (eds.) Theory and Algorithms for Cooperative Systems, pp. 157–167. World Scientific, New Jersey (2004)
3. Frew, E., Lawrence, D.: Cooperative stand-off tracking of moving targets by a team of autonomous aircraft. In: AIAA-2005-6363. AIAA Guidance, Navigation, and Control Conference, San Fancisco, California (August 2005)
4. Fawcett, J.A.: Effect of course maneuvers on bearings-only range estimation. IEEE Transactions on Acoustics, Speech, and Signal Processing 36(8), 1193–1199 (1988)
5. Hammel, S.E., Liu, P.T., Hilliard, E.J., Gong, K.F.: Optimal observor motion for localization with bearing measurements. Computers and Mathematics with Applications 18(1-3), 171–180 (1989)
6. Logothetis, A., Isaksson, A., Evans, R.J.: Comparison of suboptimal strategies for optimal own-ship maneuvers in bearings-only tracking. In: American Control Conference, Phiadelphia, Pennsylvania (June 1998)
7. Passerieux, J.M., Van Cappel, D.: Optimal observer maneuver for bearings-only tracking. IEEE Transactions on Aerospace and Electronic Systems 34(3), 777–788 (1998)
8. Oshman, Y., Davidson, P.: Optimization of observer trajectories for bearings-only target localization. IEEE Transactions on Aerospace and Electronic Systems 35(3), 892–902 (1999)
9. Frew, E.W., Rock, S.M.: Trajectory generation for constant velocity target motion estimation using monocular vision. In: IEEE International Conference on Robotics & Automation, Taipei, Taiwan (September 2003)
10. Watanabe, Y., Johnson, E.N., Calise, A.J.: Vision-based guidance design from sensor trajectory optimization. In: AIAA-2006-6607. AIAA Guidance, Navigation, and Control Conference, Keystone, Colorado (August 2006)

11. Grocholsky, B.: Information-Theoretic Control of Multiple Sensor Platforms. Ph.D thesis, University of Sydney, Sydney, Australia (2002)
12. Ousingsawat, J., Campbell, M.E.: Optimal cooperative reconnaissance using multiple vehicles. Journal of Guidance, Control, and Dynamics 30(1), 122–132 (2007)
13. Sinclair, A.J., Prazencia, R.J., Jeffcoat, D.E.: Simultaneous localization and planning for cooperative air munitions. In: Hirsch, M.J., Pardalos, P.M., Murphey, R., Grundel, D. (eds.) Advances in Cooperative Control and Optimization. LNCIS, vol. 369, pp. 81–94. Springer, Heidelberg
14. Stengel, R.F.: Optimal Control and Estimation. Dover, New York (1986)
15. Crassidis, J.L., Junkins, J.L.: Optimal Estimation of Dynamic Systems. Chapman & Hall/CRC, Boca Raton (2004)
16. Bellman, R.E., Dreyfus, S.E.: Applied Dynamic Programming. Princeton University Press, Princeton (1962)

A New Multi-objective Control Design for Autonomous Vehicles

Jiangmin Chunyu[1], Zhihua Qu[1], Eytan Pollak[2], and Mark Falash[2]

[1] School of Electrical Engineering and Computer Science,University of Central Florida, Orlando, FL 32816, USA
[2] Strategic Technologies Group at L-3 Communications, Link Simulation & Training, 12351 Research Pkwy, Orlando, FL 32826, USA

Abstract. In this chapter, a nonlinear control design is proposed for a team of wheeled mobile robots to cooperate in a dynamically evolving environment to track their virtual leader(s), while avoiding static and dynamic obstacles. Toward this end, a multi-objective control problem is formulated, and the control is synthesized by generating a potential field force for each objective and combining them through analysis and design. To the best of our knowledge, the proposed design is the first systematic approach to accommodate and achieve the multiple objectives of cooperative motion, tracking virtual command vehicle(s), obstacle avoidance, and oscillation suppression. Basic conditions and key properties are derived using rigorous Lyapunov analysis and theoretical proof. The results are illustrated by several simulation examples including cooperative motion of a team of vehicles moving through urban settings with static and moving obstacles, as well as narrow passages.

Keywords: cooperative motion, tracking of virtual leader, obstacle avoidance, oscillation suppression.

1 Introduction

Many future missions such as cooperative robot reconnaissance [1], marine minesweeping [6], and formation flight control [3,10] will be implemented with distributed autonomous systems, which require formation movement capability. To achieve this goal, the central and difficult issues are:

- cooperative formation movement control of multi-robots;
- collision avoidance naturally arising in the dynamically evolving environment;
- coupling between the above two areas.

1.1 Formation Movement Control of Multi-robots

Most existing methods dealing with formation control use one of three strategies: behavior based, virtual structure, or leader-follower.

In the behavior based approach [1,7], a series of primitive goal oriented behaviors (e.g., move-to-goal, avoid-static-obstacle, avoid-robot and maintain-formation) are proposed to each robot. A weighting factor indicates the relative importance of the individual behaviors. The high-level combined behavior is generated by multiplying the outputs of each primitive behavior by its weight, then summing and normalizing the results. The advantage of behavior based approaches is that each primitive behavior has its physical meaning and the formation feedback can be incorporated into the group dynamics by coupling the outputs of each individual behavior. The disadvantage is that it is difficult to formalize and analyze the group dynamics mathematically, consequently it is difficult to study the convergence of the formation to a desired geometric configuration.

The virtual structure approach [2,3,4,5] is inspired by the rigid body motion of a physical object with all points in the object maintaining a fixed geometric relationship via a system of physical constraints. The robot formation is considered as a single virtual rigid structure. Thus desired trajectories are not assigned to each single robot but to the entire formation as a whole by a trajectory generator. The formation is maintained by minimizing the error between the virtual structure and the current robot position. The advantage of virtual structure approaches is that it is quite easy and straightforward to prescribe the coordinated behavior of the whole team. The disadvantage is that the virtual structure's position is controlled by the positions of the robots, which makes the formation itself, be the centralized control.

In the leader-follower approach [8,9,10], some robots are designed as leaders moving along predefined reference trajectories. The remaining robots are followers and are required to maintain a desired posture (distance and orientation) relative to their own leader. Generally, the leader-follower controls take the following forms: (1) a single leader vehicle and multiple follower vehicles or (2) a "chain" of vehicles each following the preceding vehicle (such as in automated control of highway systems). The advantage of leader-follower is the controls reduce to a tracking problem which can be designed and analyzed using standard control theoretic techniques. The disadvantage is that the formation does not tolerate leader faults, since the leader's predefined trajectory is independent of the motion of each associated follower.

1.2 Collision Avoidance

Obstacle avoidance is a fundamental issue in mobile robotics. This problem has been studied extensively at the navigation system level (path planning) because in the real application, the objective of obstacle avoidance is always combined with target tracking. Most existing methods of path planning use one of two strategies: graph methods and potential field methods. Graph methods are based on a geometrical cell-decomposition of the entire workspace and generate an optimal path with respect to objective criteria, such as finding the shortest collision-free path or the minimum energy cost path. The main criticism to graph methods is that these methods require large computational resources. In the potential field method, the target applies an attractive force to the robot while the obstacles exert a repulsive force onto the robot. The resultant force determines the motion of the robot. The potential field method is particularly useful because of its simplicity, elegance and high efficiency. Some inherent limitations of potential field method have been pointed out in [11], including the following: (1) trap situations due to local minima; (2) no passage between closely spaced obstacles; (3) oscillations in the presence of obstacles; and (4) oscillations in narrow passages.

Obstacle avoidance can also be solved directly in the dynamics controller, which is normally called “avoidance control.” We define avoidance control as a control design which guarantees that every trajectory that emanates from outside of the prescribed avoidance set of a given dynamical system will never intersect the set. The potential field and Lyapunov methods are applied in the design of avoidance controls. The problem of avoidance control for a single dynamical system has been pioneered and extensively studied by Leitmann and his coworkers [13,14]. Sufficient avoidance conditions were given to avoid the set for all time in [13]. In later work [14], two special cases of avoidance are considered: the set must be avoided during a prescribed time interval (finite-time avoidance), or the set must be avoided for all time after some quantifiable or prescribed time interval (ultimate avoidance). Sufficient conditions are presented for these two kinds of avoidance. A generalization of avoidance control for multi-agent dynamic systems is studied in [12]. Sufficient conditions are provided for a class of nonlinear dynamic systems with a special decomposed structure.

1.3 Coupling of Formation Control and Collision Avoidance

Two frameworks are presented in the literature to solve the problem. One is the aforementioned behavior based method, in which avoidance of obstacles as well as other robots is designed as primitive behaviors. As previously mentioned, it is difficult to formalize and analyze the group dynamics mathematically. Consequently, it is difficult to prove convergence to the desired formation and improve the robot’s transient performance.

The other framework is leader-follower formation control based on potential field and Lyapunov direct methods (e.g., [15,16,17]). Potential fields yield interaction forces between neighboring robots to enforce a desired minimum space for any pair of robots. A virtual leader is a moving reference point that exerts forces on its neighboring robots by means of additional similar potential field. The purpose of the virtual leaders is to introduce the mission: to direct, herd and/or manipulate the vehicle group behavior [15]. A properly designed potential field function yields global asymptotic convergence of a group of mobile robots to a desired formation, and guarantees no collisions among the robots [16]. These two methods do not consider the obstacle avoidance issue. The leader-follower strategy essentially transforms the formation control problem into a tracking problem. Based on this, the decentralized controls are designed to achieve collision avoidance and target tacking for a single robot is proposed. It is then extended to address the problem of coordinated tracking of a group of robots [17]. This method does not consider the moving obstacle and only guarantees the tracking with a bounded error.

1.4 Outline of This Chapter

Cooperative formation control with collision avoidance is addressed in this chapter. We first investigate the target tracking and collision avoidance problems for a single agent. Instead of directly extending the single agent controls to the multi-agents case, we incorporated it with the cooperative control design described in [18]. The proposed decentralized control is reactive, considers the formation feedback, and allows topological changes in the sensing and communication networks. Since the proposed control is based on a potential field method, its inherent oscillation problem is also addressed to improve group transient performance.

The rest of this chapter is organized as follows: in Section 2 we formulate the problem of achieving a specified formation amongst a team of cooperative mobile robots, tracking their virtual leader(s), avoiding the static/moving obstacles as well as each other, and suppressing the excessive oscillations. In Section 3 we propose a novel analytical control design for a single point-mass agent to achieve target tracking and collision avoidance, unifying time-varying potential field, nonlinear damping, and velocity-scaled force control. Basic conditions and key properties are derived using rigorous Lyapunov analysis and theoretical proof. In Section 4, the results are extended for networked agents by incorporating an existing cooperative control design [18]. Section 5 presents examples and their simulations to illustrate the design process, to demonstrate its effectiveness, and to show performance of proposed controls . Finally, in Section 6 we conclude the chapter and suggest some future research directions.

2 Problem Formulation

Consider a collection of point-mass agents whose dynamics are given by

$$\dot{q}_{r\mu} = v_{r\mu}, \quad \dot{v}_{r\mu} = u_{r\mu}, \ (\mu = 1, \ldots, m) \tag{1}$$

where $q \overset{\Delta}{=} [x, y]^T$ denotes the center position, $v \overset{\Delta}{=} [v_x, v_y]^T$ represents the velocity, and u is the control input. Thus we can define the states $S(t) = (q\,(t)\,, v\,(t))$. Subscripts r, g and o indicate the vehicle, goal and obstacle respectively.

Given the initial configurations $S_{r\mu}(t_0) = (q_{r\mu}(t_0), v_{r\mu}(t_0))$, as shown in Figure 1, the objective of this chapter can be summarized as follows:

- tracking the specified virtual leader $S_{g\mu}\,(t) = (q_{g\mu}\,(t)\,, v_{g\mu}\,(t))$;
- avoiding the n obstacles $S_{oi} = (q_{oi}\,(t)\,, v_{oi}\,(t))\,(i = 1, 2, \cdots n)$;
- avoiding the remaining $(m-1)$ agents $S_{rj} = (q_{rj}(t), v_{rj}(t))$, $(j = 1, 2, \cdots, \mu - 1, \mu + 1, \cdots, m)$;
- suppressing the oscillation of the system trajectory.

To solve the problem, one can make the following choices without loss of generality:

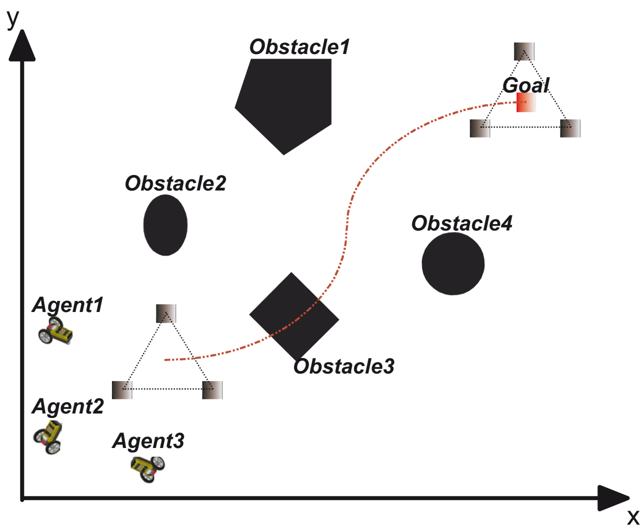


Fig. 1. Illustration of cooperative formation movement with collision avoidance(Three agents are required to maintain a triangular formation and track the goal in the presence of obstacles)

- The μth agent under consideration is represented by a 2-D circle with the center at $q_{r\mu}(t)$ and of radius R. The range of its sensors is also described by a circle centered at $q_{r\mu}(t)$ and of radius R_s.
- The ith static/moving obstacle will be represented by a convex object of any shape (such as circle, ellipse, or polygon).

3 Target Tracking and Collision Avoidance for a Single Agent

First, we derive a decentralized feedback control using Lyapunov-type analysis that guarantees collision avoidance and tracking of a virtual leader for a single robot. Then in Section 4, we extend this result to the case of networked agents by incorporating cooperative control [18]. We then propose a novel cooperative formation control design with collision avoidance.

To achieve these design objectives, two potential field functions are used to generate reactive forces. Specifically, consider the following composite potential function:

$$P(q_r - q_o, q_r - q_g) = P_a(q_r - q_g) + P_r(q_r - q_o), \tag{2}$$

where $P_a(\cdot)$ is the attractive potential function and $P_r(\cdot)$ is the repulsive potential function. Intuitively and necessarily, potential functions should have the properties that

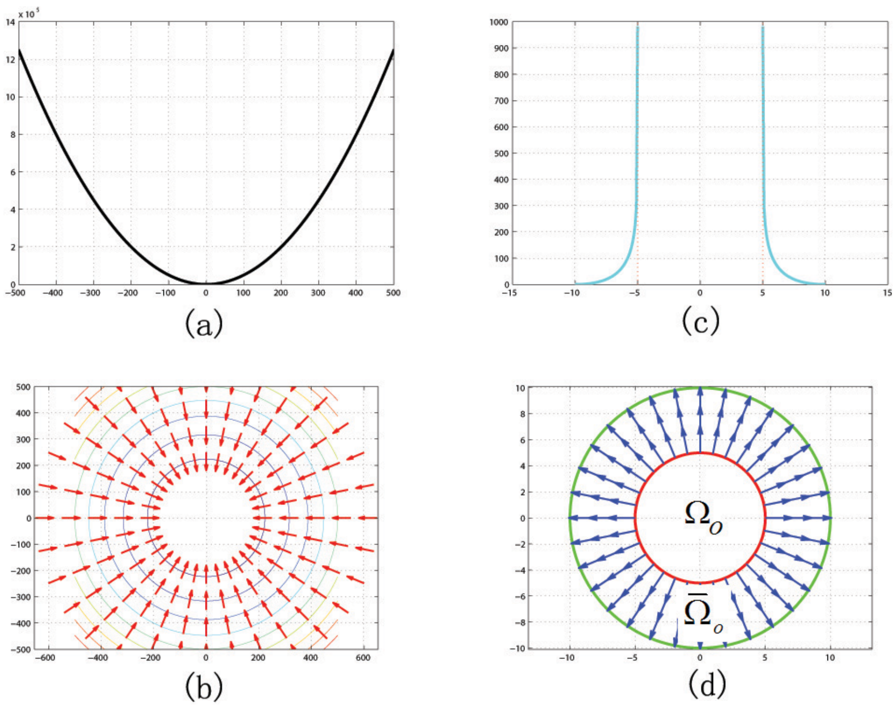


Fig. 2. Typical attractive potential function versus repulsive potential function (a:attractive potential field function; b:contour lines of attractive potential field function; c:repulsive potential field function; d:contour lines of repulsive potential field function)

$$\begin{cases} P_a(0) = 0, \quad \nabla P_a\,(s)\,|_{s=0} = 0, \\ 0 < P_a(s) < \infty \text{ if } s \neq 0 \text{ and } \|s\| \text{ is finite,} \\ \|\nabla P_a\,(s)\| < +\infty \text{ if } \|s\| \text{ is finite,} \end{cases} \tag{3}$$

and

$$\begin{cases} P_r(s) = +\infty & \text{if } s \in \Omega_o, \\ P_r(s) = 0 & \text{if } s \notin \overline{\Omega}_o, \\ P_r(s) \in (0, \infty) & \text{if } s \in \overline{\Omega}_o \text{ but } s \notin \Omega_o, \\ \lim\limits_{s \to \Omega} \|\nabla P_r\,(s)\| = +\infty & \text{if } s \notin \Omega_o, \end{cases} \tag{4}$$

where $\Omega_o \subset \Re^2$ is a compact set representing the 2-dimensional shape of the obstacle, $\overline{\Omega}_o$ is the compact set which is an enlarged version of Ω_o and in which repulsive force becomes active. The above defined attractive potential function and repulsive potential function are exemplified by Figure 2.

Furthermore, commonsense dictates that an additional detour force could easily drive vehicle to make a detour and reach its goal. Thus we introduce a novel conception "unit detour force vector" $T(q_r - q_o)$, which has the properties that

$$\nabla P_r^T(s)\,T(s) = 0, \quad -\nabla P_a^T(s)\,T(s) \geq 0, \quad \textit{and} \quad \|T(s)\| = 1. \tag{5}$$

Let the vehicle control to be a reactive control of the form

$$\begin{aligned} u_\mu = &-\nabla P_a(q_{r\mu} - q_{g\mu}) - \nabla P_r(q_{r\mu} - q_{oi}) + f_d(q_{r\mu} - q_{oi})\,T(q_{r\mu} - q_{oi}) \\ &-\xi(q_{r\mu} - q_{g\mu})(v_{r\mu} - v_{g\mu}) + \dot{v}_{g\mu} - \dot{\eta}(q_{r\mu} - q_{oi})\,\|v_{g\mu} - v_{oi}\|^2 \\ &-2\eta(q_{r\mu} - q_{oi})(v_{g\mu} - v_{oi})^T(\dot{v}_{g\mu} - \dot{v}_{oi}), \end{aligned} \tag{6}$$

where the term $\nabla P_a(q_{r\mu} - q_{g\mu})$ and $\nabla P_r(q_{r\mu} - q_{oi})$ are the standard reactive control components, $\xi(\cdot) > 0$ is a locally uniformly bounded function designed to ensure stability and damp oscillations, $f_d(s) > 0$ $\left(f_d(s) = 0, \text{ if } s \notin \bar{\Omega}_o\right)$ is a locally uniformly bounded function designed to generate detouring force in the vicinity of the obstacle, and $\eta(\cdot)$ is the force to resolve the potential conflict between goal tracking and collision avoidance. Vector function $\eta(\cdot)$ has the properties that

$$\eta(s) \triangleq \begin{bmatrix} \eta_1(s) \\ \eta_2(s) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \text{if } s \notin \bar{\Omega}_o, \tag{7}$$

and

$$\lim_{s \notin \Omega\ ,\ s \to \Omega} \frac{\eta^T(s)\,\nabla P_r(s)}{\|\nabla P_r(s)\|} = +\infty. \tag{8}$$

3.1 Tracking of a Virtual Leader

The tracking problem is to ensure that the μth agent will converge to the goal position $q_{g\mu}(t)$ of the μth virtual leader. The following lemma provides the basic result.

Lemma 1: *The state of system (1) under control (6) converges asymptotically to that of the virtual vehicle provided that, after a finite time instant t^*, $[q_g(t) - q_o(t)] \notin \overline{\Omega}_o$ for all $t \geq t^*$. If $[q_g(t) - q_o(t)]$ stays in or intermittently returns to $\overline{\Omega}_o$, there is no convergence of $[q_r(t) - q_g(t)] \to 0$.*

Proof. Proof: It follows from (1), (2) and (6) that the tracking error system is

$$\begin{aligned} \dot{e}_{1\mu} &= e_{2\mu} \\ \dot{e}_{2\mu} &= -\nabla P_a(e_{1\mu}) - \nabla P_r(e_{1\mu} + q_{g\mu} - q_{oi}) \\ &\quad + f_d(e_{1\mu} + q_{g\mu} - q_{oi})\,T(e_{1\mu} + q_{g\mu} - q_{oi}) - \xi(e_{1\mu})e_{2\mu} \\ &\quad - \begin{bmatrix} \nabla\eta_1(e_{1\mu} + q_{g\mu} - q_{oi})^T \\ \nabla\eta_2(e_{1\mu} + q_{g\mu} - q_{oi})^T \end{bmatrix}(e_{2\mu} + v_{g\mu} - v_{oi})\,\|v_{g\mu} - v_{oi}\|^2 \\ &\quad -2\begin{bmatrix} \eta_1(e_{1\mu} + q_{g\mu} - q_{oi}) \\ \eta_2(e_{1\mu} + q_{g\mu} - q_{oi}) \end{bmatrix}(v_{g\mu} - v_{oi})^T(\dot{v}_{g\mu} - \dot{v}_{oi}), \end{aligned}$$

where $e_{1\mu} = q_{r\mu} - q_{g\mu}$ and $e_{2\mu} = v_{r\mu} - v_{g\mu}$. It is straightforward to verify that, if $[q_{g\mu}(t) - q_{oi}(t)] \in \overline{\Omega}_o$, $e_{1\mu} = e_{2\mu} = 0$ is not an equilibrium point of the error system and hence no convergence can be achieved.

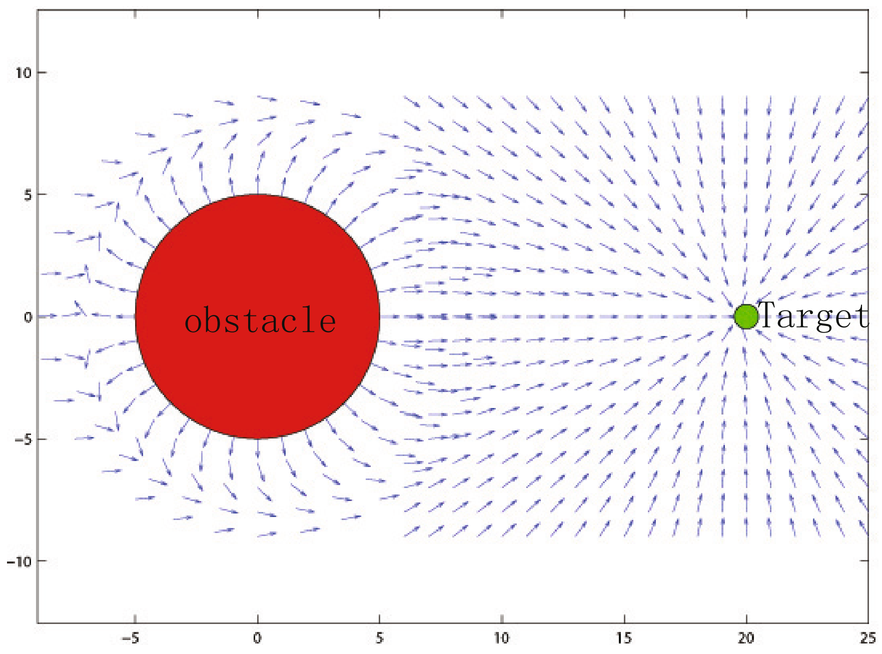


Fig. 3. Illustration of the vector field yielded by the proposed control $((q_{r\mu} - q_{oi}) \notin \bar{\Omega}_o)$

Moreover, from the geometric viewpoint, the resultant force vector field rendered by the proposed control can be illustrated in Figure 3 when $(q_{r\mu} - q_{oi}) \notin \bar{\Omega}_o$. As we had mentioned in Section 2, the obstacle is assumed to be a convex object. As shown in Figure 3, once the robot is in the set $\bar{\Omega}_o$, it will be pushed away from the obstacle and make a detour to reach its goal. Finally, the agent will not stay in or intermittently be in $\bar{\Omega}_o$ which means $[q_{r\mu}(t) - q_{oi}(t)] \notin \overline{\Omega}_o$ for all $t \geq \bar{t}^*$ $(\bar{t}^* > t^*)$.

In this case, by properties (4) and (7), the system error reduces to

$$\dot{e}_{1\mu} = e_{2\mu}, \quad \dot{e}_{2\mu} = -\nabla P_a\left(e_{1\mu}\right) - \xi(e_{1\mu})e_{2\mu}.$$

Consider the Lyapunov function

$$L_1(t) = P_a(e_{1\mu}) + \frac{1}{2}\|e_{2\mu}\|^2.$$

It follows that

$$\begin{aligned}\dot{L_1} &= e_{2\mu}^T \nabla P_a\left(e_{1\mu}\right) + e_{2\mu}^T\left[-\nabla P_a\left(e_{1\mu}\right) - \xi(e_{1\mu})e_{2\mu}\right] \\ &= -\xi\left(e_{1\mu}\right)\left\|e_{2\mu}\right\|^2,\end{aligned}$$

which is negative semi-definite. Asymptotic stability of $e_{1\mu}$ and $e_{2\mu}$ can be concluded using LaSalle's invariant set theorem [19]. $\square$

3.2 Obstacle Avoidance

The obstacle avoidance problem is to ensure that the agent will not enter the given compact set Ω_o provided that its initial position is not in the set. The following lemma provides the basic result.

Lemma 2: *Suppose that potential field function (2) satisfies properties (3) and (4). Then, as long as the initial condition is not in set Ω_o, system (1) under control (6) is collision-free provided that $v_g(t)$ and $v_o(t)$ are uniformly bounded and that $q_o(t)$ is uniformly bounded.*

Proof. Let us choose the following Lyapunov function candidate:

$$V_1(t) = \frac{1}{2}\left\|v_{r\mu} - v_{g\mu} + \eta(q_{r\mu} - q_{oi})\|v_{g\mu} - v_{oi}\|^2\right\|^2 + P(q_{r\mu} - q_{g\mu}, q_{r\mu} - q_{oi}).$$

It follows from (1) and (6) that

$$\begin{aligned}
\dot{V}_1 &= \left[v_{r\mu} - v_{g\mu} + \eta\left(q_{r\mu} - q_{oi}\right)\left\|v_{g\mu} - v_{oi}\right\|^2\right]^T \left[\dot{v}_{r\mu} + \dot{\eta}\left(q_{g\mu} - q_{oi}\right)\left\|v_{r\mu} - v_{oi}\right\|^2\right. \\
&\quad \left. -\dot{v}_{g\mu} + 2\eta\left(q_{r\mu} - q_{oi}\right)\left(v_{g\mu} - v_{oi}\right)^T\left(\dot{v}_{g\mu} - \dot{v}_{oi}\right)\right] \\
&\quad + \left(v_{r\mu} - v_{g\mu}\right)^T \nabla P_a\left(q_{r\mu} - q_{g\mu}\right) + \left(v_{r\mu} - v_{oi}\right)^T \nabla P_r\left(q_{r\mu} - q_{oi}\right) \\
&= \left[v_{r\mu} - v_{g\mu} + \eta\left(q_{r\mu} - q_{oi}\right)\left\|v_{g\mu} - v_{oi}\right\|^2\right]^T \left[-\nabla P_a\left(q_{r\mu} - q_{g\mu}\right)\right. \\
&\quad \left. -\nabla P_r\left(q_{r\mu} - q_{oi}\right) + f_d\left(q_{r\mu} - q_{oi}\right) T\left(q_{r\mu} - q_{oi}\right) - \xi\left(q_{r\mu} - q_{g\mu}\right)\left(v_{r\mu} - v_{g\mu}\right)\right] \\
&\quad + \left(v_{r\mu} - v_{g\mu}\right)^T \nabla P_a\left(q_{r\mu} - q_{g\mu}\right) + \left(v_{r\mu} - v_{oi}\right)^T \nabla P_r\left(q_{r\mu} - q_{oi}\right) \\
&= \omega\left(q_{r\mu} - q_{g\mu}, q_{r\mu} - q_{oi}, v_{r\mu} - v_{g\mu}, v_{r\mu} - v_{oi}\right) \\
&\quad -\xi\left(e_{1\mu}\right)\left\|e_{2\mu}\right\|^2 - \eta^T\left(q_{r\mu} - q_{oi}\right)\nabla P_r\left(q_{r\mu} - q_{oi}\right)\left\|v_{g\mu} - v_{oi}\right\|^2, \qquad (9)
\end{aligned}$$

where

$$\begin{aligned}
&\omega\left(q_{r\mu} - q_{g\mu}, q_{r\mu} - q_{oi}, v_{r\mu} - v_{g\mu}, v_{r\mu} - v_{oi}\right) \\
&= \left(v_{g\mu} - v_{oi}\right)^T \nabla P_r\left(q_{r\mu} - q_{oi}\right) + \left(v_{r\mu} - v_{g\mu}\right)^T f_d\left(q_{r\mu} - q_{oi}\right) T\left(q_{r\mu} - q_{oi}\right) \\
&\quad -\eta^T\left(q_{r\mu} - q_{oi}\right)\nabla P_a\left(q_{r\mu} - q_{g\mu}\right)\left\|v_{g\mu} - v_{oi}\right\|^2 \\
&\quad -\eta^T\left(q_{r\mu} - q_{oi}\right)\xi\left(q_{r\mu} - q_{g\mu}\right)\left(v_{r\mu} - v_{g\mu}\right)\left\|v_{g\mu} - v_{oi}\right\|^2 \\
&\quad +\eta^T\left(q_{r\mu} - q_{oi}\right) f_d\left(q_{r\mu} - q_{oi}\right) T\left(q_{r\mu} - q_{oi}\right)\left\|v_{g\mu} - v_{oi}\right\|^2 .
\end{aligned}$$

Recall that $\|v_{g\mu} - v_{oi}\|$ and $\|q_{oi}(t)\|$ are uniformly bounded. It follows from (3) that $\nabla P_a\left(q_{r\mu} - q_{oi}\right)$ and $\xi\left(q_{r\mu} - q_{g\mu}\right)$ are uniformly bounded for $\left(q_{r\mu} - q_{oi}\right) \in \overline{\Omega}_o$. Hence, there exist constants $c_1, c_2 \geq 0$ such that

$$\begin{aligned}
&\left|\omega\left(q_{r\mu} - q_{g\mu}, q_{r\mu} - q_{oi}, v_{r\mu} - v_{g\mu}, v_{r\mu} - v_{oi}\right)\right| \\
&\leq c_1\left\|\nabla P_r\left(q_{r\mu} - q_{oi}\right)\right\| + c_2\left\|\eta^T\left(q_{r\mu} - q_{g\mu}\right)\right\|\left\|v_{r\mu} - v_{g\mu}\right\| .
\end{aligned}$$

Therefore, we know from (8) and (9) that,

$$\lim_{(q\quad - q\ )\to\Omega} \dot{V}_1 < 0 .$$

Similarly we also have

$$\lim_{v \to\infty} \dot{V}_1 < 0 \, .$$

We can draw the conclusion $V_1(t)$ is finite in any finite region for any finite initial condition $(q_r(t_0), q_g(t_0), q_o(t_0), v(t_0), v_g(t_0))$. Thus, $V_1(t)$ will stay finite under the initial and collision-free conditions mentioned in the lemma. □

3.3 Oscillation Suppression

In this section, we first investigate the nature of the inherent oscillation problem of potential field methods. From this base, we illustrate why the proposed control is a remedy for this problem.

Oscillation Analysis. The causes of oscillations can be classified into the following three types:

1. Potential field functions, especially the interaction between the attractive potential field function (to the goal) and repulsive potential field function (around the obstacle).
If the distance between an obstacle and the goal is small compared to the distance between either and the vehicle, the two similar but opposite force fields can result in oscillation.
2. Insufficient damping, especially in the nonlinear setting.
Damping serves two purposes: (1) Stabilize the system; (2) Suppress the oscillation. For example, if $\xi(\cdot) > 0$ is set to be zero, then the system (1) will never converge to the goal unless the initial condition is trivially given by $(q_{r\mu}(t_0) = q_{g\mu}(t_0), v_{r\mu}(t_0) = v_{g\mu}(t_0))$.
3. Sampling and the gradient descent method.
In the gradient descent approach, the control law for a robot navigating in a potential field uses the negative gradient direction to determine a vector that points toward the target. Whenever we consider a discrete system and the potential contour is not perfectly circular, solutions tend to exhibit oscillation, especially in proximity to obstacles or in narrow passages.

Quasi-Monotone Convergence. We begin with the following definitions.

Definition 1: Let $\alpha(t)$ be a scalar function. Function $\alpha(t)$ is (strictly) monotone decreasing over an interval if $\alpha(t_2) \leq \alpha(t_1)$ $(\alpha(t_2) < \alpha(t_1))$ for any $t_2 > t_1$ within the interval. Function $\alpha(t)$ is (strictly) monotone increasing if $-\alpha(t)$ is (strictly) monotone decreasing. Function $\alpha(t)$ is (strictly) monotone if $\alpha(t)$ is either (strictly) monotone increasing or (strictly) monotone decreasing. Function $\alpha(t)$ is (strictly) *monotone convergent* if it is (strictly) monotone and if $\lim_{t\to\infty} \alpha(t) = 0$.

Definition 2: Let $\alpha(t)$ be a scalar function. $\alpha(t)$ is one-swing quasi-monotone over an interval if the interval can be divided into two subintervals over each of which $\alpha(t)$ is monotone. Function $\alpha(t)$ is called to be *one-swing quasi-monotone convergent* if it is one-swing quasi-monotone and if $\lim_{t\to\infty} \alpha(t) = 0$.

Consequently, any monotone function is also one-swing quasi-monotone. In general, the converse does not always hold true. The nontrivial case of $\alpha(t)$ being one-swing quasi-monotone convergent is that $\alpha(t)$ is convergent while $|\alpha(t)|$ is monotone increasing for $t \in [t_0, t_1]$ and monotone decreasing for $t \in [t_1, \infty)$. For dynamic systems of order higher than one, quasi-monotone convergence is generally best achievable upon successfully suppressing all the oscillations. The following lemma provides such a result.

Lemma 3: *Suppose that differentiable function $\beta(\cdot) \in \Re^2$ exists to satisfy the following properties:*

$$\beta(-s) = -\beta(s), \quad \beta^T(s)\beta(s) = \frac{P_a(s)}{2}, \quad \frac{\partial \beta(s)}{\partial s} = \frac{\xi(s)}{2} I. \tag{10}$$

Assume that after a finite time instant t^*, $[q_g(t) - q_o(t)] \notin \overline{\Omega}_o$ for all $t \geq t^*$. Then, the error between the state of system (1) under control (6) and that of the virtual vehicle is one-swing quasi-monotone convergent.

Proof. Defining the state transformation

$$z_1 = e_1, \quad z_2 = e_2 + \beta(e_1),$$

we can rewrite the error dynamics as, as long as $[q_g(t) - q_o(t)] \notin \overline{\Omega}_o$

$$\begin{aligned}
\dot{z}_1 &= -\beta(z_1) + z_2 \\
\dot{z}_2 &= -\frac{\partial P_a(z_1)}{\partial z_1} + \frac{\partial \beta(z_1)}{\partial z_1}[z_2 - \beta(z_1)] - \xi(z_1)[z_2 - \beta(z_1)] \\
&= -\frac{\partial P_a(z_1)}{\partial z_1} - \frac{\partial \beta(z_1)}{\partial z_1}\beta(z_1) + \xi(z_1)\beta(z_1) \\
&\quad - \left[\xi(z_1) I - \frac{\partial \beta(z_1)}{\partial z_1}\right] z_2.
\end{aligned}$$

It follows from (10) that

$$\dot{z}_1 = -\beta(z_1) + z_2, \quad \dot{z}_2 = \frac{\xi(z_1)}{2} z_2,$$

from which one-swing quasi-monotone convergence can be concluded using Lemma 4. □

Remark 1: We can choose the set $\bar{\Omega}_o$ to be small, thus the impact of the obstacle is confined to a small area to suppress the oscillation. On the other hand, the $\bar{\Omega}_o$ can not be too small due to the numerical calculation.

Remark 2: The term $f_d T$ in (6) yields the detour force which can decrease the chance of oscillation, while speeding up the convergence to the target.

Remark 3: The controller sampling rates must be small to suppress the oscillation.

Lemma 4: *Consider the differential equations*

$$\dot{\alpha}_1(t) = -f_1(\alpha_1) + \alpha_2(t), \quad \dot{\alpha}_2(t) = -f_2(\alpha_2), \tag{11}$$

where functions $f_i(s)$ have the properties that $f_i(0) = 0$ and $df_i(s)/ds > 0$ for all $s \neq 0$. Then, solution $\alpha_1(t)$ is one-swing quasi-monotone convergent.

Proof. It follows from the property of $f_2(\cdot)$ that

$$\frac{d\alpha_2^2}{dt} = -2\alpha_2 f_2(\alpha_2) < 0$$

and hence $\alpha_2(t)$ is strictly monotone convergent. Using the property of $f_1(\cdot)$, we know that

$$\epsilon(t) = f_1^{-1}(\alpha_2(t))$$

is well defined, that $\epsilon(t)$ is also strictly monotone convergent (either $\dot{\epsilon}(t) \geq 0$ with $\epsilon(t_0) \leq 0$ or $\dot{\epsilon}(t) \leq 0$ with $\epsilon(t_0) \geq 0$), and that

$$\dot{\alpha}_1(t) = -f_1(\alpha_1) + f_1(\epsilon(t)) \tag{12}$$

$$\frac{d}{dt}[\alpha_1(t) - \epsilon(t)] = -[f_1(\alpha_1) - f_1(\epsilon(t))] - \dot{\epsilon}(t). \tag{13}$$

Choose $q > 1$ and function $h_2(s)$ such that $h_2(s)$ is a strictly monotone increasing function with $h_2(0) = 0$ and that

$$|h_2(s) f_2(s)|^{\frac{1}{}} \geq |s|.$$

Let $p > 1$ be the constant such that $1/p + 1/q = 1$, and select $h_1(s)$ such that $h_1(s)$ is a strictly monotone increasing function with $h_1(0) = 0$ and that

$$|f_1(s)|^{\frac{1}{}} \geq \frac{1}{p}|h_1(s)|^{\frac{1}{}}.$$

Consider Lyapunov function

$$V = \frac{1}{p}\int_0^{\alpha_1} h_1(s)ds + \frac{2}{q}\int_0^{\alpha_2} h_2(s)ds.$$

It follows from Holder's inequality $a^p/p + b^q/q \geq ab$ that, along the solution of (11),

$$\begin{aligned}
\dot{V} &= -\frac{1}{p}h_1(\alpha_1)f_1(\alpha_1) + \frac{1}{p}h_1(\alpha_1)\alpha_2 - \frac{2}{q}h_2(\alpha_1)f_2(\alpha_2) \\
&\leq -|h_1(\alpha_1)f_1(\alpha_1)|^{\frac{1}{}}|h_2(\alpha_2)f_2(\alpha_2)|^{\frac{1}{}} + \frac{1}{p}h_1(\alpha_1)\alpha_2 \\
&\quad -\frac{1}{q}h_2(\alpha_1)f_2(\alpha_2) \\
&\leq -\frac{1}{q}h_2(\alpha_1)f_2(\alpha_2) \leq 0,
\end{aligned}$$

from which convergence of $\alpha_2(t)$ can be concluded. One-swing quasi-monotone convergence is analyzed below by studying four distinct cases.

Case 1: $\dot{\epsilon}(t) \leq 0$ and $\alpha_1(t_0) \geq \epsilon(t_0) \geq 0$. In this case, we know from (13) that $\alpha_1(t) \geq \epsilon(t)$ for all $t \geq t_0$ and consequently from (12) that $\alpha_1(t)$ is monotone decreasing (and convergent).

Case 2: $\dot{\epsilon}(t) \leq 0$ and $\alpha_1(t_0) < \epsilon(t_0)$. In this case, we know from (12) that $\alpha_1(t)$ is monotone increasing while $\epsilon(t)$ is monotone decreasing. Hence, there exists time instant $t_1 \in [t_0, \infty]$ such that $\alpha_1(t_1) = \epsilon(t_1)$, that $\alpha_1(t) < \epsilon(t)$ for $t \in [t_0, t_1)$, and that evolution of $\alpha_1(t)$ over $[t_1, \infty)$ becomes that in case 1. Hence, $\alpha_1(t)$ is one-swing quasi-monotone convergent.

Case 3: $\dot{\epsilon}(t) \geq 0$ and $\alpha_1(t_0) \leq \epsilon(t_0) \leq 0$. This case is analogous to case 1 except that $\alpha_1(t)$ is monotone increasing (and convergent).

Case 4: $\dot{\epsilon}(t) \geq 0$ and $\alpha_1(t_0) > \epsilon(t_0)$. This case is parallel to case 2 except that, while convergent, $\alpha_1(t)$ is first monotone decreasing and then monotone increasing.

The proof is completed by summarizing all the cases. □

4 Cooperative Formation Control of Networked Agents with Collision Avoidance

4.1 Cooperative Control for Networked Systems of Canonical Form

Consider a group of networked dynamic systems given by the following canonical form

$$\dot{X}_i = A_i X_i + B_i U_i, \quad Y_i = C_i X_i, \quad \dot{\eta}_i = g_i(\eta_i, X_i), \tag{14}$$

where $i = 1, \cdots, q$, $l_i \geq 1$ is an integer, $X_i \in \Re^{l\ m}$, $\eta_i \in \Re^{n\ -l\ m}$, $I_{m\times m}$ is the m dimensional identity matrix, $\otimes$ denotes the Kronecker product, J_k is the kth order Jordan canonical form given by

$$J_k = \begin{bmatrix} -1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & -1 & 1 & \ddots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & -1 & 1 & 0 \\ 0 & 0 & 0 & \cdots & -1 & 1 \\ 0 & 0 & 0 & \cdots & 0 & -1 \end{bmatrix} \in \Re^{k\times k},$$

where $A_i = J_{l} \otimes I_{m\times m} \in \Re^{(l\ m)\times(l\ m)}, B_i = \begin{bmatrix} 0 \\ I_{m\times m} \end{bmatrix} \in \Re^{(l\ m)\times m}, C_i = \begin{bmatrix} I_{m\times m} & 0 \end{bmatrix} \in \Re^{m\times(l\ m)}, Y_i \in \Re^m$ is the output, $U_i \in \Re^m$ is the cooperative control law to be designed, and subsystem $\dot{\eta}_i = g_i(\eta_i, X_i)$ is input-to-state stable.

We consider the general case where exchange of output information among the vehicles occurs only intermittently and locally. To capture this information flow, let us define the following sensing/communication matrix and its corresponding time sequence $\{t_k^s : \ k = 0, 1, \ldots\}$

$$S(t) = \begin{bmatrix} S_1(t) \\ S_2(t) \\ \vdots \\ S_q(t) \end{bmatrix} = \begin{bmatrix} s_{11}(t) & s_{12}(t) & \cdots & s_{1q}(t) \\ s_{21}(t) & s_{22}(t) & \cdots & s_{2q}(t) \\ \vdots & \vdots & \vdots & \vdots \\ s_{q1}(t) & s_{q2}(t) & \cdots & s_{qq}(t) \end{bmatrix}, \begin{cases} S(t) = S(t_k^s), \ \forall t \in \left[t_k^s, t_{k+1}^s\right) \\ S(k) \overset{\Delta}{=} S(t_k^s) \end{cases}, \tag{15}$$

where $s_{ii}(t) \equiv 1$; $s_{ij}(t) = 1$ if the jth vehicle is known to the ith vehicle at time t, and $s_{ij}(t) = 0$ otherwise, and $t_0^s \overset{\Delta}{=} t_0$. Time sequence $\{t_k^s\}$ and the corresponding changes in the row $S_i(t)$ of matrix $S(t)$ are detectable instantaneously and locally at the ith vehicle, but they are not predictable, prescribed or known *a priori* or modeled in any way.

Cooperative controls proposed in this chapter are in the class of linear, piecewise constant, local feedback controls with feedback gain matrices $G_i(t) \overset{\Delta}{=} [G_{i1}(t), \cdots, G_{iq}(t)]$, where $i = 1, \cdots, q$,

$$\begin{aligned} &G_{ij}(t) = G_{ij}(t_k^s), \ \forall t \left[t_k^s, t_{k+1}^s\right); \\ &G_{ij}(k) \overset{\Delta}{=} G_{ij}(t_k^s) \overset{\Delta}{=} \frac{s_{ij}(t_k^s)}{\sum_{\eta=1}^{q} s_{i\eta}(t_k^s)} K_c, \quad j = 1, \cdots, q; \end{aligned} \tag{16}$$

where $s_{ij}(t)$ are piecewise-constants as defined in (15) and $K_c \in \Re^{m \times m}$ is a constant, nonnegative, and row stochastic matrix. That is, cooperative controls are of form

$$U_i \overset{\Delta}{=} \sum_{j=1}^{q} G_{ij}(t) \left[s_{ij}(t) y_j\right] = G_i(t) Y \tag{17}$$

where $Y = \left[Y_1^T, \cdots Y_q^T\right]^T$. Although $S(t)$ is not known *a priori* nor can it be modelled, $S(t)$ is piecewise constant, diagonally positive and binary, and the value of row $S_i(t)$ is known at time t to the ith vehicle. The above choice of the feedback gain matrix block $G_{ij}(t)$ in terms of $s_{ij}(t)$ ensures that matrices $G_i(t)$ are row stochastic and that control is always local and implementable with only available information.

Theorem 1: Consider dynamics system in (14) and under cooperative control (17). Then systems of (14) exhibit a single cooperative behavior as,

$$X_{ss} = 1_N \ cX(t_0) = c_0 1_N , \quad \text{and} \quad Y_{ss} = c_0 1_m, \quad c \in \Re^{1 \times N}, \ c_0 \in \Re, \tag{18}$$

where $N_q = m \sum_{i=1}^{q} l_i$ provided that
i) Gain matrix K_c is chosen to be irreducible and row stochastic.
ii) Systems in (14) have a sequentially complete sensing/communication.

Proof. Please refer to [18] for a detailed proof. □

The single cooperative behavior described in (18) does not necessarily mean that, if $Y_{ss}^d = c_0^d 1_m$, the desired behavior represented by constant c_0^d is achieved. In order to ensure $c_0 = c_0^d$ in (18), we must employ an adaptive version of cooperative control (17). To this end, a virtual vehicle representing a hand-off operator is introduced as

$$\dot{X}_0 = -X_0 + U_0, \quad Y_0\left(t\right) = X_0\left(t\right), \quad U_0 = K_c X_0\left(t\right),$$

where $X_0 \in \Re^m$ with $X_0\left(t_0\right) = c_0^d 1_m$. Communication from the virtual vehicle to the physical vehicles is also intermittent and local, thus we can introduce the following augmented sensor/communication matrix and its associated time sequence $\{\bar{t}_k^s : \ k = 0, 1, \cdots\}$ as:

$$\bar{S}\left(t\right) = \begin{bmatrix} 1 & 0 \ \cdots \ 0 \\ s_{10} & \\ \vdots & S\left(t\right) \\ s_{q0} & \end{bmatrix} \in \Re^{(q+1)\times(q+1)}, \quad \begin{cases} \bar{S}\left(t\right) = \bar{S}\left(\bar{t}_k^s\right), & \forall t \in \left[\bar{t}_k^s, \bar{t}_{k+1}^s\right) \\ \bar{S}\left(k\right) \triangleq \bar{S}\left(\bar{t}_k^s\right), & \end{cases} \tag{19}$$

Accordingly, cooperative control is modified from (17) to the following adaptive version:

$$U_i\left(t\right) = \sum_{j=0}^{q} \frac{s_{ij}\left(t\right)}{\sum_{\eta=0}^{q} s_{i\eta}\left(t\right)} K_c\left[s_{ij}\left(t\right) Y_j\right], \quad i = 1, \cdots, q, \tag{20}$$

where $s_{ij}\left(t\right)$ are piecewise-constant entries of (19). Applying Theorem 1 to the resulting augmented closed loop system renders the following corollary.

Corollary 1: Under the adaptive version cooperative control (20) with irreducible and row stochastic matrix K_c, systems of (14) exhibit the desired cooperative behavior Y_{ss}^d, i.e.,

$$X_{ss} = 1_{L\ \ +1} \otimes Y_{ss}^d, \quad \text{and} \quad Y_{i,ss} = Y_{ss}^d, \ (L_q = \sum\nolimits_{i=1}^{q} l_i),$$

if $Y_{ss}^d = c_0^d 1_m$ for $c_0^d \in \Re$ and if their augmented sensor/communication sequence $\left\{\bar{S}\left(k\right)\right\}$ defined by (19) is sequentially complete.

4.2 Formation Calculation

A formation is defined in a coordinate frame that moves with the desired trajectory relative to some other, fixed, coordinate frame. Let $o_j\left(t\right) \in \Re^3$ $(j = 1, 2, 3)$ be the orthonormal vectors which form the moving frame $F(t)$. Let $O_\mu = [x_\mu, y_\mu, z_\mu] \in \Re^3$ be the location of the μth agent and $O_d = \left[x_d\left(t\right), y_d\left(t\right), z_d\left(t\right)\right] \in \Re^3$ be any desired trajectory of the origin of the moving frame. A formation consists of m agents in $F(t)$, denoted by $\{O_1, \cdots, O_m\}$,where

$$O_\mu = d_{\mu 1}\left(t\right) o_1\left(t\right) + d_{\mu 2}\left(t\right) o_2\left(t\right) + d_{\mu 3}\left(t\right) o_3\left(t\right), \quad \mu = 1, \cdots, m, \tag{21}$$

with $d_\mu\left(t\right) = \left[d_{\mu 1}\left(t\right), d_{\mu 2}\left(t\right), d_{\mu 3}\left(t\right)\right] \in \Re^3$ being the coordinate values of the μth agent in the formation. The desired position for the μth agent is then

$$O_\mu^d(t) = O_d(t) + d_{\mu 1}^d o_1(t) + d_{\mu 2}^d o_2(t) + d_{\mu 3}^d o_3(t). \tag{22}$$

where the constant vector $d_\mu^d = \left[d_{\mu 1}^d, d_{\mu 2}^d, d_{\mu 3}^d\right] \in \Re^3$ is the desired relative position for the μth agent in the formation. Meanwhile, $o_j(t)\,(j = 1, 2, 3)$ generally can be chosen based on the attitude angles of the virtual leader(yaw ψ, pitch θ, roll ϕ). Certainly, the choice of $o_j(t)\,(j = 1, 2, 3)$ is not unique. For example, $o_j(t)$ can also be determined by the velocity vector of the virtual leader.

4.3 Mapping from Formation Control Problem to Cooperative Control Problem

Through state transformations, the formation control problem for (1) can be recast as the cooperative control design problem (14). Let the transformation be

$$X_\mu = O_\mu(t) - O_\mu^d(t), \tag{23}$$

Then we introduce the canonical model with $X_\mu = [X_{\mu 1}, X_{\mu 2}, X_{\mu 3}]^T \in \Re^3$, $U_\mu \in \Re^3$, and $Y_\mu \in \Re^3$.

$$\dot{X}_\mu = \lambda * (A_\mu X_\mu + B_\mu U_\mu), \; Y_\mu = C_\mu X_\mu. \tag{24}$$

where A_μ, B_μ and C_μ are given by

$$A_\mu = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}, \; B_\mu = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \; C_\mu = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

To this end, if we can design the cooperative control U_μ such that states X_μ for all μ converge to the same steady state X_{ss}, then it follows form Corollary 1 that

$$O_\mu \to X_{ss} + O_\mu^d(t),$$

from which it can be seen that the desired formation is achieved for the whole group, while the agents move along the desired trajectory shape.

4.4 Vehicle Level Control for Nonholonomic Agents

Consider the following kinematic and dynamic model of a unicycle,

$$\begin{cases} \dot{x} = v\cos\theta \\ \dot{y} = v\sin\theta \\ \dot{\theta} = \quad \omega \\ \dot{v} = \quad \frac{F}{M} \end{cases}, \tag{25}$$

where θ is the orientation, v is the linear velocity, ω is the angular velocity, F is the applied force and M is the mass. The top view of the unicycle is shown in Figure 4.

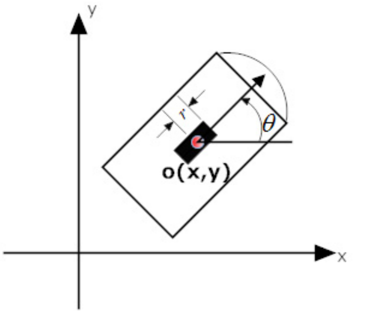


Fig. 4. Relevant variables for the unicycle (top view)

Consider the following dynamic compensator:

$$\begin{cases} \omega = \frac{u_2 \cos\theta - u_1 \sin\theta}{v} \\ F = M\left(u_1 \cos\theta + u_2 \sin\theta\right) \end{cases} . \tag{26}$$

Note the following facts:
1. The sign of linear velocity v will determine forward or backward motion of the vehicle.
2. Transformation (26) is singular at $v = 0$, i.e., when the mobile robot is not moving.
Substituting (26) into (25) yields the following transformed system:

$$\begin{cases} \ddot{x} = u_1 \\ \ddot{y} = u_2 \end{cases} . \tag{27}$$

4.5 Formation Control with Collision Avoidance

Considering the dynamic system (24), where we aim to make $\|X_i - X_j\| \to 0$ while at the same time ensure that $\|O_i - O_j\|^2 \geq \rho_s$ for some positive constant ρ_s. The following condition is then imposed:

$$\left\|O_i^d(t) - O_j^d(t)\right\|^2 \geq \rho_s.$$

To address the collision avoidance problem, let us consider the control to be given by

$$\begin{aligned} U_\mu^* &= U_\mu + \sum_{j=1, j\neq\mu}^{q} \Big[-\nabla P_r\left(O_{r\mu} - O_{rj}\right) - \dot{\eta}\left(O_{r\mu} - O_{rj}\right)\|v_{g\mu} - v_{rj}\|^2 \\ &\quad + f_d\left(O_{r\mu} - O_{rj}\right)T - \eta\left(O_{r\mu} - O_{rj}\right)\left(v_{g\mu} - v_{rj}\right)^T\left(v_{g\mu} - v_{rj}\right)\Big] \\ &\quad + \sum_{l=1}^{n} \Big[-\nabla P_r\left(O_{r\mu} - O_{ol}\right) - \dot{\eta}\left(O_{r\mu} - O_{ol}\right)\|v_{g\mu} - v_{ol}\|^2 \\ &\quad + f_d\left(O_{r\mu} - O_{ol}\right)T - \eta\left(O_{r\mu} - O_{ol}\right)\left(v_{g\mu} - v_{ol}\right)^T\left(v_{g\mu} - v_{ol}\right)\Big] . \end{aligned} \tag{28}$$

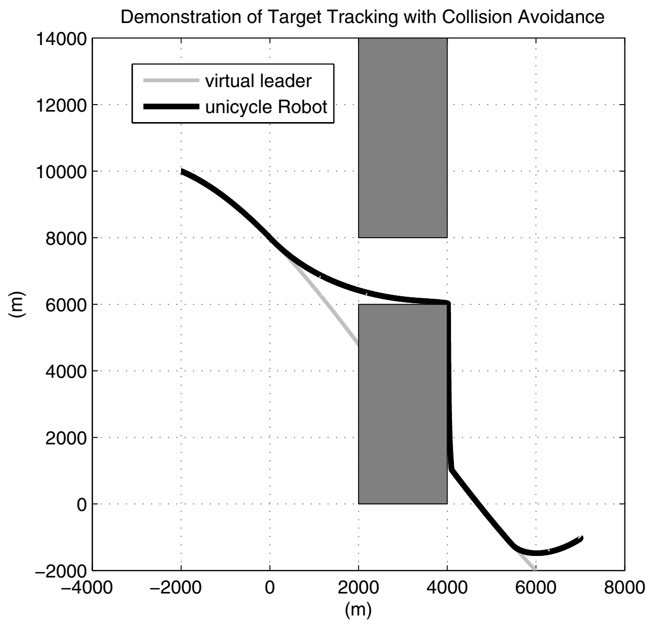


Fig. 5. Experiment on a single unicycle vehicle

5 Simulation

Two simulation scenarios are presented to show the effectiveness of the proposed control in Sections 3 and 4 respectively.

A. Target Tracking and Collision Avoidance for A Single Agent
There are only two rectangular static obstacles, (3000,11000,2000,6000)[1] and (3000,3000,2000,6000). The initial location of the virtual leader is (6000,-2000) with the following waypoints: (3000,3000), (0,8000), and (-2000,10000). The initial location of the unicycle vehicle is (7000,-1000). The simulation result is shown in Figure 5.

B. Cooperative Formation Control of Networked Agents with Collision Avoidance
Three agents are required to execute the formation movement with the desired triangular formation shown in Figure 6. The initial location of the virtual leader is (1850,-1000) with the following waypoints: (1800,3000), (1900,11000), and (1850,15000). The initial location of the above three agents are: (1850,-960), (1930,-1040), and (1770,-1040). In the workspace, there are six rectangular static obstacles, (700,3000,2000,6000), (700,11000,2000,6000), (3000,3000,2000,6000),

[1] Data format:(center position, width, length). For example, (3000,11000) denotes the center position. The width is 2000 and the length is 6000.

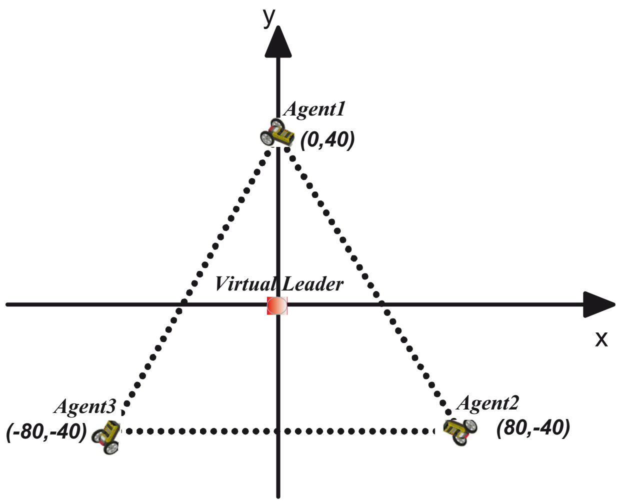


Fig. 6. Illustration of the desired formation

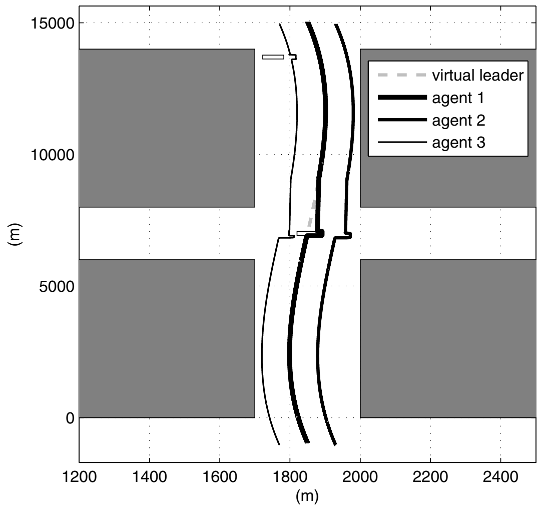


Fig. 7. Cooperative formation movement with avoiding static obstacles

(3000,11000,2000,6000), (1850,7000,60,150), and (1752.5,13700,60,150). In addition, one circular moving obstacle of radius being 10 is also considered. The simulation result considering only the static obstacles is shown in Figure 7. The simulation result considering all obstacle, static and moving, is shown in Figure 8. Figure 8 is zoomed in to shown the successful avoidance of the moving obstacle, depicted in Figure 9.

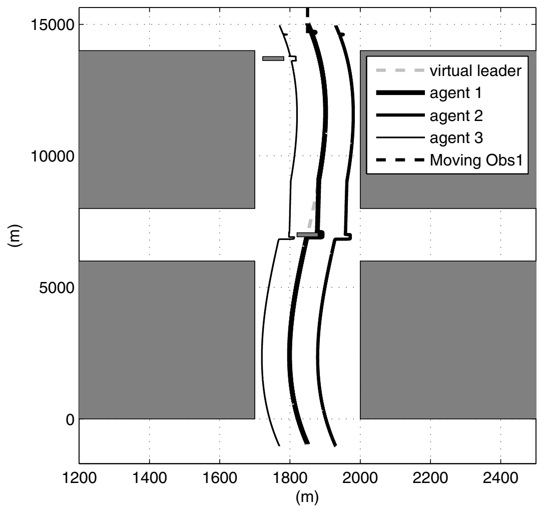


Fig. 8. Cooperative formation movement with avoiding static/moving obstacles

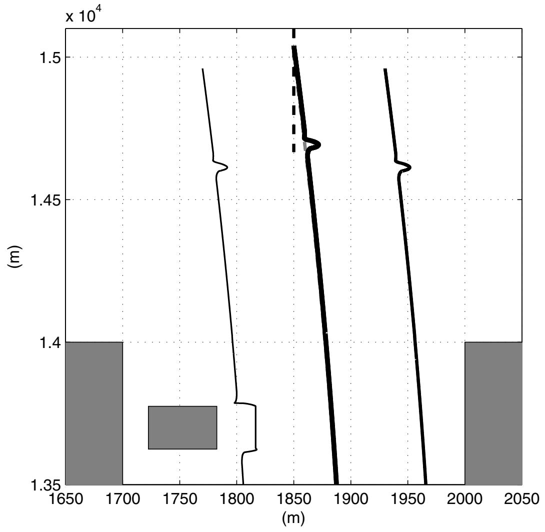


Fig. 9. Cooperative formation movement with avoiding static/moving obstacles (enlarged view)

6 Conclusions

In this chapter, we proposed a systematic approach to accommodate and achieve multiple objectives of cooperative motion, tracking of virtual command vehicle(s), collision avoidance and oscillation suppression. Simulation example A confirms the effectiveness of Lyapunov Design of multi-objective control for the single agent proposed in Section 3. Rigorous proof of incorporation of the proposed control for the single agent with the cooperative formation control is still needed. The effectiveness of the incorporation has been validated by the simulation example B. In addition, we plan to consider a variety of different feedback controllers such as dynamic, adaptive types of controllers to improve the overall performance.

References

1. Balch, T., Arkin, R.C.: Behavior-based formation control for multi-robot team. IEEE Transactions on Robotics and Automation 14(6), 926–939 (1998)
2. Lewis, M.A., Tan, K.H.: High Precision Formation Control of Mobile Robots Using Virtual Structures. Autonomous Robots 4(4), 387–403 (1997)
3. Ren, W., Beard, R.W.: Virtual Structure based Spacecraft Formation Control with Formation Feedback. In: AIAA Guidance and Control Conference, Monterey, CA, AIAA Paper no. 2002-4963 (August 2002)
4. Do, K.D., Pan, J.: Nonlinear formation control of unicycle-type mobile robots. Robotics and Autonomous Systems 55(3), 191–204 (2007)
5. Young, B.J., Beard, R.W., Kelsey, J.M.: A Control Scheme for Improving Multi-Vehicle Formation Maneuvers. In: American Control Conference, Arlington, VA, June 25-27, pp. 704–709 (2001)
6. Edwards, D.B., Bean, T., Odell, D., Anderson, M.J.: A Leader-Follower Algorithm for Multiple AUV Formations. In: Proceedings of 2004 IEEE/OES Autonomous Underwater Vehicles, Sebasco Estates, Maine, June 17-18 (2004)
7. Lawton, J., Beard, R., Young, B.: A Decentralized Approach To Formation Maneuvers. IEEE Transactions on Robotics and Automation 19(6), 933–941 (2003)
8. Tanner, H.G., Pappas, G.J., Kumar, V.: Leader-to-Formation Stability. IEEE Transactions on Robotics and Automation 20(3), 433–455 (2004)
9. Consolini, L., Morbidi, F., Prattichizzo, D., Tosques, M.: A Geometric Characterization of Leader-Follower Formation Control. In: Proc. of the IEEE International Conference on Robotics and Automation, Rome, April 10-14, 2007, pp. 2397–2402 (2007)
10. Mesbahi, M., Hadaegh, F.Y.: Graphs, matrix inequalities, and switching for the formation flying control of multiple spacecraft. In: American Control Conference, San Diego, CA, June 2-4, 1999, pp. 4148–4152 (1999)
11. Koren, Y., Borenstein, J.: Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation. In: Proceeding of the IEEE International Conference on Robotics and Automation, pp. 1398–1404 (1991)
12. Stipanovic, D.M., Shankaran, S., Tomlin, C.J.: Multi-agent avoidance control using an M-matrix property. Electronic Journal of Linear Alebra 12, 64–72 (2005)
13. Leitmann, G., Skowronski, J.: Avoidance control. Journal of Optimization Theory and Applications 23, 581–591 (1977)
14. Leitmann, G., Skowronski, J.: A note on avoidance control. Optimal Control Applications & Methods 4, 335–342 (1983)
15. Leonard, N.E., Fiorelli, E.: Virtual leaders, artificial potentials and coordinated control of groups. In: Proc. IEEE Conference on Decision and Control, Orlando, Florida, USA, December 2001, vol. 3, p. 2968–2973 (2001)

16. Do, K.D.: Bounded Controllers for Formation Stabilization of Mobile Agents With Limited Sensing Ranges. IEEE Transactions on Automatic Control 52, 569–576 (2007)
17. Mastellone, S., Stipanovic, D.M., Graunke, C.R., Intlekofer, K.A., Spong, M.W.: Formation Control and Collision Avoidance for Multi-agent Non-holonomic Systems: Theory and Experiments. The International Journal of Robotics Research 27(1), 107–126 (2008)
18. Qu, Z., Wang, J., Hull, R.A.: Cooperative Control of Dynamical Systems with Application to Autonomous Vehicles. IEEE Transactions on Automatic Control (to appear, May 2008)
19. Khalil, H.K.: Nonlinear Systems, 3rd edn. Prentice-Hall, Upper Saddle River

Maximizing Search Coverage Using Future Path Projection for Cooperative Multiple UAVs with Limited Communication Ranges

Pedro DeLima and Daniel Pack

Department of Electrical and Computer Engineering
United States Air Force Academy, CO, USA
{pedro.lima,daniel.pack}@usafa.edu

Abstract. In this chapter, we present Future Path Projection (FPP) as a novel method for multiple Unmanned Aerial Vehicles (UAVs) with limited communication ranges to cooperatively maximize the coverage of a large search area. For multiple cooperative UAVs to perform an effective search mission, the critical status and sensor information collected by each UAV must be shared with all other UAVs in the group. In an ideal environment where there is no communication limitation, all involved UAVs can share the necessary information without any constraints. In a more realistic environment, UAVs must deal with limited communication ranges. The communication range limitation, however, introduces a challenging problem for multiple UAVs to effectively cooperate. In the proposed method, each UAV constructs an individual probability distribution map of the search space which reflects predictions of the future paths of UAVs as they move beyond their communication ranges. The probability distribution map describes the likelihood of detecting targets within the search space. The overall, collective UAV search patterns are governed by decisions made by each UAV within the group, based on each individual probability distribution map. We show that the collective search patterns generated by cooperative UAVs using the proposed method significantly improve the search area coverage when compared to similar search patterns produced by other mitigation strategies designed to overcome the communication range limitation. We validate the effectiveness of the proposed path projection method using simulation results.

Keywords: cooperative, unmanned aerial vehicles, limited communication, future path projection.

1 Introduction

Significant research has been conducted on mobile multi-agent systems directed at solving problems such as target search [1], target observation [2], and cooperative transportation [3]. Of particular interest is the development of multi-agent systems composed of Unmanned Aerial Vehicles (UAVs) capable of covering vast areas using a wide range of sensors. These systems are ideal for applications such as surveillance, reconnaissance, rescue, and emergency site monitoring [4].

For an increasing number of applications, multiple UAV systems provide superior performance compared to single UAV systems by taking advantage of the redundancy, robustness, and cooperation potential of multiple systems. However, to gain the advantages of no centralized control unit, cooperation intrinsically requires some degree of communication between UAVs [5]. In practice, the cooperation potential is often not fully achieved due to restricted communication capabilities, such as limited communication ranges. Moreover, the bigger the volume of information a cooperative algorithm requires to be transferred between UAVs, the greater will be the necessary communication bandwidth. In most protocols, increased bandwidth usage ultimately leads to communication delays. When UAVs operate based on delayed information about the states of other cooperating UAVs, the environment, or the status of the global mission, the entire system performance can be degraded or its stability compromised [6], [7].

Recently, strategies that aim at mitigating the impact of communication limitations on the cooperation performance of multi-agent systems have been the focus of significant research activity. Current efforts can be generally classified into two groups: uninterrupted communication strategies that restrict the mobility of the fully connected cooperating UAVs; and unrestricted mobility strategies that provide the agents with freedom of movement but temporarily increase the volume of information exchanged when two or more UAVs happen to fly within communication range of each other.

An example of an uninterrupted communication strategy can be found in [8], where a formation control framework is applied to a set of multiple agents with the goal of balancing the intent of each unit to contribute to the collective mission and the requirement to maintain a single communication network by restricting any single agent from moving beyond the communication range of the group. Another such strategy is introduced in [9], where occasional nonlocal interactions determined by an acute angle switching algorithm are shown to generate mobile networks that robustly preserve system-wide connectivity while seeking to cover a number of regions of interest. Approaches such as these have the benefit of allowing agents to operate under a single network. If the volume of exchanged information is not prohibitive, it allows for all agents to share a common knowledge database during the entire mission. However, to achieve and maintain a single network, the mobility of each individual agent becomes limited, which can compromise the performance of the group.

In unrestricted mobility strategies, individual performance is not compromised. However, cooperation performance is generally affected negatively since, for the lack of a system-wide network, the UAVs must operate without access to a common knowledge database. Without such information, the capability of a UAV to effectively cooperate with the team becomes limited. Therefore, unrestricted mobility strategies focus on techniques that provide additional information when communication opportunities occur as two or more UAVs fly within a communication range of each other. Typical approaches involve the sharing of information in the form of past states and/or past sensor readings. In practice, the frequency and duration of such encounters tend to decrease with smaller communication ranges, the use of a smaller number of agents, and/or larger mission areas. With only short intervals to transmit the knowledge accumulated between encounters, greater bandwidth is necessary, which can lead to the negative outcomes previously outlined.

In this chapter we propose a novel unrestricted mobility strategy called Future Path Projection (FPP) that aims to mitigate the adverse effects of operating a cooperative control algorithm under scenarios with limited communication ranges. Our work suggests that the FPP strategy outperforms the existing approaches without limiting UAV mobility or increasing the volume of information shared among UAVs during their sporadic encounters. This is accomplished by having each UAV draw a probabilistic projection of the future paths of the other UAVs based on their operational behavior as they move beyond the communication range.

To compare the effectiveness of the proposed FPP strategy, we implemented two other strategies. The first is an uninterrupted communication strategy in which UAVs have their mobility restricted by an algorithm that sets maximum UAV-to-UAV distances in order to generate a flexible, reconfigurable chain structure that maintains the connection among all UAVs and the ground station. The second strategy represents a classical unrestricted mobility strategy, which we named Past Path Sharing (PPS). Similar to the proposed FPP strategy, the PPS data exchange takes place only when two or more UAVs happen to fly, based on their independent decentralized goals, within each other's communication ranges. Unlike the FPP strategy, in the PPS strategy, UAVs operate based only on deterministic past data (as opposed to probabilistic extrapolations of future behavior), but require increased data transfer rates in order to share both current and past data.

To measure the impact on the capability of multiple UAVs to cooperate under different communication ranges and to quantify the level of success of the different mitigation strategies, we challenge a set of UAVs to perform a search for mobile ground targets over a wide area using short range sensors. An efficient solution for this task requires UAVs to cooperate by coordinating their flight paths in order to provide coverage [10] by scanning the entire search area at least once and by revisiting every section as often as possible and with similar frequencies. In this manner, total area coverage and its evolution through time become two measures of the degree of cooperation of a team of UAVs.

For the experiments, we make use of the approach introduced in [11] and [12]. This algorithm seeks to maximize the likelihood of detecting a mobile ground target, and, therefore, maximize the coverage efficiency, by providing a cost function which each UAV uses to determine its next waypoint. The key characteristic of this cost function that directly impacts the efficiency of the group coverage is its ability to balance each UAV's intention to fly towards the location where the probability of detecting a target is greater, to fly away from other UAVs in order to maximize the spread of the search, and to remain inside the designated search area. Although the algorithm requires only that the position information be exchanged between UAVs, as the communication range is reduced and UAVs have fewer opportunities to exchange information, the cost function will not be able to account for information obtained by all other UAVs and, therefore, the collective search will not be as effective. More importantly, the lack of knowledge on the positions of UAVs outside the communication range deteriorates the accuracy of the target detection probability map carried by each individual UAV. The decrease in the spreading of UAVs and in the accuracy of the individual probability distribution maps makes it easy to discern the connection between the communication range, the degree of cooperation, and the ultimate sweep coverage efficiency of the group.

This chapter is organized as follows. In Section II, the prototype search algorithm is described and a series of simulation results demonstrate the negative impact of running it under progressively smaller communication ranges without any mitigation strategy. Section III describes the uninterrupted communication strategy, and Section IV introduces two unrestricted mobility strategies: PPS and the proposed FPP. In this section, both independent and comparative simulation results are provided for all three mitigation strategies. Section V closes the chapter with some final observations and conclusions.

2 Cooperative Decentralized Search Algorithm

In this chapter, we assume a target is detected when the distance between a UAV and the target is less than a given detection range. If the targets of interest are stationary, it is possible to measure the effectiveness of the search effort by the total area covered by the sensors onboard all involved UAVs and by how fast this coverage was achieved. Once the entire area is covered, the probability of detecting such targets becomes one. However, since our interest is in mobile targets, another relevant factor is how often each location is revisited.

In order to demonstrate the impact of a limited communication range to a mobile target search effort, we implemented the cooperative, decentralized search algorithm introduced in [11]. As a distributed approach, cooperative search patterns are generated by having each UAV determine its own flying path based on the group dispersion pattern, the search history of the immediate neighborhood, and the fuel consumption necessary for possible maneuvers. These factors are captured in search cost C_s shown in Equation (1). By evaluating C_s for different locations around its current position, a UAV continuously flies in the direction of the location with the minimum search cost.

$$C_S = (1 - P(i,j)) \left(\frac{1}{\sum(D_k)} + \frac{1}{\sum(D_l)} \right) |\Delta\phi_{\mathrm{UAV}}|. \tag{1}$$

Equation (1) is composed of four decision variables. The sum of the distances from a UAV to its peers and the sum of the distances between a UAV and the search area boundaries are represented by $\sum(D_k)$ and $\sum(D_l)$, respectively. The change in the heading angle required to reach a particular location is represented by $\Delta\phi_{\mathrm{UAV}}$. Finally, P is the probability matrix defined over the search area, and each element $P(i,j)$ corresponds to the probability of detecting a target in cell

i, j of the search area. Before the search starts, P is initialized with the same value for all cells, which is a function of the number of targets and the size of the search area. As a UAV flies over a cell, subjects to its sensors, and does not detect a target; the probability of a target existing in the cell drops to zero. However, as we consider mobile targets, as a UAV moves away from a visited cell, the probability of a target existing in an already searched cell gradually increases over time, making it more attractive to be revisited as time goes by.

Assuming unlimited communication range and bandwidth, by minimizing C_S each UAV attempts to fly to destinations that offer a greater probability of detecting a target (i.e., areas that have not been inspected for longer times), while maximizing distances from other UAVs, therefore maximizing the coverage of the overall search area. At the same time, each UAV also tries to remain inside the search area and maintain a constant heading, thereby minimizing fuel consumption. Note, however, that for each UAV to maintain an accurate representation of the current probabilistic distribution of the target locations in the matrix P, each UAV must know the positions of all other UAVs participating in the search effort at all times. As can be expected, when a limited communication range is considered, each UAV operates on an incomplete, inaccurate, individual probability distribution matrices P^k (where k stands for each UAV's identification number) that can differ greatly from the true probability distribution matrix P, unless some mitigation strategy is applied.

To demonstrate the impact of limited communication ranges on the performance of the cooperative search when no mitigation strategy is implemented, simulation results were collected where six UAVs attempt to cover a rectangular area of 150 km by 112.5 km. Each UAV is equipped with a sensor that is assumed to detect a target with probability one if the target is within 5 km of the UAV, and probability zero if the distance is larger. The UAVs fly at 100 km/h and calculate the cost C_S every 10 seconds at seven points equally spaced around its current location. The UAV then chooses the point with minimum C_S as its next waypoint, but its actual heading change is limited by a maximum turn rate of 2 degrees per second to simulate actual UAV dynamics limits. Since no mitigation strategy is implemented, UAVs receive only the current position information of other UAVs within their communication range, positions which are then incorporated into each individual target detection probability matrix P^k. During each simulation run, the six UAVs were launched sequentially at five minutes intervals and allowed to fly for six hours in order to allow long-term effects to be observed. All results presented in this chapter pertain to averages over 10 simulations for each scenario considered. A snapshot of a typical run showing both the *true* collective probabilistic distribution map P and the individual P^k built by one of the UAVs can be seen in Figure 1.

As previously mentioned, since the target is mobile, the probability of a target moving into an area that has already been searched increases over time, making it more likely to detect a target when the same location is searched again. However, since typically the speed of ground mobile targets is less than the speed of the UAVs, it is safe to assume that the probability of finding a target in a location

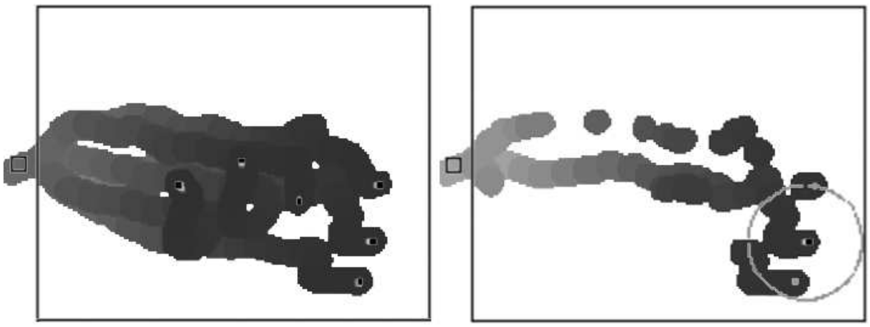


Fig. 1. Starting from an airfield (small square on the left), six UAVs (dark squares) attempt to search the designated area (wide rectangle). The complete probability matrix P is overlaid over the search area in the left frame, where the lighter the area, the greater the probability of detecting a target. The individual probability matrix P^1 built by UAV Number 1 is shown in the right frame. The circle around the UAV Number 1 indicates its maximum communication range.

that has not yet been searched will at all times be greater than a previously visited location. Therefore, instead of measuring the success of the cooperative effort as the time to locate a particular target, no targets were simulated and the degree of success of the effort is measured by observing the evolution over time of the percentage of the total area that has been searched at least once by at least one UAV. We refer to this quantity as the aggregated coverage, shown in Figure 2 as the average result of 20 simulations for each scenario. As expected, the aggregated coverage of the team decreased as the communication range was reduced, reflecting the failure of the UAVs to efficiently cooperate due to the increased disparities between each individual target detection probability map and the true probability map P. At the end of a six hour simulation run, UAVs without any communication range limitations achieved an average aggregated coverage of 98.0%, while UAVs that operated with a 20 km communication range achieved only 90.6%, those that were restricted to a 15 km communication range achieved 73.8%, and those that were restricted to a 10 km communication range achieved only 67.7%.

3 Uninterrupted Communication Strategy

The goal of this control strategy is to allow each UAV to freely move as long as a constant communication chain linking all UAVs and the ground station is maintained. In this manner, even though at times a UAV may not be able to reach a desired location within the search space, the preserved communication ensures that the true collective target detection probability map P is available to all UAVs. Since the only critical information to be transferred within the network is the position and heading of each UAV and the estimated position of

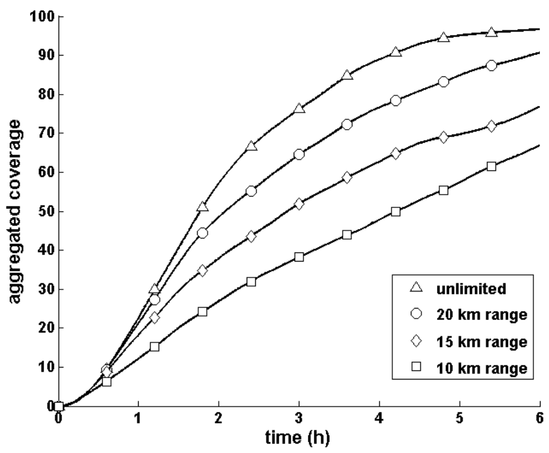


Fig. 2. Evolution of aggregated search coverage over time for scenarios with no communication limit (triangles), a 20 km communication range (circles), a 15 km communication range (diamonds), and a 10 km communication range (squares). No mitigation strategy applied.

detected targets, we assume that a single communication link between UAVs is capable of handling all the bandwidth required.

Before we introduce the modified search algorithm that will prevent UAVs from breaking the network connectivity while searching an area, we must first establish a network ranking system. The purpose of this ranking scheme is to allow each UAV to know which communication link is critical to maintain its communication with the ground station and, by extension, all other deployed UAVs. To determine its rank, each UAV must first calculate the distances between itself and the ground station, as well as all other UAVs within its communication range. If, for a particular UAV, the distance to the ground station is less than the distance to any other UAV with a lower or equal rank, then a direct link to the ground station is established and such UAV gains rank one. On the other hand, if a subset of UAVs with lower or equal ranks is closer to its location than the ground station, the UAV in question assumes a rank equal to one plus the rank of the UAV that is closer to its present location. An example of a possible configuration is shown in Figure 3. For this ranking scheme to accurately represent the necessary number of hops required for a UAV to communicate with the ground station, it is necessary that the UAVs perform periodic rank changes one at a time. For this reason, a token passing control mechanism is implemented to guarantee that the rank change occurs in a sequential manner. Since all UAVs are launched from the vicinity of the ground station, all are initialized with rank one. With each UAV periodically ranked according to the previously discussed procedure, a new term, $N(d)$, is introduced to the search cost as shown in Equation (2), defining the modified search cost C_S'.

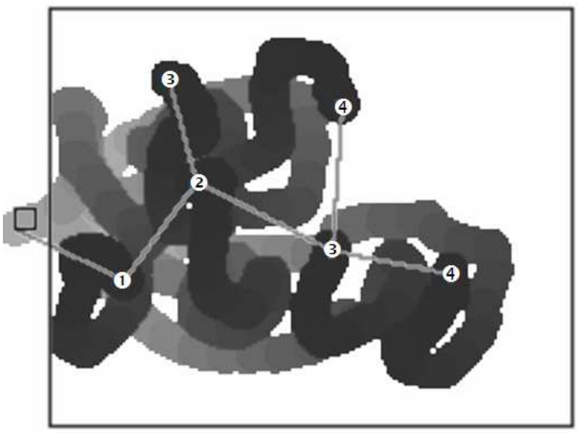


Fig. 3. A snapshot of the uninterrupted communication strategy managing a network of UAVs with a 40 km maximum communication range. The lines connecting the UAVs indicate the current information paths connecting all six UAVs and the ground station. The numbers over the UAVs are their dynamically adjusted ranks.

$$C_S^{'} = (1 - P(i,j)) \left(\frac{1}{\sum(D_k)} + \frac{1}{\sum(D_l)} \right) |\Delta\phi_{UAV}|(1 + N(d)). \tag{2}$$

The purpose of $N(d)$ is to prevent a UAV from choosing a heading that will cause a break in the communication network while preserving most of the search mobility autonomy. As shown in Figure 4, the value of the term N changes depending on the value of d, the distance from a UAV to the closest UAV with smaller rank (or distance to the ground station if the UAV's rank is one). For a distance d beyond the communication range cr, the cost factor $N(d)$ starts at a very large number γ and linearly increases with d as shown in Equation (3). For $N(d)$ between zero and the point α, $N(d)$ is zero and therefore imposes no restriction to the search process. The region between α and cr generates a softer early response region that provides gradual cost impact to a UAV in order to allow it to plot feasible flight trajectories that allow it to remain within the communication range. To minimize the mobility restrictions imposed by this strategy, the value of α is platform dependent and should be set as high as admissible by the dynamics of the UAV. The values that $N(d)$ assumes within this region are given by Equation (3).

$$N = \begin{cases} 0 & \text{, if } d \le \alpha \\ min\left[\gamma, tan\left(\frac{(d-\alpha)*\pi}{2*(cr-\alpha)}\right)\right] & \text{, if } \alpha < d \le cr \\ \gamma(1 + d - cr) & \text{, otw.} \end{cases} \tag{3}$$

Performing simulations with the same parameters used in the previous section, using this uninterrupted communication strategy the average final aggregated coverage was 95.0% for a maximum communication range of 50 km and 89.7%

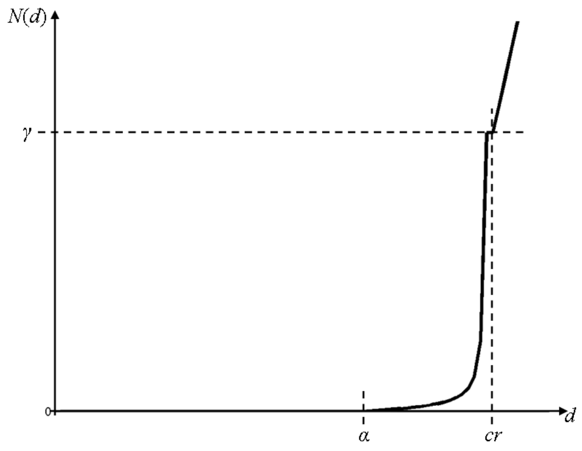


Fig. 4. Shape of $N(d)$ curve for arbitrary α, cr, and γ

for a range of 40 km. For communication ranges of 30 km and smaller, even if the algorithm resorts to a single line of UAVs stretching from the ground station, it is not geometrically possible to reach full aggregated coverage given the dimensions of the search area. If attempted, an aggregated coverage of 34.8% is reached for the 20 km communication range scenario, and only 8.45% for the 10 km one.

4 Unrestricted Mobility Strategies

For the search of vast areas with multiple UAVs subject to limited communication, a different set of strategies provides unrestricted search mobility by allowing communication to occur only sporadically when the search paths of two UAVs, determined by the original search cost introduced in Section II, happen to be close enough to allow data to be transferred. Due to the sporadic characteristic of the communication, all UAVs do not have access to the status of the complete search coverage effort of the group. If the content of the communications among UAVs is limited to their current positions and headings, as implemented in the previous section, it is clear to see that as the communication range is reduced, each individual probability distribution map $P^k(i,j)$ will become increasingly different from the other individual maps as well as from the true collective probability distribution map P. Two strategies, PPS and FPP, are proposed to mitigate the loss of search efficiency, caused by the disparities between each individual probability map, while still maintaining full mobility for the minimization of C_s. Both are described in detail in the following subsections.

4.1 Past Path Sharing

The PPS strategy approaches the problem by taking advantage of the chance encounters between UAVs to share past knowledge in order to improve the accuracy of their respective individual probability maps $P^k(i,j)$. Ideally, the most complete and accurate knowledge would be attained if the communicating UAVs could share their entire individual probability maps $P^k(i,j)$. However, in order to provide the minimum level of local accuracy, such maps become very large matrices in order to represent entire search areas, resulting in a severe increase in communication bandwidth requirements. Instead, in the proposed PPS, during such chance encounters each UAV transmits within each communication interval its current position and a set of past known visited positions (by itself or others) along with the time they were visited, which is then transformed by the receiving UAV into modifications in its own $P^k(i,j)$.

Although this approach requires an increase in the amount of information transferred between UAVs, the size of the set of past locations and sensing times transferred within each interval can be adjusted according to the bandwidth limitations of the communication network. Furthermore, after a packet of data is successfully sent, the next packet includes the UAV's updated current position and the next set of past values that pertain to even more distant instants in time. In this manner, even for scenarios that require low bandwidth, the accuracy of the individual $P^k(i,j)$ of the communicating UAVs is periodically improved throughout the time the UAVs remain within communication range of each other.

In order to store and provide the past information, each UAV maintains a table of past visited positions. Each line of the table pertains to a particular moment in time and contains the positions of the individual UAV and the positions of all other UAVs with known positions, leaving empty the cells for which no knowledge is available. Knowledge of a UAV position can be gained either directly by receiving position data from a communicating UAV, or indirectly through the sharing of tables among communicating UAVs. With a set frequency, synchronized among all UAVs, all cells of the table are shifted down, the oldest line at the bottom of the table is discarded, and the current known positions are added to the top of the table.

In actual implementation, when two UAVs happen to fly within the communication range of each other, each exchanges packets containing its current position, heading, and lines from the table, starting from the top (most recent) and moving to the bottom with each communication opportunity. Note that as the communication range decreases with respect to the overall search area, less knowledge can be obtained and shared, causing the tables to become increasingly sparse, a situation in which the communication toll can be reduced by the sharing of only non-empty cells. Also, increasing the maximum length of the table allows more knowledge to be stored from times further in the past while requiring additional memory usage by each UAV. This also allows more information to be exchanged when the UAVs' flight paths happen to remain close to each other for extended periods.

To demonstrate the impact of the PPS method on the final coverage capabilities of the group, the strategy was implemented in the same simulation environment used in the previous sections. Figure 5 shows a snapshot of the coverage as a UAV updates its map with past positions of a UAV within its communication

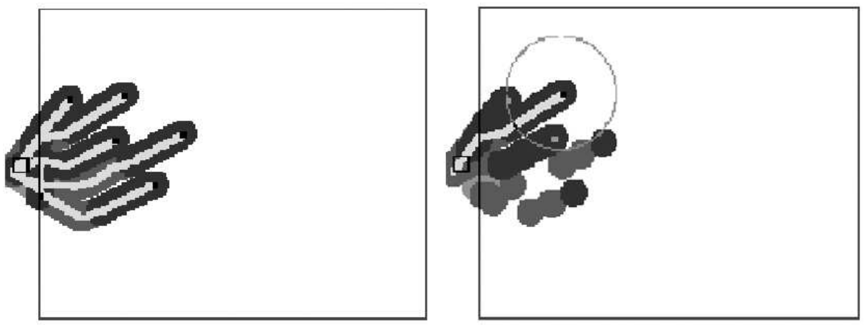


Fig. 5. A simulation snapshot for the PPS strategy. The left frame shows the total P, and the right frame shows the P^k of one of the UAVs where past information on locations visited by other UAVs are being obtained through communication with its neighbors.

range. As we vary the communication range, an aggregated coverage of 95.0% was reached when a 20 km communication range was available, while only 78.2% was achieved at the 10 km communication range.

4.2 Future Path Projection

Different from the PPS strategy, which required some level of additional communication load, in the FPP strategy it is not necessary that any information, other than their current positions and headings, be exchanged between UAVs during their chance encounters. In the PPS strategy, past information is used with the goal of making each individual $P^k(i,j)$ approach the collective probabilistic map P. On the other hand, FPP focuses on the instant when one UAV departs from the communication range of another and modifies its $P^k(i,j)$ based on the chance of the departing UAV visiting areas in the future, given the last received position and heading. This process is illustrated in Figure 6.

Since UAVs are subject to dynamic limitations on their turn radius, and since the original search cost C_S has as one of its goals to maintain the same heading in order to conserve fuel, it is reasonable to assume that the more likely path of a UAV is in the direction of its final received heading. To also incorporate the course changes that can occur due to the other factors within C_S, the projected path fans out from the last received heading direction with a likelihood that is inversely proportional to the required angular change in the heading. Finally, since the projection becomes less accurate the further it extends into the future, the likelihood that an area is visited in the future also is reduced proportional to its distance from the last received position. The end result is the cone shape shown in Figure 6 with maximum likelihood (intensity) in the area immediately ahead of the point of last communication and with gradually reduced likelihood as either the angle of the heading, or the distance from the position of last communication is increased over time.

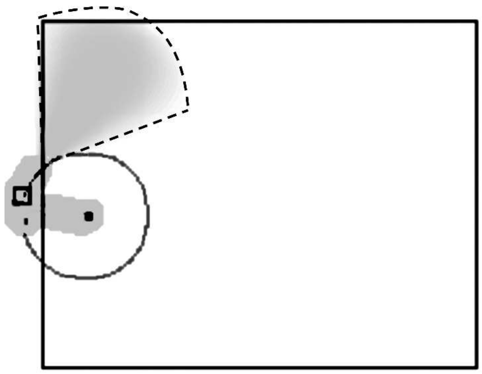


Fig. 6. As one UAV departs from the communication range of another, the FPP strategy augments P^k with a probabilistic projection (dotted line area in the picture) of the departing UAV's future path

A simulation with the same parameters used for the test of PPS was conducted using FPP. In this simulation, the FPP strategy was allowed to extend the projection up to 60 km away from the point at which communication was lost with an included angle of 30 degrees. At the end of the six hours of simulation, the application of the FPP mitigation strategy resulted in a comparable 93.9% average aggregated coverage for the 20 km communication range scenario, and a vastly superior response of 92.1% for the 10 km communication range scenario. All coverage values for both the PPS and FPP cases were obtained from the aggregated effective coverage represented by P, as opposed to the individual perception of the target detection probability map of each P^k.

Having introduced all three mitigation strategies, Figure 7 shows a comparison of the evolution of the aggregated coverage over time for a more demanding scenario where communication only takes place between UAVs less than 10 km apart. For reference, Figure 7 also displays the performance of the cooperative decentralized search algorithm operating with unlimited communication range and with the restricted communication range but without any mitigation strategy. As expected for communication ranges at or below 30 km, the uninterrupted communication strategy performs even worse than when no mitigation strategy is applied, since at this range the network, although flexible, is geometrically incapable of stretching over the entire area without losing communication with the ground base. Applying both PPS and FPP strategies in parallel over the same Pk map resulted in a performance level between the full communication and no mitigation strategy scenarios, with FPP providing a superior average aggregated coverage throughout the entire six hours. When PPS and FPP were applied simultaneously, the end result was statistically identical to the result when FPP was applied alone, suggesting that if FPP is applied at this communication range there is no benefit from increasing the communication bandwidth usage to implement PPS.

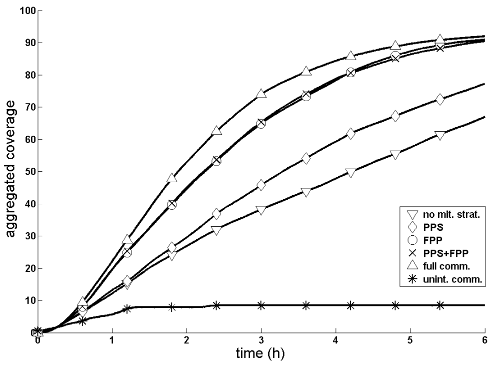


Fig. 7. Comparison of the performance of the cooperative sweep search with a 10 km maximum communication range using the different mitigation strategies: uninterrupted communication (asterisk), PPS (diamond), FPP (circle), and the combination of PPS and FPP (×). For reference, curves from the system operating without any mitigation strategy (down-facing triangle) and with full communication (up-facing triangle) are also provided.

Figure 8 shows results equivalent to when a 20 km communication radius was applied. As before, the uninterrupted communication strategy under-performs since the very limited communication range still prevents it from achieving full coverage. Comparing the performance of PPS and FPP, we notice that at this communication range FPP outperforms PPS until approximately three and half hours after the search starts. After that point, PPS provides superior aggregated coverage. This behavior can be explained by the fact that in the beginning of

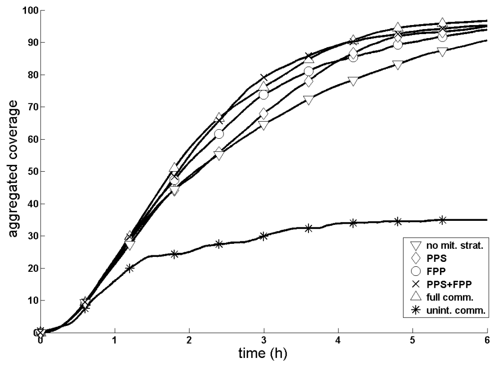


Fig. 8. Comparison of the performance of the cooperative sweep search with a 20 km maximum communication range using the different mitigation strategies: uninterrupted communication (asterisk), PPS (diamond), FPP (circle), and the combination of PPS and FPP (×). For reference, curves from the system operating without any mitigation strategy (down-facing triangle) and under full communication (up-facing triangle) are also provided.

the simulation UAVs are less likely to deviate from the paths predicted by FPP based on their knowledge of the search area since they all depart from the same airbase. In such situations, the choice of future waypoints will be directly influenced by the projection of future paths of its neighbors, whereas knowledge of a neighbor's past positions will only impact the decision making of a UAV when it changes course and if it approaches the vicinity of such locations. Different from the results for the shorter communication range, at 20 km the benefit of combining both PPS and FPP strategies becomes evident as the end result shows the rapid initial increase in the aggregated coverage brought by FPP and the ultimately superior result of PPS. This result suggests that combining the two techniques can be significantly advantageous on scenarios with extremely constrained communication ranges. Nevertheless, it is important to note that any usage of PPS comes with an increase in the bandwidth demand, which may not be acceptable due to the negative consequences mentioned earlier in the chapter.

5 Conclusion

In this chapter, we showed the validity and effectiveness of the FPP strategy to update search maps which are used by a set of cooperative UAVs to collectively, in a probabilistic context, maximize the coverage of a large search area under communication range limitations. The unique search problem involves mobile ground targets emitting radio frequency signals that can turn on and off. The nature of the targets demands a search technique that causes the cooperative UAVs to not only efficiently cover the search area, but also to revisit areas previously searched with unexpected frequencies and patterns. The proposed method has been shown to be effective in maintaining good search coverage with a wide variety of communication ranges, demonstrating its value especially when the communication range for each UAV is relatively small. We presented a comparative study on the performance of the proposed method against two other methods to accommodate the communication limitation problem, one based on an uninterrupted communication principle and the second on an uninterrupted mobility principle. Using the search task as a testbed, we demonstrated that the proposed FPP algorithm managed to provide a group of decentralized UAVs operating under a limited communication range with a degree of cooperation comparable to the one attained under the assumption of unlimited communication range.

References

1. Bourgault, F., Furukawa, T., Durrant-Whyte, H.T.: Coordinated decentralized search for a lost target in a bayesian world. In: Proc. IEEE/RJS Conference on Intelligent Robotic Systems, Las Vegas, NV, pp. 48–53 (October 2003)
2. Parker, L., Emmons, B.: Cooperative multi-robot observation of multiple moving targets. In: Proc. International Conference on Robotics and Automation, Albuquerque, NM, April 1997, pp. 2082–2089 (1997)

3. Miyata, N., Ota, J., Arai, T., Asama, H.: Cooperative transport by multiple mobile robots in unknown static environments associated with real-time task assignment. IEEE Transactions on Robotics and Automation 18(5), 769–780 (2002)
4. Pack, D., Sward, R., Cooper, S., York, G.: Searching, detecting, monitoring, and tracking emergency sites using multiple Unmanned Aerial Vehicles. In: Proc. AIAA Infotech at Aerospace Conference on Collaboration in UAV Systems, Washington, D.C (September 2005)
5. Klavins, E.: Communication complexity of multi-robot systems. In: Proc. Workshop on Algorithmic Foundations of Robotics, Nice, France (December 2002)
6. Yook, J.K., Tilbury, D.M., Soparkar, N.R.: Trading computation for bandwidth: reducing communication in distributed control systems using state estimators. IEEE Transactions on Control Systems Technology 10(4), 503–518 (2002)
7. Chan, H., Ozguner, U.: Closed-loop control of systems over a communication network with queues. International Journal of Control 62(3), 493–510 (1995)
8. Balch, T., Arkin, R.C.: Behavior-based formation control for multirobot teams. IEEE Transactions on Robotics and Automation 14(6), 926–939 (1998)
9. Shucker, B., Murphey, T., Bennett, J.K.: A method of cooperative control using occasional non-local interactions. In: Proc. International Conference on Robotics and Automation, Orlando, FL, pp. 1324–1329 (May 2006)
10. Gage, D.W.: Command control for many-robot systems. In: Proc. AUVS Technical Symposium, Hunstville Alabama, USA, pp. 22–24 (June 1992); Unmanned Systems Magazine 10(4), 28–34 (reprint, 1992)
11. York, G., Pack, D., Harder, J.: Comparison of cooperative search algorithms for mobile RF targets using multiple Unmanned Aerial Vehicles. In: Grundel, D., Murphey, R., Pardalos, P., Prokopyev, O. (eds.) Advances in Cooperative Control and Optimization, pp. 387–403. Kluwer Academic Publishers, Amsterdam (2006)
12. Pack, D.J., York, G.W.P.: Developing a control architecture for multiple unmanned aerial vehicles to search and localize RF timevarying mobile targets: part I. In: Proc. International Conference on Robotics and Automation, Barcelona, Spain, April 2005, pp. 3965–3970 (2005)

Operator Aided Decision Processes for UAVs in a Stochastic Environment

John J. Baker[1], Raymond Holsapple[2], Anouck Girard[1], Meir Pachter[3], and Phillip Chandler[2]

[1] The University of Michigan
[2] Air Force Research Laboratory
[3] Air Force Institute of Technology

Abstract. In this chapter, we consider an Intelligence, Surveillance, and Reconnaissance (ISR) scenario where a human operator is tasked to provide feedback regarding the nature of some objects of interest (OOI). The feedback is relayed to the stochastic controller of an unmanned aerial vehicle (UAV), which must determine an appropriate mission plan. A small aerial vehicle (SAV) loiters at a high altitude where it may survey a large territory. An operator decides which objects in the SAV's field of view are of interest and which are not. Then a team of micro aerial vehicles (MAVs) are assigned individual tours to survey the OOI at a low altitude. As a MAV flies over an OOI, the operator must decide if the OOI has a feature that defines it as a target. The key parameters are the operator's response and the time taken for the operator to respond. The stochastic controller takes these into account and performs an analysis to compute expected information gain of a revisit. In previous studies automatic target recognition (ATR) was used for making some decisions in the SAV and the MAVs. This chapter investigates the use of human feedback alone for target recognition. Different methods for calculating expected information gain are examined and compared against a maximum operator delay revisit threshold.

Keywords: Dynamic Programming, Stochastic, ISR, Task Assignment.

1 Introduction

1.1 COUNTER Scenario

First it is necessary to discuss COUNTER, the scenario these algorithms are designed for. COUNTER [1] is an acronym for Cooperative Operations in UrbaN TERrain, an Air Force program using a team of UAVs to investigate task assignment and path planning algorithms for use in ISR missions in urban areas. The COUNTER team is a collection of Air Force military, Air Force civilian and contractor engineers and is based at the Air Force Research Laboratory's (AFRL) Control Sciences Center of Excellence.

COUNTER uses a team of UAVs, one small aerial vehicle (SAV) and four micro aerial vehicles (MAVs). The SAV loiters over the urban area at 1000-1500 feet above ground level (AGL), while an operator surveys the live video feed from the SAV for objects of interest. We assume that the objects remain stationary. After an operator selects a collection of OOI, a task assignment algorithm assigns a tour to each MAV that is to be launched. The MAVs fly at a much lower altitude (50-150 feet AGL) allowing them to investigate the OOI close up and at an acute angle which may permit them to see into vehicles and under tarpaulins and camouflage nets. Like the SAV, each MAV is equipped with front and side facing video cameras. This video feed is relayed back to a ground control station where an operator attempts to classify the objects in real time as the MAVs survey the collection of OOI.

At this point in the scenario a very important assumption becomes active. The operator is not asked to give a response whether or not the OOI is a target or a non-target based on his or her inspection of the video. Instead, the operator is asked whether or not he or she has seen a unique distinguishing feature that has been described to him or her prior to the mission. The operator may even have a sample picture of such a feature to refer to during the mission, as was done in a prior Air Force research effort [2]. The assumption about this feature is that it uniquely separates targets from non-targets. An example of such a feature may be a large gun mounted in the rear of a truck, but there are many such possible features.

1.2 Stochastic Controller

It was anticipated that a human operator would be overwhelmed if expected to manage MAV tour reassignments while simultaneously attempting to classify OOI as targets or non-targets, so a stochastic controller was developed to pick up this high level responsibility [3]. Dynamic programming is used to solve this decision making with uncertainty problem. Developed with great detail in [3], the state of the dynamic program is the amount of remaining fuel reserve for revisiting an OOI, which we shall just refer to as reserve.

As mentioned before, the OOI are stationary. One way to extend this work to non-stationary objects might be to amend the way the cost of a revisit is computed. For instance, one might consider the cost to be an expected value, using some probability distribution of the location of the OOI based on prior measurements. Although this is not considered in this effort, it is a possible future direction.

In addition to the reserve, the stochastic controller also makes use of the operator's response, the amount of time the operator took to make the response (operator delay), and the number of remaining objects in the MAV's tour. Given the operator's decision, the controller performs an information gain analysis where it computes an expected reward for performing a revisit using a priori probabilities that were determined experimentally [2]. These probabilities characterize the target density and operator's decision behavior. Given the operator's response, operator delay, reserve and remaining objects, the controller will decide whether a MAV should revisit an object and modify the MAV's flight plan accordingly.

Revisits are often useful because they can provide additional information regarding an OOI at a different approach angle. For example, a feature may only be visible from the rear of the OOI, so if the MAV approaches from the front, it may never see the definitive feature. In this case there may (or may not) be sufficient information gain to perform a revisit, depending on the values of the inputs to the controller. In the case of extremely low target density, there might be sufficient information gain to perform a revisit even in cases where the operator says that he or she has seen the distinguishing feature on the initial inspection. This is due to the uncertainty in the operator's responses, which will be discussed later.

It should be pointed out that each MAV has a finite fuel reserve set aside for revisits. This reserve is the state of the dynamic program that is formulated to solve this optimization problem. The solution of the dynamic program generates a matrix of cost thresholds that will be used in the decision process. At the moment an operator gives a response (which coincides with the moment of decision by the controller), the expected cost of a revisit is compared to the cost threshold which was computed in the information gain analysis. At this point the control decision is simply a table look up. If the expected cost is less than the cost threshold, the MAV will revisit the OOI.

1.3 Motivation

In previous publications [3,4,5,6,7] a probabilistic method was developed to determine the reward values that are used in the cost function of the dynamic program. The probabilities were developed based off of the assumption that the MAV had an automatic target recognition (ATR) device used for classifying OOI. The MAV would only defer to an operator for classification if the ATR encountered an ambiguous feature.

Optical feature recognition would be problematic in application because of additional payload constraints, communication unreliability, and highly variable lighting conditions. Additionally, and perhaps more practically, the MAVs used for COUNTER are simply not equipped with such a device. This was the motivation for a system that instead of employing an ATR, relies solely on an operator for OOI classification.

1.4 Original Contributions

In this chapter, three different expected reward functions will be considered. These functions will rely on the operator's response from the flyover of an OOI instead of utilizing an ATR. The number of a posteriori probabilities needed for the reward functions will be developed. Then an analysis will be performed comparing the reward methods against a benchmark and each other. Performance of each method will be discussed and a recommendation for future work will be made.

2 Formulation

2.1 Definition of Terms

In this system there are three main events, and they can be treated as boolean operators: Feature Visibility, Operator Response, and Target Truth Status (whether or not the OOI is an actual target). These are considered boolean because only absolutes are considered. For example, the feature is either visible or not visible, the operator indicates feature or no feature, and an OOI is either a target or it is not. Each MAV visit can be thought of in terms of some combination of said events. The first and second visit of a MAV will be treated as independent events, so the subscripts denoting the visit are only important when probabilities involve events from both visits. Below is a list of nomenclature used throughout the chapter.

$$\begin{aligned} v &= \text{Feature Visibility} \\ r &= \text{Operator Response} \\ t &= \text{Target Truth Status} \\ \theta &= \text{Feature Visibility Interval} \\ \text{subscript}_1 &= \text{First Visit} \\ \text{subscript}_2 &= \text{Second Visit} \\ \text{subscript}_T &= \text{Boolean True} \\ \text{subscript}_F &= \text{Boolean False} \\ P_{TO} &= \text{Probability of Feature Detection} \\ 1 - P_{FTO} &= \text{Probability of Feature False Alarm} \end{aligned}$$

2.2 A Priori Probabilities

The probability that an object of interest is a target is assumed a priori:

$$P(t_T) = p\,, \tag{1}$$
$$P(t_F) = 1 - p\,. \tag{2}$$

In real scenarios, the probability that an OOI is an actual target is very low. It could be on the order of one out of ten thousand in extremely cluttered urban areas, Baghdad for example. For the purposes of this chapter, a much higher probability of one out of twenty was used so that simulation time would be minimal. Moreover, this is the a priori probability used in COUNTER flight tests due to practicality issues. An operator confusion matrix developed in [3] is a way of depicting the stochastic behavior of a human operator given a collection of probabilistic events. In [3,4,5,6,7], the probabilistic event corresponding to any given operator response was simply the target truth status. This truth status is of course unknown to the operator but has a probability distribution described by equations (1) and (2).

Table 1. Operator Confusion Matrix

	r_T	r_F
$P(r \mid v_T \cap t_T)$	P_{TO}	$1 - P_{TO}$
$P(r \mid v_T \cap t_F)$	undefined	undefined
$P(r \mid v_F \cap t_T)$	$1 - P_{FTO}$	P_{FTO}
$P(r \mid v_F \cap t_F)$	$1 - P_{FTO}$	P_{FTO}

The operator confusion matrix in this chapter is different from the previous version because it accounts for an additional stochastic event, the feature visibility. It is important to note that a feature cannot be visible on an OOI that is not an actual target so the row of $v = T$ and $t = F$ is undefined by nature. A design choice was made that the probability that an operator responds that the OOI has the feature, when no feature is actually visible and the OOI is in fact a target, would simply be the probability of a false alarm. The issue is that even if the operator's implication (the OOI is a target) is technically correct, it is based on no visual evidence and should be treated as a false alarm, which is how it will be modeled in this chapter. The operator confusion matrix actually represents a collection of conditional probabilities, and it is shown in Table 1.

Here we should note a couple of things. First, P_{TO} and P_{FTO} are functions of the assumed target distribution given by equations (1) and (2). Secondly, it is assumed that P_{TO} and P_{FTO} are affected by the operator's workload. Nominally we suggest that as an operator's workload increases, the probability of detection should decrease while the probability of false alarm increases. Experiments performed by the Human Effectiveness Directorate of AFRL have investigated this very topic [2]; however, the results of the report are limited to exclude the public for now.

Next consider the possible feature visibility outcomes from a MAV flying over an OOI. Each actual target is modeled as having an angle range over which the definitive feature is visible, θ. The range of visibility divided by the total range of angles the OOI can be viewed from is the conditional probability that a feature is visible given that it is a target. Table 2 lists this set of a priori conditional probabilities.

In the case of two visits, the system is modeled such that the MAVs perform their second visit from the opposite angle of approach. This means that if a feature was visible on the first pass, it should not be on the second and vice versa. From this, the conditional probabilities of feature visibility for two visits

Table 2. Visibility Given Target Truth Status

	v_T	v_F
$P(v \mid t_T)$	$\theta/2\pi$	$1 - \theta/2\pi$
$P(v \mid t_F)$	0	1

Table 3. Visibility Given Target Truth Status

	v_{1T}, v_{2T}	v_{1T}, v_{2F}	v_{1F}, v_{2T}	v_{1F}, v_{2F}
$P\left(v_1 \cap v_2 \vert t_T\right)$	0	$\theta/2\pi$	$\theta/2\pi$	$1-\theta/\pi$
$P\left(v_1 \cap v_2 \vert t_F\right)$	0	0	0	1

given the target truth status may be inferred. These conditional probabilities are given in Table 3.

2.3 Reward Multiplier Probabilities

The motivation of the following exercise is to determine the probability of the operator's decision and target feature visibility on a second visit given the decision and visibility from the first visit. These probabilities will be used as gains applied against reward values so that they are weighted according to the probability that they will occur. To do this, the probabilities will be broken down into their constituent a priori sub-probability combinations. Before describing how that is done, we first note that we seek the following:

$$\begin{aligned} P(r_2 \cap v_2 | r_1 \cap v_1) &= \frac{P((r_2 \cap v_2) \cap (r_1 \cap v_1))}{P(r_1 \cap v_1)} \\ &= \frac{P(r_2 \cap v_2 \cap r_1 \cap v_1 \cap t_T) + P(r_2 \cap v_2 \cap r_1 \cap v_1 \cap t_F)}{P(r_1 \cap v_1 \cap t_T) + P(r_1 \cap v_1 \cap t_F)} . \end{aligned} \tag{3}$$

The terms in the denominator can be resolved into their a priori constituent parts as follows:

$$\begin{aligned} P(r_1 \cap v_1 \cap t) &= P(r_1 | v_1 \cap t) P(v_1 \cap t) \\ &= P(r_1 | v_1 \cap t) P(v_1 | t) P(t) && (4) \\ &\equiv \widetilde{P}(t) \, . && (5) \end{aligned}$$

The terms in the numerator may also be resolved to a priori constituents. To do this we will begin with a definition.

Definition 1. Two events E_1 and E_2 are conditionally independent of event E_3 if and only if

$$P(E_1 \cap E_2 | E_3) = P(E_1 | E_3) P(E_2 | E_3) \, , \tag{6}$$

or equivalently

$$P(E_1 | E_2 \cap E_3) = P(E_1 | E_3) \, . \tag{7}$$

While breaking the terms in the numerator into their constituent parts, we must assume conditional independence several times. In the equations below, conditional independence is assumed as we proceed from equation (8) to equation (9) and from equation (10) to equation (11). This assumption is not counter intuitive, as it makes sense to assume that the operator's response and feature visibility pairs from the first and second visits should be conditionally independent of each other given the target truth status. If conditional independence is not asserted and conventional conditional probability rules are applied, then the operator's response and feature visibility from the first visit would depend on those of the second visit which violates temporal causality. With all this considered, the terms in the numerator are broken down as follows:

$$\begin{aligned}
P(r_2 \cap v_2 \cap r_1 \cap v_1 \cap t) &= P((r_2 \cap v_2) \cap (r_1 \cap v_1)|t)P(t) \qquad (8)\\
&= P(r_2 \cap v_2|t)P(r_1 \cap v_1|t)P(t) \qquad (9)\\
&= \frac{P(r_2 \cap v_2 \cap t)}{P(t)} \frac{P(r_1 \cap v_1 \cap t)}{P(t)} P(t)\\
&= \frac{P(r_2 \cap v_2 \cap t)P(r_1 \cap v_1 \cap t)}{P(t)}\\
&= \frac{P(r_2|v_2 \cap t)P(v_2 \cap t)P(r_1|v_1 \cap t)P(v_1 \cap t)}{P(t)}\\
&= P(r_2|v_2 \cap t)P(r_1|v_1 \cap t)\frac{P(v_2|t)P(t)}{P(t)}P(v_1|t)P(t)\\
&= P(r_1|v_1 \cap t)P(r_2|v_2 \cap t)P(v_1|t)P(v_2|t)P(t) \qquad (10)\\
&= P(r_1|v_1 \cap t)P(r_2|v_2 \cap t)P(v_1 \cap v_2|t)P(t) \qquad (11)\\
&\equiv \widehat{P}(t)\,. \qquad (12)
\end{aligned}$$

Finally, we may use equations (4) and (11) to assemble the a priori constituent forms of the numerator and denominator in equation (3). For brevity we will write (3) using the equivalent definitions given by equations (5) and (12),

$$P(r_2 \cap v_2|r_1 \cap v_1) = \frac{\widehat{P}(t_T) + \widehat{P}(t_F)}{\widetilde{P}(t_T) + \widetilde{P}(t_F)}\,. \qquad (13)$$

2.4 Reward Probabilities

The two conditional probabilities used to compute the reward values must also be determined. These two probabilities are $P(t|r_1 \cap v_1)$ and $P(t|r_1 \cap v_1 \cap r_2 \cap v_2)$, and they are decomposed into constituents in the equations below. Using equations (4) and (5), we have

$$\begin{aligned}
P(t|r_1 \cap v_1) &= \frac{P(t \cap r_1 \cap v_1)}{P(r_1 \cap v_1)}\\
&= \frac{P(r_1 \cap v_1 \cap t)}{P(r_1 \cap v_1 \cap t_T) + P(r_1 \cap v_1 \cap t_F)}\\
&= \frac{\widetilde{P}(t)}{\widetilde{P}(t_T) + \widetilde{P}(t_F)}\,. \qquad (14)
\end{aligned}$$

Furthermore, using equations (11) and (12), the more complicated of the two conditional probabilities becomes

$$\begin{aligned} P(t|r_1 \cap v_1 \cap r_2 \cap v_2) &= \frac{P(t \cap r_1 \cap v_1 \cap r_2 \cap v_2)}{P(r_1 \cap v_1 \cap r_2 \cap v_2)} \\ &= \frac{P(r_2 \cap v_2 \cap r_1 \cap v_1 \cap t)}{P(r_2 \cap v_2 \cap r_1 \cap v_1 \cap t_T) + P(r_2 \cap v_2 \cap r_1 \cap v_1 \cap t_F)} \\ &= \frac{\widehat{P}(t)}{\widehat{P}(t_T) + \widehat{P}(t_F)} \,. \end{aligned} \tag{15}$$

Since each event is represented by a boolean value, the number of equations needed to describe the system is simply 2^n, where n is the number of boolean events involved. For example, $n = 5$ in equation (15), so there are thirty-two equations like equation (15).

2.5 Reward Functions

Two information theory reward functions from a previous effort [3] and an additional method, where discrete reward values are assigned, will be considered and evaluated. The range of values from the rewards is essentially arbitrary but provides a basis for comparison of possible outcomes. These rewards will then be scaled respectively by equation (13) from the previous section. For the purpose of brevity while describing the three methods, let $A = r_1 \cap v_1$ and $B = r_2 \cap v_2$.

Method 1. We begin by defining some conditional probabilities:

$$P_{11} = P(t_T|A)\,, \tag{16}$$
$$P_{12} = P(t_F|A)\,, \tag{17}$$
$$P_{13} = P(t_T|A \cap B)\,, \tag{18}$$
$$P_{14} = P(t_F|A \cap B)\,. \tag{19}$$

Then the reward value using Method 1 can be expressed using equations (16) - (19) and is given by the following:

$$\mathrm{R}_1 = \log\left(\frac{P_{13}}{P_{14}} + \frac{P_{14}}{P_{13}}\right) - \log\left(\frac{P_{11}}{P_{12}} + \frac{P_{12}}{P_{11}}\right)\,. \tag{20}$$

Method 2. We begin by defining some conditional probabilities:

$$P_{21} = P(t_T \cap A)\,, \tag{21}$$
$$P_{22} = P(t_F \cap A)\,, \tag{22}$$
$$P_{23} = P(t_T \cap A \cap B)\,, \tag{23}$$
$$P_{24} = P(t_F \cap A \cap B)\,. \tag{24}$$

Then the reward value using Method 2 can be expressed using equations (1), (2), (16) - (19), (21) - (24) and is given by the following:

$$\mathrm{R}_2 = \left(P_{23}\log\left(\frac{P_{13}}{p}\right) + P_{24}\log\left(\frac{P_{14}}{1-p}\right)\right) - \left(P_{21}\log\left(\frac{P_{11}}{p}\right) + P_{22}\log\left(\frac{P_{12}}{1-p}\right)\right) . \tag{25}$$

Method 3. For method three, discrete values were chosen for the sixteen combinations of operator response and feature visibility for both visits. A value of zero was assigned if the outcome was impossible, such as the feature being visible on both passes. A small reward was given if the operator was incorrect on both passes but the situation was possible, or when the operator was correct on the first visit but incorrect on the second visit. Moderate rewards were assigned for the operator being incorrect on the first pass, but correct on the second. The largest reward was given when the operator was correct on both visits. One benefit of this method is that there is never a negative reward value, which is possible with information theory and causes saturation within the revisit threshold function which is discussed in the following subsection.

Benchmark. A comprehensive study using Monte Carlo simulations was done to determine the mean operator response delay. A time threshold slightly greater than the mean delay was chosen so that it would envelope a majority of the operator delay times. Thus, if the operator delay was less then the threshold, the UAV would perform a revisit.

2.6 Threshold Surface Plots

Analyzing the surfaces provided by the threshold function provides a preliminary indication of how the system will respond. The threshold surface is primarily based upon the operator's response, how much reserve fuel remains, the operator delay, how many OOI are left to visit and the expected reward [5,6,7]. These values locate the threshold for a specific revisit. Essentially, the system determines a cost for revisit, and if the cost is less than the local threshold value on the surface, the MAV will perform the revisit. Having a deep intuitive understanding of how the shape of the threshold surface impacts the response of the system is useful but not necessary. In this analysis, we are ultimately concerned with any saturation that occurs along the threshold value axis. Large amounts of saturation indicate a bias towards a certain stochastic controller decision.

For Method 1, Figure 1 indicates that the case where the operator responds with true results in a very saturated threshold surface. This means that when the operator responds that they see a target, the stochastic controller will virtually always perform a revisit. Whereas the case where the operator responds false seen in Figure 2 is not as predictable.

For Method 2, Figure 3 indicates that if the operator responds with true, then the stochastic controller will most likely not make a revisit, whereas if

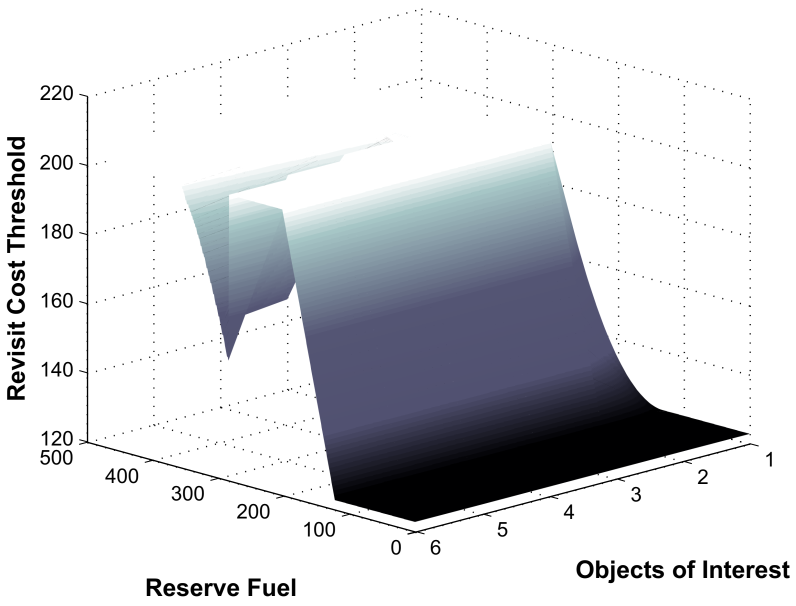


Fig. 1. Method 1 Threshold Surface Plot, Operator Response is True

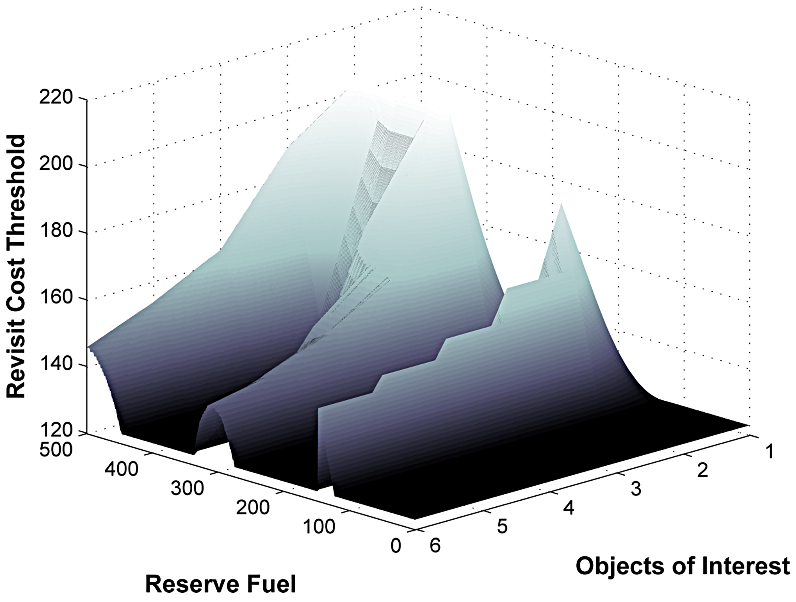


Fig. 2. Method 1 Threshold Surface Plot, Operator Response is False

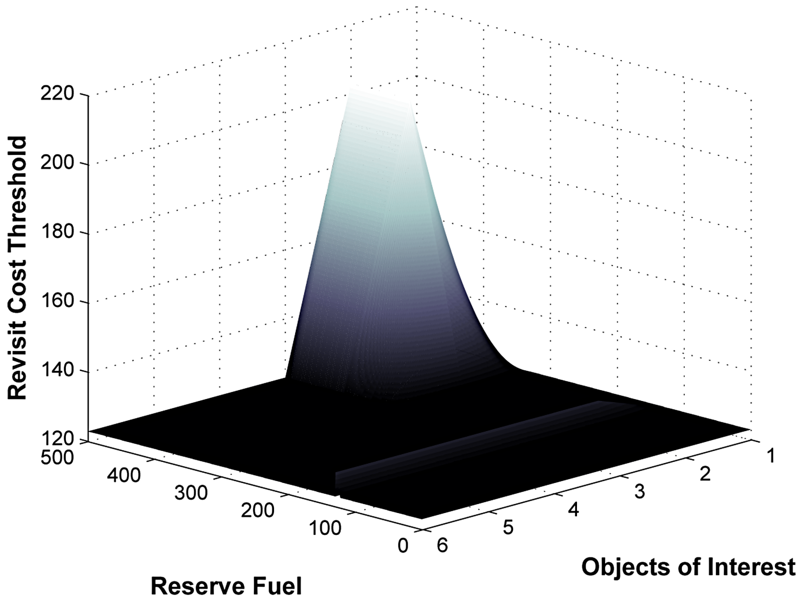


Fig. 3. Method 2 Threshold Surface Plot, Operator Response is True

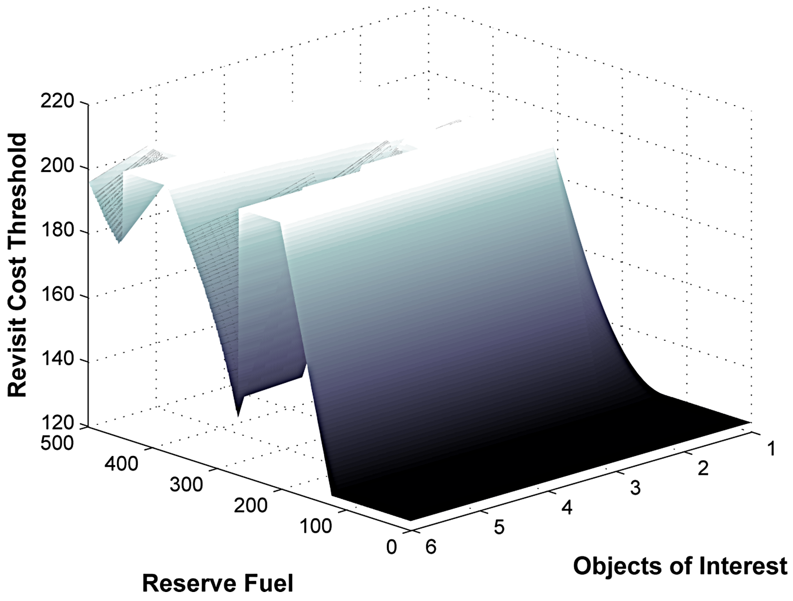


Fig. 4. Method 2 Threshold Surface Plot, Operator Response is False

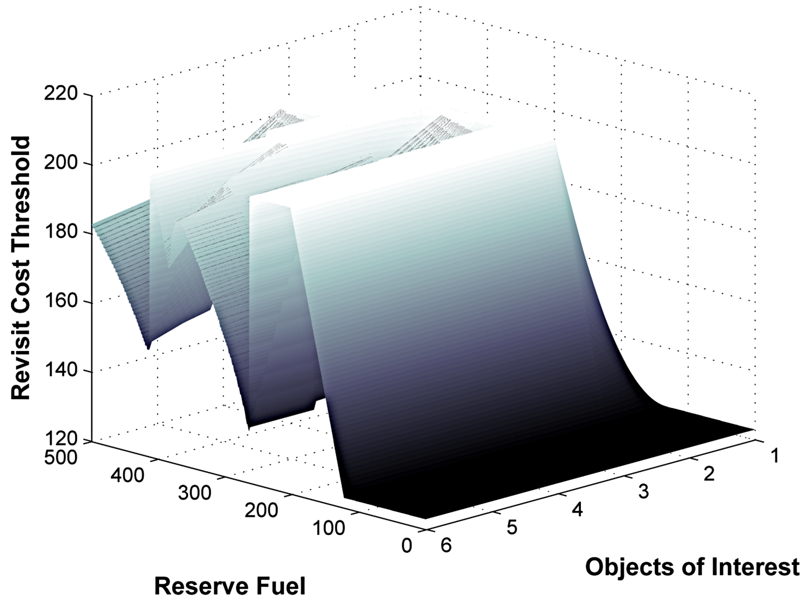


Fig. 5. Method 3 Threshold Surface Plot, Operator Response is True

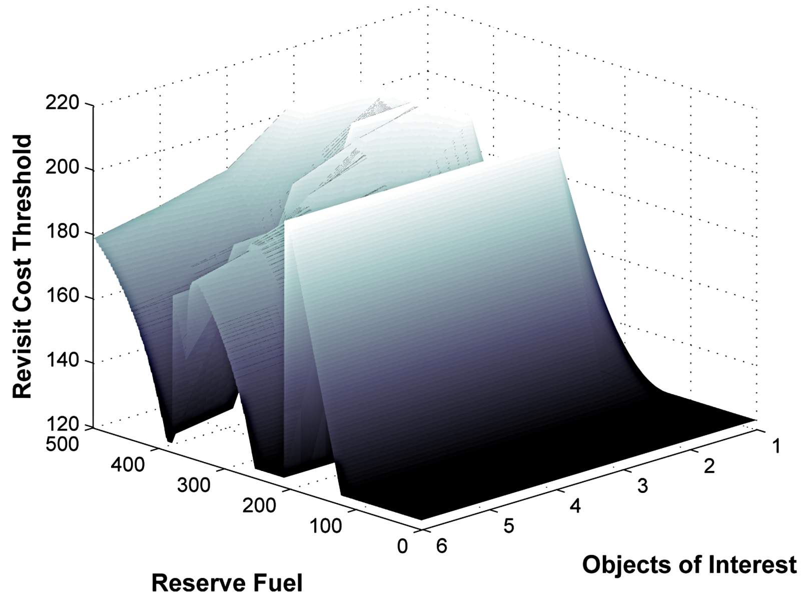


Fig. 6. Method 3 Threshold Surface Plot, Operator Response is False

the operator responds false, the stochastic controller will most likely perform a revisit as shown in Figure 4.

For Method 3, there is not enough saturation in Figures 5 and 6 to definitively say what the stochastic controller will decide to do given the operator's response.

3 Simulation and Results

Simulations were conducted to test the various reward methods. Each method underwent 100 trials, each trial being 1200 simulation seconds long, with twenty objects of interest and four UAVs on tour. The location, orientation, and target truth status of the OOI were randomized for each trial. A log was kept of the stochastic controller's decisions throughout all of the trials, the data from which acts as a basis for comparison between the various reward methods.

To compare the different methods, a rating system was devised. For all cases where the stochastic controller had a UAV revisit an OOI, it added a point if either of the following occurred:

– the operator response was true on one visit and false on the other visit and the target truth status was true,
– the operator response was false on both visits and the target truth status was false.

These are the best case scenarios where everything the operator indicates coincides with reality. The points tallied for the case where target truth status is true and the case where it was false was kept separate because they have a different probability of occurrence. The overall point tally for each case is then divided by the probability of that case to determine a normalized point system. The points for each target truth status case can then be added together to determine an overall score for that method. Table 4 summarizes the results.

From Table 4, it can be seen that Method 2 outperforms the other methods for the case where the operator responds false. This results from Method 2 having a strong preference to opt for a revisit if the operator responded false, in

Table 4. Simulation Results

	Benchmark	Method 1	Method 2	Method 3
True Score Mean	0.190	0.610	0.500	0.760
True Standard Deviation	0.419	0.803	0.674	0.842
False Score Mean	0.410	0.070	2.080	0.520
False Standard Deviation	0.911	0.432	2.977	1.453
Adjusted True	1.900	6.100	5.000	7.600
Adjusted False	0.456	0.078	2.311	1.531
Overall Score	2.356	6.178	7.311	8.178

combination with the fact that a majority of the time the operator will respond false on both visits. Method 3 outperforms the other methods for the case where the operator responds true, and its overall score indicates that it is the best of the three methods. This is because of its lack of saturation seen in Figures 5 and 6. Although it has no strong preference for either response, it performs well on average for both, whereas Methods 1 and 2 only perform well for one of the possible operator responses.

4 Conclusion

The effectiveness of the stochastic controller depends strongly on the reward function. To determine the effectiveness of a variety of reward methods, hundreds of simulation trials were performed and the resulting data were analyzed by a scoring algorithm. The results of the different reward methods are strongly related to the threshold surfaces. Although threshold surface saturation is not necessarily bad, it is not optimal. The method where saturation was mostly avoided was the method that performed the best. It was also shown that a discrete reward method can outperform an information theoretical method.

Essentially, the reward function outputs two values, expected rewards for the cases where the operator responds true or false. It may not be necessary to have a function dedicated to the determination of these values based on probabilities in the scenario. Instead, the two expected reward values could be determined using an optimization program that modulates the values over a series of simulations.

References

1. Gross, D., Rasmussen, S., Chandler, P., Feitshans, G.: Cooperative Operations in UrbaN TERrain (COUNTER). In: Proceedings of the SPIE, Orlando, FL, May 2006. Defense Transformation and Network-Centric Systems, vol. 6249 (2006)
2. Warfield, L., Carretta, T., Patzek, M., Gonzalez-Garcia, A.: Multi-Aircraft Video-Human/Automation Target Recognition (MAVHTR) studies: Unaided target acquisition involving multiple micro air vehicle (MAV) videos. Technical Report AFRL-HE-WP-TR-, -0036, Air Force Research Laboratory/Human Effectiveness Directorate/Warfighter Interface Division, Wright-Patterson AFB, OH, distribution limited to DoD and U.S. DoD contractors only (2007)
3. Pachter, M., Chandler, P., Darbha, S.: Optimal Control of an ATR Module Equipped MAV-Human Operator Team. In: Cooperative Networks: Control and Optimization - Proceedings of the 6th International Conference on Cooperative Control and Optimization. Edward Elgar Publishing, Northampton (2006)
4. Pachter, M., Chandler, P., Darbha, S.: Robust control of unmanned aerial vehicles in uncertain environments. International Journal of Robust and Nonlinear Control 18(2) (January 2008)

5. Pachter, M., Chandler, P., Darbha, S.: Optimal sequential inspections. In: Proceedings of the IEEE Conference on Decision and Control, San Diego, CA (December 2006)
6. Girard, A., Pachter, M., Chandler, P.: Optimal decision rules and human operator models for UAV operations. In: Proceedings of the AIAA Guidance, Navigation and Control Conference, Keystone, CO (August 2006)
7. Girard, A., Darbha, S., Pachter, M., Chandler, P.: Stochastic dynamic programming for uncertainty handling in UAV operations. In: Proceedings of the American Control Conference, New York, NY (July 2007)

Dynamics and Control of Surface Exploration Robots on Asteroids

Julie Bellerose[1], Anouck Girard[2], and Daniel J. Scheeres[3]

[1] PhD Candidate, Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI
`juliebel@umich.edu`

[2] Assistant Professor, Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI
`anouck@umich.edu`

[3] Professor, Department of Aerospace Engineering, University of Colorado at Boulder, CO

Abstract. Over the past decade, a few solo-robotic landing missions have been sent to asteroids at modest cost, providing a basic understanding of their environment. These missions can diversify and be improved upon by having multiple landers which also contribute to increasing the overall mission reliability. Since the gravity on an asteroid is low, a wheeled vehicle would likely bounce back from hitting the surface, and be difficult to control. Instead we consider hopping robots. We develop a first order model of the dynamics of hoppers to estimate the total time and distance covered from an initial bounce to a stop due to friction and restitution coefficients. From this dynamical model, hoppers could easily investigate the surface by controlling their initial velocity; one would just need to estimate to desired distance to be covered. To extend the single hopper control law to collaborative landers, we apply sliding-mode control to discrete formation control.

1 Introduction

Of the small body population (asteroids and comets), there are 5000 Near-Earth Objects. Since these small bodies are not eroded and are believed to have formed in the early stage of our solar system, studying small bodies on a closer scale may answer some fundamental questions about the formation and evolution of our solar system. Over the past decade, robotic missions such as NEAR, Stardust, Rosetta, Hayabusa (Muses-C), and DAWN, have been sent to small bodies, providing a basic understanding of their environment. So far, only single spacecraft or probes have been sent to these small bodies. For future scientific investigations, rendezvous missions carrying small lander(s) become more interesting as landers can perform autonomous tasks on the surface. In addition, current plans are to include surface robots and/or multiple probes in order to assess the soil composition, do internal and surface mapping, take multiple images, and to increase the overall reliability of the mission.

There are challenges in sending spacecraft and landers to small bodies. First, there are many sources of perturbations such as solar pressure or uneven gravity fields due to non-spherical bodies. Robots have shown to be very useful in hazardous situations on Earth, as well as on other planets. For instance, Spirit and Opportunity are still sending back data from the surface of Mars. The meteorological station Phoenix is scheduled to arrive to Mars in May 2008. Hence, as part of the planetary exploration effort, we propose a robotic mission composed of a mother ship and a number of cooperative robots for surface exploration. Collaborative rover applications are being studied for planetary exploration [1]. However, for asteroid applications, one needs to design for very low gravity fields, about 1/100 of the Earth's gravity. In such environments, wheeled rovers might be difficult to control and navigate. However, a hopper would be able to investigate a larger area in quicker time.

Previous work has looked at the dynamics of spacecraft or particles near asteroids and comets, for single asteroids as well as binary asteroid systems [16]. Several methods have been developed to simplify the problem while keeping interesting features. For surface motion, one needs to develop a model that predicts the distance traveled and the time it takes in order to design a control law. We design a leaderless control taking reference on sliding-mode control techniques [13,23]. In Section 2, we summarize the dynamical characteristics of asteroid systems and we derive a prediction model for surface dynamics. In Section 3, we develop a discrete control law for single and cooperative hoppers.

2 Underlying Dynamics of Asteroids

2.1 Asteroid Orbiters

Before looking at surface motion on asteroids, we first briefly summarize the dynamics of particles, or spacecraft, in the gravitational field of an asteroid. As a first approximation, we neglect the effect of the sun and we assume a uniform gravitational field for the asteroid. One of the principal characteristic of asteroids is their irregular shape. Current and past work have looked into methods to compute the potential of an irregular body. Werner, Scheeres and Fahnestock [18] have derived polyhedron potential computations to model irregularly shaped bodies. Other methods involve the use of Lie Group computations [10] or spherical harmonics [9].

As shown in Figure 1, we model the asteroid as a tri-axial ellipsoid. We can write the dynamics of a spacecraft in an asteroid-fixed frame as

$$\ddot{\boldsymbol{r}}_{\boldsymbol{e}} + 2\boldsymbol{\omega} \times \dot{\boldsymbol{r}}_{\boldsymbol{e}} + \boldsymbol{\omega} \times (\boldsymbol{\omega} \times \boldsymbol{r}_{\boldsymbol{e}}) = \frac{\partial U_e}{\partial \boldsymbol{r}_{\boldsymbol{e}}}, \tag{1}$$

where $\ddot{\boldsymbol{r}}_{\boldsymbol{e}}$ is the position vector of the spacecraft relative to the asteroid center of mass, $\boldsymbol{\omega}$ is the angular velocity of the ellipsoid, and U_e is the potential expression of the ellipsoidal body. U_e can be written in terms of an elliptic integral [3,12,4] and has the form

$$U_e = \frac{3}{4}\int_{\lambda}^{\infty} \phi(\boldsymbol{r}, v)\frac{dv}{\Delta(v)} \tag{2}$$

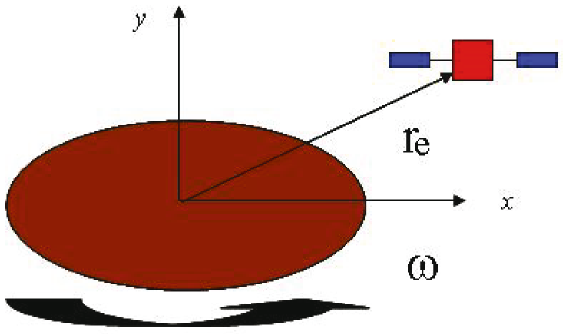


Fig. 1. Representation of the asteroid orbiter problem

with

$$\phi(\boldsymbol{r}, v) = 1 - \frac{(x + \nu r)^2}{1 + v} - \frac{y^2}{\beta^2 + v} - \frac{z^2}{\gamma^2 + v} \tag{3}$$

and

$$\Delta(v) = \sqrt{(1 + v)(\beta^2 + v)(\gamma^2 + v)}, \tag{4}$$

where $0 < \gamma \leq \beta \leq 1$, γ and β correspond to the z and y radii of the ellipsoid, and λ satisfies $\phi(\boldsymbol{r}, \lambda) = 0$.

This system also allows for one integral of motion [5], the Jacobi integral, written as

$$J = \frac{1}{2}(\dot{x}^2 + \dot{y}^2 + \dot{z}^2) - \frac{1}{2}\omega^2(x^2 + y^2) - U_e, \tag{5}$$

which gives an indication of the allowable regions of motion for a spacecraft.

2.2 Binary Asteroid Systems Orbiters

Some small body systems are found to be in pairs, where two asteroids orbit each other. It is currently estimated that about sixteen percent of the Near Earth Asteroid population may be binary systems [11]. Current interests in asteroid missions are to better characterize the external and internal composition and to better understand asteroid and asteroid systems formation and evolution. Hence, sending a mission to a binary asteroid system may provide an opportunity to get returns on both the geology and the dynamics of asteroids. Studies have looked at the dynamics of these systems [15] and the dynamics of a spacecraft close to them [17]. The binary asteroid orbiter is pictured in Figure 2, where M_1 is defined as the mass of the spherical shape and M_2 as the mass of the ellipsoid. The mass fraction of the system is defined as,

$$\nu = \frac{M_1}{M_1 + M_2}. \tag{6}$$

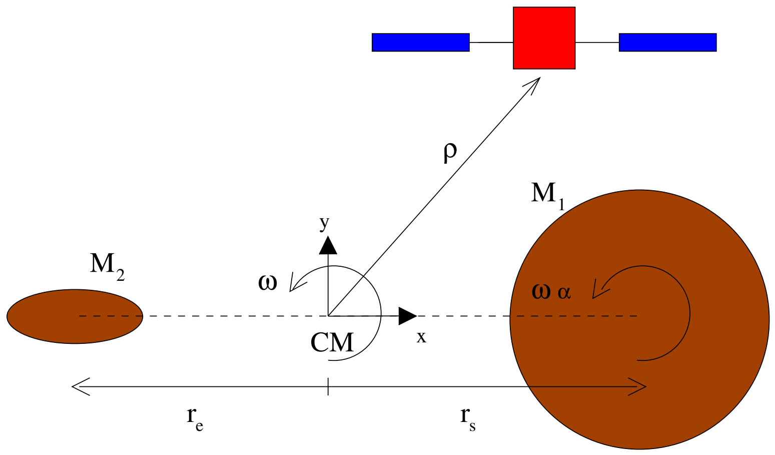


Fig. 2. The Restricted Full Three-Body Problem

We use $\tilde{\boldsymbol{\rho}}$ for the nondimensional position vector of the spacecraft relative to the binary system center of mass, in a frame fixed to the ellipsoid. In this case, the equations of motion of the spacecraft are written as

$$\ddot{\tilde{\boldsymbol{\rho}}} + 2\boldsymbol{\omega} \times \dot{\tilde{\boldsymbol{\rho}}} + \boldsymbol{\omega} \times (\boldsymbol{\omega} \times \tilde{\boldsymbol{\rho}}) = \frac{\partial U_{12}}{\partial \tilde{\boldsymbol{\rho}}}, \tag{7}$$

where U_{12} is the potential expression from both the sphere and the ellipsoid. It is expressed as,

$$U_{12} = \frac{\nu}{|\tilde{\boldsymbol{\rho}} - (1-\nu)\boldsymbol{r}|} + (1-\nu)U_e(\tilde{\boldsymbol{\rho}} + \nu\boldsymbol{r}) \tag{8}$$

with U_e being the ellipsoid potential given by Equations (2-4).

Studies have looked at special conditions where the two bodies are locked in space, keeping the same orientation, as well as general dynamics with small perturbations near these equilibrium configurations. When the system is locked in an aligned configuration, the system allows for one integral of motion, the Jacobi integral, which can be written as

$$C = \frac{1}{2}v_R^2 - V, \tag{9}$$

where v_R is the speed of a particle or a spacecraft relative to the rotating frame.

2.3 Surface Motion Dynamics

Once the dynamics of asteroid system orbiters are developed, we can look at the dynamics happening at the surface. Since there is about 1/100 of the Earth's gravity on an asteroid, a vehicle will bounce every time it tries to move on the

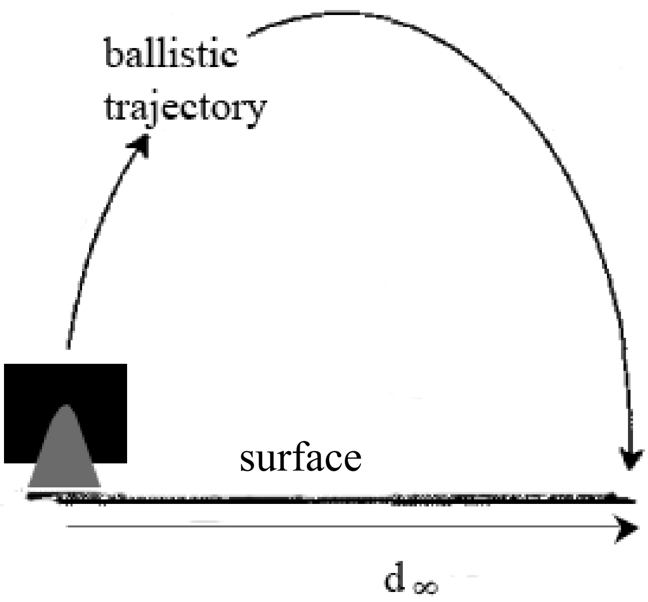


Fig. 3. Hopping motion on asteroids

surface, and hence loose traction. We develop a first order model of the dynamics on the surface to estimate the total time and distance covered from an initial jump to a stop due to friction and restitution coefficients. Being able to estimate the total distance covered, we will be able to design a control system for such applications. The hopping motion is sketched in Figure 3.

We model the surface motion in three dimensions, with a tangential $\hat{\boldsymbol{t}}$, normal $\hat{\boldsymbol{n}}$, and cross track vectors, $\hat{\boldsymbol{d}}$. This frame is fixed at the initial impact point on the surface. The normal is defined as

$$\hat{\boldsymbol{n}} = \frac{\boldsymbol{\nabla} S}{|\boldsymbol{\nabla} S|}, \tag{10}$$

where S is the ellipsoid surface function expressed as $S = x^2 + \frac{y^2}{\beta^2} + \frac{z^2}{\gamma^2} - 1 = 0$. Note that the normal is the unit vector in the direction of the surface gradient. For better tracking, the tangential direction is defined as the unit vector perpendicular to the velocity,

$$\hat{\boldsymbol{t}} = \frac{\hat{\boldsymbol{t}} \times \boldsymbol{V}}{|\hat{\boldsymbol{t}} \times \boldsymbol{V}|}, \tag{11}$$

where $\boldsymbol{V}$ is the impact velocity expressed in the binary fixed frame. And then, the cross track unit vector is obtained from orthogonality of the tangential and normal vectors,

$$\hat{\boldsymbol{d}} = \hat{\boldsymbol{t}} \times \hat{\boldsymbol{n}}. \tag{12}$$

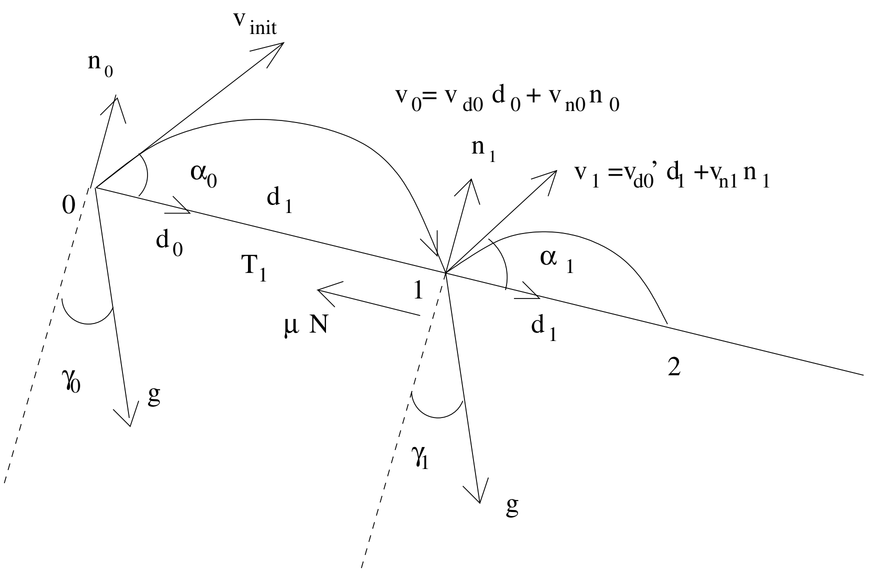


Fig. 4. Dynamics of collisions for a particle on an inclined surface with restitution and friction coefficient c_r and μ respectively

Note that the total acceleration acts in the normal - cross track plane. The geometry for modeling the impact dynamics is shown in Figure 4. At the impact point, an object is subjected to a local coefficient of restitution and surface friction factor, c_r and μ respectively. The velocity $\boldsymbol{v_n}$ at the nth jump, just before the next impact, is influenced by the general gravity vector $\boldsymbol{g}$ and the rotational acceleration. By approximating the local surface as a flat surface, we can find general expressions for the time, t_n, distance, d_n, and velocity components between jumps as function of the initial conditions. The total distance covered and the time of travel are then a summation of the intervals, which are given by

$$d_{d,\infty} = \frac{2v_{n_0}v_{d_0}}{g_{n_0}(1-c_r)} + \frac{2v_{n_0}^2}{g_{n_0}(1-c_r)^2}\left(-\mu c_r + \frac{g_{d_0}}{g_{n_0}}\right), \tag{13}$$

and

$$t_\infty = \frac{2v_{n_0}}{g_{n_0}}\left(\frac{1}{1-c_r}\right), \tag{14}$$

where the subscripts n and d in this case represent components in the normal and cross track directions, respectively. Note that the magnitude and direction of the gravity vector are obtained from the dynamics of the asteroid system, that is, using Equation (7), and are then transformed to the surface frame.

We can validate this approximation model using the true asteroid dynamics. In the numerical simulations, we define a new surface frame at each impact. The velocity components after impact are then converted back into the asteroid system frame for numerical integration until the next impact. Having defined the unit vectors at the surface of the ellipsoid with respect to the RF3BP fixed frame, we can write the transformation from the surface frame to the RF3BP frame. In general, for a jump distance below 5 meters, we find that the analytical model agrees within 1 % with the numerical simulations.

2.4 Effect of the Ellipsoid Shape on the Surface Dynamics

An ellipsoidal body has points on its surface where an object could stay in equilibrium. The stability of the equilibrium points depend on its spin and on its shape parameters. One would expect that the dynamics of vehicles on the surface of a small body are affected by these equilibria and their stability, motivating control of the surface dynamics in order to investigate specific regions.

In [6], the authors use classical dynamics and geometrical analysis to investigate the stability of surface equilibrium points for a rotating ellipsoid. For small perturbations, a moving object on the surface of a rotating ellipsoid will tend to stay closer to a stable equilibrium point. On the other hand, it will stay far from an unstable region. In fact, this explains current observations of asteroids where material is accumulated near the equator while some other asteroids have accumulation near the poles [6]. Figure 5 shows an example where the polar regions are stable. The curve shown is made of a series of hops under perfect surface conditions. For asteroids with faster spin, moving vehicles tend to stay toward the equator.

3 Robotic Mission to an Asteroid

We can integrate the underlying dynamics of asteroids described in the previous sections to design a robotic mission for surface exploration. MINERVA, originally designed for the Hayabusa mission is a good example of a "hopper" application, using a torque driving system as the main driver [8,22]. Ball Aerospace also designed a spherical robot having three side openings for stability at the surface [19]. In all cases, the motion of these hoppers looks like the ballistic trajectory shown in Figure 3. Hoppers could easily investigate the surface by controlling their initial bounce velocity and orientation from estimating the jumping distance to be covered for a set of desired locations.

In the current work, we leave the details of a robotic mission hardware as future studies and we will concentrate on the dynamics and control of these cooperative robots. We assume a spacecraft in orbit about an asteroid from which landers could be ejected for further surface exploration. A hopper would easily move on the surface, investigating the region while taking scientific data. The goal in sending robots for surface investigation is to maximize the search area for mapping, imaging, and taking geological/scientific measurements or samples. The landers could be released at one end of the asteroid, sent on a predefined grid and move to the other end. With a spacecraft staying on a stable orbit, they could all update their position to a central unit on the spacecraft as well as communicate with each other through sensors in order to perform common tasks.

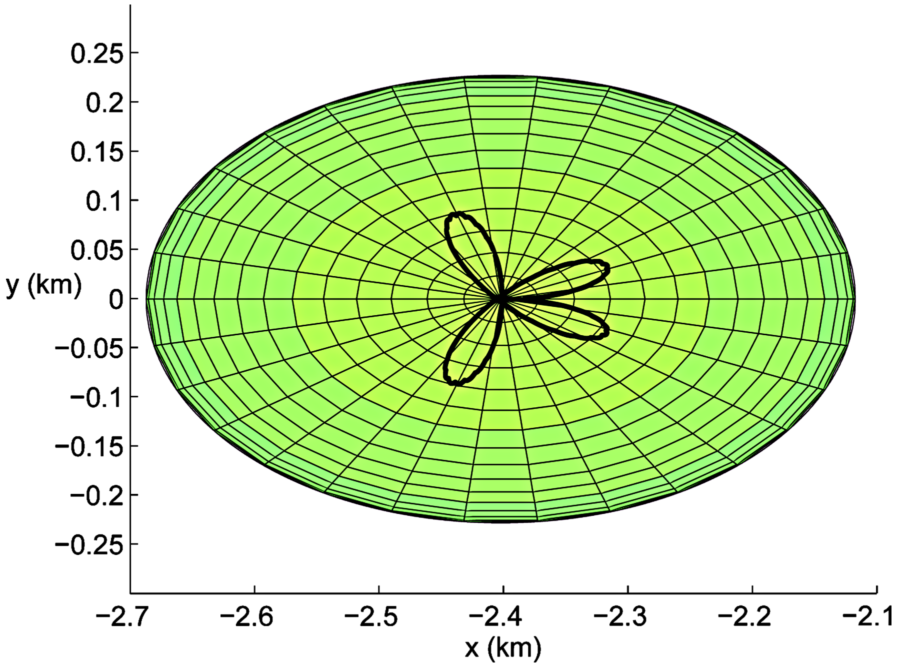


Fig. 5. Top view of the dynamics close to a stable pole. The trace is made of hops under ideal conditions.

There are many variables to consider in making efficient moves: time and distance to travel, fuel consumption, external disturbances, obstacles, collaborative tasks, etc. Too many jumps would consume time and energy while high jumps could be hard to track. For efficient surface investigation, we develop a discrete control law for a single vehicle acting on its initial conditions, and we give an extension for multiple cooperative vehicles. Many control strategies have been considered, especially for unmanned aerial vehicles. Some of them involve leader-follower approaches [7] or string and mesh-stable approaches [20,14]. Other possible approaches include virtual structures [2,21] and potential methods [24]. These methods are difficult to implement in asteroid applications. The following sections describe the collaborative control developed in applying ideas from sliding-mode control methodology [13,23].

3.1 Control of Hoppers

Using the surface dynamics model from Section 2.3, we attempt to minimize the error associated with the travel. For doing so, we develop a discrete control law by defining a control parameter affecting the error between hops. We let the position at the n-th jump be represented in the cross track-tangential plane, $\hat{\boldsymbol{d}} - \hat{\boldsymbol{t}}$, by

$$\eta^n = \begin{bmatrix} d_d \\ d_t \end{bmatrix}. \tag{15}$$

The $(n+1)$-st location is then the sum of the previous location and the distance covered from jumping

$$\eta^{n+1} = \eta^n + \Delta\eta, \tag{16}$$

where we want $\Delta\eta$ to be solved in such a way that the error on the position decreases in time. If we define the error at the n-th jump as

$$\epsilon^n = \eta^n - \eta_d, \tag{17}$$

where η_d is the desired end position, then we want to choose a control parameter K such that the error decreases over each bounce, that is,

$$\epsilon^{n+1} = e^{-K}\epsilon^n. \tag{18}$$

Substituting for ϵ^{n+1} and ϵ^n in Equation (18), we get

$$\eta^{n+1} - \eta_d = (\eta^n - \eta_d)e^{-K} \tag{19}$$

or, substituting Equation (16) and solving for $\Delta\eta$,

$$\Delta\eta = (\eta^n - \eta_d)\left[e^{-K} - 1\right]. \tag{20}$$

Hence, Equation (20) gives the distance to be covered given a desired position and control parameter K. Using this result in the surface dynamics model, we can invert Equation (13) to find the initial velocity component. Hence, the cross track velocity component is given by

$$v_0 \ = \frac{g_{n_0}(1-c_r)}{2v_{n_0}}\left[d_d + \frac{2v_{n_0}^2\mu c_r}{g_{n_0}(1-c_r)^2} + \frac{2v_{n_0}^2 g_{t_0}}{g_{n_0}^2(1-c_r)^2}\right]. \tag{21}$$

Therefore, using Equation (20), we compute a jump distance $\Delta\eta$ with a given value of the control parameter K. Then, from Equation (21) we can find the initial velocities to achieve that distance $\Delta\eta$. Some results are shown in Figure 6, for a single hopper. We compare three different controlled dynamics, $K = 0.75$, $K = 1.5$ and $K = 2.5$, under the same surface conditions, with $c_r = 0.5$ and $\mu = 0.5$. The desired end position is set at $x = -2.3\ km$ in the binary frame labeled η_d in Figure 6. We see that increasing the control K reduces the number of hops necessary to reach η_d. However large values of the control parameter, such as $K = 2.5$, makes the hopper overshoot η_d. Undershooting a target might be a better strategy. For small hops, a hopper would need to relocate itself and estimate its required distance to reach η_d again. In this case, more jumps are needed in order to reach the objective, which takes more time but increases the chance of success.

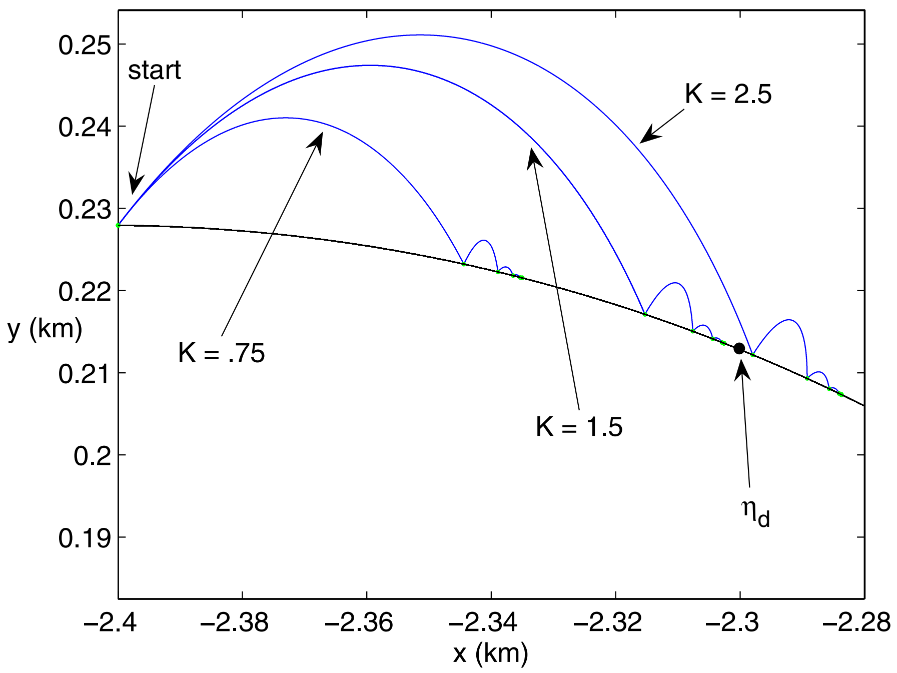


Fig. 6. Controlled dynamics of a single rover with coefficient of restitution and friction factor set to 0.5. K is the control variable and η_d is the desired end position.

3.2 Control of Cooperative Hoppers

For collaborative hoppers, we adopt a sliding-mode control strategy first developed for collaborative unmanned aerial vehicles. As described in [23], we want a formation to navigate to a desired end point while maintaining or achieving a certain configuration. For collaborative robots, we will use the notation η_i^n for the i-th hopper at the n-th location. The $(n+1)$-st location is then

$$\eta_i^{n+1} = \eta_i^n + \Delta\eta_i, \tag{22}$$

where, again, we want the distance, $\Delta\eta_i$, to be solved in such a way that the error on the position decreases in time. For the current application, we want the error to take into account the absolute error, i.e., the error with respect to its desired location, as well as the relative error for each hopper relative to its neighbors. We define the error term as,

$$\begin{aligned}\epsilon_i^n =& (\eta_i^n - \eta_{d,i}^n) + K_r(\eta_{i,j}^n - \eta_{i-1,j}^n + \eta_{d,i-1,j}) \\ &+ K_r(\eta_{i,j}^n - \eta_{i+1,j}^n + \eta_{d,i+1,j}) + K_r(\eta_{i,j}^n - \eta_{i,j-1}^n + \eta_{d,i,j-1}),\end{aligned} \tag{23}$$

where K_r is a weight factor giving precedence on the absolute or relative hopper end position.

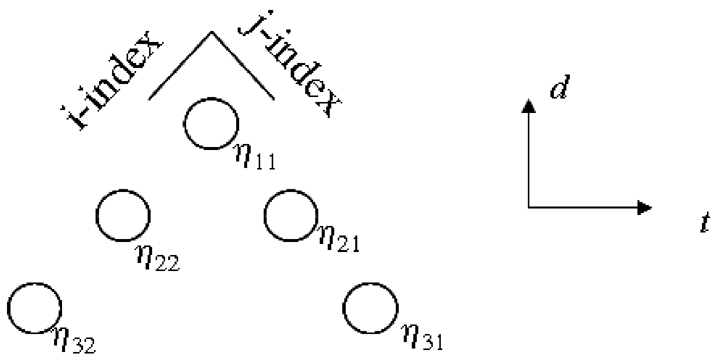


Fig. 7. Triangular Formation for Collaborative Hoppers

Earlier work has shown mesh stability of a triangular formation as shown in Figure 7 [23]. Hence, for the current work, as a first step we started by investigating a triangular formation of three hoppers. As for the single hopper, we want to choose a control parameter K such that the error decreases in time,

$$\epsilon_i^{n+1} = e^{-K}\epsilon_i^n. \tag{24}$$

And, using again the control parameter on the relative position of the hoppers, we write the error expression for the three hoppers as

$$\epsilon_1 = (\eta_1 - \eta_{d,1}) + K_r(\eta_1 - \eta_2 - \eta_{d,12}) + K_r(\eta_1 - \eta_3 - \eta_{d,13}), \tag{25}$$

$$\epsilon_2 = (\eta_2 - \eta_{d,2}) + K_r(\eta_2 - \eta_1 - \eta_{d,21}) + K_r(\eta_2 - \eta_3 - \eta_{d,23}), \tag{26}$$

and

$$\epsilon_3 = (\eta_3 - \eta_{d,3}) + K_r(\eta_3 - \eta_1 - \eta_{d,31}) + K_r(\eta_3 - \eta_2 - \eta_{d,32}). \tag{27}$$

Substituting for ϵ^{n+1} and ϵ^n for all three hoppers in Equation (24), we get

$$\begin{aligned}(\eta_1^{n+1}-\eta_{d,1}) + K_r(\eta_1^{n+1} - \eta_2^{n+1} - \eta_{d,12}) + K_r(\eta_1^{n+1} - \eta_3^{n+1} - \eta_{d,13}) = \\ e^{-K}\left[(\eta_1 - \eta_{d,1}) + K_r(\eta_1 - \eta_2 - \eta_{d,12}) + K_r(\eta_1 - \eta_3 - \eta_{d,13})\right],\end{aligned} \tag{28}$$

$$\begin{aligned}(\eta_2^{n+1}-\eta_{d,2}) + K_r(\eta_2^{n+1} - \eta_1^{n+1} - \eta_{d,21}) + K_r(\eta_2^{n+1} - \eta_3^{n+1} - \eta_{d,23}) = \\ e^{-K}\left[(\eta_2 - \eta_{d,2}) + K_r(\eta_2 - \eta_1 - \eta_{d,21}) + K_r(\eta_2 - \eta_3 - \eta_{d,23})\right],\end{aligned} \tag{29}$$

and

$$\begin{aligned}(\eta_3^{n+1}-\eta_{d,3}) + K_r(\eta_3^{n+1} - \eta_1^{n+1} - \eta_{d,31}) + K_r(\eta_3^{n+1} - \eta_2^{n+1} - \eta_{d,32}) = \\ e^{-K}\left[(\eta_3 - \eta_{d,3}) + K_r(\eta_3 - \eta_1 - \eta_{d,31}) + K_r(\eta_3 - \eta_2 - \eta_{d,32})\right].\end{aligned} \tag{30}$$

We can solve for $\Delta\eta_1$, $\Delta\eta_2$, and $\Delta\eta_3$ using Equation (22). Then, Equations (28-30) become

$$\begin{aligned}\Delta\eta_1(1{+}2K_r) - K_r\Delta\eta_2 - K_r\Delta\eta_3 = \\ \eta_1^n(1+2K_r)(e^{-K}-1) + \eta_2^n K_r(1-e^{-K}) + \eta_3^n K_r(1-e^{-K}) \\ + n_{d,1}(1-e^{-K}) + n_{d,12}K_r(1-e^{-K}) + n_{d,13}K_r(1-e^{-K}),\end{aligned} \tag{31}$$

$$\begin{aligned}\Delta\eta_2(1{+}2K_r) - K_r\Delta\eta_1 - K_r\Delta\eta_3 = \\ \eta_2^n(1+2K_r)(e^{-K}-1) + \eta_1^n K_r(1-e^{-K}) + \eta_3^n K_r(1-e^{-K}) \\ + n_{d,2}(1-e^{-K}) + n_{d,21}K_r(1-e^{-K}) + n_{d,23}K_r(1-e^{-K}),\end{aligned} \tag{32}$$

and

$$\begin{aligned}\Delta\eta_3(1{+}2K_r) - K_r\Delta\eta_1 - K_r\Delta\eta_2 = \\ \eta_3^n(1+2K_r)(e^{-K}-1) + \eta_1^n K_r(1-e^{-K}) + \eta_2^n K_r(1-e^{-K}) \\ + n_{d,3}(1-e^{-K}) + n_{d,31}K_r(1-e^{-K}) + n_{d,32}K_r(1-e^{-K}).\end{aligned} \tag{33}$$

Hence, solving for the three $\Delta\eta$'s gives the required distance the three hoppers should jump to. As for the case of a single hopper, given a distance to cover, we can invert Equation (13) to find the initial velocity components for the required move. The overall motion is governed by the control parameters K and K_r.

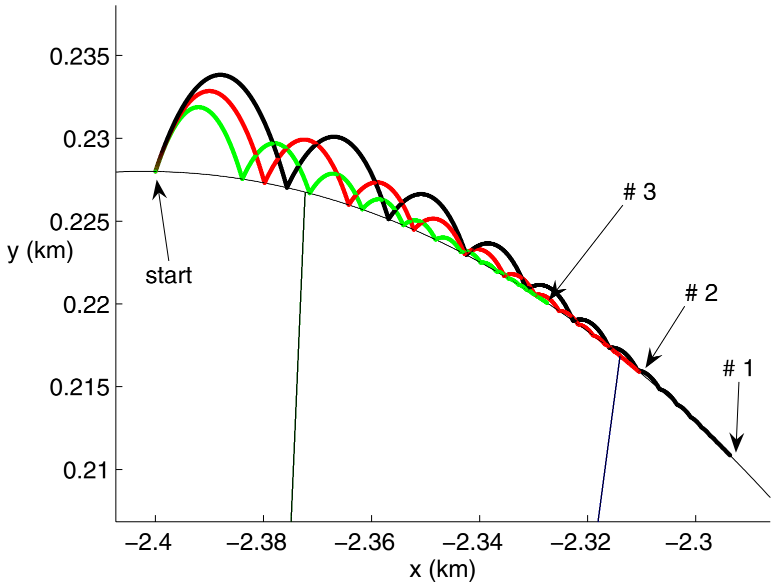


Fig. 8. Controlled dynamics of a linear robot formation with coefficient of restitution and friction factor set to 0.5 and control parameters K=1.5 and K_r=0.5

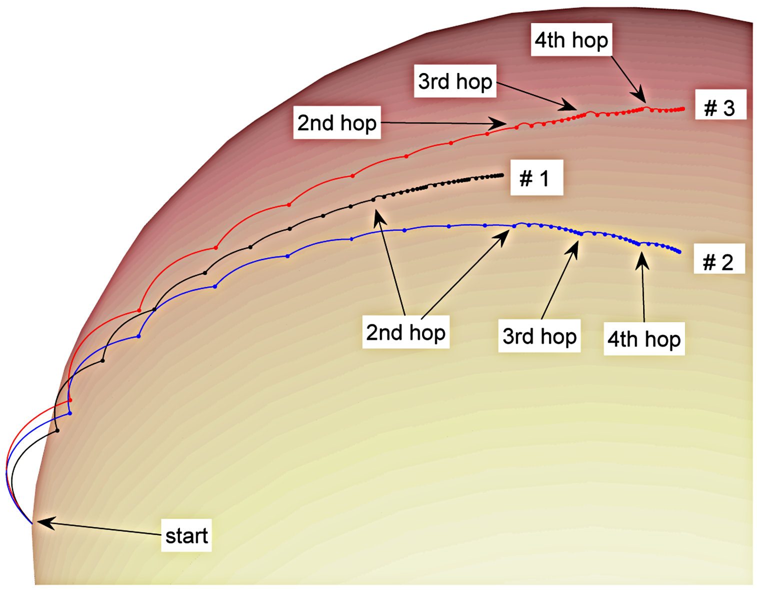


Fig. 9. Controlled dynamics of a triangular robot formation with coefficient of restitution and friction factor set to 0.8 and 0.1, control parameters of K=1.5 and K_r=0.5, and a longitudinal distance to cover less than 100 meters. Hoppers 1, 2, and 3 have reached their final position.

Preliminary results are shown in Figures 8-10 for a collinear motion test and a triangular formation of three robots. In Figure 8, the three hoppers are released from a common point and move along the equator, each having a different set of final positions as indicated in the figure. Note that the control scheme assumes the hoppers can update their position using star trackers and sensors before making the next hop. In Figure 9, the hopper formation achieves the desired configuration and position within two hops using control parameters $K = 1.5$ and $K_r = 0.5$, with a separation distance of less than 100 meters. We note that reducing K_r makes the triangular path wider before reaching the desired position.

In order to have more insight in the control efficiency, the distance to cover was doubled in the simulation shown in Figure 10. Having a desired position far from the starting point, we see that the dynamics of the asteroid has a strong effect on the dynamics and control of the hoppers. In the case shown in Figure 10, the asteroid has stable polar regions. It is clear that hoppers 2 and 3, which are closer to these regions, are attracted to them as their trajectories bend toward the poles. Even though the following hop attempts to correct the situation, the hoppers are again attracted to it. Note that the paths shown in Figures 9-10 are made of a series of jumps until the final position and configuration are achieved. For large formation, it may be impossible to reach the desired position and configuration.

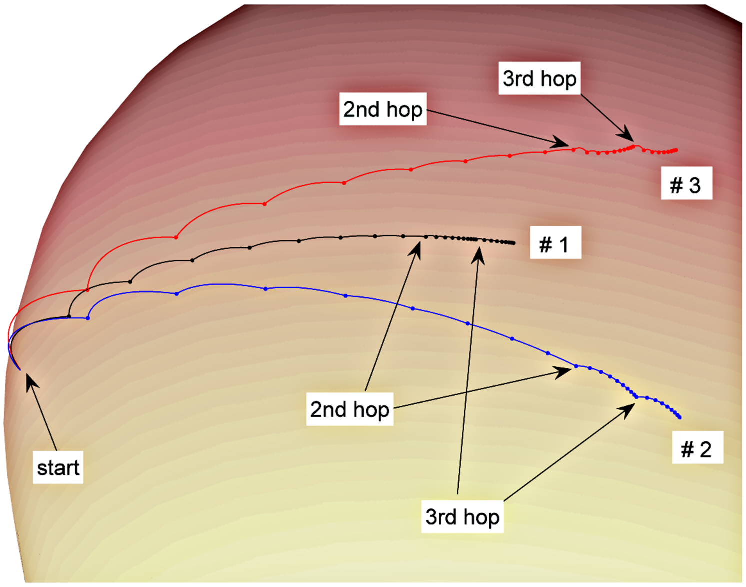


Fig. 10. Controlled dynamics of a triangular robot formation with coefficient of restitution and friction factor set to 0.8 and 0.1, and control parameters K=1.5 and K_r=0.5, with a distance to cover of 200 meters while the polar region is stable. Hoppers 1, 2, and 3 have reached their final position.

Hence, the control developed above give good results for small motion, within 100 meters. For large distance to cover, having a higher gain K_r will keep the formation tight before reaching the final desired location, which may prevent the hoppers to have their trajectory deflected by stable or unstable regions of the asteroid. However, further investigation is needed in order to design an optimal control system valid for longer travel distances and to possibly counteract stronger nonlinear perturbations on asteroids.

4 Conclusion

In this work, we use the dynamics of asteroid systems to develop a model for the dynamics and control of collaborative hoppers for surface exploration on asteroids. Many perturbations exist from having a rotating body in space. Some regions may be difficult to access due to stable or unstable equilibrium points, or for a rapidly rotating body, which gives a motivation for developing a control of surface landers suitable for asteroid applications. Collaborative robots or probes can be useful for more accurate and reliable science investigations and to access particular regions. We developed a control law and showed simulations of single and collaborative hopping robots in triangular formation taking into account an absolute desired position and a relative configuration.

Future work should involve the investigation of different regions on asteroids for a range of asteroid systems, and for different formation configurations. Also, one needs to include bounds on velocities to preventing the surface lander from physically escaping the asteroid, as well as including environment uncertainties. In addition, the control scheme can be optimized using the time of travel and fuel consumption for different configurations and tasks.

Acknowledgements

This research was funded, in part, by a grant from the Jet Propulsion Laboratory/California Institute of Technology Director's Research and Development Fund. Julie Bellerose acknowledges support via a Fellowship from the Natural Sciences and Engineering Research Council of Canada.

References

1. Barfoot, T.D., Clark, C.M.: Motion planning for formations of mobile robots. Robotics and Autonomous Systems 46, 65–78 (2004)
2. Beard, R., Lawton, J., Hadaegh, F.: A coordination architecture for spacecraft formation control. IEEE Transactions on Control Systems Technology 9(6), 777–790 (2001)
3. Danby, J.M.A.: Fundamentals of Celestial Mechanics, 2nd edn. Willmann-Bell, VA (1992)
4. Flannery, B.P., Press, W.H., Teukolsky, S.A., Vetterling, W.T.: Numerical Recipies in C, The Art of Scientific Computing, 2nd edn. Cambridge University Press, Cambridge (1996)
5. Greenwood, D.T.: Principles of Dynamics, 2nd edn. Prentice-Hall, New York (1988)
6. Guibout, V., Scheeres, D.J.: Stability of surface motion on a rotating ellipsoid. Celestial Mechanics and Dynamical Astronomy 87, 263–290 (2003)
7. Hedrick, J., Tomizuka, M., Varaiya, P.: Control issues in automated highway systems
8. Kawaguchi, A.F.J., Kuninaka, H., Uesugi, T.: Muses-c, its launch and early orbit operation. Acta Astronautica 59, 669–678 (2006)
9. Kaula, W.M.: Theory of Satellite Geodesy. Blaisdell Publishing Company, Waltham (1966)
10. Lee, T., Leok, M., McClamroch, N.H.: Lie group variational integrators for the full body problem in orbital mechanics. Celestial Mechanics and Dynamical Astronomy 98(2), 121–144 (2007)
11. Margot, J.-L., Nolan, M.C., Benner, L.A.M., Ostro, S.J., Jurgens, R.F., Giorgini, J.D., Slade, M.A., Campbell, D.B.: Binary asteroids in the near-earth object population. Science 296(5572), 1445–1448 (2002)
12. Moulton, F.R.: An Introduction to Celestial Mechanics. Dover (1970)
13. Olfati-Saber, R., Murray, R.: Graph rigidity and distributed formation stabilization of multi-vehicle systems 3, 2965–2971 (2002)
14. Pant, A., Seiler, P., Hedrick, K.: Mesh stability of look-ahead interconnected systems. IEEE Transactions on Automatic Control 47(2), 403–407 (2002)
15. Scheeres, D.J.: Stability of relative equilibria in the full two-body problem. Annals of the New York Academy of Sciences 1017, 81–94 (2004)

16. Scheeres, D.J.: Relative equilibria for general gravity fields in the sphere-restricted full 2-body problem. Celestial Mechanics and Dynamical Astronomy 94(3), 317–349 (2006)
17. Scheeres, D.J., Augenstein, S.: Spacecraft Motion about Binary Asteroids. Astrodynamics 2003, Part II, Advances in the Astronautical Sciences Series, vol. 116, pp. 991–1010 (2003)
18. Scheeres, D.J., Fahnestock, E.G.: Simulation of the full two rigid body problem using polyhedral mutual potential and potential derivatives approach. Celestial Mechanics and Dynamical Astronomy 96(3-4), 317–339 (2006)
19. Shiga, D., (January 22, 2007), `http://space.newscientist.com/article/dn11001`, `NewScientist.com`
20. Swaroop, D., Hedrick, J.K.: String stability of interconnected systems. IEEE Transactions on Automatic Control 41(3), 349–357 (1996)
21. Tillerson, M., Breger, L., How, J.: Distributed coordination and control of formation flying spacecraft 2, 1740–1745 (2003)
22. Yoshimitsu, T., Kubota, T., Nakatani, I., Adachi, T., Saito, H.: Micro-hopping robot for asteroid exploration. Acta Astronautica 52, 441–446 (2003)
23. Zheng, Z., Girard, A.: Leaderless formation control using dynamic extension and sliding control. In: IFAC World Congress (2008)
24. Zohdi, T.: Computational design of swarms. International Journal for Numerical Methods in Engineering 57, 2205–2219 (2003)

Trajectory Generation for Relative Guidance of Merging Aircraft

Félix Mora-Camino[1,2], Karim Achaibou[2], and Baba Ouattara[3]

[1] LARA/ENAC, The French Civil Aviation Institute,
7, Av. Edouard-Belin, 31055, Toulouse, France
felix.mora@enac.fr
[2] LAAS du CNRS, Laboratoire d'Architecture et d'Analyse des Systèmes,
7 Av. Edouard Belin, 31077, Toulouse, France
achaibou@ipst.fr
[3] EAMAC, The African Civil Aviation Institute,
B.P. 746, Niamey, Niger
bouattara@eamac.ne

Abstract. In this chapter is considered the problem of generation of time efficient trajectories for N aircraft performing merging maneuvers behind a leading aircraft. The resulting multi trajectory generation problem is first considered as a minimum time control problem. The analysis of the resulting set of complex optimality conditions shows that the minimum time trajectories are produced by bang-bang control laws and can be characterized by few geometric parameters. Based on these considerations, a merging sequence is established for the N aircraft. Then regular minimum time convergence trajectories can be defined for each aircraft following a target generated by its own predecessor. For practical considerations it appears necessary to solve on-line this problem. An off-line exhaustive solution approach, based on reverse dynamic programming is proposed to cover a large set of initial relative positions. Then feed forward neural networks can be trained to generate on board each aircraft guidance directives to perform efficiently the overall merging maneuver.

Nomenclature

- θ_{ij}, ψ_i : relative and absolute headings of aircraft.
- d_{ij}, d_{min} : distance between aircraft and minimum horizontal separation.
- V_i, V_0 : speeds of trailing and leader aircraft.
- ϕ_i, r_i : bank angle and yaw rate of trailing aircraft.
- $\lambda_d, \lambda_\theta, \lambda_\psi$: co-state variables.
- μ_{ij}, v_{ij}, H : dual variables, slack variables and Hamiltonian.
- l_i, ω_i : length and turn angle of i^{th} minimum time path section.
- t, t_f, t_n : current time, final time and time to point n.
- $P_n, S_n^*, \tilde{S_n}$: n^{th} problem and associated solutions sets.
- x_i, y_i : coordinates of the i^{th} reference point for a minimum time trajectory.
- R_{min} : minimum turn radius.

- δ_{ij} : minimum safe spatial separation.
- a_{max} : maximum aircraft acceleration.
- ρ : relative separation margin.

1 Introduction

In this chapter is considered the problem of on line generation of minimum time trajectories to be followed by N aircraft to achieve relative convergence maneuvers. With the development of free flight and autonomous aircraft concepts in Civil Aviation in the recent years, this problem has received increased interest. Considering the current and predicted levels of congestion of air traffic, studies related to the delegation to the flight crew of some tasks currently performed by air traffic controllers are actively tackled today [1]. Among these studies, relative guidance between aircraft has appeared to be a promising way to increase air traffic capacity.

The objective of this chapter is to propose a solution based on cooperative control to automatically perform simultaneous merging maneuvers. From an operational point of view, and assuming normal operations, the air traffic controller should be relieved of providing instructions to the trailing aircraft for merging behind the leading aircraft and maintaining safe spacings between them (see Figure 1 for an example of traffic situation). Thus, the expected benefit of such new capabilities is an increase of air traffic capacity and safety. Enhancement of flight crew airborne traffic situational awareness is also expected with associated safety benefits. The feasibility of such a relative guidance device is based on the present ability of modern aircraft to broadcast and receive suitable navigation data thanks to Automatic Dependent Surveillance-Broadcast (ADS-B) [2]. Among those transmitted navigation data, identification, position, speed and heading are essential to achieve efficient merging maneuvers.

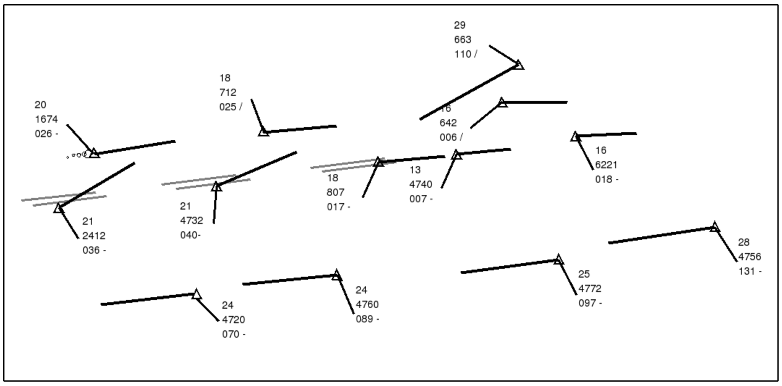


Fig. 1. Example of radar image at Paris-Charles de Gaulle

Here the trajectory generation problem is first considered as a minimum time control problem and optimality conditions (Pontryagine's maximum principle [3]) are derived from the mathematical formulation of a global control problem. The analysis of the corresponding optimality conditions shows that the minimum time trajectories resulting from optimal bang-bang control laws can be characterized by few geometric parameters while regular minimum time convergence trajectories can be defined. Then a new mathematical programming problem can be formulated from the original optimal control problem. Since the evolution of aircraft is in general subject to perturbations (winds, navigation and guidance errors, etc), it appears necessary to solve on line this mathematical programming problem according with the current situation.

Considering the complexity of the problem and since aircraft navigation dynamics can produce rapid relative situation changes, the effective on line resolution of this problem is not feasible. However a practical solution strategy, composed of three steps, is proposed in this communication: First, taking advantage of the properties of the regular minimum time trajectories, a merging sequence is established for the N aircraft, then an off line exhaustive solution approach, based on reverse dynamic programming, is developed to cover a large set of initial relative position, finally, feed forward neural networks are built and trained to associate to current relative positions of each aircraft, guidance directives so that they perform efficient merging maneuvers.

The chapter is organized as follows: In Section 2 the trajectory generation problem is described and formulated, then the optimality conditions are analyzed and the regular minimum time trajectories are introduced in Section 3, a sequencing method between merging aircraft is described in Section 4, the generation of a learning data base through reverse dynamic programming is discussed in Section 5 , the resulting feed-forward neural networks are displayed and finally simulations studies are presented in Section 6.

2 Problem Formulation

The case where N trailing aircraft must converge towards the trajectory of another aircraft declared to be the leader is considered in this study. The aircraft are supposed to maintain their common flight level and their respective speeds until the convergence maneuver is completed.Then they are supposed to adopt progressively the speed of the leader aircraft. This maneuver is considered to be completed once the trailing aircraft satisfy the following conditions: their speed is parallel to the leader's speed, they follows the same track and their separations between themselves and with the leader are larger than a minimum separation and smaller than a maximal separation for successive aircraft on the final common track. Here, to avoid redundancies, E is the set of $\frac{(N+1)N}{2}$ pairs given by equation (1):

$$E = \{(1,0),(2,0),...,(N,0),(2,1),(3,1)...,(N-1,N-2),(N,N-1)\} \qquad (1)$$

The adopted minimum separations take into account the minimum separation δ_{ij} imposed by the rules of Civil Aviation and the difference of speed between aircraft:

$$d_{ij}^{min} = \begin{cases} \delta_{ij} & if \quad V_j \geq V_i \\ \delta_{ij} + \frac{(V \ - V \ )^2}{a} & if \quad V_j < V_i \end{cases} \qquad (i,j) \in E \tag{2}$$

The maximal separations at convergence are such that:

$$d_{ij}^{max} = d_{ij}(0)(1 + \rho_{ij}) \qquad with \quad \rho_{ij} > 0 \qquad (i,j) \in E \tag{3}$$

These bounds allow the aircraft to avoid too wide dispersion along the common track at convergence by limiting orthogonal convergence towards the common track. The equations describing the relative motion of the aircraft are given by:

$$\dot{\theta}_{ij} = (-V_j \sin(\theta_{ij} - \psi_j) + V_i \sin(\theta_{ij} - \psi_i))/d_{ij} \qquad (i,j) \in E \tag{4}$$

$$\dot{d}_{ij} = V_j \cos(\theta_{ij} - \psi_j) - V_i \cos(\theta_{ij} - \psi_i) \qquad (i,j) \in E \tag{5}$$

$$\dot{\psi}_i = r_i \qquad i \in \{1, ..., N\} \tag{6}$$

where V_0 and V_i are respectively the speed of the leading aircraft and the speed of the i^{th} trailing aircraft, $d_{ij}(t)$ is the instantaneous separation between aircraft i and j, ψ_0 and ψ_1 are respectively the headings of the leader aircraft and of the i^{th} trailing aircraft and θ_{ij} is the angle of sight of aircraft j from aircraft i (see Figure 2).

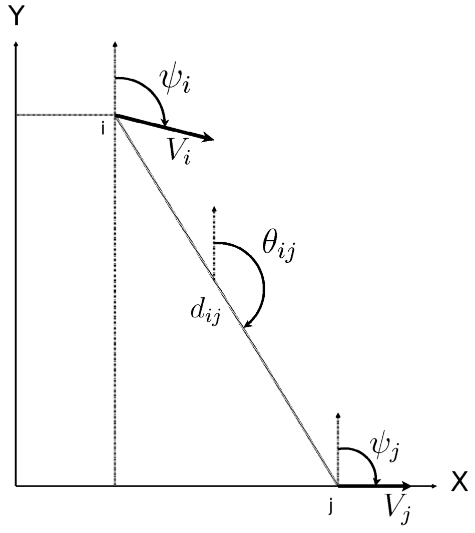


Fig. 2. Relative positions of aircraft

A measure of the effectiveness of such a maneuver can be given by its total duration and a minimum time convergence problem whose solution will provide feasible convergence trajectories for the trailing aircraft is then considered. This problem can be formulated as:

$$min \int_0^t \, dt \tag{7}$$

under the dynamical constraints (4), (5) and (6) and the limit conditions:

$$-\phi_i^{max} \leq \phi_i(t) \leq \phi_i^{max} \quad and \qquad d_{min} - d_{ij}(t) \leq 0 \qquad i,j \in \{1,...,N\} \tag{8}$$

where $\phi_i, i \in \{1,...,N\}$ are the bank angles of the trailing aircraft and $\phi_i^{max}, i \in \{1,...,N\}$ are their maximum values.

Initial conditions must be met:

$$d_{ij}(0) = d_{ij}^0 \qquad with \qquad d_{ij} \geq d_{ij}^{min}, \qquad \theta_{ij}(0) = \theta_{ij}^0 \qquad (i,j) \in E \tag{9}$$

$$\psi_i(t_f) = \psi_0 \qquad i \in \{1,...,N\} \tag{10}$$

as well as final convergence conditions :

$$\psi_i(t_f) = \psi_0 \qquad i \in \{1,...,N\} \tag{11}$$

and

$$\theta_{ij}(t_f) = \psi_0 \qquad and \qquad d_{ij}^{max} \geq d_{ij}(t_f \geq d_{ij}^{min}, \qquad (i,j) \in E \tag{12}$$

The independent variable here is time t with final value t_f and the control variables appear here to be the yaw rates r_i of the trailing aircraf. But since it is supposed that all turn maneuvers are performed in stable conditions, the following relation holds [10]:

$$r_i = (g/V_i).tg\phi_i \qquad i \in \{1,...,N\} \tag{13}$$

and the bank angles of the trailing aircraft ϕ_i can be taken ultimately as the actual control variables which must be set by the lateral function of the autopilot of the trailing aircraft. It is also worth observing that the separation at final convergence has not been imposed here since the ordering of the trailing aircraft is not known beforehand.

3 Characterization of Minimum Time Trajectories

Following the classical results of optimal control theory [3], a Hamiltonian H is associated to the above minimum time optimization problem:

$$\begin{aligned} & H(\theta, d, \psi_i, r_i, \lambda_\theta \;, \lambda_d \;, \lambda_\psi \;, \mu_{ij}, \nu_{ij}) = \\ & 1 + \sum_{(i,j)\in E} (\lambda_d \; (V_j \cos(\theta_{ij} - \psi_j) - V_i \cos(\theta_{ij} - \psi_i))) + \\ & \lambda_\theta \; (-V_j \sin(\theta_{ij} - \psi_j) + V_i sin(\theta_{ij} - \psi_i))/d_{ij} + \lambda_\psi \; r_i + \\ & \mu_{ij}^l (d_{min} - d_{ij} + {\nu_{ij}^l}^2) + \mu_{ij}^h (d_{ij}^{max} - d_{ij} - {\nu_{ij}^h}^2)) \end{aligned} \tag{14}$$

where λ_d , λ_θ and λ_ψ are the co-state variables associated to the differential equations (4), (5) and (6) which provide the rates of change of the state variables d_{ij}, θ_{ij} and ψ_i. When considering the minimum separation constraints, dual variables μ_{ij} are introduced as well as slack variables ν_{ij}, which should be such that:

$$\mu_{ij}^{\alpha}(t) \geq 0 \qquad and \qquad \mu_{ij}^{\alpha}\nu_{ij} = 0 \qquad (i,j) \in E \qquad \alpha \in l,h \tag{15}$$

The optimality conditions can then be established. They are composed of the following terms:

- The Hamilton's canonical equations:

$$d\underline{X}/dt = \partial H/\partial\underline{\lambda} \qquad and \qquad d\underline{\lambda}/dt = -\partial H/\partial\underline{X} \tag{16}$$

with

$$\underline{X}' = (\underline{d}', \underline{\theta}', \underline{\psi}') \qquad and \qquad \underline{\lambda}' = (\underline{\lambda}_d', \underline{\lambda}_\theta', \underline{\lambda}_\psi') \tag{17}$$

- The transversality conditions dealing here with initial and final conditions for the state vector (relations (9), (10), (11) and (12)).

- The minimum principle which states that "the optimal solution is the one which minimizes the Hamiltonian while meeting the constraints".

Then to find the solution of this minimum time convergence problem, a sixth order non-linear two-point boundary problem should in theory be solved. The direct resolution of such problem is in numerical grounds very complex [4], [5] and is not suited for a real time airborne implementation, especially when transportation aircraft are concerned. However, the analysis of the minimization conditions of the associated Hamiltonian with respect to the control variable r_i provides some insight over the optimal trajectory and related operational conditions. Indeed, the above Hamiltonian presents an affine form with respect to the control variable since it can be written such as:

$$H = f(\underline{\theta}, \underline{d}, \underline{\psi}, \underline{\lambda}_\theta, \underline{\lambda}_d, \underline{\mu}, \underline{\nu}) + \sum_{i=1}^{N} \lambda_\psi \, r_i \tag{18}$$

Then its minimization with respect to the control variables leads to a solution of the "bang bang" type (here the attached asterisk indicates that the values are those of an optimal solution)where for $i = 1$ to N:

$$r_i^* = g\tan(\phi_{max})/V_i \qquad if \quad \lambda_\psi \, (t) < 0 \tag{19}$$

$$r_i^* = 0 \qquad and \qquad \phi_i^* = 0 \qquad if \qquad \lambda_\psi^* \, (t) = 0 \tag{20}$$

$$r_i^* = -g\tan(\phi_{max})/V_i \qquad if \quad \lambda_\psi \, (t) > 0 \tag{21}$$

This type of solution has been studied in detail in [7]. It can be shown that the corresponding optimal trajectories are composed of maximum left or right bank angle turns according to the sign of the co-state variable $\lambda_\psi^* \, (t)$, and by straight

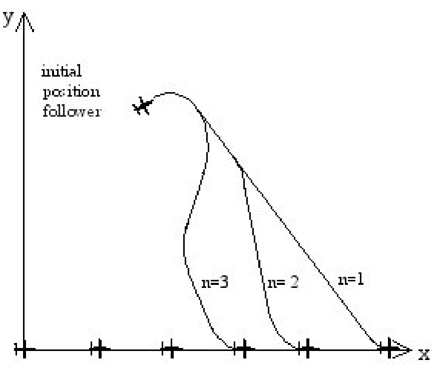


Fig. 3. Examples of regular convergence trajectories with different orders

evolutions at zero bank angle when this variable maintains the zero value. The radius associated with the maximum bank angle turns, when performed in a coordinated way [10], are given by:

$$R_i^{min} = V_i^2/(g.tan(\phi_i^{max})) \qquad i = 1 \quad to \quad N \tag{22}$$

The number and the duration of turns and linear segments depend of the initial relative positions of the aircraft as well as of their respective speeds. Three main cases can be then considered:

- One where direct convergence is possible. In this case the optimal convergence trajectory is composed of a straight-line segment followed by a final turn maneuver to join the route of the leading aircraft.

- One where an initial turn is necessary to insure a converging track for the trailing aircraft towards the route of the followed aircraft. In this case, the minimum time convergence trajectory is composed of two opposite maximum bank angle turns separated by a straight-line segment.

- One where a preliminary separation maneuver is necessary before starting convergence. In this last case the convergence trajectory will be composed of three maximum bank angle turns separated by two straight line segments (one of them can be reduced to an inflexion point).

A regular convergence trajectory of order n (see Figure 3 for examples of different orders) can then be defined as a trajectory composed of a succession of $(n+1)$ pairs, each pair being composed of a straight line segment and a maximum bank angle turn.

The regular convergence trajectory displayed in Figure 4 is characterized by a succession of pairs (ω_i, l_i). Here ω_i is the angular value of the i^{th} maximum rate turn and l_i is the length of the straight line segment leading to the i^{th} maximum bank angle turn.

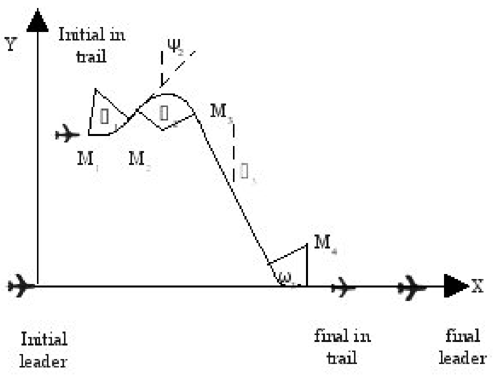


Fig. 4. Representation of a regular convergence trajectory

4 Scheduling of the Merging Aircraft

The numerical solution of the global minimum time problem considered in the previous sections is difficult to be obtained, mainly when considering working conditions, so a practical solution scheme is developed here. It is supposed that the Air Traffic Control- ATC is responsible for establishing the schedule of the merging aircraft in a given area. Two main situations can be considered:

- One in which, ATC builds up an ordered stream of aircraft from the start by assigning to the newest arriving aircraft the last rank in the stream. In this case, given the optimized trajectory of the preceding aircraft, the follower optimizes its own trajectory.

- One in which, given a preexisting bundle of aircraft, ATC decides to organize it in a unique stream of aircraft behind a given leader aircraft. In this case, it is necessary to assign a rank to each aircraft, so that each aircraft shouldl merge behind the preceding aircraft on the common final track. In this situation, the decision problem to be solved is in general quite difficult and a trade-off must be established between the optimality and the complexity of the solution. Indeed, complex solutions may increase the difficulty for aircraft to perform safely the resulting maneuvers while it will be difficult for ATC to monitor the changing traffic situation. What is proposed here, is to solve for each follower aircraft ($i = 1 \quad to \quad N$) the problem of its merging towards the leader aircraft (aircraft 0), ignoring the other aircraft, and then to assign rank 1 to the aircraft performing the fastest merging maneuver. Then the rank 1 aircraft is taken as the reference to be followed for the other merging aircraft and the process is repeated until a rank is assigned to all merging aircraft. However, since the new leader aircraft is not necessarily on the final track when its direct follower starts the convergence maneuver, to avoid the generation of complex trajectories, it appears of interest to built a shadow target representing the new leader aircraft along the final track.

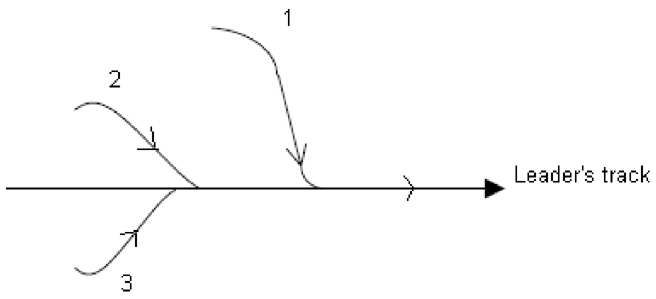


Fig. 5. Scheduling of trailing aircraft (first aircraft)

To lessen the difficulty of the overall ranking problem, a simplification is adopted so that instead of solving N minimum time merging optimization problems involving aircraft i ($i \in 1, ..., N$) and aircraft 0, a simple geometric problem is considered: It is supposed that each aircraft performs a direct merging maneuver towards the track of aircraft 0. This constant speed maneuver is taken to be composed of a first turn at maximum rate, followed by a straight line segment and a final turn. Of course, initial and final turns can be reduced to nothing depending on the initial relative situation. With respect to the target point behind the leading aircraft, to avoid unrealistic solutions, two situations can be considered:

In the first situation the follower aircraft could violate the minimum separation with the leading aircraft, in that case, the first turn will be pursued until minimum separation is guaranteed at convergence.

In the second case, a common reference relative heading, say $\pm\pi/4$ is adopted during the straight part of the convergence maneuver. In both cases, if the trailing aircraft has a speed different to the one of the leading aircraft, an additional segment along the final track must be introduced so that the trailing aircraft can adopt the speed of the leader. This process is represented in Figure 5.

Let x_{i0}, t_{i0} be respectively for aircraft of rank i, the linear position at final convergence in heading ($\psi_F = \psi_L$) and in speed ($V_F = V_L$) and the time of final convergence. Then this aircraft generates for its followers a shadow target of speed V_0 and of position:

$$\tilde{x}_i(t) = x_{i0} - \delta x_{i0} + V_0(t - t_{i0}) \qquad for \quad t \geq 0 \tag{23}$$

where δx_{i0} is an additional separation which can be chosen by the trailing aircraft considering bad weather or navigation accuracy limitations. Then, the minimum time merging trajectory of each trailing aircraft towards its leader may be computed.

In the next section this problem is first considered in an off-line context and its solution is extended, through an intensive computing effort necessary to train a neural network structure, to the on-line context.

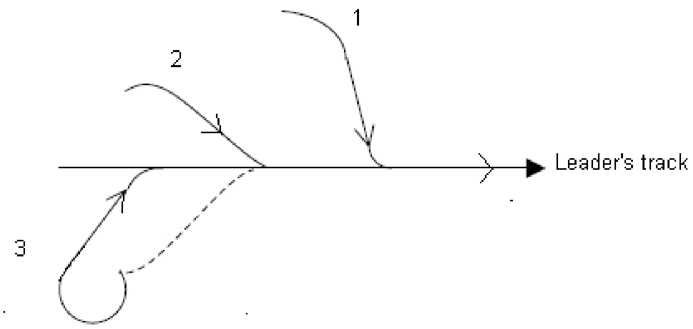


Fig. 6. Representation of a regular convergence trajectory (second trailing aircraft)

5 Off Line Generation of Individual Trajectories

Let $P_n((l_i, \omega_i), i = 1 \quad to \quad n-1)$ be the problem of definition of a minimum time regular convergence trajectory of order n between a leader aircraft L and a follower aircraft F, which, starting from the given initial conditions, satisfies the merging constraints and the minimum separation constraints. Let $S_n^* = ((l_i, \omega_i), i = 1 \quad to \quad n-1)$ be the solution of this problem. From this solution it is possible to build trivially feasible solutions of higher orders. For instance, for P_{n+1}, a feasible solution $\tilde{S}_{n+1}$ is given by:

$$\tilde{S}_{n+1} = ((l_i, \omega_i), i = 1 \quad to \quad n-1) \oplus (l_n, \omega_n) \quad with \quad l_n^n = 0 \quad and \quad \omega_n^n = 0 \tag{24}$$

where $\oplus$ is a concatenation operator for chains of the above pairs.

These solutions having the same duration, it appears that problem P_{n+1} must have a performance at least equal to the one of problem P_n. This result may appear paradoxical but it is useful to observe that the final convergence point is not fixed and that minimum separation constraints can become active. However, in the case in which the final convergence point is fixed, the solution of problem P_n remains interesting since it can determine if the convergence at this given point is feasible with an n order trajectory. It is also desirable to limit as much as possible the number of elementary maneuvers to perform the overall minimum time convergence maneuver so that the comfort of passengers and crews remain at an acceptable level and the workload of air traffic controllers does not become excessive. Then the proposed trajectory generation procedure adopts the following steps:

First choose a maximum order n_{max} for candidate regular convergence trajectories, then solve problem P_n . If this problem has no feasible solution, the intended maneuver is not possible and the considered aircraft must be deleted from the merging list.

Considering the solution of the optimization P_n, figure 4 shows that the duration of a regular convergence maneuver (from point M_n to point M_1) is given by:

$$t_n = \sum_{k=1}^{n-1}(l_k + R_{min}\omega_k)/V_F \tag{25}$$

The convergence conditions can be rewritten as:

$$d^{max} \geq x_L(0) + V_L t_n - (x_F(0) + \sum_{k=1}^{n-1}(l_k sin(\psi_0 + \sum_{h=1}^{k-1}\omega_h) + R_{min}(cos(\psi_0 + \sum_{h=1}^{k}\omega_h) - cos(\psi_0 + \sum_{h=1}^{k-1}\omega_h)) \geq d^{min} \tag{26}$$

$$y_F(0)+(\sum_{k=1}^{n-1}(l_k cos(\psi_0+\sum_{h=1}^{k-1}\omega_h)+R_{min}(sin(\psi_0+\sum_{h=1}^{k-1}\omega_h)-sin(\psi_0+\sum_{h=1}^{k}\omega_h))) = 0 \tag{27}$$

$$\psi_F(0) + \sum_{h=1}^{n-1}\omega_h = \psi_L \tag{28}$$

The minimum separation constraints take the continuous form:

$$\sqrt{(x_F(t) - x_L(t))^2 + (y_F(t) - y_L(t))^2} \geq d_{min} \qquad \forall t \in [0, t_n] \tag{29}$$

and can be approximated by the set of discrete constraints:

$$\sqrt{(x_F(t_n^k) - x_L(t_n^k))^2 + (y_F(t_n^k) - y_L(t_n^k))^2} \geq d_{min} \qquad k = 1toK \tag{30}$$

$$with \quad t_n^0 = 0, \quad t_n^k = k\triangle T \quad k = 1 \quad to \quad K \quad and \quad t_n^K = t_n \tag{31}$$

The minimization of t_n (equation (25) under constraints (26), (27), (28), (30) and (31) is a non-convex mathematical programming problem. An exact method which seems of interest here once the problem has been discretized is Dynamic Programming [8]. This method is known to be quite efficient to construct recursively a solution for a separable optimization problem. To turn separable the above problem, constraints (30) can be changed to:

$$\begin{aligned} |x_F(t_n^k - x_L(t_n^k)| \geq d_{min} \quad for \quad t_n^k \in T_X \\ and \quad |y_F(t_n^k - y_L(t_n^k)| \geq d_{min} \quad for \quad t_n^k \in T_Y \end{aligned} \tag{32}$$

where $T_X$and T_Y constitute a partition of $(t_n^0, t_n^1, ..., t_n^K)$. The optimality conditions established in Section 3 have not allowed to get a practical numerical solution to an instance (initial relative positions, aircraft speeds and final relative positions) of the minimum time convergence problem, however, they have provided a good insight into the regular structure of optimal convergence trajectories. The retained formulation for problem P_n does not lead as well to a solution process which can be effectively run on board the aircraft. Nevertheless,

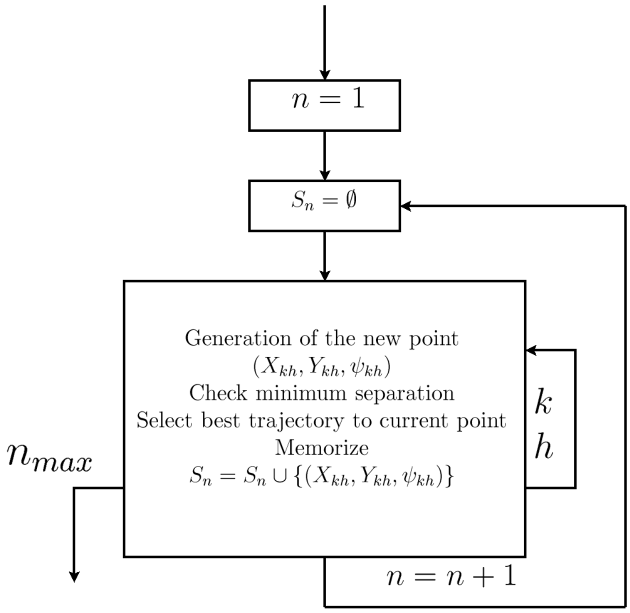

$n = 1$
$S_n = \emptyset$
Generation of the new point
$(X_{kh}, Y_{kh}, \psi_{kh})$
Check minimum separation
Select best trajectory to current point
Memorize
$S_n = S_n \cup \{(X_{kh}, Y_{kh}, \psi_{kh})\}$
k
h
n_{max}
$n = n + 1$

Fig. 7. Inverse generation of optimal convergence trajectories sets

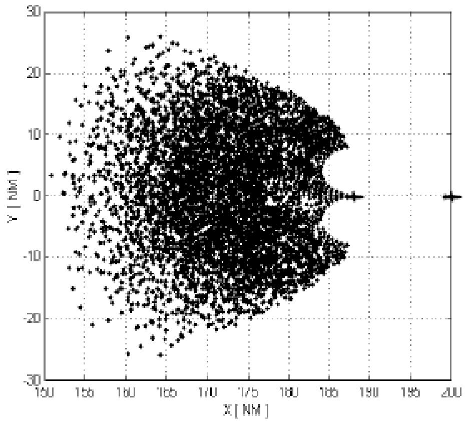


Fig. 8. Discretization of the convergence manoeuver relative space

if initial conditions are relaxed from problem P_n , it is possible to solve it step by step, starting from final imposed convergence conditions, using a reverse dynamic programming process. Then, it is possible to generate in sequence and in a reverse way, the set of triplets (x_0^s, y_0^s, ψ_0^s) associated to the corresponding n_{max} order feasible optimal convergence trajectories. Then, if the angular range of ω is discretized into N_ω values, such as $\omega_k = (k-1)\triangle\omega, k = 1 \quad to \quad N_\omega$,

Table 1. Size of merging parameters neural network generators

Parameter	Number of neurones in the hidden layer
Length of initial progression	35
Angle of first turn	40
Length of intermediate progression	35
Angle of second turn	40
Length of the third progression	20
Angle of final turn	30

and if the length of the straight segments is discretized into N_l values such as $l_h = (h-1)\triangle l, \quad h = 1 \quad to \quad N_l$, at each stage, the number of generated triplets will be multiplied by $N_l(2N_\omega - 1)$. The recursive structure of the algorithm used for the generation of the convergence trajectories is displayed in Figure 7. The set of generated trajectories builds a tree whose root is given by $(-D, 0, \psi_L)$ in a reference frame of $\mathcal{R}^2 \times [0, 2\pi]$, attached to the final estimated position of the leading aircraf. Here D is the final separation. To each point in this space are associated the starting parameters of an optimal regular trajectory of minimal order. This input-output data can be memorized in a data base to give grounds to an on-line interpolation by the relative guidance computers of the merging aircraft(Figure 8).

6 On-Line Generation of Guidance Directives

It has been shown above that possible solution trajectories present a large diversity of regular shapes. In general a convergence trajectory will last at least some minutes during which the aircraft can be submitted to wind effects and to inaccuracies of the navigation and guidance systems. Also, during this period, the leading aircraft can be driven by the air traffic control system to modify its flight plan or its guidance references. Then, once the convergence maneuver is started, it will be necessary to restart repeatedly the problem of generation of new minimum time convergence trajectories. Then it appears interesting to designe a new system able to perform an on line generation of the current guidance parameters necessary to perform a whole convergence trajectory.

Then at regular time intervals (some seconds), the relative guidance computer of a trailing aircraft must assess its relative position with respect to its predecessor, then it will check in the database of the new guidance system which absolute guidance parameters have to be adopted momentarily to goes on efficiently with the convergence maneuver. Since in many situations, no database input is exactly identical with the current relative situation of a trailing aircraft, an interpolation appears necessary. This can be achieved using a classical feed forward neural network device as it has been proved to be effective in [9] (see Figure 9 where x and y are the position, ψ is the heading and ϕ is the bank angle of the trailing aircraft).

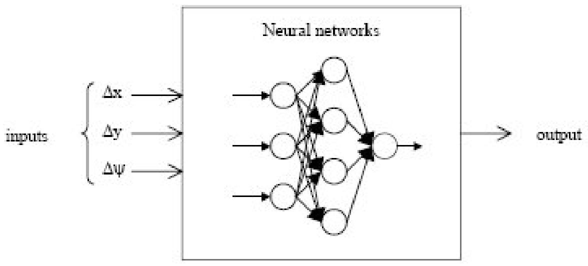


Fig. 9. Adopted neural network structure

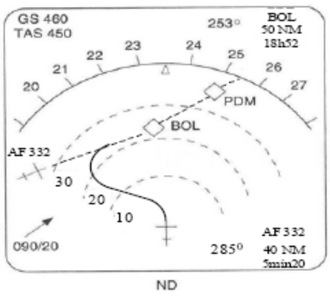


Fig. 10. Example of possible convergence trajectory representation on a Navigation Display (ND) Global view

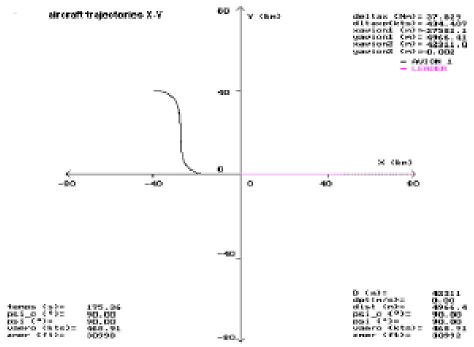


Fig. 11. Example of steady convergence maneuver

In Figure 10 the navigation display (ND) of an in trail aircraft is presented. This in trail aircraft is to start a convergence maneuver towards flight AF332 which is estimated to last 5 minutes and 20 seconds over a distance of 40 nautical miles. Various numerical simulation experiments involving accurate wide body simulation models have been performed. The new guidance function has been

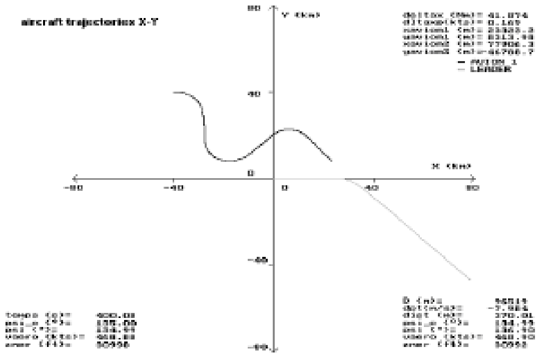


Fig. 12. Example of swirling convergence maneuver

integrated into the simulated autopilot of the trailing aircraft and has provided smooth convergent trajectories. Different case studies have been considered, including some in which the leading aircraft follows a straight route (Figure 11) and some in which the leading aircraft performs unexpected lateral maneuvers (Figure12).

7 Conclusion

In this chapter the merging manoeuver of N aircraft behind a leading aircraft has been considered. The whole problem has been first formulated as a minimum time control problem. Optimality conditions have been derived and their analysis has shown that the minimum time convergence trajectories can be characterized by some few geometric parameters. Then a new mathematical programming problem can be formulated from the original optimal control problem to generate sets of regular convergence trajectories. This allows ATC to choose simply a merging sequence for the N aircraft and the global merging problem can be split in N dual leader-follower relative guidance problems. Since the flight evolution of each aircraft is in general subject to perturbations it appeared of interest to introduce a neural network structure to provide in real time updated directives to the guidance system of each merging aircraft.

Here the adopted coordination principle is based on the characteristics of the minimum time convergence trajectories since they allow the ATC to establish the merging sequence and then they provide to each merging aircraft a guideway to perform autonomously its own merging maneuver.It appears that the proposed solution is compatible with modern on board guidance systems ([11] and [12]) and may contribute to enhance booth safety and capacity of the air traffic system.

References

1. European commission & Eurocontrol: CARA/ASAS Activity 5 description of a first package of GS/AS applications. version 2.2 (September 2002)
2. Ivanescu, D., Hoffman, E., Zeghal, K.: Impact of ADS-B link characteristics on the performances of in-trail following aircraft. In: AIAA Guidance, Navigation and Control Conference, Monterey, USA (August 2002)
3. Bryson, A.E., Ho, Y.C.: Applied Optimal Control. Hemisphere Publishers, Washington (1975)
4. Culioli, J.C.: Introduction á l'optimisation Ellipses Editeurs, Paris, pp. 225–246 (1994)
5. Betts, J.T.: Survey of Numerical Methods for Trajectory Optimization. Journal of Guidance, Control and Dynamics 18(1), 151–159 (1995)
6. Hagelauer, P., Mora-Camino, F.: Evaluation of Practical Solutions for On Board Aircraft Fourth Dimensional Guidance. Journal of Guidance, Control and Dynamics 20(5), 1052–1054 (1997)
7. Shahzad, M.: Contribution á l'optimisation du guidage relatif des aéronefs. PhD Dissertation, LAAS du CNRS, Toulouse, pp. 208–229 (2000)
8. Bertsekas, D.P.: Dynamic Programming and Optimal Control, vol. 1. Athena Scientific, Belmont (1995)
9. Slama, J.G., Shazad, S., Mora-Camino, F.: A Neural Adaptive Approach for Relative Guidance of Aircraft. In: 7th MDP Conference, Cairo, pp. 123–130 (2000)
10. Mora-Camino, F.: Systémes de Conduite Automatique et de Gestion du, Toulouse, vol. ENAC (1993)
11. Lu, W.C., Mora-Camino, F., Achaibou, K.: Differential Flatness and Flight Guidance: a Neural Adaptive Approach. In: Proceedings of AIAA Guidance, Navigation and Control Conference, San Francisco, August 14-18 (2005)
12. Miquel, T., Mora-Camino, F., Casaux, F., Loscos, J.M.: A Simplified Backstepping Design for 3D Time-based Aicraft Relative Guidance. In: Proceedings of AIAA Guidance, Navigation and Control Conference, Providence (August 2004)

A Comparative Study of Task Assignment and Path Planning Methods for Multi-UGV Missions*

Johan Thunberg[1], David A. Anisi[2], and Petter Ögren[1]

[1] Department of Autonomous Systems,
Swedish Defence Research Institute (FOI),
SE-164 90 Stockholm, Sweden
{johan.thunberg,petter.ogren}@foi.se
[2] Optimization and Systems Theory,
Royal Institute of Technology (KTH),
SE-100 44 Stockholm, Sweden
anisi@kth.se

Abstract. Many important problems involving a group of unmanned ground vehicles (UGVs) are closely related to the multi traviling salesman problem (m-TSP). This paper comprises a comparative study of a number of algorithms proposed in the litterature to solve m-TSPs occuring in robotics.

The investigated algoritms include two mixed integer linear programming (MILP) formulations, a market based approach (MA), a Voronoi partition step (VP) combined with the local search used in MA, and a deterministic and a stocastic version of the granular tabu search (GTS).

To evaluate the algoritms, an m-TSP is derived from a planar environment with polygonal obstacles and uniformly distributed targets and vehicle positions.

The results of the comparison indicate that out of the decentralized approaches, the MA yield good solutions but requires long computation times, while VP is fast but not as good. The two MILP approaches suffer from long computation times, and poor results due to the decomposition of the assignment and path planning steps. Finally, the two GTS algorithms yield good results in short times with inputs from MA as well as the much faster VP. Thus the best performing centralized approach is the GTS in combination with the VP.

1 Introduction

Often missions with systems of unmanned ground vehicles (UGVs) translate into $\mathcal{NP}$-hard problems. A way to simplify the problem is to decompose the mission into a set of subproblems. One such decomposition is to design a task assignment

* Funded by the Swedish defence materiel administration (FMV) and the Swedish armed forces through the Technologies for Autonomous and Intelligent Systems (TAIS) project. 297316-LB704859.

problem which allocates tasks to the vehicles, and a path planning box which solves the path planing problem for each vehicle. Beneath the path planning box low level boxes dealing with trajectory planing and vehicle actuators can be used.

If task assignment and path planning occurs in a cooperative manner, *i.e.* there is no clear dividement between task assignment and path planning, it will be referred to as cooperative task assignment and path planning (CTAPP) or a one-box approach. On the contrary if task assignment and path planning occurs sequentially, *i.e.* there is no feedback from path planning to task assignment, it will be referred to as sequential task assignment and path planning (STAPP). In [11] a taxonomy is presented over many multivehicle task assignment problems.

A multivehicle system have many axis of freedom [13] such as centralized system *vs.* decentralized system and explicit comunication *vs.* implicit communication. A good algorithm should be able to handle these different aspects. Many algorithms have been suggested over the years, but comparative studies of the their performance are still lacking, presenting such a comparative study is the main contribution of the paper in hand. In [15] a comparative study over UAV path planning, can be found.

Some of the main research directions include meta heuristic approaches such as tabu search [4, 9, 17, 18] simulated annealing or ant colony optimization [9]. Market based approaches where the system of vehicles shall resemble a market economy, have also drawn much attention lately [5, 16, 22, 26, 27]. Often the missions can be formulated as mixed integer linear programming (MILP) problems [1, 6, 21]. Some MILP approaches are introduced in [14] that solve task assignment and path planning sequentially (STAPP) for a problem similar to the multi traveling salesman problem (m-TSP). In missions including threats, voronoi diagrams are commonly used [2, 12]. Voronoi diagrams can also be used in task assignment [8] or in obstacle avoidance [19].

The outline of this paper is as follows. In Section 2 the problem will be defined on which the algorithms are compared and in Section 3, the algorithms will be presented. Then, in Section 4 it will be described how they are implemented and results will be presented. Finally, conclusions are drawn in Section 5.

2 Comparative Study

There are assumed to be a group of N vehicles that live on a connected compact subset of $\mathbb{R}^2$, for simplicitly a square. This set, which will be referred to as Q, will have polygonal holes representing obstacles. Targets or tasks are defined as points in Q that must be visited to fulfill an objective.

In this setting the algorithms will be compared on the following problem.

Problem 1 (Multi-vehicle multi-task problem). Consider N vehicles and M targets uniformly distributed on Q. Each vehicle has the capacity to visit all targets and each target must be visited by one vehicle and one vehicle only. Find the optimal paths for the vehicles so that all targets are visited and either the mission completion time (a) or the fuel consumption (b) is minimized.

Constant vehicle speed is assumed and minimizing completion time is hence equivalent to minimizing the length of the longest path for any vehicle. Therfore, Problem 1a is refered to as the *min-max* objective. Minimizing fuel consumption for the vehicle group is further more assumed to be the same as minimizing the total distance traveled for the vehicle group. Problem 1b is thus referred to as the *min-sum* objective.

3 Algorithms

In this section we first list the algorithms being compared, and then give a more detailed description of them. The first algorithm is the market based approach, presented in [26]. The next two methods are two of the mixed integer linear programming (MILP) methods presented in [14] and the fourth method is a voronoi partitioning algorithm inspired by [8]. Also included in this study is the granular tabu search [25] and a stochastic modification of it. The algorithms are slightly modified to fit Problem 1 and compared in Monte carlo simulations. The methods are not only chosen on the basis of their closeness to optimality, characteristics like computational time and possibilities to work in different systems such as decentralized or centralized ones are also important.

3.1 Market Based Approach

This market based algorithm is presented in [26] and is a CTAPP algorithm. The algorithm will be referred to as MA. The mission is to allocate a group of targets to a group of vehicles. For that purpose bidding rules are introduced. Assume there are N vehicles $r_1, ..., r_N$ and M currently unallocated targets $t_1, ..., t_M$. Assume further that a set of targets T_i is assigned to each vehicle r_i, *i.e.* there is a set $\{T_1, ..., T_N\}$. Let PC(r_i,T_i) denote the minimum path cost of vehicle r_i when visiting all targets in T_i. The values of PC(r_i,T_i) are calculated locally using a meta-heuristic procedure. The two team objectives are

– Problem 1a (min-sum): $\min_T \sum_j \text{PC}(r_i,T_i)$, and
– Problem 1b (min-max): $\min_T \max_j \text{PC}(r_i,T_i)$

Each unallocated target is bidden upon by vehicles in a so-called bidding round according to a bidding rule. The vehicle with the lowest bid will win, and the target will be allocated to that vehicle.

Bidding Rule. Robot r bids on target t the difference in performance for the given team objective between the current allocation of targets to vehicles and the allocation that results from the current one if robot r is allocated target t. (Unallocated targets are ignored.)

The advantage of having this auction based system, is that each vehicle is decoupled from the others in the bidding process, which makes the system robust. The process will continue if a vehicle suddenly stops responding, due to a software or hardware failure. The targets that were allocated to that vehicle will be unallocated once again. Since an auctioneer has to be present, the system is not fully decentralized. If the auctions are administrated by a single unit, there will be a single point of failure. However the process can be made more robust if the auction is administrated by a group of vehicles.

Each vehicle maintains a path visiting the targets that are allocated to it. A bid on a target is performed in the following steps. First the target is inserted into all possible positions of the path associated with the vehicle, and new temporary paths are created.

Each temporary path, will now be improved by two sequentially performed local search heuristics, or hill climbing algorithms. The first algorithm uses the 2-opt improvement rule, which takes the path and reverses the order of each of its continuous subpaths, takes the best path and repeats the process until no improvement is possible [10, 23, 26]. The second algorithm is performed on the path that is generated by the first algorithm and uses the 1-target 3-opt improvement rule, which takes a target and puts it between two other targets [3, 26].

3.2 Two MILP Approaches

The mixed integer linear programming (MILP) algorithms described below can be found in [14]. The algorithms will be referred to as MILP1 and MILP2 and are STAPP algorithms. In accordance with [26], these algorithms are only applied to Problem (1a), i.e. the min-max objective, and we use the notation of [15] below.

The objective is to calculate T^*, where

$$T^* \equiv \min_{j} \max_{A(j)} T(j)_i^*,$$

is the least maximum traveling cost among all vehicles finishing all its subassignments. The assignment $A(j) \in A$ is one of the feasible assignments, $A(j)_i$ is the the subassignment given to the i^{th} vehicle, and $T(j)_i^*$ is the traveling cost for the i^{th} vehicle finishing all its subassignments.

Two problems are described in [15], one in which the vehicles need to return to their start positions and one in which they do not. Only the latter one will be considered, since it coincide with Problem 1 for the min-max objective. The cost $C_0(i,k)$ is the cost of traveling from the start position of the i^{th} vehicle to the position of the k^{th} target, and C_0 is a matrix. The function $\eta(\upsilon,\omega)$ is defined as $\eta(\upsilon,\omega) = (\upsilon-1)N - \upsilon(\upsilon-1)/2 + \omega - \upsilon$, where $\eta \leq \upsilon$. It is a mapping from $\{1,\dots,N\} \times \{1,\dots,N\} \to \{1,\dots,N(N-1)\}$ and is used to convert a two index variable, x_{ij} into a single index variable, $\bar{x}_k$ with $k = \eta(i,j)$. The cost vector $c(\eta(\upsilon,\omega))$ is the cost of traveling between the target υ and ω.

An exact formulation E and two non-exact algorithms H_1 and H_2 are proposed for the problem. The two non-exact methods are STAPP algorithms. The exact MILP formulation E is:

$$F_E\text{: minimize } r$$

$$\sum_{k=1}^{M} a_{ij}^{k} \leq 1 \quad \forall i, j, \tag{1}$$

$$\sum_{i=1}^{N}\sum_{j=1}^{q} a_{ij}^{k} = 1 \quad \forall k, \tag{2}$$

$$a_{ij}^{k} \in \{0,1\} \quad \forall i, j, k, \tag{3}$$

$$\sum_{k=1}^{M} a_{i(j+1)}^{k} \leq \sum_{k=1}^{M} a_{ij}^{k} \quad \forall i, j, \tag{4}$$

$$a_{ij}^{\upsilon} + a_{i(j+1)}^{\omega} + a_{ij}^{\omega} + a_{i(j+1)}^{\upsilon} = 2y_{i\eta(\upsilon,\omega)}^{j} + \tilde{y}_{i\eta(\upsilon,\omega)}^{j} \quad \forall i, j, \upsilon, \omega, \tag{5}$$

$$y_{i\eta(\upsilon,\omega)} = \sum_{j} y_{i\eta(\upsilon,\omega)}^{j} \quad \forall i, \upsilon, \omega, \tag{6}$$

$$y_{i\eta(\upsilon,\omega)}^{j} \in \{0,1\}, \quad \tilde{y}_{i\eta(\upsilon,\omega)}^{j} \in [0,1] \quad \forall i, j, \upsilon, \omega, \tag{7}$$

$$\sum_{k=1}^{M} C_0(i,k) a_{i1}^{k} + \sum_{\upsilon=1}^{n-1}\sum_{\omega=\upsilon+1}^{M} c(\eta(\upsilon,\omega)) y_{i\eta(\upsilon,\omega)} \leq r \quad \forall i, \tag{8}$$

where $i \in \{1, ..., M\}$, $j \in \{1, ..., q\}$ for (1), (2) and (3) or $j \in \{1, ..., q-1\}$ for (4), (5), (6) and (7). Each vehicle has a capacity limit of handling q targets. The binary variable a_{ij}^{k} only equals 1 when the k^{th} target is contained in the j^{th} so-called *room* of the i^{th} vehicle. Each room can at most contain one vehicle, and every vehicles must be contained in some room, (1) and (2), respectively. The rooms of a vehicle must be filled in ascending order, (4). In (5) the binary variable $y_{i\eta(\upsilon,\omega)}^{j}$ equals 1 if target υ and ω are contained in the j^{th} and the $(j+1)^{\text{th}}$ room of the i^{th} vehicle. If only one room is filled, $\tilde{y}_{i\eta(\upsilon,\omega)}^{j}$ equals 1. In (6) the room index is reduced and the binary variable $y_{i\eta(\upsilon,\omega)}$ is used in the total cost of each vehicle, (8). Equation (8) is the total cost of each vehicle when finishing all its tasks. The amount of binary variables needed for E is of order MN^2q.

The only modifications that need to be done to adapt this formulation to the formulation in Problem 1, is that the room size of each vehicle, q, equals M. This algorithm is only implemented for the min-max objective.

The two nonexact MILP formulations H_1 and H_2 are two step procedures. In the first step a partitioning MILP will run and allocate targets to vehicles, resulting in a target set for each vehicle. In the second step F_E will run on each vehicle-target set, *i.e.* F_E will run N times in total. The two different partitioning methods are presented below.

Partition $\boldsymbol{H_1}$

$$F_1 : \quad \text{minimize} \quad r$$

subject to:

$$\sum_{j=1}^{M} x_{ij} \leq q \quad \forall i, \tag{9}$$

$$\sum_{i=1}^{N} x_{ij} = 1 \quad \forall j, \tag{10}$$

$$C_0(i,j)x_{ij} \leq r \quad \forall i, \tag{11}$$

$$x_{i,j} \in \{0,1\} \quad \forall i,j, \tag{12}$$

where vehicle index $i \in \{1,...,N\}$ and target index j $\in \{1,...,M\}$. This MILP requires only MN binary variables and reduces the computation time significantly. However in this partitioning the maximum distance between a target and a vehicle is minimized, which means that an optimal solution not necessarily is the voronoi partitioning. The voronoi partitioning is however always guaranteed to be an optimal solution to the problem above. The binary variable x_{ij} equals one if target j is allocated to vehicle j.

Partition $\boldsymbol{H_2}$

$$F_2 : \quad \text{minimize} \quad r$$

subject to:

$$\sum_{j=1}^{M} x_{ij} \leq q \quad \forall i, \tag{13}$$

$$\sum_{i=1}^{N} x_{ij} = 1 \quad \forall j, \tag{14}$$

$$y_{i\eta(j,k)} \leq \frac{x_{ij} + x_{ik}}{2} \leq y_{i\eta(j,k)} + \frac{1}{2} \quad \forall i,j,k \ (j < k) \tag{15}$$

$$c(\eta(j,k))y_{i\eta(j,k)} \leq r \quad \forall i,j,k \ (j < k), \tag{16}$$

$$C_0(i,j)x_{ij} \leq r \quad \forall i, \tag{17}$$

$$x_{i,j}, y_{i\eta(j,k)} \in \{0,1\} \quad \forall i,j,k \ (j < k), \tag{18}$$

where $i \in \{1,...,N\}$ and $j \in \{1,...,M\}$. This program also minimizes the distance between the two targets that are assigned to the same vehicle and are furthest away from each other. This MILP requires M^2N binary variables. The two methods of solving a partitioning problem H_1 and then E, and H_2 and then E will from now on be referred to as MILP1 and MILP2 respectively.

3.3 Voronoi Partitioning

This algorithm is a STAPP algorithm. First a task assignment process is performed using a so-called Voronoi partitioning [8], then the local search described in (MA) above is applied.

This algorithm is inspired by [8], but is not an exact implimentation. The same partitioning technique is used. The idea is to first allocate each target to its nearest vehicle. This is the same as constructing a so-called Voronoi diagram. After this partitioning has been performed, each vehicle will solve a traveling salesman problem (TSP) on its subset of targets, using the local search of (MA). The Voronoi partitioning is polynomial in time.

3.4 Granular Tabu Search

This is a slightly adjusted version of the granular tabu search (GTS), described in [25]. The algorithm is applicable to a broad class of combinatorial optimization problems, but is implemented for Problem 1. Some new features are added to make the algorithm a bit more effective, and the algorithm is adjusted to fit both objectives; 1a (min-max) and 1b (min-sum). A stochastic variant of this algorithm was also evaluated. These algorithms are CTAPP algorithms, and will be referred to as GTS and SGTS respectively.

This is a meta heuristic method that is applied to a feasible starting guess, the result of another algorithm. First one of the other algorithms are applied to Problem 1, and a solution is generated. This solution is the input to GTS, which aims at improving the solution further. The only constraint on the input solution is that it is feasible. However it will work better on a good solution because of the granularity threshold explained below.

The solution paths of all vehicles are represented by two vectors of size $N+M$, and are denoted IN and OUT. The first N elements of the vectors correspond to the vehicle positons and the last M elements correspond to the target positions. Each vehicle and target has a unique id-number in the interval $(0,...,N+M-1)$. The elements $IN(i)$ and $OUT(i)$ are the positions that are visited before and after target i on the vehicle path that visits target i.

A virtual depot position is created, which has id-number $N+M$. The elements $IN(i) = N+M \quad \forall i \in \{0,...,N-1\}$. The output, *i.e.* OUT, of the targets that are the last targets on a vehicle path also equals $N+M$. The virtual depot is an easy way to see where a path starts and ends. Of course, the inputs of the vehicles must always be the virtual depot.

The main idea behind tabu search heuristics are that local search or hill climbing algorithms are used, but to avoid local minima tabu lists are introduced. In these lists recent moves are stored and regarded as tabu, *i.e.* they cannot be used. Also when a local minimum has been reached, the algorithm is forced to make a *move* even though it will not improve the solution, see Figure 1. When the solution has not been improved for n iterations the algorithm stops and the best solution achieved during the algorithm is collected. In this algorithm an overriding criteria is used, where moves in the tabu lists can be used if it leads to a new global minimum.

The term *move* needs to be explained further. Moves or exchange operators operate on arcs, where an arc in this context is the directed closest path from one target or vehicle to another target. In a k-exchange or k-opt move, k arcs are removed and replaced by k other arcs, hence the cardinality of the neighborhod of all possible k-exchanges is $\mathcal{O}((N+M)^k)$. Special exchange operators that are 2-opt, 3-opt and 4-opt will be used that only has the neighborhod cardinality $\mathcal{O}((N+M)^2)$.

The following exchange/move operators will be used: The *two exchange*, the *target insertion*, the *two target insertion* and the *swap*. These are illustrated in Figure 2.

There are $(N+M)^2$ arcs, however the arcs between the vehicles are given a very large cost. Not all arcs are used in the algorithm, which is the core aspect of this algorithm. Only arcs that have an associated cost that is smaller than a certain granularity threshold υ. The threshold υ is calculated as

$$\upsilon = \beta \frac{z}{N+M},$$

where z is the cost of the solution of the input algorithm to GTS, *e.g.* the market based algorithm. Running GTS with MA as input algorithm will be referred to as GTS-MA. The threshold is a factor β of the average arc cost in the input solution to GTS. Various values of β can be used. If the solution is not improved for a certain number of steps, β is increased to a higher value during some consecutive iterations, and then decreased to a lower value again.

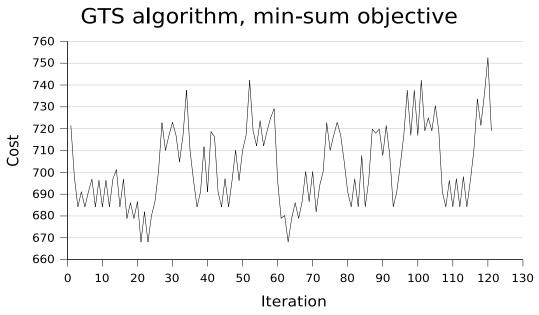


Fig. 1. The cost at each iteration is shown for the tabu search algorithms (GTS) in a scenario with 20 vehicles and 40 targets

When using the vectors IN and OUT the change of the cost of the total solution when applying a move is calculated in constant time. Just to demonstrate this fact, the exchange operator with the min-sum objective will be looked at. The exchange is applied on arc $(a1, b2)$ in Figure 2. The change in total cost will be

$$-cost(a1, OUT(a1)) - cost(IN(b2), b2) +$$

$$cost(a1, b2) + cost(IN(b2), OUT(a1)),$$

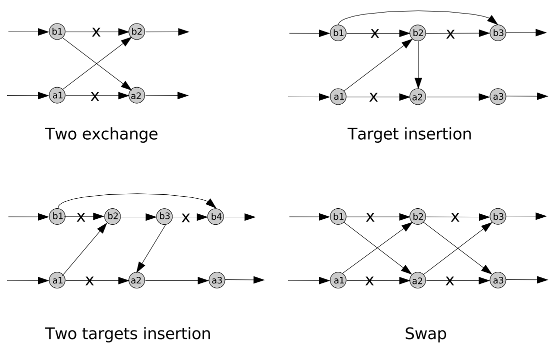


Fig. 2. The four exchange operators. Here a and b are two different vehicle paths.

where $OUT(a1) = a2$ and $IN(b2) = b1$. In the stochastic version of this algorithm, the best moves are not always chosen in the local search process. With some probability, other moves can also be chosen.

This concludes the algorithm descriptions.

4 Implementation and Results

In this section we will describe the simulation environment and the results of running the algoritms.

4.1 Simulation Environment

The simulation environment consists of two parts. In the first part a problem instance in terms of an environment is generated, and in the second part the algorithms are applied to this problem instance.

In the environmental generation part, first a scenario is generated according to Problem 1. From the scenario a visibility graph is generated [20]. The third and last step of the environmental generation process is to calculate a cost matrix. This is a matrix where the (i,j):th entry is the optimal cost of traveling between vehicle/target i and vehicle/target j, where element (i,j) is equal to element (j,i). The minimum cost path between two nodes is calculated using the A*-graph search algorithm [7].

In the second part the algorithms are compared on the same cost matrix. All the algorithms except for the MILP methods are implemented in C++. The MILP methods are implemented in the matlab interface of CPLEX 8.0.

4.2 Results

The results are presented in Figures 3-8. First we describe the technical details of the data set and then we discuss the results one figure at a time.

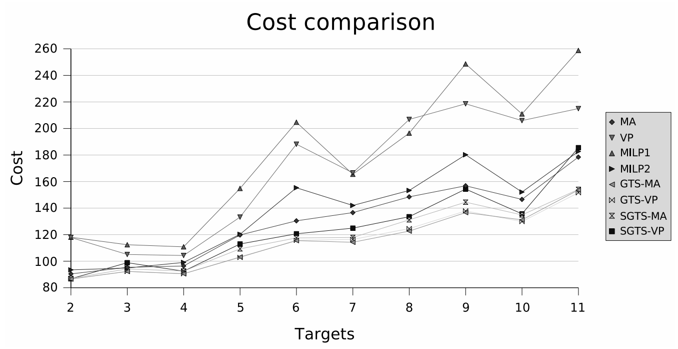


Fig. 3. Cost comparison for 4-vehicle problems using the min-max objective. Note that VP and MILP1 find the highest costs, followed by MILP2 and MA while most of the time, all (S)GTS find lower costs.

In all scenarios there are 12 rectilinear possibly nonconvex polygonal buildings. For each combination of vehicles and targets there are 50 generated scenarios in figure 3, and 25 generated scenarios in Figures 5-8. The MILP algorithms are not included in Figures 5-8, because of the high computational times on large problems. The computational times presented in Figure 4 are found using CPLEX 8.0. on an Intel Pentium 4 2.40GHz with 512 MB ram, while the times in Figures 6 and 8 are from simulations on an Intel Pentium M 1.6 GHz with 1280 MB ram.

Running the algoritms on a small problem of 4 vehicles and up to 11 targets we get the costs depicted in Figure 3, where it can be seen that the GTS and SGTS find lower costs than the rest. The corresponding computation times are an order of magnitude higher for the two MILP approaches. These are

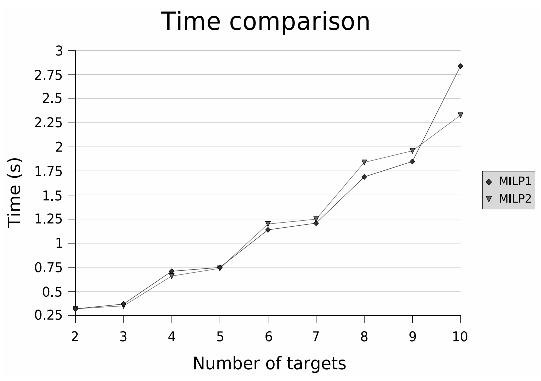


Fig. 4. Time comparison for MILP algorithms on the 4-vehicle problems using the min-max objective. Computation times for the other algorithms on a larger problem can be found in Figure 6.

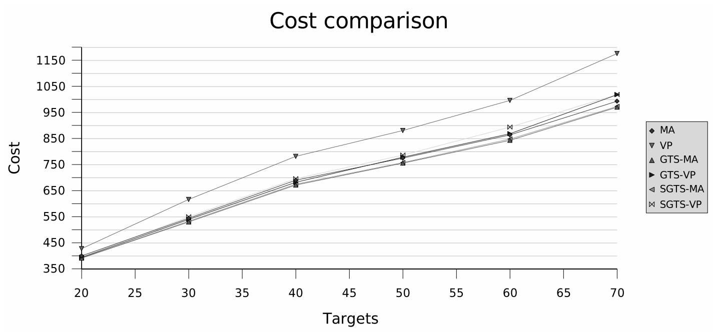


Fig. 5. Cost comparisons for 20-vehicle problems using the min-sum objective. Note that the MA is consistently better than the VP, both as stand alone algorithms and as inputs to the (S)GTS. The best costs are found using MA-GTS and MA-SGTS.

therefore shown separately in Figure 4, where the median computation times are shown. In fact the worst computation times on these problems were as large as 200 seconds for the MILP1 algorithm on a problem with 4 vehicles and 10 targets.

The results from a larger problem with 20 vehicles and up to 70 targets can be found in Figures 5-6. Here it can be seen that while the VP is faster than the MA, the MA finds lower costs than VP. Both of MA and VP are easily to implement in a decentralized manner to improve overall system robustness.

If, on the other hand, centralized computations can be made, the GTS and SGTS algorithms can improve the results of the VP considerably. Using MA or VP as inputs to GTS or SGTS the resulting costs differ only slightly, while the combined computation times are still much longer for the algorithms using MA as starting solution.

The differences between the GTS and the SGTS are small. Most of the time the GTS find better solutions than the SGTS, but sometimes the opposite holds.

Fixing the number of targets at 20 and varying the number of vehicles from two to ten you get the results depicted in Figures 7-8. The overall trends are the same as in Figures 5-6.

As seen above, the division of the combined task assignment and path planning problem (CTAPP) into two separate problems (STAPP), as in VP and MILP1,2 can give lower computation times, but at the price of solution quality. The solutions can however be improved upon using heuristics, such as the (S)GTS, if centralized computations can be performed.

We were surprised by the performance of the algorithm proposed in [25] and we think that heuristics such as the GTS should be considered, before trading solution quality for computation time by partitioning hard problems into subproblems with different optimal solutions. For more results see [24].

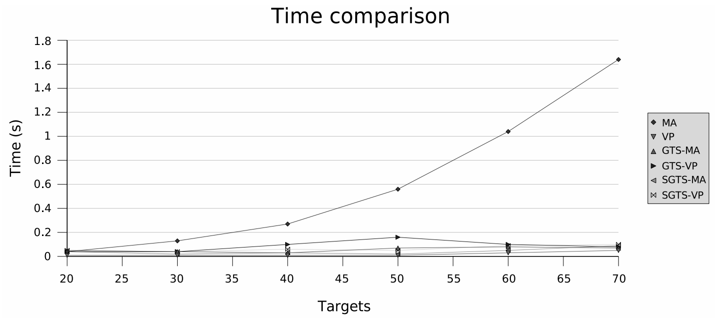


Fig. 6. Time comparison for 20-vehicle problems using the min-sum objective. All VP algorithms finish well under 0.2 s, while the MA ones need up to 1.8 s. Note that the total running time of e.g.the GTS-MA is the sum of the MA and the GTS-MA curves.

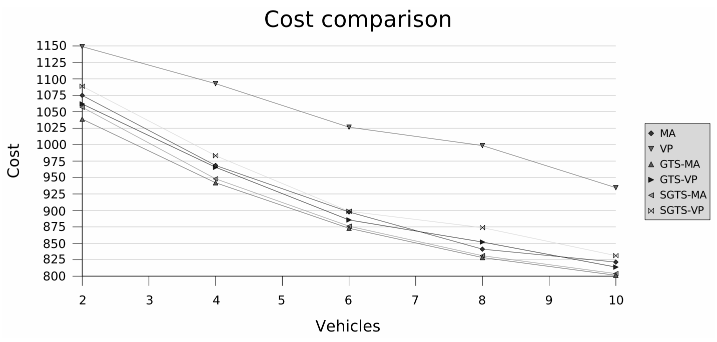


Fig. 7. Cost comparison for 20-target problems using the min-sum objective. As in Figure 5 above, the best costs are found using MA-GTS and MA-SGTS.

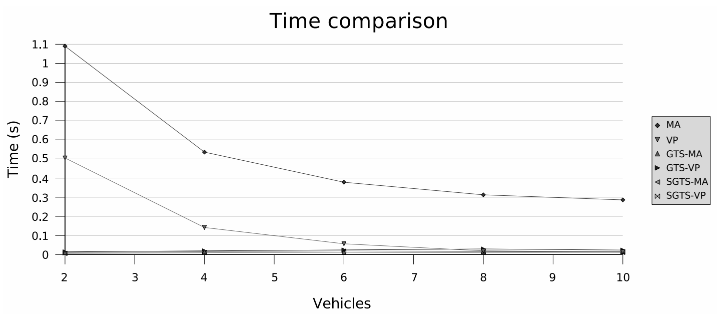


Fig. 8. Time comparison for 20-target problems using the min-sum objective. As in Figure 6, the total computational times of the MA algorithms are larger then the VP ones.

5 Conclusion

From the results it can be deduced that the STAPP algorithms VP, MILP1 and MILP2 are outperformed by the CTAPP algorithms MA, GTS and SGTS. The best algorithm was the granular tabu search, also when considering computational time. The market based algorithm (MA) is robust and easy to implement in a decentralized system, if it is possible to run the algorithms centralized, GTS is preferably used. On most problems, the computational times are low enough for the best algorithms to run on-line to respond to changes in the environmental information.

References

1. Alighanbari, M., Bertuccelli, L.F., How, J.P.: Filter-Embedded UAV Task Assignment Algorithms for Dyanamic Environments. In: AIAA Guidance, Navigation, and Control Conference and Exhibit, pp. 1–15 (2004)
2. Beard, R.W., McLain, T.W., Goodrich, M.A., Anderson, E.P.: Coordinated target assignment and intercept for unmanned air vehicles. IEEE Transactions on Robotics and Automation 18(6), 911–922 (2002)
3. Bellmore, M., Nemhauser, G.L.: The traveling salesman problem: a survey. Operations Research 16(3), 538–558 (1968)
4. Bent, R., van Hentenryck, P.: A Two-Stage Hybrid Local Search for the Vehicle Routing Problem with Time Windows. Transportation Science 38(4), 515–530 (2004)
5. Berhault, M., Huang, H., Keskinocak, P., Koenig, S., Elmaghraby, W., Griffin, P., Kleywegt, A.: Robot exploration with combinatorial auctions. In: Proceedings of 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003), vol. 2 (2003)
6. Bertuccelli, L.F., Alighanbari, M., How, J.P.: Robust planning for coupled cooperative UAV missions. In: Proc. of the 43rd IEEE Conference on Decision and Control, 2004, CDC, vol. 3 (2004)
7. Dechter, R., Pearl, J.: Generalized best-first search strategies and the optimality of A*. Journal of the ACM (JACM) 32(3), 505–536 (1985)
8. Frazzoli, E., Bullo, F.: Decentralized algorithms for vehicle routing in a stochastic time-varying environment. In: Proc. of the 43rd IEEE Conference on Decision and Control, CDC (2004)
9. Gambardella, L.M., Taillard, É., Agazzi, G.: MACS-VRPTW: a multiple ant colony system for vehicle routing problems with time windows. Advanced Topics In Computer Science Series, pp. 63–76. McGraw-Hill, New York (1999)
10. Gendreau, M., Laporte, G., Potvin, J.Y.: Metaheuristics for the Capacitated VRP. In: The Vehicle Routing Problem, pp. 129–154 (2002)
11. Gerkey, B.P., Mataric, M.J.: A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems. The International Journal of Robotics Research 23(9), 939 (2004)
12. Guo, Y., Parker, L.E., Madhavan, R.: Towards collaborative robots for infrastructure security applications. In: Proceedings of International Symposium on Collaborative Technologies and Systems, pp. 235–240 (2004)

13. Jones, C., Shell, D., Mataric, M.J., Gerkey, B.: Principled Approaches to the Design of Multi-Robot Systems. In: Proc. of the Workshop on Networked Robotics, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004) (2004)
14. Kim, Y., Gu, D.W., Postlethwaite, I.: Real-time optimal mission scheduling and flight path selection. IEEE Transactions on Automatic Control (submitted) (accepted January 12, 2007)
15. Kim, Y., Gu, D.W., Postlethwaite, I.: A comprehensive study on flight path selection algorithms. In: The IEE Seminar on Target Tracking: Algorithms and Applications, pp. 77–90 (2006)
16. Lagoudakis, M.G., Markakis, E., Kempe, D., Keskinocak, P., Kleywegt, A., Koenig, S., Tovey, C., Meyerson, A., Jain, S.: Auction-based multi-robot routing. Robotics: Science and Systems (2005)
17. Laporte, G., et al.: Classical and Modern Heuristics for the Vehicle Routing Problem. Blackwell Synergy, Malden (1999)
18. Le Bouthillier, A., Crainic, T.G.: A Cooperative Parllel Meta-Heuristic for the Vehicle Routing Problem with Time Windows. Computers & Operations Research 32(7), 1685–1708 (2005)
19. Lindhe, M., Ögren, P., Johansson, K.H.: Flocking with Obstacle Avoidance: A New Distributed Coordination Algorithm Based on Voronoi Partitions. In: Proceedings of the 2005 IEEE International Conference on Robotics and Automation, 2005, pp. 1785–1790 (2005)
20. Overmars, M.H., Welzl, E.: New methods for computing visibility graphs. In: Proceedings of the fourth annual symposium on Computational geometry, pp. 164–171 (1988)
21. Panton, D.M., Elbers, A.W.: Mission planning for synthetic aperture radar surveillance. Interfaces 29(2), 73–88 (1999)
22. Pikovsky, A., Shabalin, P., Bichler, M.: Iterative Combinatorial Auctions with Linear Prices: Results of Numerical Experiments. In: E-Commerce Technology, 2006. The 8th IEEE International Conference on and Enterprise Computing, The 3rd IEEE International Conference on E-Commerce, and E-Services, p. 39 (2006)
23. Renaud, J., Boctor, F.F.: A sweep-based algorithm for the fleet size and mix vehicle routing problem. European Journal of Operational Research 140(3), 618–628 (2002)
24. Thunberg, J.: Task assignment and path planning for systems of surveillance unmanned ground vehicles. Master's thesis, The Royal Institute of Technology, KTH (2007)
25. Toth, P., Vigo, D.: The Granular Tabu Search and Its Application to the Vehicle-Routing Problem. INFORMS Journal on Computing 15(4), 333–346 (2003)
26. Tovey, C., Lagoudakis, M.G., Jain, S., Koenig, S.: The generation of bidding rules for auction-based robot coordination. In: Proceedings of the 3rd International Multi-Robot Systems Workshop, pp. 14–16 (2005)
27. Zlot, R., Stentz, A.: Market-based Multirobot Coordination for Complex Tasks. The International Journal of Robotics Research 25(1), 73 (2006)

Optimal Control of the Weapon Operating Characteristic with Control Inequality Constraints[⋆]

John Bode II, David Jacques, and Meir Pachter

Air Force Institute of Technology
2950 Hobson Way, Bldg 640
Wright-Patterson Air Force Base, OH, 45433, USA
{John.Bode,David.Jacques,Meir.Pachter}@Afit.edu

Abstract. Optimal employment of autonomous wide area search munitions or unmanned aerial vehicles is considered. The air vehicle searches a battle space for stationary targets in the presence of false targets and/or clutter. A control problem is formulated to achieve optimal scheduling of the control variables P_{TR}, the probability of target report, which is a reflection of the sensor's threshold setting, and Q, the area coverage rate. These optimal control problem formulations yield solutions that maximize the probability of target attack while at the same time also satisfying hard bounds on the probability of false target attacks. Analysis has shown that the optimal solutions are sensitive to the particular relationship between the sensor's receiver operating characteristic parameter, c, and area coverage rate, Q. In this chapter, several different heuristics for this particular relationship are studied and the optimal dynamic schedules of the sensor threshold and area coverage rate are obtained.

1 Introduction

The use of Unmanned Aerial Vehicles (UAVs) in both surveillance and attack missions is ever increasing. Much of this can be attributed to the promises of reduced materiel costs and the growing uneasiness of Americans to accept casualties in military operations. UAVs have successfully been employed in reconnaissance; and recently as attack platforms in the Middle-East and Afghanistan. A recent article from Balad Air Base, Iraq, demonstrates this point. "The airplane is the size of a jet fighter, powered by a turboprop engine, able to fly at 300 mph and reach 50,000 feet. It's outfitted with infrared, laser and radar targeting, and with a ton and a half of guided bombs and missiles. The Reaper is loaded, but there's no one on board.... The arrival of these outsized U.S. "hunter-killer" drones, in aviation history's first robot attack squadron, will be a watershed moment even in an Iraq that has seen too many innovative ways to hunt and kill," says the regional U.S. air commander [1]. Plans are being developed for an ever expanding role of these unmanned aircraft in current and future conflicts.

[⋆] The views expressed in this chapter are those of the authors and do not reflect the official policy or position of the United States Air Force, the Department of Defense, or the U.S. Government. This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

The search and destroy mission involves searching, detecting, classifying, and attacking targets. Different types of UAVs using Autonomous Target Recognition (ATR) may be used. A sensor craft could autonomously locate and classify objects, then pass the target information to a separate attack aircraft that is the shooter. In a similar way, an Unmanned Combat Aerial Vehicle (UCAV) could autonomously locate, classify, and destroy targets. An expendable autonomous Wide Area Search Munition (WASM) locates, classifies, and destroys a target as well as itself in the process.

In this chapter, UAVs with different quantities of warheads are considered. The infinite-warhead case is used to mathematically model the zero-warhead sensor craft case since it is not limited in the number of target attacks (although it is fuel/time limited). The sensor crafts mission does not terminate because it runs out of warheads, but it continues to search and classify until the search area is covered. Therefore, a sensor craft is mathematically equivalent to a vehicle having an infinite number of warheads.

For the scenarios considered herein, any objects reported as targets (correctly or incorrectly) are attacked. We are obviously interested in increasing the probability of true target attacks. This is achieved by lowering the sensor's threshold, namely, increasing the conditional probability of target report given encounter, P_{TR}. Unfortunately, the conditional probability of false target report given encounter, P_{FTR}, then decreases, increasing the false alarm rate, $1 - P_{FTR}$. This is disastrous, in particular when the ratio of targets to false targets is low.

We are interested in the optimal employment of UAVs. On the one hand, we are trying to correctly identify and attack as many true targets as possible. This is constrained by attempting to keep at the same time the number of misidentified targets attacked below a prescribed maximum. In this chapter, we expand on the analytic derivation of the previously established Weapon Operating Characteristic (WOC) [2] by including control inequality constraints and using the analytic expression for the Receiver Operating Characteristic (ROC) parameter, c.

The number of false target attacks during a mission is denoted m, and the number of target attacks is denoted n. System effectiveness can be quantified using the probability of at least $\bar{m}$ false target attacks, denoted $P(m \geq \bar{m})$, and the probability of at least $\underline{n}$ true target attacks, denoted $P(n \geq \underline{n})$. The values for $\bar{m}$ and $\underline{n}$ are specified by the mission planner. From this, we can formulate the general optimal control problem

$$\begin{aligned} \text{Max: } & P(n \geq \underline{n}) \\ \text{s.t.: } & P(m \geq \bar{m}) \leq b \\ \text{m,n: } & \in \mathbb{Z}^{+} \end{aligned}$$

where the upper bound b is set by the mission planner. Note that m and n could take on the value of zero but that would result in a trivial solution resulting from fact that the conditional probability would be exactly equal to 1.

2 Search and Destroy Mission

As previously discussed, the search and destroy mission involves search, detection, classification, and attack. A"linear" search pattern is considered, as depicted in Figure 1.

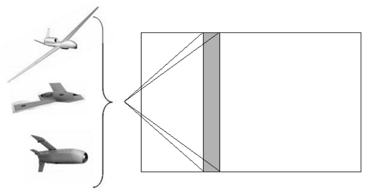


Fig. 1. Linear Battle Space Search Pattern

Figure 1 represents the three different types of UAVs; a sensor craft, a UCAV, or a WASM searching the battle space in a swath of area at a time. Figure 1 shows a constant swath width, however the pattern could be dynamically set during the course of a mission. Further, any irregular-shaped battle space can be covered by a union of rectangular regions, closely approximating the shape of the region.

Area coverage rate can be varied in a number of ways. For constant-altitude flight, area coverage rate is the product of velocity and swath width. Thus, changing either parameter changes the area coverage rate. Swath width is constrained by the sensor gimbal limits and/or the sensor field of view. For a fixed sensor field of view and a fixed velocity, increasing altitude increases the area coverage rate while at the same time reducing the number of pixels on the target. In the present formulation, the flight condition in terms of altitude, velocity, and swath width will all be rolled up into area coverage rate, Q, which will be one of the two control variables. The second control variable is P_{TR}, which is directly determined by the sensor threshold setting.

Six different scenarios were considered by Jacques and Pachter [3]. These scenarios allow for different search patterns, true and false target distributions, and also a finite number of targets uniformly distributed. This chapter will consider the scenario where a single target is uniformly distributed in a space among a Poisson field of false targets (Scenario 1 in [3]).

The sensor reports a detected object as either a target, thereby authorizing an attack, or a false target, thereby commanding no attack. Sensor performance is judged by how often the sensor is correct. The probability of a target report is the conditional probability that the system correctly reports a *target* when a *target* is encountered. Selecting a sensor threshold is tantamount to selecting the probability of target report, P_{TR}. The sensor can also be wrong and a target can go undetected with probability $1-P_{TR}$. Similarly, the probability of false target report, P_{FTR}, is the conditional probability that a sensor reports a *false target* when a *false target* is encountered. A false target is any entity with characteristics similar to that of a real target as seen by the sensor, e.g., clutter or a decoy. By using the Receiver Operating Characteristic, ROC, to model sensor performance, the conditional probability $1-P_{FTR}$ of attacking a false target can be expressed in terms of the conditional probability of a target report. Therefore, the second control variable will be the conditional probability of a target report, P_{TR}.

The operating point on a ROC curve specifies P_{TR} and P_{FTR}. Together, P_{TR} and P_{FTR} determine the entries of the “confusion matrix” shown in Table 1 which can be used to determine the outcome of a random draw each time an object is encountered. It is “binary” in the fact that only one of two decisions can be arrived at for any encounter, an object is either declared a target or declared a false target.

Table 1. Binary Confusion Matrix

	Encountered Object	
Declared Object	Target	False Target
Target	P_{TR}	$1-P_{FTR}$
False Target	$1-P_{TR}$	P_{FTR}

P_{FTR} is a dependent variable, related to P_{TR} via the ROC. A mathematical representation of the ROC curve produces a graph of true positive fraction P_{TR} versus false positive fraction $1-P_{FTR}$ that starts at $(0,0)$, and monotonically increases to $(1,1)$. A ROC model adapted from medical research [4] is used in this chapter

$$(1-P_{FTR}) = \frac{P_{TR}}{(1-c)P_{TR}+c}, \tag{1}$$

where the parameter $c \in [1,\infty)$, which represents the signal to noise ratio, is an indication of the quality of the sensor. It will also depend on the vehicle speed, which inversely affects processing time, and engagement geometry, which includes flight altitude and look angle. The ROC is a family of curves parameterized by c as shown in Figure 2. As c increases, the ROC improves. As $c \to \infty$, the area under the curve approaches unity indicating perfect classification.

To calculate the probability of at least a certain number of target attacks, one needs to know the probability of an exact number of attacks. Decker [5] showed that one must distinguish between the case where all warheads are used and the case where warheads are left over. Both cases need to be considered to correctly calculate the overall

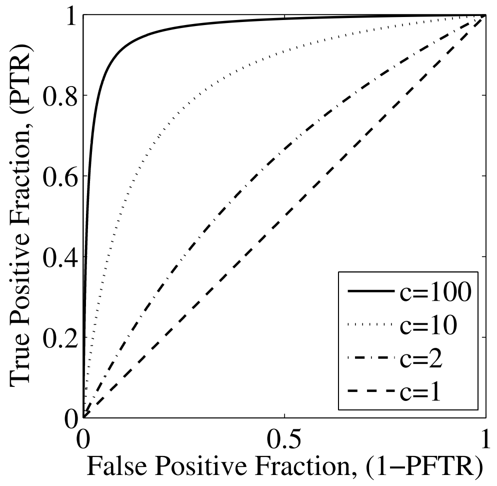


Fig. 2. Family of ROC Curves

probability. For the fixed warhead case, what can happen in the future is dependent on what happened in the past. Calculating the probability of at least a certain number of attacks involves summing all the possible mutually exclusive probabilities. Kish et al. [2] showed that this can be done for all the scenarios, and greatly simplifies for Scenario 1. Scenario 1, recall, entails one target uniformly distributed in a Poisson field of false targets.

3 Inequality Constraint

The analysis from Kish et al. [2] shows the control can and often does saturate by reaching its upper limit. The control variable $u = P_{TR}$ is a probability; therefore, a hard control inequality constraint, $P_{TR} = u \leq 1$, must be enforced. If the control saturates at 1, a discontinuity in the derivative of the control signal, $P_{TR}(t)$, will occur at an interior point known as a corner point [6].

The notation of Hull [6] will be used throughout this chapter. No constraint is placed on the location of the corner point for this example (natural, or free corner conditions). The corner conditions discussed by Hull require that the Hamiltonian, H, and Lagrange multiplier, λ, must be continuous across the corner. The corner time at which the control saturation occurs, t_c, can be calculated based on the relationships of the on-boundary and off-boundary sub-arcs. Denoting the Hamiltonian where $t_c \leq t \leq T$ as H_{c+}, the Hamiltonian where $0 \leq t \leq t_c$ as H_{c-}, and applying the natural corner condition $H_{c+} = H_{c-}$, we look for solutions for both the off-boundary and on-boundary portions of the problem. The solution is composed of an off-boundary sub-arc followed by an on-boundary sub-arc. The on-boundary arc is used in the calculations of t_c.

3.1 Modeling

The first scenario examined will assume fixed endurance, T, fixed area coverage rate, Q, variable P_{TR} and a Poisson field of false targets with spatial arrival rate α. The UAV flies along a straight path with area coverage rate Q. Therefore, the only control variable is P_{TR}. Constant Q thus implies the ROC parameter c is constant. We wish to maximize the probability of target attack, P_{TA}. Thus our problem is

$$\max_{P_{TR}} \quad P_{TA}, \tag{2}$$

where the expression for P_{TA} in terms of P_{TR} will be developed subsequently.

We consider a rectangular battle space of area A_t, which is searched up to time t. Thus,

$$A_t = wvt, \tag{3}$$

where A_t is the area searched, w is the width of battle space, v is the velocity of the UAV and t is the search time. The total battlespace area, A_s, is the area searched during $0 \leq t \leq T$, where T is the search duration.

$$A_T = wvT. \tag{4}$$

The Poisson random variable has a sample space, S, of all integers greater than or equal to zero, and the probability of exactly k encounters is given by the Poisson probability distribution function:

$$P(k) = e^{-\lambda}\frac{\lambda^k}{k!}, \quad k = 0,1,2\ldots \text{ and } \lambda > 0. \tag{5}$$

We need to express the non-dimensional Poisson distribution parameter, λ, in terms of the number of false target encounters per unit area. Using α for the spatial arrival rate of false targets in the battle space,

$$\lambda = \alpha A_T. \tag{6}$$

The false target arrival rate, α will be assumed known ahead of time from battlefield intelligence or the enemy's order of battle. At any time t, the expected number of false target encounters will be

$$\lambda_t = \frac{t}{T}\lambda. \tag{7}$$

In order to be able to recognize and engage the target at time t, the munition can not have previously engaged a false target. The probability of not having encountered a false target is

$$P_{(k=0)} = e^{-(1-P_{FTR})\lambda\frac{t}{T}}. \tag{8}$$

It is understood that for the remainder of the derivation, P_{TR}, P_{FTR}, u and x will all be functions of t. To simplify the notation, the explicit relationship will be omitted.

The time of true target attack, t, is a random variable and $f(t)$ is its probability density function. Using the expression for $P_{(k=0)}$, the target attack probability density function [7] is

$$f(t) = \frac{1}{T}P_{TR}e^{-(1-P_{FTR})\lambda\frac{t}{T}}. \tag{9}$$

An expression for the false target attack probability density function is similarly obtained and given by

$$g(t) = \left[(1 - P_{TR}\frac{t}{T})\right]\left[e^{-(1-P_{FTR})\lambda\frac{t}{T}}\right]\left[\frac{1}{T}\lambda(1 - P_{FTR})\right]. \tag{10}$$

From these two fundamental probability density functions, $f(t)$ and $g(t)$, the probabilities of mission success and of no engagement can be derived as shown by Jacques and Pachter [3]. Using these expressions, we can now start to complete the unconstrained optimal control problem statement, leading to an objective function

$$J \equiv P_{TA} = \int_0^T f(t)dt. \tag{11}$$

We now express $1 - P_{FTR}$ in terms of the control variable P_{TR} while using the ROC relationship. The optimization problem may now be stated as

$$\max_{P_{TR}} \; P_{TA} = \max_{P_{TR}} \int_0^T \frac{1}{T}P_{TR}e^{-\int_0^t \lambda\left(\frac{P_{TR}}{c-(c-1)P_{TR}}\right)\frac{d\tau}{T}}dt. \tag{12}$$

Next, we normalize by setting $T = 1$ and we obtain

$$\max_{P_{TR}} = \int_0^1 P_{TR} e^{-\int_0^t \lambda (\frac{P_{TR}}{c-(c-1)P_{TR}}) d\tau} dt. \tag{13}$$

Our control variable is P_{TR}, the state variable, x, is introduced and the nonlinear dynamics are

$$\dot{x} = \frac{P_{TR}}{c-(c-1)P_{TR}}. \tag{14}$$

Our optimal control problem is now restated as

$$J^* = \max_u \quad P_{TA} = \int_0^1 u e^{-\lambda x} dt, \tag{15}$$

subject to the dynamics

$$\dot{x} = \frac{u}{c-(c-1)u}. \tag{16}$$

A control inequality constraint is enforced because the control is a probability. However, a slack variable, ζ, is used to obtain an equality constraint, i.e.,

$$u \le 1 \Rightarrow u - 1 + \zeta^2 = 0, \tag{17}$$

along with the initial and terminal conditions:

$$t_0 = 0,\ x_0 = 0,\ t_f = 1,\ x_f = x(f). \tag{18}$$

The augmented Hamiltonian, $\hat{H}$, is given by

$$\hat{H} = u e^{-\lambda x} + \lambda_x \left(\frac{u}{c-(c-1)u}\right) + \lambda_u (u - 1 + \zeta^2). \tag{19}$$

We will first consider the off-boundary case where $\lambda_u = 0$ and $0 \le t \le t_c$ where t_c is the corner point at which saturation sets in. Because the control is monotonically increasing and will saturate at the upper limit, it cannot decrease and therefore it will only have one corner. Applying the steps outlined by Hull [6], we are able to use the first-order differential conditions to solve the following equations using the off-boundary segment

$$\begin{aligned}
\dot{\lambda}_x &= -\hat{H}_x = \lambda u e^{-\lambda x}, \\
\lambda_x &= \int_{t_0}^{t_f} \dot{\lambda}_x dt = \int_{t_0}^{t_f} \lambda u e^{-\lambda x} dt, \\
x &= \int_{t_0}^{t_f} \dot{x} dt = \int_{t_0}^{t_f} \frac{u}{c-(c-1)u} dt, \\
\zeta^2 &= 1 - u, \\
\hat{H}_u &= e^{-\lambda x} + \lambda_x \frac{c}{[c-(c-1)u]^2} + \lambda_u = e^{-\lambda x} + \lambda_x \frac{c}{[c-(c-1)u]^2} = 0.
\end{aligned}$$

Solving for the optimal control variable, we obtain

$$u^*(t) = \frac{-(\sqrt{-\lambda_x(t)c})e^{\frac{1}{2}\lambda x} + c}{c-1}. \tag{20}$$

We know from the initial conditions that $x(0) = 0$, so

$$u(0) = \frac{-(\sqrt{-\lambda_x(0)c}) + c}{c-1}. \tag{21}$$

We are interested in solving for our control, u, as a function of time. To arrive at that, we need $\lambda_x(t)$ and $x(t)$ to solve for $u^*(t)$ in Equation 20. To ensure that the control, $u(t)$ remains positive, we must use the negative $\sqrt{-\lambda_x(t)c}$ root. We know that $\lambda_x(t) \leq 0$ because $\lambda_x(t)$ is monotonically increasing when $c \in [1,\infty)$ and that $\lambda_x(t_f) = 0$. However, we are unable to analytically determine values for $\lambda_x(t)$ and therefore we can not get a closed-form solution for the off-boundary case to solve for t_c.

For further insight, we investigate the on-boundary portion of the trajectory where $\zeta = 0$ and $t_c \leq t \leq T$. Applying the no slack condition, we see

$$\zeta = 0 \Rightarrow u = 1, \tag{22}$$

the control is saturated and we are able to use the first-order differential conditions to solve for the following equations using the on-boundary segment

$$\begin{aligned}
\dot{\lambda}_x &= -\hat{H}_x = \lambda u e^{-\lambda x} = \lambda e^{-\lambda x}, \\
\dot{x} &= \frac{u}{c-(c-1)u} = 1, \\
x(t) &= \int_{t_c}^{t} \dot{x} dt = \int_{t_c}^{t} d\tau = t - t_c \Rightarrow t_c = t - x(t).
\end{aligned}$$

We know from the boundary conditions that $x(1) = 1 - t_c$. Rearranging the previous equation yields $t_c = 1 - x(1)$; however, we don't know the value of $x(1)$. We do know that at the point of discontinuity, t_c is a constant so by setting both equations for t_c equal to each other and solving for $x(t)$ will yield

$$x(t) = t - 1 + x(1). \tag{23}$$

We can not use the continuity expression of $H_{C+} = H_{C-}$ to solve the problem because the form of u_{C-} would not allow for a closed-form solution. However, if we use the on-boundary side conditions to solve for t_c and make the appropriate substitution for $x(t)$ in the expression

$$\dot{\lambda}_x = -\hat{H}_x = \lambda_{FT} u e^{-\lambda_{FT} x} = \lambda_{FT}(1)e^{-\lambda_{FT} x} = \lambda_{FT} e^{-\lambda_{FT} x}, \tag{24}$$

we obtain

$$\dot{\lambda}_x = \lambda_{FT} u e^{-\lambda_{FT}(t+x(t)-1)}, \tag{25}$$

and then integrate using $t = t_c$, we may solve for t_c and obtain the following explicit expression

$$t_c = 1 - \frac{1}{\lambda} \ln\left(\frac{c}{c-1}\right). \tag{26}$$

This shows that the time of saturation, t_c, is only a function of the false target density parameter, λ, and the receiver operating characteristic parameter, c.

After solving the unconstrained case, we may solve the constrained case in a similar fashion. The basic problem may be set up as follows: The cost to be maximized is

$$J = \int_0^T ue^{-\lambda x} dt \tag{27}$$

subject to the dynamics

$$\begin{aligned} \dot{x} &= \frac{u}{c-(c-1)u},\ x(0) = 0,\ 0 \leq t \leq 1 \\ \dot{y} &= u,\ y(0) = 0,\ 0 \leq t \leq 1 \end{aligned}$$

and the additional constraint imposed by the probability of false target attack,

$$P_{FTA} = \int_0^1 \frac{u\lambda(1-y)}{c-(c-1)u} e^{-\lambda x} dt \leq b. \tag{28}$$

We form our augmented Hamiltonian as before including the additional constraint

$$\hat{H} = ue^{-\lambda x} + \lambda_c \frac{u\lambda(1-y)}{c-(c-1)u} e^{-\lambda x} + \lambda_x \left(\frac{u}{c-(c-1)u}\right) + \lambda_y u + \lambda_u (u - 1 + \zeta^2). \tag{29}$$

In this particular formulation the constraint is appended as an equality constraint in Equation 29. This will result in a solution that will force the resulting P_{FTA} to a specific value.

Let us begin by examining the off-boundary portion of the problem where $\lambda_u = 0$ and $0 \leq t \leq t_c$. Applying the first-order conditions yield

$$\dot{\lambda}_x = -\hat{H}_x = \lambda ue^{-\lambda x} + \lambda \lambda_c \frac{u\lambda(1-y)}{c-(c-1)u} e^{-\lambda x},\ \ \lambda_x(1) = 0 \tag{30}$$

$$\dot{\lambda}_y = -\hat{H}_y = \lambda_c \frac{u\lambda}{c-(c-1)u} e^{-\lambda x},\ \ \lambda_y(1) = 0 \tag{31}$$

$$\zeta^2 = 1 - u \tag{32}$$

$$\hat{H}_u = 0 \Rightarrow u = \frac{\sqrt{c}}{c-1} \left[\sqrt{c} - \sqrt{\frac{\lambda_c(1-y)\lambda_{FT} + \lambda_x e^{\lambda x}}{1 + \lambda_y e^{\lambda x}}}\right]. \tag{33}$$

As with the unconstrained case, we need expressions for $\lambda_c(t), \lambda_x(t), \lambda_y(t)$ in order to solve for $u(t)$. We are unable to solve for those analytically so a closed-form solution using the off boundary conditions is not possible.

The on-boundary case conditions where $\zeta = 0$ can be used as shown previously in the unconstrained case. The solution method will be the same and it yields a similar result for the corner point t_c.

4 Dynamic Area Coverage

We consider Scenario 1 [3], where the number of actual targets is one, with a Poisson distribution of false targets parameterized by the density, $\alpha(t)$. We assume that the UAV flies along a straight path with area coverage rate $Q(t)$. P_{TR} and Q are the control variables. For fixed duration missions, a lower coverage rate increases the sensor performance but decreases the total area searched.

For the munition problem (single warhead, $k = 1$), with dynamic area coverage rate Q, and dynamic P_{TR}, the states are

$$x(t) = \int_0^t \alpha Q(\tau)\left[1 - P_{FTR}(\tau)\right] d\tau \tag{34}$$

$$y(t) = \int_0^t \frac{1}{A_B} Q(\tau) P_{TR}(\tau) d\tau \tag{35}$$

$$z(t) = \int_0^t \alpha Q(\tau)\left[1 - P_{FTR}(\tau)\right]\left[1 - y(\tau)\right] e^{-x(\tau)} d\tau. \tag{36}$$

The corresponding integrand in the Lagrange optimal control problem , L, is

$$L = -\frac{1}{A_B} Q(t) P_{TR}(t) e^{-x(t)} \tag{37}$$

where A_B is the battle space area. Thus, the derivative state equations are

$$\dot{x}(t) = \alpha Q(t)\left[1 - P_{FTR}(t)\right], \tag{38}$$

$$\dot{y}(t) = \frac{1}{A_B} Q(t) P_{TR}(t), \tag{39}$$

$$\dot{z}(t) = \alpha Q(t)\left[1 - P_{FTR}(t)\right]\left[1 - y(t)\right] e^{-x(t)}. \tag{40}$$

Equation (1) becomes

$$\left[1 - P_{FTR}(t)\right] = \frac{Q(t) P_{TR}(t)}{Q(t) P_{TR}(t) + Q_n\left[1 - P_{TR}(t)\right]}, \tag{41}$$

and the augmented Hamiltonian is now

$$\hat{H} = \left(\lambda_y - e^{-x}\right)\frac{1}{A_B} Q P_{TR} + \alpha \frac{Q^2 P_{TR}}{Q P_{TR} + Q_n(1 - P_{TR})}\left[\lambda_x + \lambda_z(1 - y) e^{-x}\right]. \tag{42}$$

Taking the partial derivative of $\hat{H}$ with respect to the control variable P_{TR} results in

$$\frac{\partial \hat{H}}{\partial P_{TR}} = Q\left\{\left(\lambda_y - e^{-x}\right)\frac{1}{A_B} + \frac{\left[\lambda_x + \lambda_z(1 - y) e^{-x}\right] Q_n \alpha Q}{\left[Q P_{TR} + Q_n(1 - P_{TR})\right]^2}\right\}. \tag{43}$$

Taking the partial derivative of $\hat{H}$ with respect to the decision variable Q results in

$$\frac{\partial \hat{H}}{\partial Q} = P_{TR}\left\{\left(\lambda_y - e^{-x}\right)\frac{1}{A_B} + \alpha\left[\lambda_x + \lambda_z(1 - y) e^{-x}\right]\frac{Q^2 P_{TR} + 2 Q_n Q(1 - P_{TR})}{\left[Q P_{TR} + Q_n(1 - P_{TR})\right]^2}\right\}. \tag{44}$$

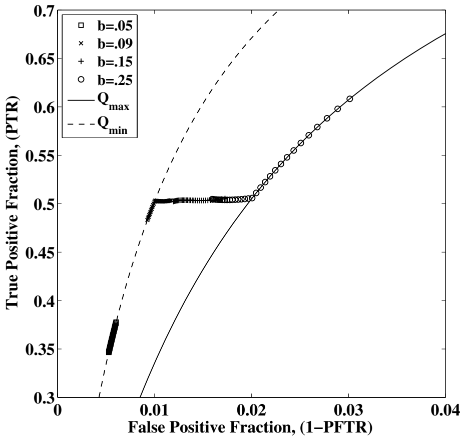


Fig. 3. ROC plot. Scenario 1 with dynamic-Q, dynamic-P_{TR}, $k = 1$, $\alpha = 2$ [1/km^2], $Q_n = 1000$, $Q_{min} = 10$, $Q_{max} = 20$, and $T = 0.5$ [hr].

This produces the necessary condition

$$\left[\lambda_x + \lambda_z (1-y) e^{-x}\right] \left[Q P_{TR} - 2 Q_n P_{TR} + Q_n\right] = 0. \tag{45}$$

Thus, either

$$P^*_{TR}(t) = \frac{Q_n}{2Q_n - Q^*(t)}, \tag{46}$$

where Q^* is the optimal area coverage rate, or

$$\lambda_x(t) = -\lambda_z [1 - y(t)] e^{-x(t)}. \tag{47}$$

This leads to the interesting result

$$P^*_{TR}(t) > 0.5, \tag{48}$$

for $Q_n >> Q^*$ (which is the case when $c = 100$), or better stated as $P^*_{TR} \approx 0.5$ for all time. In addition, the relation between P^*_{TR} and Q^* does not depend on α or the value of b.

The expression for c, which determines how well a munition can discriminate between true and false targets at a given sensor threshold setting, is dependent on several factors. If area coverage rate Q is allowed to vary, the operating point continually moves from one ROC curve to another. Intuition dictates that better performance is expected at slower area coverage rates since longer dwell times allow for better sensing and data processing. When the UAV travels at a higher speed and the coverage rate increases, worse performance can be expected. It was also assumed that experimentation would yield a scaled nominal area coverage rate, Q_n. To model these characteristics, Kish et al. [2] used the expression for the receiver operating characteristic parameter

$$c = \frac{Q_n}{Q}. \tag{49}$$

There is still the requirement of $c > 1$, thus $Q < Q_n$. The goal is to find the optimal combination of P_{TR} and Q either constant, or as a function of elapsed time or area covered. This optimal combination of P_{TR} and Q establishes the optimal Weapon Operating Characteristic (WOC). Figure 3 from [2] demonstrates the effect of varying the false target attack constraint on the optimal values of Q and P_{TR}. At low values of b, P_{TR} increases while the value of Q remains at its minimum limit. As the parameter b is allowed to increase, P_{TR} levels at approximately at 0.5 while Q increases, producing a corner. As b is allowed to further increase, Q increases until it reaches its maximum and P_{TR} increases causing a second corner in the two input control problem. As expected, the false positive fraction increases throughout the process.

5 The ROC Parameter c

One of the results of Kish et al. [2] was that for several scenarios, the value of P^*_{TR}, was nearly equal to 0.5 when $Q_n >> Q^*$ (which is the case when $c = 100$). This would equate to the probability of correctly identifying a true target for only about half the encounters. This did not seem like a practical solution or match with any known actual sensor data. We have shown through the analytical approach with inequality constraints that the original methodology was correct as we were able to reproduce the results arrived at with similar mathematical techniques. Thus, the discrepancy must lie elsewhere and the obvious choice was to revisit the original model for the Receiver Operating Characteristic (ROC) parameter c.

The parameter c is monotonically decreasing in Q in the region of interest, where the value of c approximately equal to 100 was chosen with values of Q in the range of 5 to 25 when a Q_n of 1000 was selected for scaling purposes. This is shown in Figure 4. Different shaped curves that met the appropriate end point condition were tested and the most promising were that of the linear and quadratic curve fits. Figure 5 shows the linear and quadratic curve fits along with the Kish et al. [2] expression for c. A family of curves may be generated that meet the criteria for the values of c in the range of interest. These may be varied as desired by using generic expressions for the curves of

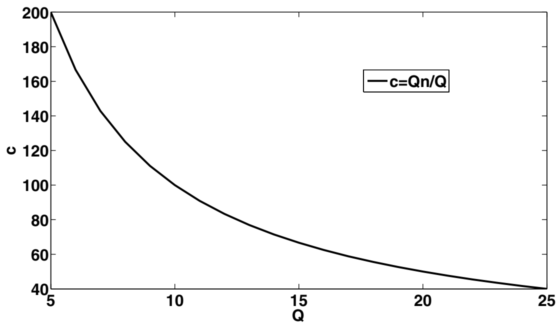


Fig. 4. c vs Q, Q_n=1000

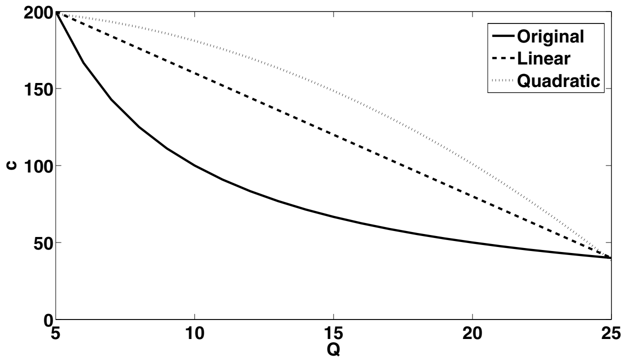


Fig. 5. Family of c vs Q curves, Q_n=1000

Table 2. Generic Expressions for c and P^*_{TR}

Class	c	P^*_{TR}
Original	$\frac{Q_n}{Q}$	$\frac{Q_n}{2Q_n-Q}$
Linear	$\frac{Q_n}{a}-bQ-d$	$\frac{abQ}{-a+ad+Q_n}$
Quadratic	$\frac{Q_n}{a}-AQ^2+BQ+D$	$\frac{-abQ+2aAQ^2}{-a+aD+aQ^2+Q_n}$

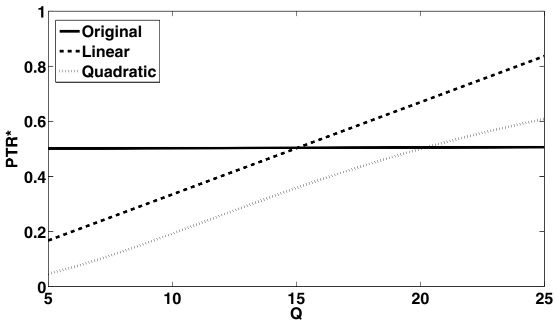


Fig. 6. $P_{TR}*$ vs. Q, Q_n=1000

the form shown in Table 2. Expressions for P^*_{TR} may be similarly by obtained and are also shown in Table 2.

Using the same values for the coefficients that generated the curves in Figure 5, we are able to now see the effect on P^*_{TR} of differing values of the expression c in Figure 6. The linear and quadratic cases show that P^*_{TR} values in excess of 0.5 Kish et al. [2] may be obtained and compare favorably to real-world values of P^*_{TR}. The key is that the solutions obtained are sensitive to the chosen model for the parameter c which is dependent on Q. The original relationship of c and Q was based upon intuition. The research presented herein shows that there is a relationship between the values of

P^*_{TR} and the expression for c. Note that the determination of which is the appropriate formulation is left for further research.

6 Conclusions

The original formulation of the WOC optimization problem lacked control inequality constraints to analyze the existence and location of corner points due to the saturation of the control variables. In this chapter, using the methods of optimal control theory, the original solution methodology is proven correct using the control inequality constraint formulation and may be applied to all the scenarios, for both the scalar and multi-input optimal control problems. The original results were strongly dependent on the heuristic for the Receiver Operating Characteristic (ROC) parameter, c. Our work shows that different heuristics for that parameter yield results more closely aligned with real-world values for the optimal control schedules. We believe that the quadratic expression relating the signal to noise ratio parameter c to the area coverage rate Q may be closer to the actual characteristics of ATR modules because the value of c doesn't change much near the bottom of the region of acceptable Q values and then drops off sharply as they increase. Additional expressions for the signal-to-noise ratio parameter c, based on actual ATR module characteristics for the active and passive radar cases are needed to obtain more desirable results.

References

1. Hanley, C.J.: US prepares to strike with unmanned ‘Reapers’. Dayton Daily News (July 22, 2007)
2. Kish, B.A., Jacques, D.R., Pachter, M.: Optimal Control of Sensor Threshold for Autonomous Wide-Area-Search Munitions. Journal of Guidance, Control and Dynamics 30, 1239–1248 (2007)
3. Jacques, D.R., Pachter, M.: A Theoretical Foundation for Cooperative Search, Classification and Target Attack. In: Cooperative Control: Models, Applications and Algorithms, vol. 2. Kluwer, Dordrecht (2003)
4. Moses, L.E., Shapiro, D.E., Littenberg, B.: Combining Independent Studies of a Diagnostic Test into a Summary ROC Curve: Data-Analytic Approaches and Some Additional Considerations. In: Statisics in Medicine. Wiley, New York (1993)
5. Decker, D., Jacques, D.R., Pachter, M.: A Theory of Wide Area Search Engagement. Military Operations Research 12, 37–57 (2007)
6. Hull, D.: Optimal Control Theory for Applications. Springer, New York (2003)
7. Rosario, R.A.: Optimal Sensor Theshold Control and the Weapon Operation Characteristic for Autonomous Search and Attack Munitions. Thesis, Air Force Institute of Technology (2007)

Autonomous Target Detection and Localization Using Cooperative Unmanned Aerial Vehicles

Youngrock Yoon, Scott Gruber, Lucas Krakow, and Daniel Pack

Unmanned Systems Laboratory
Department of Electrical and Computer Engineering
US Air Force Academy
US Air Force Academy, CO 80840, USA
{youngrock.yoon,scott.gruber,lucas.krakow,daniel.pack}@usafa.edu

Abstract. In this chapter we present sensor implementation issues encountered in developing a team of cooperative unmanned aerial vehicles (UAVs) for intelligence, surveillance and reconnaissance missions. UAVs that compose the cooperative team are equipped with heterogeneous sensors and onboard processing capability. The sensor information collected by each UAV is constantly shared among the neighboring UAVs and processed locally using Out-of-Order Sigma-Point Kalman Filtering (O^3SPKF) techniques. Results from flight experiments support the effectiveness of the cooperative autonomous UAV technologies.

1 Introduction

Unmanned mobile agents used to detect and localize ground targets have numerous applications ranging from intelligence collection, surveillance and reconnaissance (ISR) in the military domain to search-and-rescue missions for civilian purposes. We are particularly interested in developing a network of inexpensive off-the-shelf mobile sensors that collectively deliver superior sensing performance compared with a small number of expensive mobile sensors [1]. In a network of cooperative sensors, optimally combining sensor information collected by the sensor nodes is critical. Other challenges may also exist: 1) covert/passive sensing; 2) unknown target dynamics; 3) episodically mobile targets; 4) intermittently occluded targets; and 5) out-of-order sensor measurements. In [2,3], we reported a sensor fusion system that offers a novel theoretical solution for addressing these issues using a Sigma-Point Kalman Filter (SPKF). In this chapter, we describe our on-going efforts to develop a mobile sensor network, focusing on hardware implementation details needed to deliver the system.

We are developing a team of unmanned aerial vehicles (UAV) that are equipped with heterogeneous sensors and onboard processing capabilities. The sensors include visual range (VR), infra-red (IR), and radio-frequency (RF) sensors. All sensor data are processed in real-time using an onboard processor before the data is sent to a ground station or other UAVs. The onboard processing capabilities enable our system to send and receive processed sensor information which is much smaller in size compared to raw data (such as images) used on many other platforms, thus reducing the communication bandwidth required. Typically, each UAV carries a camera (either VR or IR) sensor and a RF sensor. In Section 2, we present the details on how each sensor is used to detect a ground target.

When new processed sensor information is available from local sensors, each UAV broadcasts the sensor information. Sensor information from multiple UAVs are combined in a distributed manner; each UAV runs its own sensor fusion algorithm independently. In Section 3, we present the method used to represent the heterogeneous sensor information suitable for UAVs to share, followed by the implemented sensor fusion algorithm in Section 4. Experimental results are presented in Section 5 and we conclude the chapter with some summary remarks.

2 Target Detection with Visual-Range (VR) Sensors

Camera sensors (both VR and IR) identify the image location of a target when it enters into the camera field of view. When VR sensors are used, the color of the target (e.g., a red car) is used as a cue for detection. For detecting the pixels that belong to a VR image of the target, we use a one-class Gaussian classifier that takes the normalized red and green channel values of the pixels as an input. Normalized red-green-blue (RGB) values have only two-degrees of freedom. Therefore, we drop the normalized blue channel from the pixel classification. Color normalization is used to make the VR sensor less sensitive to the lighting condition changes. Let $\hat{p}_{i,j} = [\hat{r}_{i,j}, \hat{g}_{i,j}]^T$ be a vector of the normalized red and green values of a pixel $p_{i,j}$ in the input image, where (i, j) are the image coordinates. Assuming the expected color of the target has a Gaussian distribution with mean $\bar{p}$ and covariance Σ_p, we calculate the Mahalanobis distance [6], denoted $d_{i,j}$, between the observed pixel value $\hat{p}$ and the expected color value of the target as follows:

$$d_{i,j} = (\hat{p}_{i,j} - \bar{p})^T \Sigma_p^{-1} (\hat{p}_{i,j} - \bar{p}) \tag{1}$$

where $\bar{p}$ and Σ_p are estimated with a pre-sampled training set of the target images. If $d_{i,j}$ is lower than a threshold, then $p_{i,j}$ is classified to be a pixel of the target, otherwise as background. The threshold is chosen appropriately considering $d_{i,j}$ has a Chi-square distribution with two degrees of freedom. Noise in the pixel classification is suppressed by applying a morphological filter.

After each pixel in the image is classified, pixels are clustered to form a set of image blobs using a recursive connected component analysis algorithm. The algorithm scans the binary image which is the output of the pixel classification algorithm described above. When the system meets a foreground pixel, it creates a region that consists of the foreground pixel, and recursively grows the region by adding foreground pixels that are connected to the region. This process creates a set of foreground pixel blobs. The image blobs are further clustered based on a geometric distance measure on the image. For a pair of two regions, R_a and R_b, let c_a and c_b be the image coordinates of the center of mass of the two regions, respectively, and Σ_a and Σ_b be the covariances of the image coordinates of the

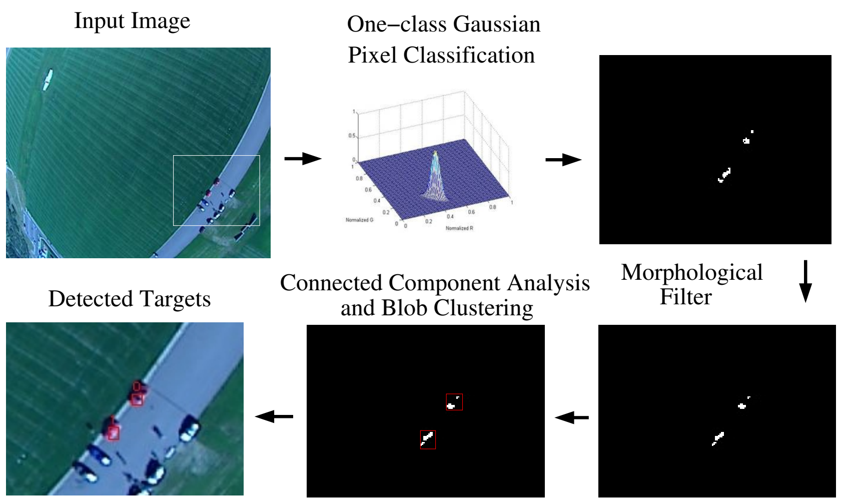

Input Image
One–class Gaussian Pixel Classification
Morphological Filter
Connected Component Analysis and Blob Clustering
Detected Targets
Normalized G
Normalized R

Fig. 1. VR sensor target detection algorithm

pixels in the two regions, respectively. Then, we can calculate a distance measure $\zeta_{a,b}$ between the two regions a and b as follows:

$$\zeta_{a,b} = (c_a - c_b)^T (\Sigma_a + \Sigma_b)^{-1} (c_a + c_b) \tag{2}$$

If $\zeta_{a,b}$ is smaller than a certain threshold, the two regions are merged. Such a merging process is repeated iteratively until no pair of regions can be merged. After the blob merging process, the foreground image will consist of image regions. Each image region denotes a detected target, and the center of the mass of the corresponding image region denotes the location of the target in the image space. As shown on Figure 1, the image location of the target is used to calculate the bearing angle to a target. In the figure, two cars are detected after the onboard processing of the captured image in the first frame.

3 Calculating Target Bearing Angles

In order to combine the sensor information, it must be transformed into a workspace common to all UAVs. To describe the required transformations, we employed three reference coordinate frames: 1) a global coordinate frame, denoted W; 2) a UAV coordinate frame, denoted U; 3) a sensor coordinate frame, denoted S. Figure 2 illustrates how these reference coordinate frames are arranged.

In our UAVs, the sensor system computes a unit direction vector , denoted as $\overrightarrow{v}_S$ in Figure 2, towards a target with respect to the sensor-specific coordinate frame S. With the given notion of reference frames, our goal is to transform $\overrightarrow{v}_S$ to a direction vector $\overrightarrow{v}_W$ which is defined with respect to the common

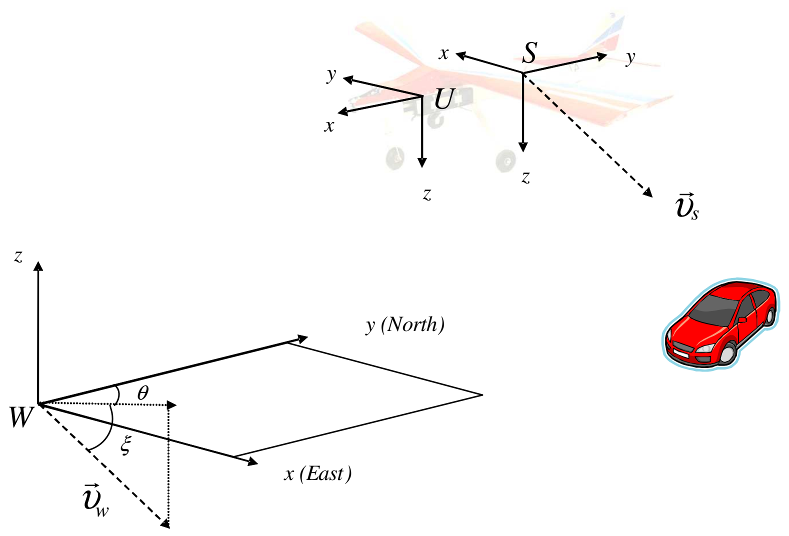


Fig. 2. Calculating target bearing angles

global reference coordinate frame W. Note that $\overrightarrow{v}_S$ and $\overrightarrow{v}_W$ both point to the target.

VR sensors must compute the direction vector using the image coordinates of the target. In the following subsection, we describe how to estimate a unit direction vector given the image coordinates of a target in a VR or IR image, followed by subsections that illustrate the necessary coordinate transformations to convert local sensor information to a global coordinate frame.

3.1 Calculating Target Bearing Vector Using Camera Sensors

For the task of target detection using a camera image, the calculation of $\overrightarrow{v}_S$ is often referred to as solving the perspective projection model [4,5] shown in Figure 3. Let (x_t, y_t, z_t) be the coordinates of the target location with respect to S, and (u_t, v_t) be the image coordinates of the target projected onto the image space. Then, these two coordinates are related by the following projection equation (called the perspective projection equation):

$$\begin{bmatrix} su_t \\ sv_t \\ s \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_t \\ y_t \\ z_t \end{bmatrix} \tag{3}$$

where (c_x, c_y) are the coordinates of the image center, and f_x and f_y are the focal lengths normalized by the pixel quantization ratio in both the x and y directions.

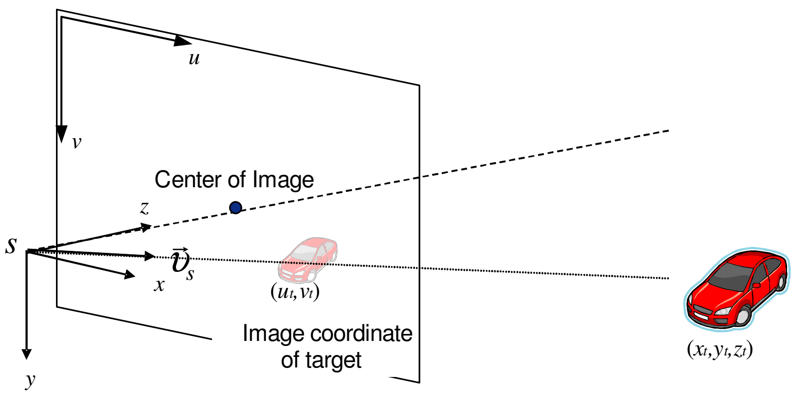


Fig. 3. Perspective projection model

These parameters are found by performing an intrinsic camera calibration [7][1]. The image coordinates (u_t, v_t) are therefore related to (x_t, y_t, z_t) as follows:

$$u_t = f_x \frac{x_t}{z_t} + c_x \tag{4}$$

$$v_t = f_y \frac{y_t}{z_t} + c_y \tag{5}$$

Unfortunately, this projection model only applies to an image without any optical distortion. Usually, camera sensors introduce radial and tangential distortion to the image. For example, our camera sensor uses a fish-eye lens, which introduces a significant amount of radial distortion as shown in Figure 4. In order to take the distortion into account in the projection model, Equations (4) and (5) can be modified as follows [7]:

$$x'_t = \frac{x_t}{z_t} \tag{6}$$

$$y'_t = \frac{y_t}{z_t} \tag{7}$$

$$x''_t = x'_t + (1 + k_1 r^2 + k_2 r^4) + 2p_1 x'_t y'_t + p_2(r^2 + 2x'^2_t) \tag{8}$$

$$y''_t = y'_t + (1 + k_1 r^2 + k_2 r^4) + 2p_2 x'_t y'_t + p_1(r^2 + 2y'^2_t) \tag{9}$$

where $r^2 = x'^2_t + y'^2_t$ and

$$u_t = f_x x''_t + c_x \tag{10}$$

$$v_t = f_y y''_t + c_y \tag{11}$$

[1] We use camera calibration software provided by the OpenCV library.

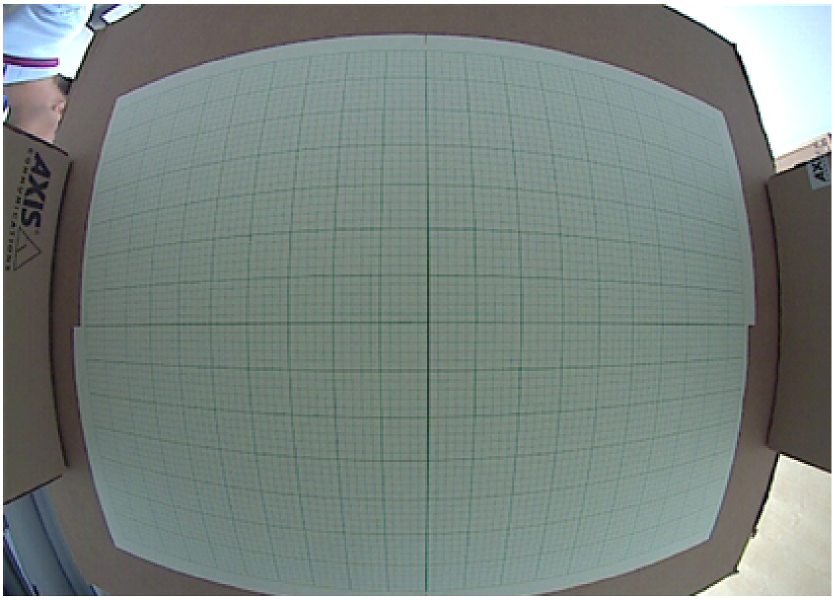


Fig. 4. Example of an image with radial distortion

Parameters k_1, k_2, p_1, and p_2 are the camera distortion parameters estimated using the camera calibration procedure. Since we are only interested in the target bearing angle with respect to S, we represent $\overrightarrow{v}_S$ as a three-dimensional unit direction vector calculated as follows:

$$\overrightarrow{v}_S = \frac{1}{\sqrt{x_t'^2 + y_t'^2 + 1}} [x_t', y_t', 1]^T \tag{12}$$

Using Equations (6) through (12), our objective of calculating $\overrightarrow{v}_S$ is met by calculating x_t' and y_t', given (u_t, v_t). To do so, the procedure described in Equations (6) through (11) must be reversed. Since this process is highly non-linear and the closed-form analytic solution to the inversion process is not always available, a numerical solution is applied as follows:

- Step 1: Calculate x_t'' and y_t'' as follows:

$$x_t'' = \frac{u_t - c_x}{f_x} \tag{13}$$

$$y_t'' = \frac{v_t - c_y}{f_y} \tag{14}$$

- Step 2: Initialize x_t' and y_t' with the value of x_t'' and y_t''

$$\begin{aligned} x_t' &\leftarrow x_t'' \\ y_t' &\leftarrow y_t'' \end{aligned} \tag{15}$$

– Step 3: Iteratively calculate the following equations until (x'_t, y'_t) converges:

$$r^2 = x'^2_t + y'^2_t \tag{16}$$

$$x'_t = \frac{x''_t - 2p_1 x'_t y'_t - p_2(r^2 + 2x'^2_t)}{1 + k_1 r^2 + k_2 r^4} \tag{17}$$

$$y'_t = \frac{y''_t - 2p_2 x'_t y'_t - p_1(r^2 + 2y'^2_t)}{1 + k_1 r^2 + k_2 r^4} \tag{18}$$

3.2 Transforming Local Sensor Information to the Global Coordinate Frame

The required transformation of a sensor-specific direction vector $\overrightarrow{v}_S$ to a global direction vector $\overrightarrow{v}_W$ is described using the following equation:

$$\overrightarrow{v}_W = {}^W R_U {}^U R_S \overrightarrow{v}_S \tag{19}$$

where ${}^W R_U$ is a 3×3 rotational matrix that denotes the relationship between coordinates U and W, and ${}^U R_S$ is a rotation transformation for coordinates in S to U.

4 Sensor Fusion Using Out-of-Order Sigma-Point Kalman Filter

As we alluded to in Section 1, the target bearing information in W is shared among neighboring UAVs to cooperatively locate a target using the Out-of-Order Sigma-Point Kalman Filter (O^{3}SPKF) [3].

In [2] we showed, using a number of simulation results, that SPKF outperforms conventional non-linear Kalman filters, such as the Extended Kalman Filter (EKF), in estimating the target location from highly non-linear sensor information. In this approach, a sequential stochastic inference model is used to estimate the state vector of a target location using sensor information. Unlike the conventional Extended Kalman Filter, which uses a Taylor series expansion to linearize the observation function, SPKF carefully chooses a set of points (*sigma points*) from *a priori* probabilistic distributions of the state vector, the process noise and the sensor noise. These points are then evaluated and the weighted mean and covariance of the output are used for the approximation of an *a posteriori* probabilistic distribution. For a more detailed description of our sensor fusion approach using SPKF, readers are referred to [2].

When a specific UAV sensor fusion system processes the sensor information shared by the neighboring UAVs, some sensor information may be out-of-order because of nondeterministic communication and/or processing latencies. The SPKF algorithm described in [2] is further extended to handle such out-of-order sensor measurements efficiently [3]. In this approach, when a UAV receives out-of-order sensor information, the system backtracks the state vector to the time when the out-of-order measurement was taken. Then, it uses the backtracked state vector to generate sigma points for evaluating the sensor output function. A detailed description of our approach to handle out-of-order sensor measurements is presented in [3].

5 Experimental Results

Although our sensor fusion algorithm is capable of combining information from heterogeneous sensors, our initial focus was to demonstrate the capability of our cooperative sensor fusion algorithm using VR sensors. We compare results of locating a ground target using one UAV and two UAVs. The UAVs were controlled by an autonomous decentralized control algorithm (presented in [10]) to cooperatively search, detect and localize a ground target. Figure 5 shows an example sensor image with a visible target. Such images are used to detect and locate the target using the camera sensor detection algorithm and the sensor fusion algorithm described earlier.

The final root-mean-square target location error for the one UAV experiment was 10.0 meters in the East direction and 24.3 meters in the North direction, while RMS error for the two UAV experiment was 2.7 meters in the East direction and 5.8 meters in the North direction. In both experiments, the target location estimates are updated for about 300 seconds. The results, in Figures 6 and 7,

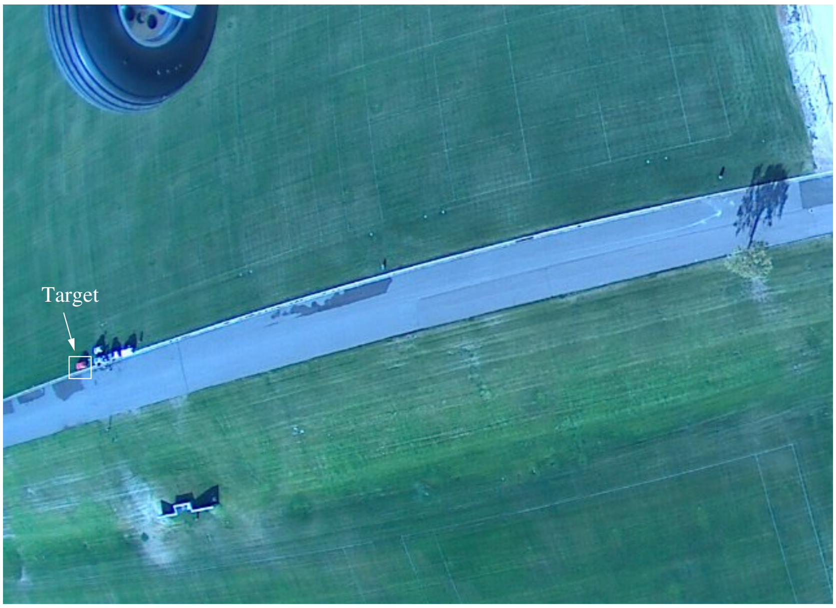
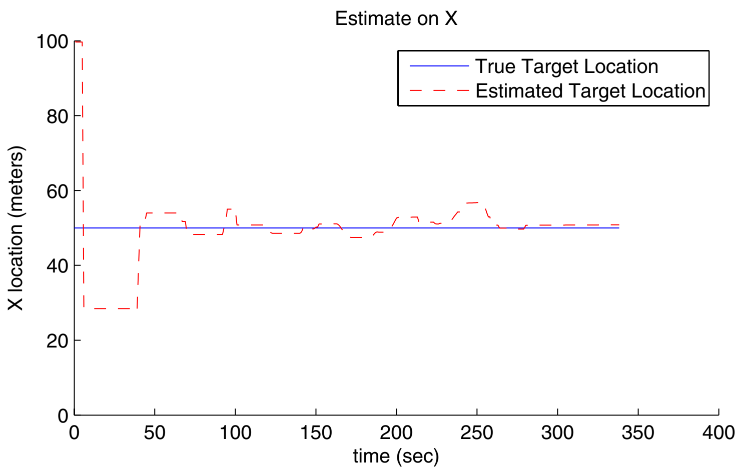
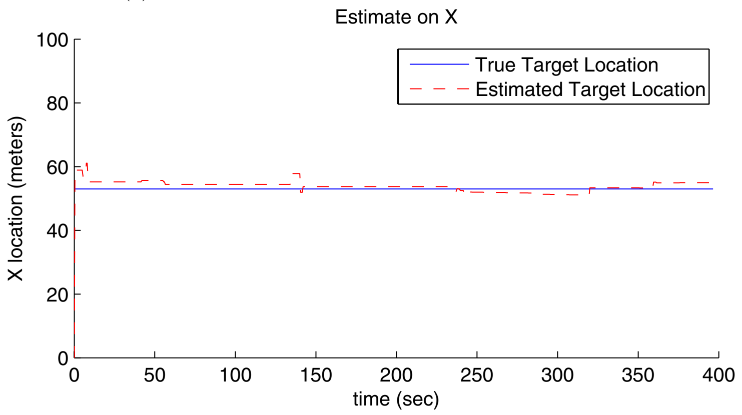


Fig. 5. A sample sensor image while UAVs are localizing the target (a red car)



(a) Location estimate in the East direction with 1 UAV



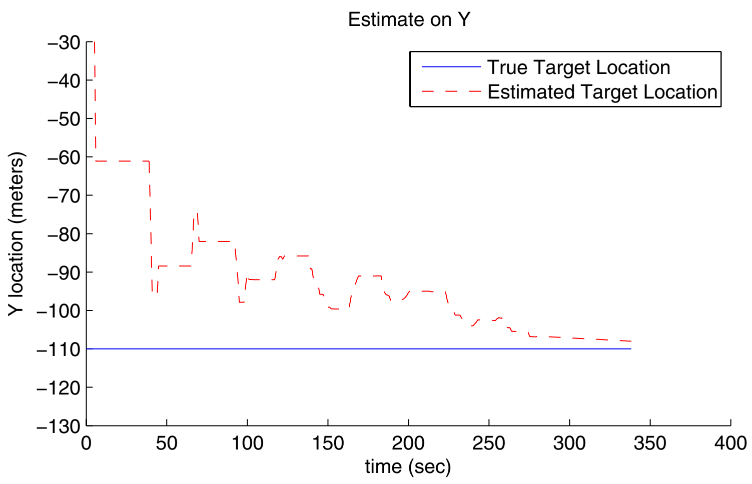
(b) Location estimate in the East direction with 2 UAVs

Fig. 6. Target location estimates in the East direction for experiments with one and two UAVs. The solid line shows the true target location.

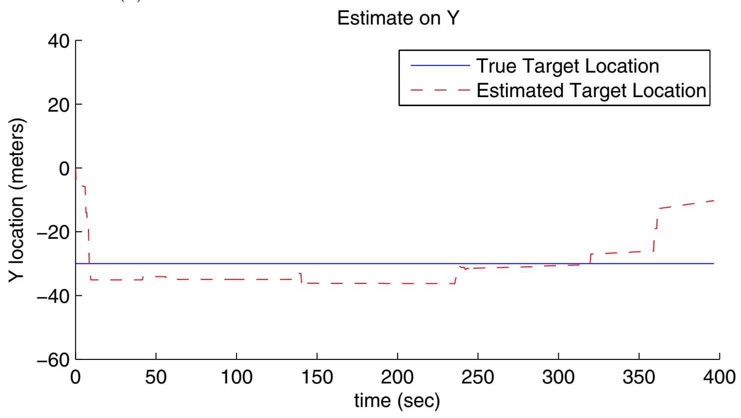
show the localization accuracy using two UAVs performed four times better than the one obtained using a single UAV.

One interesting thing to note here is that, in the experiment with two UAVs, we observe some error spikes in the target location estimates while the estimates gradually converge to the true location. After some rigorous analysis, we concluded that those errors were mainly attributed to poor synchronization of the sensor information with UAV location and attitude. Recall that UAV location and attitude play an important role in calculating the target bearing angle. Better handling of such synchronization issues will be crucial for enhancing overall accuracy of the target localization, which we plan to address in the future.

Figures 6 and 7 depict the progress of target location estimation while the UAVs are localizing the target. Note here that we intentionally moved the target



(a) Location estimate in the North direction with 1 UAV



(b) Location estimate in the North direction with 2 UAVs

Fig. 7. Target location estimates in the North direction for experiments with one and two UAVs. The solid line shows the true target location.

to the North at the end of the experiment with two UAVs. It can be observed in Figure 7-(b) that the target location estimates in the North direction increased toward the end of the experiment. The target location estimates while the target was moving were not considered for the RMS error calculation.

6 Conclusion

In this chapter, we present a heterogeneous sensor fusion system for localizing mobile ground targets using multiple cooperative UAVs. Our preliminary results validated the advantage of using multiple sensors.

Our future research will address two issues. First, we are working on enhancing overall target detectability. In order to do so, we are currently working on fast image feature extraction and pattern recognition algorithms for low-resolution images along with integrating a radio frequency detection sensor. Second, we are trying to understand how poor synchronization between sensors and UAV position affects the target localization accuracy. We are also working on improving the synchronization.

References

1. Pack, D.J., York, G., Toussaint, G.J.: Localizing Mobile RF Targets using Multiple Unmanned Aerial Vehicles with Heterogeneous Sensing Capabilities. In: Proc. NSC 2005 (2005)
2. Plett, G.L., De Lima, P., Pack, D.J.: Target localization using multiple UAVs with sensor fusion via Sigma-Point Kalman Filtering. In: Proc. 2007 AIAA, FL (2007)
3. Plett, G.L., Zarzhitsky, D., Pack, D.J.: Out-of-order sigma-point Kalman filtering for target localization using cooperating unmanned aerial vehicles. In: Hirsch, M.J., Pardalos, P.M., Murphey, R., Grundel, D. (eds.) Advances in Cooperative Control and Optimization. Lecture Notes in Control and Information Sciences, vol. 369, pp. 22–44. Springer, Heidelberg (2007)
4. Brannan, D., Esplen, M., Gray, J.: Geometry. Cambridge University Press, Cambridge (1999)
5. Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press, Cambridge (2000)
6. Mahalanobis, P.C.: On the generalized distance in statistics. Proc. National Institute of Science of India 12, 49–55 (1936)
7. Zhang, Z.: A flexible new technique for camera calibration. Microsoft Research Technical Report MSR-TR-98-71 (March 1999)
8. Johnston Jr., M.G.: Ground Object Geo-Location using UAV Video Camera. In: 25th Digital Avionics Systems Conference, pp. 5C3–1-7 (October 2006)
9. Fu, K.S., Gonzalez, R.C., Lee, C.S.G.: Robotics,Control, Sensing,Vision, and Intelligence. McGraw-Hill, New York (1987)
10. Pack, D.J., York, G.: Developing a control architecture for multiple Unmanned Aerial Vehicles to search and localize RF time-varying mobile targets: part I. In: Proc. IEEE International Conference on Robotics and Automation, Barcelona, Spain, pp. 3965–3970 (April 2005)

Cooperative Control of Multiple Swarms of Mobile Robots with Communication Constraints

Yechiel J. Crispin

Aerospace Engineering Department,
College of Engineering,
Embry-Riddle Aeronautical University,
Daytona Beach, FL 32114, USA
crispinj@erau.edu

Abstract. We study the effect of limited communication range and noise on the cooperative control of a group of swarms of mobile robots using the Particle Swarm Optimization algorithm. The advantage of multiple swarms is the parallel search for a common goal, in addition to the implicit parallelism built in each independent swarm. We demonstrate the method on a problem where a group of swarms searches for multiple similar minima or multiple similar objects in a given domain of the plane. The algorithm is robust with respect to limited communication range for range values of more than 20% of the characteristic size of the domain. The method is applied to the problem of swarms formations, where several swarms of mobile robots are initially dispersed over a given domain in the plane. The formation of the group of swarms breaks down when the communication range is less than half the typical size of the formation.

1 Introduction

We consider the problem of finding multiple minima or detecting multiple objects by a swarm of robots searching a finite domain of the plane, taking into account the effects of limited communication range and communication noise. The problem of finding the minima of a multimodal function with multiple local and global minima has received attention in the literature due to important applications of this problem in many fields of science and engineering [19,9]. With the advent of population based evolving methods, such as genetic algorithms and particle swarm algorithms, it has become possible to find approximate solutions to complex optimization problems which were previously considered hard or intractable. For example, Parsopoulos and Vrahatis [12,13] have recently considered the problem of finding all global minimizers of a multi-modal objective function using Particle Swarm Optimization (PSO). Their approach consists of applying transformations to the objective function such as deflection and stretching techniques in order to isolate the global minima. This approach can work in cases where the objective function is known in closed form a priori.

In the context of a swarm of robots searching for the minima of a physical property in a real environment using a detection device, a technique based on deflection and stretching of the objective function is not a viable approach. First, the function is not known a priori. Second, measurements may be corrupted by noise, making the function noisy and more difficult to approximate by elementary functions. Moreover, trying to generate such an objective function experimentally by scanning the domain and then applying the stretching technique is not a promising proposition, since the problem of mapping a domain or finding the distribution of a scalar property experimentally over a domain is a hard problem in its own right, especially if it is to be achieved without resorting to exhaustive search.

The original PSO algorithm is built on the premise that virtual point particles (or agents) are moving in a virtual multi-dimensional space, searching for a minimum or several minima without taking into account physical constraints on the motion of particles. At the same time, advances have been made in the field of collective search by multiple robots using various approaches, in which real-world physical effects and constraints are taken into account. For example, random search strategies with the effect of imperfect sensors have been studied by Gage [6,7]. Distributed exploration strategies with several real autonomous robots have been studied by Hayes, Martinoli and Goodman [8]. The idea of applying aspects of multi-robot search to improve the PSO has been proposed recently by Pugh, Sagapelli and Martinoli [14]. The neighborhoods of particles in the PSO algorithm were modified to incorporate physical constraints occuring in multiple robots systems. Several variants of the PSO algorithm incorporating various definitions of the robots' neighborhoods have been compared with a standard PSO with global neighborhood. It was shown that the modified PSO algorithms with local neighborhoods can display superior performance on low dimensional standard benchmark problems and degraded performance on higher dimensional problems, when compared to the standard PSO algorithm with a global neighborhood. Pugh and Martinoli [16] also presented a single swarm multiple robot search algorithm based on the PSO in which they studied the effects of the number of robots and communication range. The same authors applied the PSO to the field of robot learning using a robot control system based on neural networks [17,14].

Another approach for treating the problem of multi-modal objective functions using the PSO algorithm is to add a mechanism of niching or speciation, borrowing ideas from genetic algorithms. The PSO algorithm is started with a single swarm or population and speciation occurs as the search evolves, the environment is explored and species or niches of particles or agents develop according to their fitness function. Some niches become spatially isolated, even preventing other particles from penetrating their space. Another possibility is to have the species or niches dispersed over the whole search space and coexisting together. This type of approach has been advanced by Brits, Engelbrecht and van den Bergh [1] in their Niche PSO algorithm. A similar approach using multiple species PSO for multi-modal functions was proposed by Iwamatsu [10]. To the best of our knowledge, a multiple swarm algorithm has not been applied to the field of search with multiple swarms of robots in order to detect multiple objects scattered in a two-dimensional domain.

We propose to solve the problem of finding multiple similar local minima or, equivalently, detecting multiple similar objects, by using several swarms of autonomous robots in parallel, at the same time. Each swarm is controlled by its own coordinating mobile robot. Information can be exchanged between the coordinating mobile robots and a top-level coordinating agent. The domain is divided into several sub-domains and each swarm is assigned its own sub-domain of action by a top-level coordinating agent. In the context of function minimization by PSO, such a parallel algorithm can be implemented on a multiple processor machine, where the algorithm for each separate swarm runs on a separate processor. Coordination between the processors can be achieved by message passing. This is in addition to the implicit parallelism which might be achievable with a single swarm, where the trajectory of each particle in the swarm can be generated by a separate processor [18].

We also consider the problem of swarms formations, where several swarms of mobile robots are initially dispersed over a given domain in the plane. The group of swarms is set into controlled motion and each swarm is assigned a goal to gather at a given location by the top-level coordinating agent. Here also, each swarm is controlled by its own coordinating mobile robot, which sends the coordinates of the rendezvous point to all the robots located within its communication range. If the communication range is not too small, the information about the gathering points will eventually reach all the robots that come into range and the formation will be achieved.

In Section 2, we introduce a PSO formulation of the proposed cooperative control method of multiple swarms and study the effect of limited communications range and noise on the performance of a cooperative search for multiple minima in a two-dimensional domain. In Section 3, we study the effect of limited communication range on multiple swarms formations and we follow with a short summary and conclusions.

2 Cooperative Control of Multiple Swarms with Limited Communication Range

2.1 The Multiple Swarms Control Algorithm

The problem of controlling a single swarm of autonomous robots working towards a common goal with the effects of noise and communication delay has been treated by Crispin [5,3]. We consider the problem of controlling a group of swarms of mobile robots working towards a common goal, such as locating multiple similar objects that are dispersed in a domain. Another common goal is to bring the group of swarms to a structured formation. In both problems, we include the effect of limited communication range.

Let N_s be the number of swarms and N_r the number of robots in each swarm. Initially, the robots are mixed and randomly distributed over the domain. Let k describe the discrete time counter, where $k = 0$ is the initial time $t = 0$ when the swarms are set into motion and $k = N_f$ is the number of time steps Δt at the final time $t_f = N_f \Delta t$.

Each swarm has its own coordinating robot and communication network. Information about the robots locations within the swarm can be passed to the coordinating robot, which is also moving with its swarm. There is a central server or command station that tracks the motion of all the coordinating robots. Let's call this central server the top-level coordinating agent.

Physical effects or constraints are incorporated in order to implement the search method by actual mobile robots. The first effect imposes a limitation on the speed of the robot, or equivalently, a limit on the maximum distance $\Delta X_{\max}$ a robot can move in a given typical time step Δt. Another effect taken into account is limited range and noisy communication between the robots in a swarm and their coordinating robot. At any given time, communication with one or more robots can be lost because that robot has moved out of the range of its respective coordinating robot. The communication signal can also be corrupted by noise.

In order to develop the multiple swarms control algorithm, we start with a basic PSO algorithm for multiple swarms and we add the following constraints: a constraint on the speed, a constraint on the communication range and communication noise. The basic algorithm consists of minimizing a function f of n variables x:

$$\text{minimize} f(x), \text{where}\, x \in \Omega \subset \mathbb{R}^n \text{ and} f : \Omega \mapsto \mathbb{R}$$

subject to the side constraints

$$x_{\min} \leq x \leq x_{\max}$$

where $x_{\min}$ and $x_{\max}$ lie on $\partial\Omega$, the boundary of the domain Ω. The function $f(x)$ can have multiple minima. The search for minima by the N_s swarms is a directed random walk process described by the following system of stochastic difference equations:

$$\begin{aligned}\Delta x(s,i,k+1) &= w(k)x(s,i,k) + c_1 r_1(s,i,k)[P(s,i,k) - x(s,i,k)] \\ &\quad + c_2 r_2(s,i,k)[P_g(s,k) - X(s,i,k)] \qquad (1)\\ x(s,i,k+1) &= x(s,i,k) + \Delta x(s,i,k+1) \qquad (2)\end{aligned}$$

where $s \in [1, N_s]$, $i \in [1, N_r]$ and $k \in [0, N_f]$.

The variable s is a swarm counter, the variable i is a robot counter and the variable k is the discrete time counter. N_f is the total number of time steps. The parameters c_1 and c_2 are real constants that have been discussed in the PSO literature [11,2,12]. Here we use typical values of the constants that have been found to work well in many applications. The constant c_1 is called the cognitive parameter. It determines the weight to be given to the local search in the immediate neighborhood of the particle. The constant c_2 is called the social parameter. It determines the weight to be given to the global solution that was found by the swarm as a social group searching for a common goal. The numerical values of the parameters c_1 and c_2 are not critical to the convergence of the PSO algorithm. Numerical experiments conducted by Kennedy [11] suggest default values $c_1 = c_2 = 2$ for many problems.

The variables $r_1(s,i,k)$ and $r_2(s,i,k)$ are random variables uniformly distributed between 0 and 1. The location $P(s,i,k)$ is the best solution found by robot i in swarm s at time $t = k$ and $P_g(s,k)$ is the best solution found by swarm s at time $t = k$. Recall that the independent variable x is a vector of dimension n. Therefore, r_1, r_2, P and P_g have the same dimension as x. In this chapter, we study swarms moving in a two-dimensional domain which is a subset of the plane $\mathbb{R}^2$, so our discussion is limited to the case $n = 2$. The weight factor $w(k)$ can be either constant or time dependent. If it decreases with time, the search process can usually be improved as the search approaches a local minimum point and smaller steps are needed for better resolution. For example, the parameter $w(k)$ can be set to decrease linearly from an initial value of $w_0 = 0.8$ to a final value of $w_f = 0.2$ after N_f time steps.

The system of equations (1)-(2) describes a directed random walk for each robot i within each swarm s, similar to the Brownian motion of a tracer particle in a fluid. Whereas Brownian motion is an undirected random motion, the motion of a robot in each swarm s will start as random motion, but the distances traveled in given fixed time steps k will eventually decrease as robot i approaches a point $P(s,i,k)$ in the domain where the function reaches a local minimum and as each swarm s as a whole approaches a point $P_g(s,k)$ of the domain where the function reaches a minimum for that specific swarm. So for each swarm s, at any given time $t = k$, each robot i in swarm s has found its best solution $P(s,i,k)$ along its own trajectory. Comparing the values of the function $f(P)$ for the set of robots in each swarm s at time k, the best solution $P_g(s,k)$ for the swarm s can be selected.

$$P(s,i,k) = \underset{k'}{\text{argmin}}\{f(x(s,i,k^{'}))\},\ k^{'} \in [0,k] \tag{3}$$

$$P_g(s,k) = \underset{i}{\text{argmin}}\{f(P(s,i,k))\},\ i \in [1,N_r] \tag{4}$$

The following initial conditions are needed in order to start the solution of the system of difference equations

$$x(s,i,0) = x_{\min} + r(s,i)\Delta x_{\max} \tag{5}$$

$$\Delta x_{\max} = (x_{\max} - x_{\min})/N_{\text{x}} \tag{6}$$

where N_{x} is a typical number of grid segments along each component of the vector x and $r(s,i)$ are random numbers uniformly distributed between 0 and 1. Equations (1)-(2), describing the motion of the swarms, can be written in non-dimensional form. For example, if the domain consists of a two dimensional square domain with side $a = 1000$ m by $a = 1000$ m, then with $N_{\text{x}} = 100$, we can use a typical distance segment of $\Delta x_{\max} = a/N_{\text{x}} = 10$ m. If we take a typical speed of an autonomous robot as $V_c = 1$ m/s, then the characteristic time will be $t_c = \Delta x_{\max}/V_c = 10$ s. We measure x in units of $\Delta x_{\max}$, V in units of V_c and Δt in units of t_c. The equations have the same form in non-dimensional variables.

The next step involves adding the constraints on speed and communication range. First, a limit is placed on the magnitude of the velocity component of each robot in any given direction. Equivalently, we can impose a constraint on the magnitude of the distance traveled in any time step Δt as:

$$|\Delta x(s,i,k+1)| \leq \Delta x_{\max} \tag{7}$$

Under these assumptions, the equations of motion of each swarm become:

$$\begin{aligned}\Delta x(s,i,k+1) &= w(k)x(s,i,k) + c_1 r_1(s,i,k)[P(s,i,k) - x(s,i,k)] \\ &\quad + c_2 r_2(s,i,k)[P_g(s,k) - x(s,i,k)] \qquad (8)\\ x(s,i,k+1) &= x(s,i,k) + \text{sign}\,[\Delta x(s,i,k+1)]\,[\min(|\Delta x(s,i,k+1)|, \Delta x_{\max})] \qquad (9)\end{aligned}$$

The signum function term $\text{sign}(\Delta x(s,i,k+1))$ is added in order to keep the original direction of the motion while reducing the length of the step.

We introduce a constraint on the communication range. Suppose the maximum communication range is R_m. For a given swarm s at time k, let $R(s,i,k)$ be the distance from robot i in the swarm to the robot that is coordinating the motion of that swarm. Without loss of generality, let's choose the index of the coordinating robot as $i = 1$. Communication is maintained when condition (10) is satisfied.

$$R(s,i,k) = \|x(s,i,k) - x(s,1,k)\| \leq R_m \tag{10}$$

If the robot is out of communication range, that is, if $R(s,i,k) > R_m$, its communication link breaks down and its best solution $P(s,i,k)$ at time k cannot be transmitted to the coordinating robot. Therefore this robot does not participate in the search and cannot contribute its solution to its swarm. Also, this robot cannot receive the control signal $P_g(s,k)$, and therefore it uses the last value of the control signal it has previously received and kept in memory when it was within communication range. If it never came within communication range since the beginning of the search, it uses its own value $P(s,i,k)$ instead of the swarm's value $P_g(s,k)$ in order to pursue the search. On the other hand, if the robot is within communication range, its best solution $P(s,i,k)$ is transmitted and used by the coordinating robot in order to determine $P_g(s,k)$ for the swarm. It also uses the control signal $P_g(s,k)$ in attempting to improve its own solution.

The effect of noisy communication signals is taken into account by adding noise to the solution that was found by any robot in any swarm

$$P^{'}(s,i,k) = P(s,i,k) + \delta\mathcal{N}(0,1)\Delta x_{\max} \tag{11}$$

where $P^{'}(s,i,k)$ is the communication signal corrupted by noise, $\mathcal{N}(0,1)$ is a random number with a normal distribution, zero mean and a standard deviation of one. δ is a factor for setting the noise level in units of $\Delta x_{\max}$.

An example of a search for multiple minima in a two-dimensional domain using four swarms is given in the next section.

2.2 Cooperative Search for Multiple Objects in a 2-D Domain

The purpose of the multiple swarms is to collectively find multiple similar minima of a function using measurements of the function in the domain. As a numerical example let's consider a square domain with a side of 10 distance units (DU) by 10 DU, defined by the coordinates $x_i \in [x_{i,\min}, x_{i,\max}] = [-5, 5]$.

Consider a test function of two variables $f(x_1, x_2)$ with a cutoff at $f(x_1, x_2) = f_c$ defined by

$$f(x_1, x_2) = \min\left[g(x_1, x_2), f_c\right] \tag{12}$$

where $g(x_1, x_2) = (\cos x_1)^2 + (\sin x_2)^2$.

The function has multiple minima, with the objective value $f = 0$ at multiple points in the domain. This function is used to simulate a search for multiple similar objects with the same characteristic diameter and depth scattered in a landscape which is otherwise flat with a constant elevation $f_c = 0.2$. Decreasing the value of f_c causes the diameter of the objects to decrease.

The domain is divided into four sub-domains and four swarms are used to detect four different minima at the same time. The boundaries of the sub-domains are given in

Table 1. Parameters for the four swarms

Swarm	$(x_{1\,\min}, x_{1\,\max})$	$(x_{2\,\min}, x_{2\,\max})$	N_r	$c_1 = c_2$	w	N_f
1	$(-5, 0)$	$(0, 5)$	10	2	0.8	150
2	$(0, 5)$	$(0, 5)$	10	2	0.8	150
3	$(-5, 0)$	$(-5, 0)$	10	2	0.8	150
4	$(0, 5)$	$(-5, 0)$	10	2	0.8	150

Table 1. Each swarm has 10 robots and the search runs for 150 time steps. When this problem is solved with a single swarm with an equivalent total number of 40 robots, only one object is detected. We would like the swarms to be able to detect 4 objects with the same number of 40 robots over the same period of time. Table 1 lists the values of the parameters used in this example.

The constraint on the maximum step length $\Delta x_{1\,\max}$ or $\Delta x_{2\,\max}$ that a robot can move along the coordinates x_1 or x_2 is:

$$\Delta x_{1\,\max} = \Delta x_{2\,\max} = (x_{1\,\max} - x_{1\,\min})/N_{\mathrm{x}} = 10DU/10 = 1DU \tag{13}$$

where $N_{\mathrm{x}} = 10$ is the number of segments along each coordinate. Figure 1 shows how the search process evolved during the solution of this problem for a communication range of $R_m = 2$ or 20% of the side of the square. The chart on the upper left shows the random distribution of the 4 swarms of robots at the initial time $k = 0$. The robots of the first swarm are denoted by circles. Robots in the second swarm are denoted by the '+' symbol, those in the third swarm, by the 'x' symbol, and those in the fourth swarm, by stars.

The upper right of the figure displays the 4 minima of the function with values close to the minimum value $f = 0$ that were found by the 4 swarms at time $k = N_f = 150$. The trajectories of the best solution found by each swarm are shown in the bottom left. Here, all the robots' locations along the trajectories are denoted by dots. As the robots in a swarm start converging to the vicinity of an object, some of the robots will get closer to the object, whereas the other robots in this swarm will be located at larger distances from the object. Among the robots that are getting closer to the center of the object, some will start detecting a value of the objective function that is less than the cutoff value f_c but greater than zero, a value which occurs at the center of the object. For each swarm, sorting the robots according to the value of their respective objective function, we select the robot with minimum f and plot its trajectory. Such a robot is not of fixed identity and will vary from one run to the next. Each trajectory shown in the figure belongs to the robot that found the best solution in each of the 4 swarms for that specific run. When implementing this method in hardware, using actual robots searching the domain and measuring the values of the function at the points along their trajectories, the method is an inherently parallel process, since each swarm is autonomous and the swarms work in parallel at the same time. The motion of the robots in each swarm is controlled by a coordinating robot, which is moving together with the specific swarm it is controlling.

Figure 2 is for a case similar to that of Figure 1, but for a smaller communication range of $R_m = 1.2$ or 12% of the side of the square domain. Here only two objects are

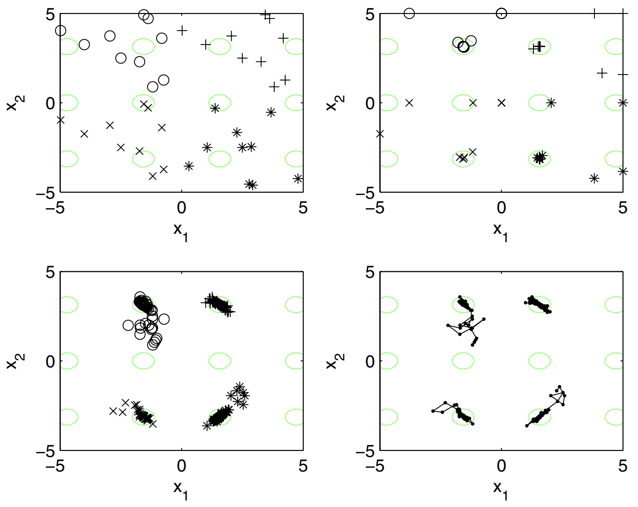


Fig. 1. Search for multiple objects using $N_s = 4$ swarms of $N_r = 10$ robots in each swarm with a communication range $R_m = 2$. Upper left: Initial distribution of the robots. Upper right: Robots locations after $N_f = 150$ time steps. Lower left: Points visited by one robot from each swarm that ended at a minimum distance from the center of an object. Lower right: Trajectories of the robots shown on the lower left chart.

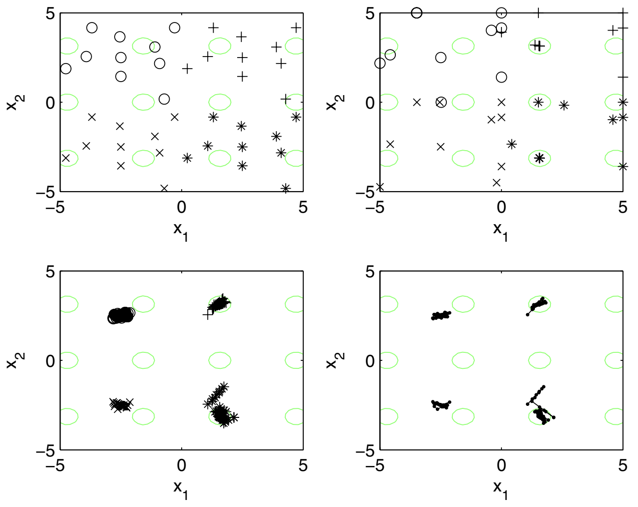


Fig. 2. Same as Figure 1 but with a smaller communication range of $R_m = 1.2$

detected. Figure 3 is for a case similar to that of Figure 1, but for a smaller communication range of $R_m = 0.5$ or 5% of the side of the square domain. In this case, no objects are detected.

Figures 4 and 5 show the combined effect of limited communication range and communication noise. Figure 4 displays a case with a range $R_m = 2$ and noise level of

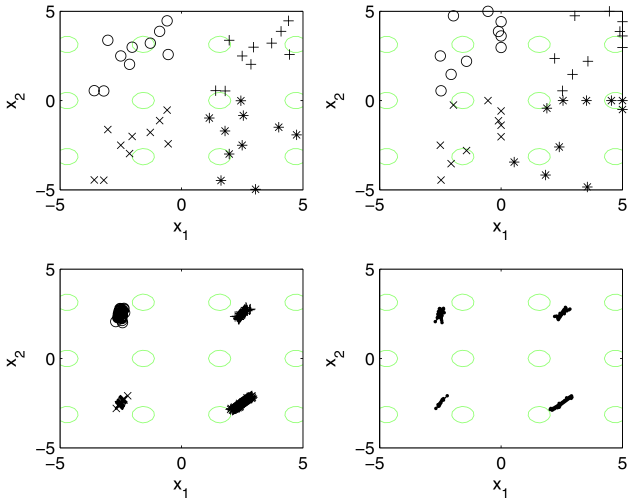


Fig. 3. Same as Figure 1 but with a smaller communication range of $R_m = 0.5$

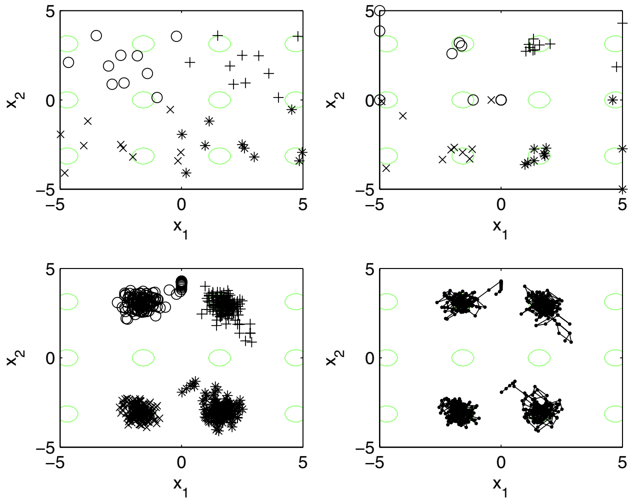


Fig. 4. Combined effect of limited communication range and noise with $R_m = 2$ and $\delta = 1$. $N_s = 4$ swarms of $N_r = 10$ robots in each swarm. $N_f = 150$.

$\delta = 1$. From the trajectories of the 4 robots, it can be seen that the robots are performing a local random search around the 4 objects with many large steps on the order of $\Delta X_{1\,\max} = 1$ DU. A similar effect is obtained for a lower communication range and the same noise level as displayed in Figure 5 for $R_m = 1.2$ and $\delta = 1$.

The algorithm is robust with respect to communication constraints for ranges $R_m > 2$, that is, range greater or equal to 20% of the side of the square domain. When the

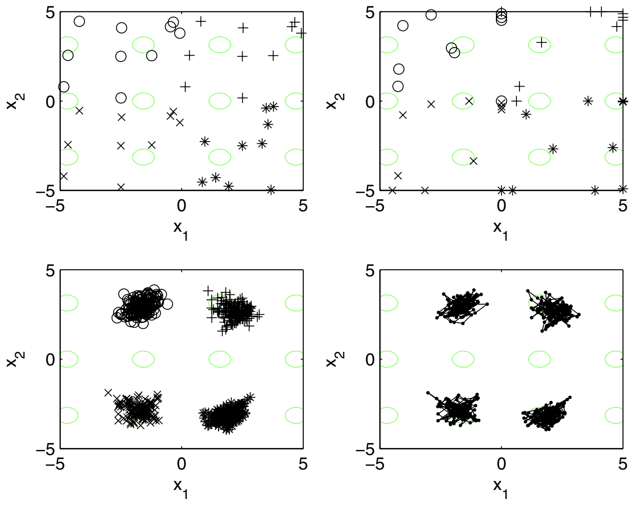


Fig. 5. Combined effect of limited communication range and noise with $R_m = 1.2$ and $\delta = 1$. $N_s = 4$ swarms of $N_r = 10$ robots in each swarm. $N_f = 150$.

range is reduced below $R_m = 2$, the search performance displays deterioration, which is further accentuated by noise.

3 Multiple Swarms Formations with Limited Communication Range

In this section we show that the method described above for finding multiple minima of a function can be applied to the problem of bringing multiple swarms into a prescribed formation. The problem of gathering a single swarm in a given point in the plane has been treated in [4]. The present work is an extension of the problem to include multiple swarms with limited communication range. In this example, we consider a two-dimensional square domain $n = 2$, of sides 1000 m by 1000 m. A case in which six swarms of 20 robots each, form a triangular configuration is presented.

The final arrangement of the swarms will be around an equilateral triangle of side $a = 600$ m. The base of the triangle is located at $y_s = -300$ m. Three swarms gather at the vertices of the triangle $P_1(0, y_s + a\sqrt{3}/2)$, $P_2(a/2, y_s)$ and $P_3(-a/2, y_s)$. The three other swarms gather at the mid-points on each side of the triangle with coordinates $P_4(-a/4, y_s + a\sqrt{3}/4)$, $P_5(a/4, y_s + a\sqrt{3}/4)$ and $P_6(0, y_s)$. These coordinates are broadcast to the six swarms and serve as their respective control signals. The parameters for this case are given in Table 2.

In this case, the maximum step size is $\Delta x_{1\,\max} = \Delta x_{2\,\max} = 20$ m= 1 DU. Figure 6 shows the motion of the swarms for a maximum communication range of $R_m = 400$ m, starting with a uniform random distribution of the 120 robots in the 6 swarms at time $k = 0$, as shown in the upper left side of the figure. The state of the swarms at time

Table 2. Parameters for the Swarms Formation Problem

$(x_{1\,\min}, x_{1\,\max})$	$(x_{2\,\min}, x_{2\,\max})$	N_x	N_s	N_r	$c_1 = c_2$	w	N_f
$(-500, 500)$	$(-500, 500)$	50	6	10	2	0.8	150

$k = 50$ is displayed on the upper right side of the figure. The lower left side shows the state of the swarms at time $k = 100$ and the lower right displays the state at $k = 200$ when the final triangular formation is achieved. We can count the number of robots that could not make it into the formation and use this number as a performance criterion. In this case, with $R_m = 400$ m, 13 robots out the 120 did not make it into formation after the given time and 6 robots remain on the boundary of the domain.

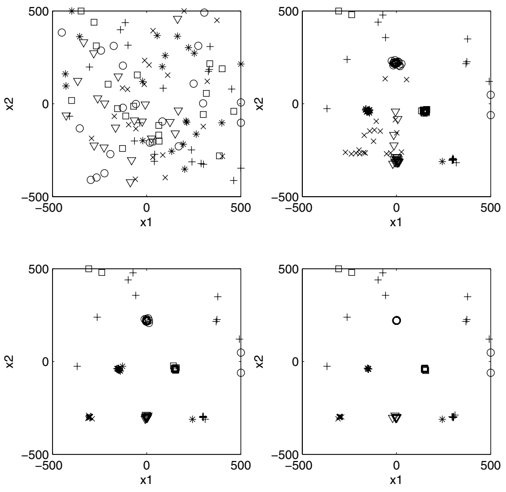


Fig. 6. Six swarms formation with a limited communication range of $R_m = 400$ m. Upper left: swarms distribution at $k = 0$. Upper right at $k = 50$. Lower left at $k = 100$. Lower right after $k = N_f = 200$ time steps.

Figure 7 displays a case similar to Figure 6 but with a lower value of the maximum communication range of $R_m = 350$ m. Here there are some 17 robots that did not reach their target points in the formation.

Figure 8 displays a case similar to Figure 6 but with a lower value of the communication range of $R_m = 300$ m. There are some 35 robots out the 120 that did not reach their target points within the given time.

With a communication range of $R_m = 250$ m it was not possible to achieve the formation after 200 time steps. The communication range has to be at least half the length of the side of the formation triangle ($a = 600$ m) in order to achieve formation.

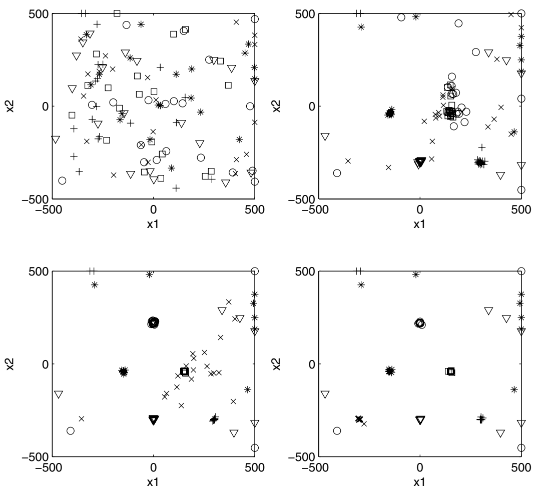


Fig. 7. Six swarms formation with a limited communication range of $R_m = 350$ m. Upper left: swarms distribution at $k = 0$. Upper right at $k = 50$. Lower left at $k = 100$. Lower right after $k = N_f = 200$ time steps.

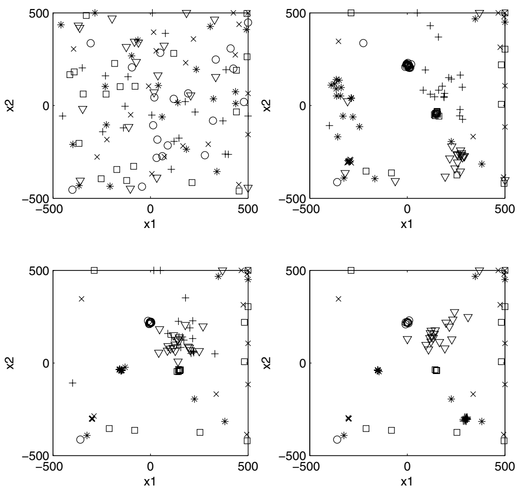


Fig. 8. Six swarms formation with a limited communication range of $R_m = 300$ m. Upper left: swarms distribution at $k = 0$. Upper right at $k = 50$. Lower left at $k = 100$. Lower right after $k = N_f = 200$ time steps.

4 Conclusion

A method for the cooperative control of a group of swarms of robots based on the PSO algorithm of swarm intelligence with the effects of limited communication range and noise has been presented. Organizing the control of robots as multiple swarms can speed up collaborative optimization with multiple minima or the detection of multiple objects in parallel. The method has been demonstrated on a problem where a group of swarms search for multiple similar minima or multiple objects in a given domain of the plane. The algorithm is robust with respect to limited communication range for range values of more than 20% of the characteristic size of the domain.

The method was used for swarm formations, by prescribing the final locations of the swarms as control signals. The algorithm is robust to limited communication range as long as it is greater than half the characteristic size of the swarms formation.

References

1. Brits, A., Engelbrecht, A.P., van den Bergh, F.: Locating Multiple Optima Using Particle Swarm Optimization. Applied Mathematics and Computation 189, 1859–1883 (2007)
2. Clerc, M.: Particle Swarm Optimization, ISTE, Newport Beach (2006)
3. Crispin, Y.: Levy Flights in Robot Swarm Control and Optimization. In: Cooperative Networks, Control and Optimization, ch. 5. Edward Elgar Publishing (in print, 2008)
4. Crispin, Y.: Levy Flights in the Stochastic Dynamics of Robot Swarm Gathering. In: Proceedings of the 2nd International Workshop on Multi-Agent Robotic Systems, MARS 2006, Setubal, Portugal (August 2006)
5. Crispin, Y.: Cooperative Control of a Robot Swarm with Network Communication Delay. In: Proceedings of the First International Workshop on Multi-Agent Robotic Systems - MARS 2005, Barcelona, Spain (2005)
6. Gage, D.W.: Randomized Search Strategies with Imperfect Sensors. In: Proceedings of the SPIE Mobile Robots, vol. 2058, pp. 270–279 (1993)
7. Gage, D.W.: Many Robots MCM Search Systems. In: Proceedings of the Symposium on Autonomous Vehicles in Mine Counter Measures, Monterey, CA, pp. 4–7 (1995)
8. Hayes, A.T., Martinoli, A., Goodman, R.: Comparing Distributed Exploration Strategies with Simulated and Real Autonomous Robots. In: Proceedings of the Fifth International Symposium on Distributed Autonomous Robotic Systems, DARS 2000, pp. 261–270 (2000)
9. Horst, R., Tuy, H.: Global Optimization, Deterministic Approaches. Springer, New York (1996)
10. Iwamatsu, M.: Multi-Species Particle Swarm Optimizer for Multi-Modal Function Optimization. IEICE Transactions on Information and Systems E89-D(3), 1181–1187 (2006)
11. Kennedy, J., Eberhart, R.C.: Swarm Intelligence. Morgan Kaufman Publishers, San Francisco (2001)
12. Parsopoulos, K.E., Vrahatis, M.N.: Recent Approaches to Global Optimization Problems Through Particle Swarm Optimization. Natural Computing 1, 235–306 (2002)
13. Parsopoulos, K.E., Vrahatis, M.N.: On the Computation of All Global Minimizers Through Particle Swarm Optimization. IEEE Transactions on Evolutionary Computation 8(3) (June 2004)
14. Pugh, J., Segapelli, L., Martinoli, A.: Applying Aspects of Multi-Robot Search to Particle Swarm Optimization. In: Dorigo, M., Gambardella, L.M., Birattari, M., Martinoli, A., Poli, R., Stützle, T. (eds.) ANTS 2006. LNCS, vol. 4150, pp. 506–507. Springer, Heidelberg (2006)
15. Pugh, J., Martinoli, A.: Multi-Robot Learning with Particle Swarm Optimization. In: Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006), pp. 441–448 (2006)

16. Pugh, J., Martinoli, A.: Inspiring and Modeling Multi-Robot Search with Particle Swarm Optimization. In: Proceedings of the IEEE Swarm Intelligence Symposium (SIS 2007), pp. 332–339 (2007)
17. Pugh, J., Martinoli, A., Zhang, Y.: Particle Swarm Optimization for Unsupervised Robotic Learning. In: Proceedings of the IEEE Swarm Intelligence Symposium (SIS 2005), pp. 92–97 (2005)
18. Schutte, J.F., Reinholt, J.A., Fregly, B.J., Haftka, R.T., George, A.D.: Parallel Global Optimization with the Particle Swarm Algorithm. International Journal of Numerical Methods in Engineering (2003)
19. Torn, A., Zilinskas, A.: Global Optimization. Springer, Berlin (1989)

Towards Optimal Positioning of Surveillance UGVs*

Ulrik Nilsson, Petter Ögren, and Johan Thunberg

Swedish Defence Research Institute (FOI),
164 90 Stockholm, Sweden
{ulrik.nilsson,petter.ogren,johan.thunberg}@foi.se

Abstract. Unmanned Ground Vehicles (UGVs) equipped with surveillance cameras present a flexible complement to the numerous stationary sensors being used in security applications today. However, to take full advantage of the flexibility and speed offered by a group of UGV platforms, a fast way to compute desired camera locations to cover an area or a set of buildings, e.g., in response to an alarm, is needed.

Building upon earlier results in terrain guarding and sensor placement we propose a way to find candidate guard positions that satisfy a large set of view angle and range constraints simulataneously. Since the original problem is NP-complete, we do not seek to find the true optimal set of guard positions. Instead, a near optimal subset of the candidate points is chosen using a scheme with a known approximation ratio of $O(\log(n))$. A number of examples are presented to illustrate the approach.

1 Introduction

As the market for surveillance continues to grow, UGV-mounted cameras are becoming a natural complement to stationary cameras and manned patrolling. The flexibility offered by UGVs is particularly important in cases of alarm response, temporary replacement of stationary cameras, or when e.g., some valuable containers are stored overnight in a large harbour terminal with only perimeter surveillance.

In this chapter we investigate how small scale UGVs, such as the one depicted in Figure 1, can be used in surveillance and security applications. In particular we will address the problem of autonomously finding a set of reachable camera locations that satisfy the requirements for a given surveillance task prompted by e.g., an alarm. We begin by giving an overview of the previous work in this field.

There are many ways to categorize the literature related to automatic camera positioning problems. The world models used are either two dimensional [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12] or three dimensional [6, 8, 13, 14, 15, 16, 17]. Most

* All three authors were funded by the Swedish defence materiel administration (FMV) and the Swedish armed forces through the Technologies for Autonomous and Intelligent Systems (TAIS) project. 297316-LB704859.

Fig. 1. Groundbot, the Surveillance UGV that will be used in real world experiments (www.rotundus.se)

of the early papers are purely theoretical, whereas more recent work include application areas such as robotics [5, 9, 11, 18, 19].

When studying camera positioning problems, the choice of an appropriate camera model is an important part of the problem statement. While many papers consider only occlusion constraints, some also deal with explicit range constraints [2, 15, 16, 18], while others incorporate limited field of view [2, 6, 9, 18, 20] or resolution constraints [2, 18].

There are also a number of different problem objectives present in the literature. Most of the early papers focus on finding a general upper bound on the number of guards, as a function of the environment complexity in terms the number of *so-called* vertices and holes. These formulations include the well known *Art gallery problem* [1].

Even though many of the proofs mentioned above are constructive and can be used to place guards in specific problem instances, other algorithms tailored to solving such specific instances have been proposed. The *Minimum point guard problem* was defined by Eidenbenz [13] as the problem of finding the minimal set of guard positions that cover an area. This problem is known to be NP-hard [21], as is the problem of finding a set of guards whose cardinality is at most $1 + \epsilon$ times the optimum [22].

Greedy approximation schemes have been proposed [3, 13] and analyzed using a transcription to the *so-called Minimum set cover problem* [13]. These greedy schemes however need a set of candidate guard positions to choose from and the choice of such positions have been studied using convex covers [23]. Other ways of finding candidate positions include vertex coloring [17] and an approximate *so-called* visibility index [16]. Tentative observer positions can also be found by partitioning the whole 3D-space using a huge set of planes or surfaces indicating where the number of visible vertices changes [9, 13]. Other approaches include randomized search [24] and random sampling [4]. A paper by Vazques et al., [19], introduce the new concept of *Viewpoint entropy* and use exhaustive search to maximize this entropy in terms of camera position. Finally, an algorithm for viewpoint computation considering an arm-mounted stereo camera was presented in a paper where the authors propose a number of interesting constraints and perform the viewpoint optimization using a genetic algorithm [18].

In this chapter, we use the idea of focusing on the surfaces where the visible set changes [9, 13] and introduce image quality constraints such as range and angle of incidence into the computations to find a set of candidate guard positions.

The organization of this chapter is as follows. In Section 2, we state our problem and propose a solution in Section 3. Examples illustrating the approach are presented in Section 4. Finally, the chapter is concluded in Section 5.

2 Problem Formulation

The problem we study is closely related to the *Minimum point guard problem* defined by Eidenbenz [13]. Here however, we incorporate explicit constraints on camera field of view, as well as image resolution in terms of pixels per meter of surveyed building. This problem is motivated by situations where one wants to either make sure that no one exits the buildings, or when movements in windows need to be monitored e.g., to look for snipers (see Remark 1). Formally we define the problem as follows

Problem 1. (Minimum wall guard problem)
Let $W = \{(p_i, q_i) : p_i, q_i \in \mathbb{R}^2\}$ be a set of line segments corresponding to the walls that needs to be surveyed. Furthermore, let $O \subset \mathbb{R}^2$ be the union of all obstacles.

The *Minimum wall guard problem* is the problem of finding a minimum set $S \subset \mathbb{R}^2$ of points on the ground plane such that every wall w_i in W is *guarded* by a point s_j in S. By *guarded* we mean that s_j and w_i satisfy the constraints in Definitions 1, 2 and 3.

Definition 1 (Visibility constraint). *A wall $w_i = (p_i, q_i) \in W$ is visible to a point guard $s_j \in S$ if the solid triangle (s_j, p_i, q_i) does not intersect the interior of the obstacle set O.*

Definition 2 (Resolution constraint). *Given a camera field of view α and image quality constants stating that every segment of length δa of the wall being surveyed must cover at least a fraction $k \in (0, 1)$ of the image, a wall $w_i = (p_i, q_i) \in W$ and a point guard $s_j \in S$ satisfies the* resolution constraint *if*

$$||r - s_j|| \leq \frac{\delta a \cos(\phi)}{k\alpha} \tag{1}$$

for all points r in w_i, where ϕ is the angle of inclination, i.e., the angle between the line (r, s_j) and the normal of w_i.

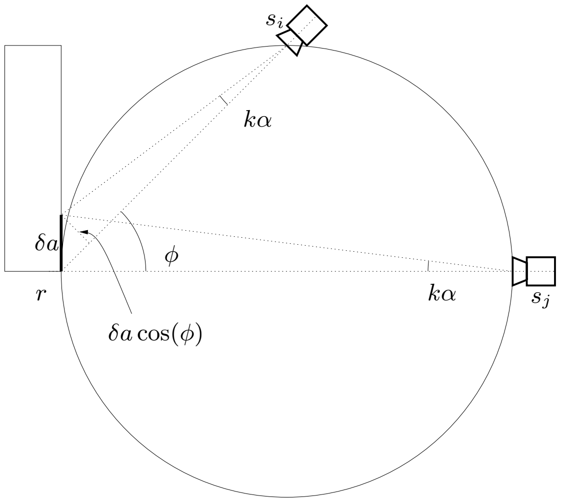


Fig. 2. Note how the boundary of the camera positions satisfying the resolution constraints in Equation (1) corresponds to a circle, denoted C_{res} in Definition 4. As seen above, the distance $||r - s_i|| = \frac{\delta a \cos(\phi)}{k\alpha}$ is smaller than $||r - s_j|| = \frac{\delta a \cos(0)}{k\alpha}$.

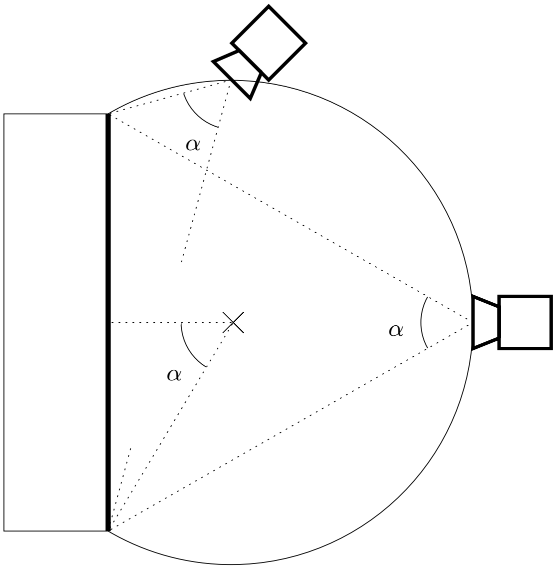


Fig. 3. Camera view angle constraint circle C_{fov} for a wall of length a and a camera field of view of α

The definition is illustrated in Figure 2, where it can be seen that the points corresponding to equality in Equation (1) trace out a circle. This circle is denoted C_{res} in Definition 4 below.

Remark 1. The above definition is motivated by the fact that computer vision algorithms often need at least some given number of pixels across a given object in order to do recognition with reasonable accuracy.

To handle field of view limitations we make the following definition.

Definition 3 (Field of view constraint). *Given a camera view angle limit α, the wall $w_i = (p_i, q_i) \in W$ and a point guard $s_j \in S$ satisfies the* field of view constraint *if*

$$\angle(p_i, s_j, q_i) \leq \alpha, \tag{2}$$

i.e., the angle between (p_i, s_j) and (q_i, s_j) is less than α.

It is well known [9] that Equation (2) will constrain the camera to be outside a circle segment as depicted in Figure 3. This circle is denoted C_{fov} in Definition 4 below.

Having defined the problem we are trying to solve, we now go on to describe the solution we propose.

3 Proposed Solution

In this section, we begin by giving an overview of the algorithm, and then describe the various steps in detail. The main idea is to carefully construct a set of candidate guard positions, and then choose a subset of these by transcribing the problem to a *Minimum set cover problem*, which in turn is solved by a known approximation method.

We propose the following algorithm to find a solution to Problem 1.

Algorithm 1

1. *Find the candidate guard set S as defined below.*
2. *Calculate the walls guarded by each $s \in S$, using Definitions 1, 2 and 3.*
3. *Transcribe the problem of finding a subset of S that guards all walls W to a Minimum set cover problem [13].*
4. *Solve the Minimum set cover problem using a greedy approach with approximation factor $O(\log(n))$ [13].*

3.1 The Candidate Set S

To specify the set S in Step 1 of the algorithm, we need the following set of definitions.

Definition 4 (Constraint circles C_{res} and C_{fov}). *Given a wall $w_i = (p_i, q_i)$: Let $C_{res}(x)$ be the set of s_j yielding equality in Equation (1), when $r = x$. Furthermore, let $C_{fov}(w_i)$ be the set of s_j yielding equality in Equation (2).*

Given these circles we go on to define the area satisfying both Definition 2 and 3, i.e., both resolution and field of view constraints.

Definition 5 (The Region R_{rf}). *Given a wall $w_i = (p_i, q_i)$ and letting h denote the convex hull of a set we define the region as follows*

$$R_{rf}(w_i) = \big(h(C_{res}(p_i)) \cap h(C_{res}(q_i))\big) \setminus h(C_{fov}(w_i)) \tag{3}$$

i.e., the set of all camera positions satisfying Definition 2 and 3 for a given wall w_i.

The set R_{rf} is depicted for two different field of view angles α, in Figure 4.

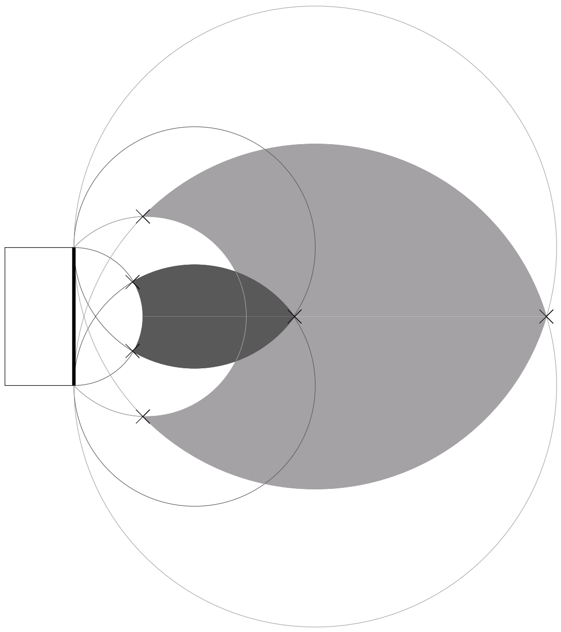


Fig. 4. View angle and resolution constraints for two different view angles. The dark gray region shows the feasible set for a camera with view angle $\alpha = 90°$ and the light gray for a camera with view angle $\alpha = 45°$.

In order to take occlusions into account below, we first note that the obstacles that affect candidate points within $R_{rf}(w_i)$ lie either inside of $R_{rf}(w_i)$, or inbetween $R_{rf}(w_i)$ and w_i. Thus we define the following

Definition 6 (R_O). *The set R_O is defined as follows*

$$R_O = h(w_i \cup R_{rf}(w_i)) \tag{4}$$

i.e., the convex hull of the wall w_i and the region satisfying Definitions 2 and 3.

The area R_O is dashed in Figure 5.

Remark 2. Note that if objects in R_O occlude vision from all of $R_{rf}(w_i)$ to the corresponding wall w_i, then Problem 1 has no feasible solution.

It is now time to define the candidate set S . Note that a good camera position obviously belongs to the set $R_{rf}(w_i)$ for one or more walls w_i. Therefore, we look at intersections of constraint boundaries inside of $R_{rf}(w_i)$.

Definition 7 (The Candidate Set S). *Let $l_{occ}(w_i)$ be the set of lines, that pass through either p_i or q_i, and intersect the boundary, but not the interior of $O \cap R_O(w_i)$. Then let $L_{occ}(w_i)$ be the set of line segments $L_{occ}(w_i) \subset l_{occ}(w_i) \cap R_{rf}(w_i)$ that are beyond the respective obstacles O, from the wall point of view. Furthermore, let Ω be as follows*

$$\begin{aligned}\Omega = &(\cup_j \{C_{res}(p_j) : (p_j, q_j) \in W\}) \\ &\cup (\cup_j \{C_{res}(q_j) : (p_j, q_j) \in W\}) \\ &\cup (\cup_j \{C_{fov}(w_j) : w_j \in W\}) \\ &\cup (\cup_j \{L_{occ}(w_j) : w_j \in W\})\end{aligned}$$

i.e., the set of all constraint boundaries in terms of arcs and line segments. The candidate set *S is defined to be the set of all intersections of an element in Ω with another element of Ω.*

The sets S and Ω for a single wall and three external obstacles are illustrated in Figure 5.

Remark 3. To obtain the set S requires $\mathcal{O}(n^2)$ computations, where n is the number of vertices of buildings and obstacles in the area.

3.2 Calculation of Guarded Walls

When the set S of candidate guard positions is found, the number of walls $w \in W$ satisfying Definitions 1, 2, and 3, are computed for each $s \in S$. If all walls that can be guarded by a single position do not fit into one single camera view due to the field of view limitation, this candidate position is duplicated and stored with all possible maximal combinations of guarded walls and corresponding viewing directions. Thus to each $s \in S$ a viewing direction $\psi(s)$ and a set of guarded walls $W(s)$ are assigned.

3.3 Transcription to the Minimum Set Cover Problem

The *Minimum set cover problem* is defined as follows [13].

Problem 2 (Minimum set cover problem). Let $E = \{e_1, \ldots, e_n\}$ be a finite set of elements, and let $\Theta = \{\theta_1, \ldots, \theta_m\}$ be a collection of subsets of E, i.e., $\theta_j \subseteq E$. The *Minimum set cover problem* is the problem of finding a minimum subset $\Theta' \subseteq \Theta$ such that every elements $e_i \in E$ belongs to at least one subset in Θ'. We say that E is covered by Θ'.

By identifying the elements with walls, $e_i = w_i$, and subsets with guarded walls $\theta_j = W(s_j)$ we see that finding the minimal set of guarding positions is the same thing as finding the minimum set cover, i.e., solving Problem 2.

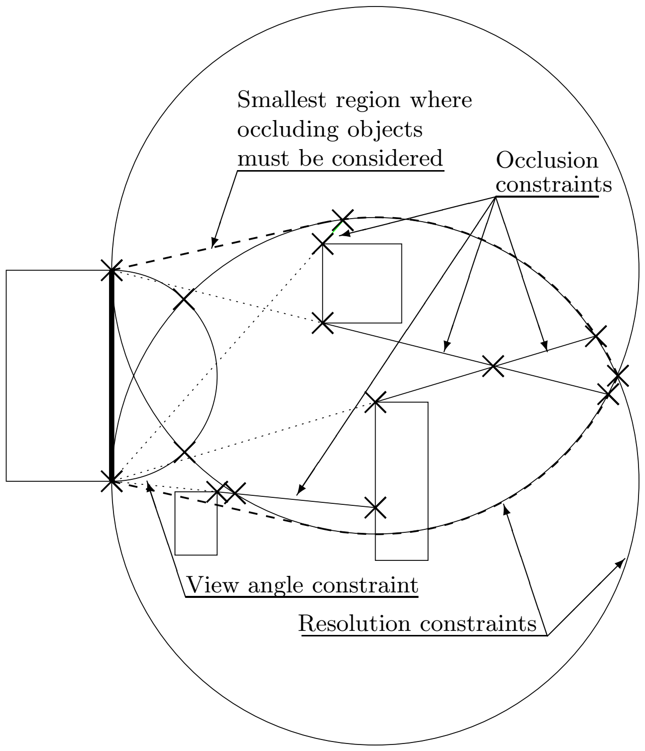


Fig. 5. Construction of the candidate set S. To reduce the size of S even further, intersections that occur between an obstacle and the wall of interest can be removed.

3.4 Solving the Minimum Set Cover Problem

Problem 2 is NP-hard [13] and hence hard to solve to optimality in practice. However, it is known that a standard greedy algorithm gives an approximation ratio of $\mathcal{O}(\log(n))$, where n is the number of subsets [13]. The greedy algorithm simply adds the guarding position in S that covers the largest set of walls first. Then these walls are removed from the list of uncovered walls, and a new guarding position is chosen that covers the largest set of walls from the remaining list. This procedure is iterated until all walls are guarded.

Remark 4. Note that the greedy approach described above is just one choice of solution to the *Minimum set cover problem*. Any other way of finding a good solution to this subproblem works just as well in the proposed approach.

Having discussed the problem formulation and proposed algorithm in detail, we move on to the example Section.

4 Example Problems

In this section we will apply the proposed algorithm to two example problems, one fairly straightforward with a single building, and one more complex with four irregularly shaped buildings.

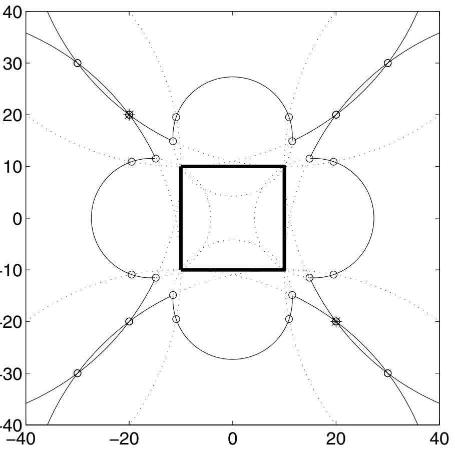


Fig. 6. A single building. The sets $R_{rf}(w_i)$ are shown in solid curves and the corresponding circles C_{res} and C_{fov} are dotted. The solution candidates are shown as circles and the chosen camera positions as asterisks (*). Note that a two guard solution is feasible due to the fact that the sets R_{rf} overlap slightly near (25,25), (-25,25), (-25,-25) and (25,-25).

The first example is depicted in Figure 6. The sets $R_{rf}(w_i)$ are shown, as well as the resulting candidate positions and final UGV positions. Note that the overlapping R_{rf} make a two guard solution feasible. The corresponding field of view cones are shown in Figure 7.

To illustrate the effect of the image resolution constraint we increase k in Equation (1), corresponding to the required number of pixels per meter of surveyed object. The new setting makes it impossible for a single UGV to see two different walls with acceptable resolution. As can be seen in Figure 8, there is no overlap between the R_{rf} sets. In this new problem, one UGV for each wall is needed to solve the problem. Again, the corresponding field of view cones are shown in Figure 9.

In Figure 10, a more complex scenario is depicted. As can be seen, seven UGVs are required to cover the 19 walls of the four buildings. Note that although no explicit obstacles are present, the buildings themselves serve as obstacles occluding the view of the UGVs.

The order in which the guard positions where chosen can be deduced from the dashed lines in Figure 10. Since the greedy algorithm picks the positions guarding a large number of walls first we see that the UGVs near (-10,37) and (80,10),

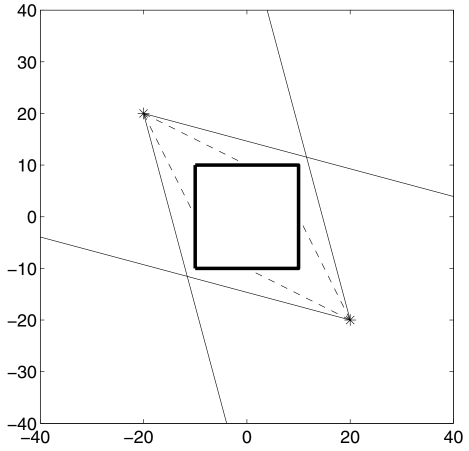


Fig. 7. The detailed solution to the problem of Figure 6. The guards are denoted by asterisks (*) and dashed lines are drawn between the guards and the walls they guard. Furthermore, the field of view cones are illustrated in solid lines.

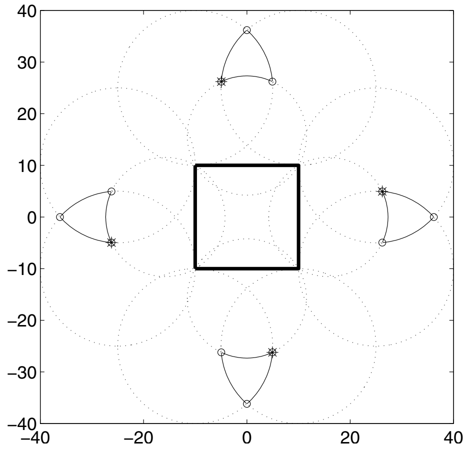


Fig. 8. The same scenario as in Figure 6, but with much tougher resolution constraints. Again, candidates are denoted by circles and the chosen UGV positions are shown using asterisks (*). Note that a two UGV solution is not feasible.

guarding four walls each, were chosen first. Then the two UGVs near (15,-20) and (39,70), guarding three walls were picked. Finally, the ones at (40,25) and (-10,0), guarding two walls, and the one at (40,-10), guarding one wall, were chosen to achieve high quality surveillance of all designated walls.

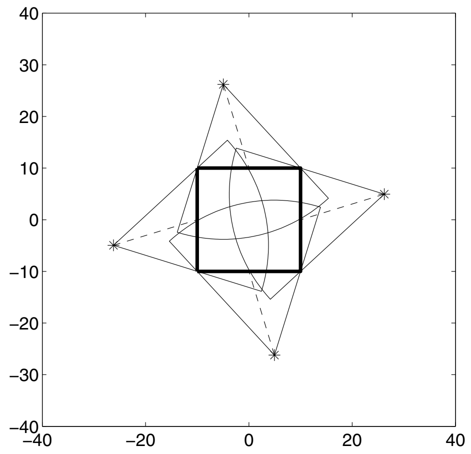


Fig. 9. The detailed solution of the problem in Figure 8. Four UGVs are required to guard all four walls with the required image quality.

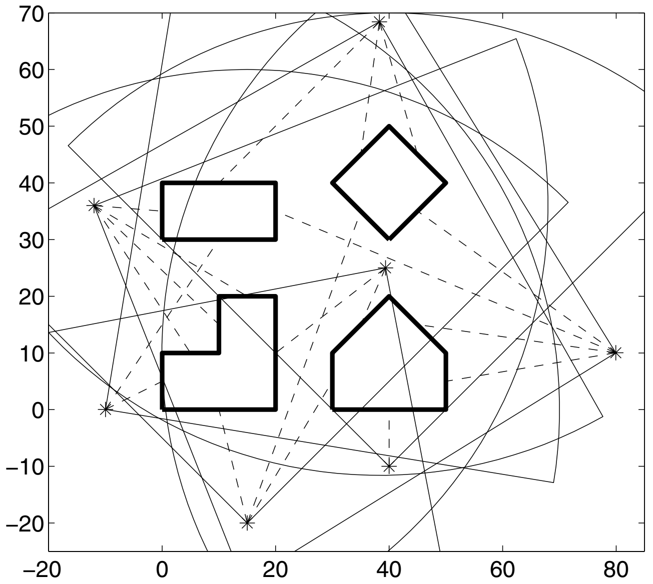


Fig. 10. A complex scenario with 19 walls to be guarded. The solution requires seven guards to guard all walls while satisfying occlusion, resolution and field of view constraints. As in Figure 7 above, asterisks (*) are guard positions, dashed lines show what walls are guarded by whom, and solid cones illustrate field of view limitations.

5 Concluding Remarks

In this chapter, the problem of positioning a team of UGVs to get a good situational awareness was studied. By extending previous work on terrain guarding we were able to include realistic field of view and resolution constraints into the formulation. The resulting problems were solved using an approximation scheme on a carefully selected set of candidate guard positions. Two examples were solved to illustrate the approach.

References

1. Chvátal, V.: A Combinatorial Theorem in Plane Geometry. Journal of Combinatorial Theory Series B 18, 39–41 (1975)
2. González-Banos, H., Latombe, J.: A Randomized Art-Gallery Algorithm for Sensor Placement. In: Proceedings of the 17th Annual Symposium on Computational Geometry, pp. 232–240 (2001)
3. Amit, Y., Mitchell, J.S.B., Packer, E.: Locating Guards for Visibility Coverage of Polygons. In: Proceedings of the 9th Workshop on Algorithm Engineering and Experiments. Proceedings in Applied Mathematics. SIAM, Philadelphia (2007)
4. Fragoudakis, C., Markou, E., Zachos, S.: How to Place Efficiently Guards and Paintings in an Art Gallery. In: Bozanis, Panayiotis, Houstis, Elias (eds.). Lecture notes in Computer Science: Advances in Informatics, pp. 145–154. Springer, Heidelberg (2005)
5. Ganguli, A., Cortes, J., Bullo, F.: Distributed Deployment of Asynchronous Guards in Art Galleries. In: Proceedings of the 2006 American Control Conference (2006)
6. Urrutia, J.: Art gallery and illumination problems. In: Sack, J.-R., Urrutia, J. (eds.) Handbook of computational geometry, pp. 973–1027. North-Holland Publishing Co., Amsterdam (2000)
7. O'Rourke, J.: Art Gallery Theorems and Algorithms. Oxford University Press, New York (1987)
8. Efrat, A., Har-Peled, S., Mitchell, J.: Approximation Algorithms for Two Optimal Location Problems in Sensor Networks. In: Proceedings of the 14th Annual Fall Workshop on Computational Geometry. MIT, Cambridge (2004)
9. Gerkey, B., Thrun, S., Gordon, G.: Visibility-Based Pursuit-Evasion with Limited Field of View. The International Journal of Robotics Research 25(4), 299–315 (2006)
10. Hoffmann, F., Kaufmann, M., Kriegel, K.: The Art Gallery Theorem for Polygons With Holes. In: Proceedings of the 32nd Annual Symposium on Foundations of Computer Science, pp. 39–48 (1991)
11. Hollinger, G., Kehagias, A., Singh, S.: Probabilistic Strategies for Pursuit in Cluttered Environments with Multiple Robots. In: IEEE International Conference on Robotics and Automation (2007)
12. Hoffmann, F.: The Art Gallery Problem for Rectilinear Polygons With Holes. B 94-22, Freie Universitat Berlin, Tech. Rep. (1994)
13. Eidenbenz, S.: Approximation Algorithms for Terrain Guarding. Information Processing Letters 82(2), 99–105 (2002)
14. Eidenbenz, S.: Optimum Inapproximability Results for Finding Minimum Hidden Guard Sets in Polygons and Terrains. In: Proceedings of the 8th Scandinavian Workshop on Algorithm Theory, pp. 60–68 (2002)

15. Franklin, W., Ray, C.: Higher isn't Necessarily Better: Visibility Algorithms and Experiments. In: Proceedings of the 6th International Symposium on Spatial Data Handling, pp. 751–763 (1994)
16. Franklin, W.: Siting Observers on Terrain. In: Symposium on Spatial Data Handling, Ottawa, pp. 109–120 (2002)
17. Marengoni, M., Draper, B.: System to Place Observers on a Polyhedral Terrain in Polynomial Time. Image and Vision Computing 18(10), 773–780 (2000)
18. Chen, S., Li, Y.: Automatic Sensor Placement for Model-Based Robot Vision. Transactions on Systems, Man and Cybernetics, Part B, IEEE 34(1), 393–408 (2004)
19. Vázquez, P., Feixas, M., Sbert, M., Heidrich, W.: Viewpoint Selection using Viewpoint Entropy. In: Proceedings of the Vision, Modeling and Visualization Conference, pp. 273–280 (2001)
20. Speckmann, B., Tóth, C.: Allocating Vertex p-Guards in Simple Polygons via Pseudo-Triangulations. Discrete and Computational Geometry 33(2), 345–364 (2005)
21. Lee, D., Lin, A.: Computational Complexity of Art Gallery Problems. IEEE Transactions on Information Theory 32(2), 276–282 (1986)
22. Eidenbenz, S.: Inapproximability Results for Guarding Polygons without Holes. In: Proceedings of the 9th International Symposium on Algorithms and Computation, pp. 427–436 (1998)
23. Eidenbenz, S., Widmayer, P.: An Approximation Algorithm for Minimum Convex Cover with Logarithmic Performance Guarantee. SIAM Journal on Computing 32, 654 (2003)
24. Efrat, A., Har-Peled, S.: Guarding Galleries and Terrains. In: Proceedings of the IFIP 17th World Computer Congress-TC1 Stream/2nd IFIP International Conference on Theoretical Computer Science: Foundations of Information Technology in the Era of Networking and Mobile Computing, pp. 181–192 (2002)

The Maximum Weight Independent Set Problem for Data Association in Multiple Hypothesis Tracking

Dimitri J. Papageorgiou and Michael R. Salpukas

The Raytheon Company
Integrated Defense Systems
Woburn, MA 01801
{Dimitri_J_Papageorgiou,Michael_R_Salpukas}@raytheon.com

Abstract. Multitarget tracking (MTT) hinges upon the solution of a data association problem in which observations across scans are partitioned into tracks and false alarms so that accurate estimates of true targets can be recovered. In this chapter, we describe a methodology for solving this data association problem as a maximum weight independent set problem (MWISP). This MWISP approach has been used successfully for almost a decade in fielded sensor systems using a multiple hypothesis tracking (MHT) framework, but has received virtually no attention in the tracking literature, nor has it been recognized as an application in the clique/independent set literature. The primary aim of this chapter is to simultaneously fill these two voids. Second, we show that the MWISP formulation is equivalent to the multidimensional assignment (MAP) formulation, one of the most widely documented approaches for solving the data association problem in MTT. Finally, we offer a qualitative comparison between the MWISP and MAP formulations, while highlighting other important practical issues in data association algorithms that are commonly overlooked by the optimization community.

Keywords: Maximum weight independent set problem, maximum clique problem, multidimensional assignment problem, multiple hypothesis tracking, multitarget tracking.

1 Introduction

Multitarget tracking (MTT) is a vital requirement of surveillance systems employing one or more sensors to collect information from an environment of interest. The goal of an MTT system is to form and maintain tracks on targets of interest using scans of detections furnished by the sensors. Two essential elements of an MTT algorithm include (1) data association, whereby scans of data are partitioned into sequences of detections, or tracks, as well as false alarms, and (2) estimation/prediction, in which quantities such as the current and future position, velocity, and acceleration of each target are computed. The primary focus

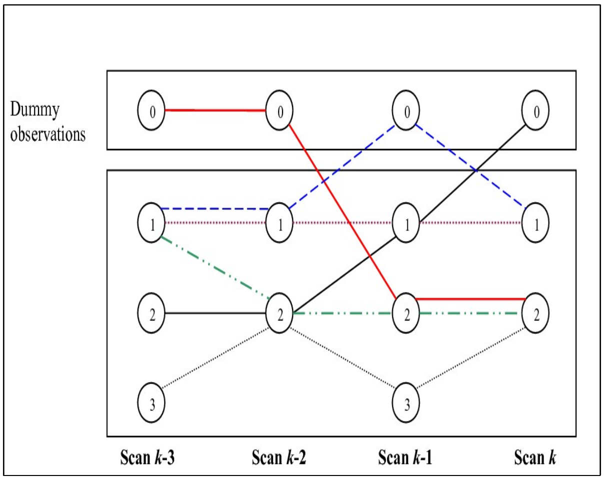


Fig. 1. Actual and "dummy" observations per scan

of this chapter is on the former issue as it arises in multiple hypothesis tracking, widely regarded as the preferred method for structuring the data association problem in modern MTT systems [6].

We assume that a surveillance system, comprised of one or more sensors, collects observations on an area of interest. An *observation*[1] refers to all the observed (or measured) quantities, e.g., kinematic data, included in a detection output from a sensor. A *scan* or *data frame* refers to any set of input observations that are produced at the same time, or within the same time interval, with the stipulation that at most one observation per target appears in a given scan. We consider S scans of observations, denoted $Z(k)$, each with m_k observations (or measurements), for $k = 1, \ldots, S$. Letting z_{i}^{k} denote the i_kth observation on scan k, we have $Z(k) = \left\{z_{i}^{k}\right\}_{i\ =1}^{m}$. For notational convenience in defining false alarms as well as tracks with missed detections, we include a so-called *dummy* observation, denoted by z_0^k, in each scan of observations $Z(k)$. We define a *track hypothesis*, or simply, a *track*, as a sequence of (actual and/or dummy) observations $Z_{i_1 i_2 \cdots i} \equiv \left\{z_{i_1}^1, \ldots, z_{i}^S\right\}$. For example, in Figure 1, assuming we have collected only $k = 4$ scans of observations, $Z_{1,1,0,1} = \left\{z_1^1, z_1^2, z_0^3, z_1^4\right\}$ represents a track of three actual observations and one missed observation (a dummy observation) on scan 3. The sequence $Z_{0,0,2,2}$ represents a track that initialized on scan 3 because the first of its two non-dummy observations occurred on scan 3.

The term *data association* in MTT refers to the manner in which observations are assigned to existing or newly initiated tracks, or are otherwise designated as spurious observations, a.k.a. false alarms. There are several ways to classify data association approaches, but the two most common are based on (1) the

[1] The terms observation, detection, measurement, report, and plot are used interchangeably in the literature.

number of observations permitted to be assigned to a track ("one-to-one" versus "many-to-one" data association) and (2) the number of data frames used in the association process ("sequential" versus "deferred" decision logic) [5]. These classes are not orthogonal, e.g., there exist one-to-one single frame and one-to-one multi-frame approaches. In many legacy and, in fact, many present day MTT systems, data association is performed sequentially, whereby observations from a scan are "permanently" assigned[2] to tracks or designated as false alarms prior to processing the subsequent scan of observations. Notable one-to-one sequential approaches include individual nearest neighbor and global nearest neighbor (sequential most probable hypothesis tracking), which relies on a two-dimensional linear assignment algorithm [8]. The fundamental deficiency with these approaches occurs when many tracks compete for the same observations in high contention scenarios resulting in an unacceptable number of misassociations, and, ultimately, poorly formed (or "noisy") tracks. Well known many-to-one sequential techniques include probabilistic data association (PDA) and joint PDA (JPDA) [1].

The most prominent (and widely implemented) deferred decision logic method is known as *multiple hypothesis tracking* (MHT), or more specifically, "track-oriented" MHT (TOMHT), which frequently involves the generation of multiple track hypotheses corresponding to the same true target. The underlying premise of the MHT framework is that by retaining more than one track per target ambiguities that arise in observation-to-track association (for example, due to crossing tracks or splitting tracks) have a better chance of being resolved as more scans of observations are collected. At the same time, storing multiple track hypotheses comes at a price, namely, increased processing, memory, and computation. Maintaining high quality tracking performance, while keeping computational costs low, is an essential requirement of any MHT algorithm, and is arguably as much of an art as it is a science. The MHT framework discussed in this chapter has been proven to work for both surface (i.e., ground and sea) and air tracking [7]. It has been used for almost a decade on several single sensor and multiple sensor programs [13]. For multisensor scenarios, each sensor reports its observations to a centralized tracker and biases in the observations are removed (to the extent possible) so that the tracker treats each scan of observations just as it would for a single sensor. If two sensors report observations at the same time, these scans are ordered according to some rule and the process continues as normal.

The first MHT framework was proposed by Reid [26] as a fully Bayesian technique for evaluating the probabilities of sequences of observations having

[2] Many authors describe the observation-to-track association used in a global nearest neighbor approach as being an "irrevocable" assignment, but this is not strictly followed in practice. Designers of tracking algorithms are well aware of the shortcomings of single hypothesis tracking in difficult scenarios and often include ad hoc logic to re-assign observations by re-processing the last several scans of observations if existing tracks do not appear to be well formed. However, since the rules and criteria for reforming tracks are more scenario- and application-dependent, this approach is rarely mentioned in the literature.

originated from targets in a surveillance region. This approach, known as "hypothesis-oriented" MHT (HOMHT) because it directly computes probabilities of global (joint) observation-to-target association hypotheses, is seldom implemented due to the complexity of treating joint hypotheses. On the other hand, the TOMHT approach has gained widespread acceptance due to the simplicity of working with (i.e., propagating, updating, and pruning) individual tracks and the availability of efficient, albeit suboptimal, algorithms for partitioning observations into tracks and false alarms [16,6,3].

The philosophy of partitioning observations into tracks and false alarms was first formalized by Morefield [17] in which he formulated a 0-1 integer programming problem, specifically a maximum weighted set packing problem, to ensure that, on each scan of an N-scan window, an observation is assigned to no more than one track. Capponi [10] took a closely related, but slightly modified, approach in which he formulated and solved a maximum weighted set partitioning problem. Today, the most well documented approach is to cast the partitioning problem as a multidimensional assignment problem (MAP) over an $(N+1)$-scan sliding window. The MAP formulation and its various solution methods have received extensive attention in the literature, see, e.g., Bar-Shalom [2], Deb et al. [12], Pasiliao et al. [19], Pattipati et al. [20], Poore [21,22], and Popp et al. [24]. To the mathematical programming community, the MAP should be understood as nothing more than an integer programming problem in which we wish to find the most likely set of tracks (based on a cost minimization to be discussed below) subject to the constraint that, on every scan, each actual observation be assigned to no more than one track or otherwise be designated as a false alarm.[3]

1.1 Contributions of This Chapter

In this chapter, we describe an alternative formulation to the data association problem in which observation partitioning in an $(N + 1)$-scan sliding window is cast as a maximum weight independent set problem. Unlike Morefield's set packing approach and the various MAP approaches in which observations in an $(N+1)$-scan window are *explicitly* partitioned into tracks and false alarms (i.e., by explicitly imposing a constraint on each observation), the MWISP approach *implicitly* partitions the observations. We believe that this contribution is significant for several reasons. First, the MWISP is a renowned discrete optimization problem that has received far more attention in the optimization literature than the MAP. More importantly, given the pervasiveness of the MWISP (and its weighted and unweighted clique and vertex packing counterparts discussed below) in numerous scientific applications completely unrelated to multitarget tracking, literally hundreds of algorithms, both exact and heuristic, have been developed, tested, and refined to solve it [4]. Second, because this approach has been proven to work in operational MHT software using non-state-of-the-art MWISP solvers, we believe that recognition of this alternative formulation could

[3] It should be noted that the MAP can be made even more general by allowing observations to be assigned to more than one track, see, e.g., Poore [21].

open the floodgates to even more efficient MHT algorithms with superior performance. To the best of our knowledge, this chapter is the first to formally recognize the MWISP formulation as a legitimate mathematical programming formulation of the observation-to-track assignment problem in an MHT framework.

As a second contribution, we establish that solving the MWISP is equivalent to solving the MAP over the same $(N+1)$-scan window. By equivalence, we mean that a solution to the MWISP is optimal if and only if it is optimal to the corresponding MAP. This fact should not be too surprising given that both implementations have been studied and used for years. The equivalence between the maximum weighted set packing and MWISP formulations is (informally) shown in Chapter 7.2 of [8] for a tracking audience, but has been known in the optimization literature for decades (see, e.g., Section 1.3 of [9]).

2 The Maximum Weight Independent Set Problem

Before describing the MHT framework, we take a brief, but necessary, tangent to present the maximum weight independent set problem (MWISP). The MWISP is an eminent combinatorial optimization problem, which has received widespread attention in the optimization literature. In this section, we formally introduce the MWISP and provide its linear integer programming formulation. For a comprehensive survey of various formulations, complexity results, algorithms, and applications, we refer the reader to Bomze et al. [4] and their extensive list of over 300 references.

We start by defining an independent set and a clique. Let $G = (V, E)$ be an arbitrary undirected graph with vertex set $V = \{1, \ldots, n\}$ and edge set E consisting of m edges. Vertices i and j are said to be adjacent if the graph contains the edge (i, j). An *independent set* (also known as a *stable set* or *vertex packing*) is any subset S of vertices whose elements are pairwise nonadjacent. The *maximum* (*cardinality* or *unweighted*) *independent set problem* (MISP) is that of finding a set of independent vertices with the largest cardinality. If each vertex also has an associated positive weight w_i, then the *maximum weight(ed) independent set problem* (MWISP) is a related problem in which we wish to find a set of independent vertices with the largest total weight. Finding a maximum weighted or unweighted independent set in an arbitrary graph is $\mathcal{NP}$-hard and the associated decision problem is $\mathcal{NP}$-complete (see, for example, [14]). As will soon become apparent, the observation partitioning problem encountered in our MHT framework arises naturally as a MWISP in which vertices correspond to tracks and edges correspond to incompatibilities between tracks.

A related problem of interest that has received even more attention in the literature than the MISP is the *maximum* (*cardinality* or *unweighted*) *clique problem* (MCP). A *clique* in a graph is a set of vertices that are pairwise adjacent. Thus, the maximum (cardinality) clique problem is that of finding the largest subset of vertices in a graph that share an edge with every other vertex in the set. The maximum weight clique problem (MWCP) arises when each vertex also has an associated positive weight w_i. Solving the MCP/MWCP is equivalent to solving the MISP/MWISP on the complement graph of G, denoted $\bar{G}$, where $\bar{E} = (i,j) \in V$, $i \neq j$ and $(i,j) \notin E$. The key point is that all algorithms developed for the MCP/MWCP are applicable to the MISP/MWISP, and vice versa, depending on whether we operate on the graph G or its complement $\bar{G}$.

The MWISP has a number of equivalent mathematical programming formulations as a linear integer programming (IP) problem or as a continuous nonconvex optimization problem. We mention the most common of the IP formulations, which is known as the *edge formulation*:

$$\begin{aligned} \max_{\mathbf{x}} \; & \sum_{i=1}^{n} w_i x_i \\ \text{s.t.} \; & x_i + x_j \leq 1, \forall (i,j) \in E \\ & x_i \in \{0,1\}, \text{for } i = 1,\ldots,n \end{aligned}$$

where the decision variable x_i takes value one if vertex i belongs to the independent set, and zero otherwise.

3 Track Scoring

In order to compare various partitions of the observations, most commonly known as *global hypotheses*, a maximum a posteriori probabilistic expression is customarily used [3]. This framework has received extensive attention in the literature, so we only highlight the essential points that surface in subsequent sections.

Associated with each track is a *track score*. Track scores serve as cost coefficients in the various discrete optimization formulations used to rank global hypotheses. The track score is simply the log likelihood ratio that the sequence of observations that form this track originated from a true target versus from false alarms. In particular, Sittler's [27] original likelihood ratio is a straightforward application of Bayes' rule:

$$LR = \frac{P\left\{z_{i_1}^1,\ldots,z_i^S \,|H_1\right\} P_0\left\{H_1\right\}}{P\left\{z_{i_1}^1,\ldots,z_i^S \,|H_0\right\} P_0\left\{H_0\right\}}$$

where H_1 (H_0) is the hypothesis that the sequence of observations originated from a single true target (false alarms); $P\left\{z_{i_1}^1,\ldots,z_i^S \,|H_i\right\}$ is the probability density function of collecting the sequence of observations $\left\{z_{i_1}^1,\ldots,z_i^S\right\}$ under the hypothesis H_i; and $P_0\left\{H_i\right\}$ is the a priori probability of hypothesis H_i, namely, the believed density of true targets in the area of interest for H_1 and that of false alarms for H_0. As MTT is an on-going process, track scores are computed recursively from scan to scan as follows. The track score at scan k is $LLR(k) = LLR(k-1) + \Delta LLR(k)$, where

$$\Delta LLR(k) = \begin{cases} \log(1-P_D) & \text{if no update on scan } k, \\ \Delta L_u(k) & \text{if track update on scan } k. \end{cases}$$

As one might expect, the track score decreases when no observation is associated with a track on scan k since the probability of detection P_D is always less than unity, hence $\log(1 - P_D) < 0$. On the other hand, the track score typically increases by the amount $\Delta L_u(k)$ when an observation is used to update the track on scan k. The magnitude of the increase depends on a number of factors including (1) the residual error between the actual observation and the predicted observation, (2) the error (covariance matrix) in the track state, (3) the expected probability density of false alarms, (4) P_D, and, possibly, (5) signal intensity (e.g., signal-to-noise ratio). See Chapter 6 of [8] for various formulae concerning the score increment $\Delta L_u(k)$. The track score is the sole component needed to determine the validity of a track. One typically prunes a track once its track score falls below a given threshold.

Having defined a score for each track, one can determine the score of a global hypothesis by summing the scores of the tracks in that hypothesis.[4] From these global hypothesis scores, we can compute global hypothesis probabilities as follows: As described in [BlP99], let T_i denote track i, L_T its score, and L_H the score of a given global hypothesis H_j, i.e., $L_H = \sum_{T \in H} L_T$. Suppose that it is possible to enumerate all global hypotheses. Then, the probability $P\{H_j\}$ of global hypothesis j can be computed using all J hypotheses[5]:

$$P\{H_j\} = \frac{\exp(L_H)}{1 + \sum_{i=1}^{J} \exp(L_H)}. \tag{1}$$

Furthermore, the probability of a particular track T_i, sometimes referred to as a *global track probability*, not to be confused with the probability of track validity defined above, is the sum of the probabilities of all hypotheses that contain the track:

$$P\{T_i\} = \sum_{j \in \{1,\ldots,J | T \in H\}} P\{H_j\}. \tag{2}$$

This global track probability could then be used for track deletion.

4 An MHT Framework

In this section, we outline a track-oriented MHT framework in which the observation-to-track association problem for determining the best global hypothesis and track probabilities is formulated as a MWISP. Reviewing this MHT framework is helpful for several reasons. First, as there is a possible exponential explosion in the number of track hypotheses formed in any MHT framework,

[4] The ability to simply sum the scores of the tracks in a global hypothesis relies on some essential independence assumptions that are commonly made, see, e.g. [21] Equations (2.7). Another standard assumption, which we follow, is that false alarms have a score of zero in any partition.

[5] Along with all J nontrivial global hypotheses, the denominator includes the term $e^0 = 1$ to account for one trivial hypothesis – that all observations are false alarms, which has a score of zero.

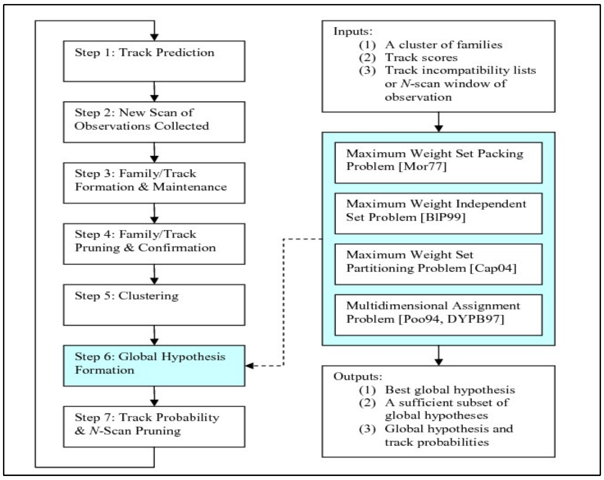

Step 1: Track Prediction
Step 2: New Scan of Observations Collected
Step 3: Family/Track Formation & Maintenance
Step 4: Family/Track Pruning & Confirmation
Step 5: Clustering
Step 6: Global Hypothesis Formation
Step 7: Track Probability & N-Scan Pruning
Inputs:
(1) A cluster of families
(2) Track scores
(3) Track incompatibility lists or N-scan window of observation
Maximum Weight Set Packing Problem [Mor77]
Maximum Weight Independent Set Problem [BIP99]
Maximum Weight Set Partitioning Problem [Cap04]
Multidimensional Assignment Problem [Poo94, DYPB97]
Outputs:
(1) Best global hypothesis
(2) A sufficient subset of global hypotheses
(3) Global hypothesis and track probabilities

Fig. 2. A high-level flowchart of a track-oriented MHT framework is shown on the left. The global hypothesis formation step can be solved using any one of the optimization formulations on the right.

real-time systems must have mechanisms for limiting the number of tracks to ease processing. Track formation, clustering, and pruning are three such mechanisms discussed below. Second, the flow of the algorithm is important because, without it, the reader familiar with the MHT framework of Poore [21,22] may not understand how the MWISP formulation arises in our MHT algorithm and how track incompatibility lists are generated and maintained. The following description parallels that given by Blackman in Chapter 16 of [8] and [6], yet is minimalist in that we avoid discussion of the numerous extensions that could be added to a TOMHT framework. Throughout, it will be helpful to refer to the flow diagram of the algorithm provided in Figure 2.

We begin by describing a hierarchy of data structures that are often maintained in MHT algorithms to encapsulate related information, to improve computational efficiency, and to aid in creating a seamless interface with the user. These data structures, shown in Figure 3, are clusters, families, tracks, and observations. Tracks and observations were introduced in the opening section, but we repeat here to highlight this hierarchy that a track is composed of (in fact, is a sequence of) observations. In a similar manner, a family (of tracks) is composed of multiple tracks corresponding to the same target. We define a *family*, sometimes called a *track family*, as a set of tracks all emanating from a single root observation. In TOMHT, multiple track hypotheses are constructed for each postulated target based on the target's dynamic model and the association of observations. For each postulated target, it is convenient both conceptually and computationally to create a family (of tracks) or what Kurien [16] refers to as a *target tree*. Initially, the root observation of a family denotes the first observation associated with a new postulated target. Branches within a family

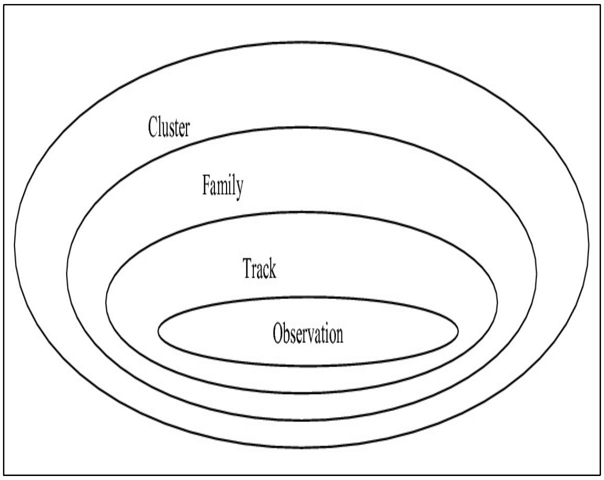


Fig. 3. Hierarchy of data structures maintained in an MHT algorithm

represent various tracks for this target and are formed as different observation-to-track assignments are hypothesized. As an example using the tracks from Figure 1, suppose that $k = 4$ and that tracks $Z_{1,1,0,1}$, $Z_{1,1,1,1}$, and $Z_{1,2,2,2}$ represent three tracks associated with the same postulated target, i.e., three branches in a family whose root observation is z_1^1. The target tree depiction of this family is shown in Figure 4(a).

In practice, a family data structure stores a list (not a tree) of tracks in the family with pointers to each of these tracks, a common root observation shared by all tracks in the track list, the status and score of the highest scoring track, and various other data. The status of a family is useful to the user and typically includes at least three categories: 'tentative,' 'confirmed,' and 'lost.' Whereas a 'tentative' track is typically a track with only a handful of observations and a low track score, a 'confirmed' track is generally considered to be a track with a sufficiently long history of observations, a high track score, and that has been found to be in at least one of the best global hypothesis formed thus far in the algorithm. The root observation is typically updated with N-scan pruning, which we describe below. Finally, clusters are composed of one or more families and are an effective data structure for decomposing the observation-to-track association problem into smaller independent problems. Intuitively, when targets are sufficiently separated such that their observations can be grouped into individual clusters, one can perform the steps in Figure 2 independently on each cluster.

All MHT algorithms face two diametrically opposed objectives: storing as few tracks as possible to keep computational efficiency high, while simultaneously retaining multiple observation-to-track association hypotheses for each postulated target until enough data is available to make an irrevocable decision. Ideally, we would like an algorithm that could simultaneously process all scans of data and produce an optimal partition of the entire data set. This approach, however, is not practical because the computational requirements grow at an exponential rate. One almost universally accepted technique for mitigating the combinatorial explosion in track hypotheses is to apply an approximation technique known as *N-scan pruning* to track families [16].

There are at least two approaches to N-scan pruning and the difference lies in the definition of the parameter N. Suppose that we have collected observations up to and including scan k, updated existing tracks with these observations, and initiated new tracks. In one approach – the one we adopt in this chapter – the assignment of observations before scan $k - N$, for $k > N$, is fixed for the remainder of the algorithm while the observations in the $N+1$ scans $k-N, \ldots, k$ are not yet permanently assigned. Upon finding the best global hypothesis, we obtain a set of families each with a single track in this optimal hypothesis. Note that some families may not have a track in the best global hypothesis. For each track in this hypothesis, we trace the track's sequence of observations from scan k back to scan $k - N + 1$ (i.e., counting scan k, we trace the track back a total of N scans). If the track (and, hence, the family) had been initiated before scan $k - N + 1$, we reset the root observation of the family to the observation $z_{i_{\;-\;+1}}^{k-N+1}$ in this track's sequence of observations and delete other branches of the prior root observation in the family. Otherwise, the family was initiated within the N-scan window $\{k - N + 1, \ldots, k\}$ and the root observation need not be updated. For families not in the best global hypothesis, the rules for N-scan pruning are typically application dependent; in this chapter, we advocate applying N-scan pruning to these families as well using the highest score track in the family. Thus, the two consequences of N-scan pruning are that observations on scan $k - N$ become permanently assigned to tracks (or are designated as false alarms) for the remainder of the algorithm and track branches are deleted.

An example of this type of N-scan pruning for a family of three tracks – $Z_{1,1,0,1}$, $Z_{1,1,1,1}$, and $Z_{1,2,2,2}$ from Figure 1 – with root observation z_1^1 is shown in Figure 4. In this example, suppose track $Z_{1,1,1,1}$ is found to be in the best global hypothesis and that an N-scan pruning technique, with $N = 3$, is used. Then, we trace track $Z_{1,1,1,1}$ back 3 scans to its observation on scan 2, which, in this case, is z_1^2, reset the root observation of the family to z_1^2, and delete the track/branch $Z_{1,2,2,2}$.

In the variant of N-scan pruning given above, the parameter N refers to the number of *scans*, counting the current scan, that we trace back to find a new root observation. Another common variant is to define N as the number of actual *observations* we trace back to find this new root observation. This variant is popular when there is a high probability of not receiving an observation for a particular target on every scan. For example, if three sensors are passing observations to a centralized MHT algorithm, it may be the case that only one or two of the sensors report observations on a particular subset of targets at a given time. The choice of N is highly dependent on the tracking application and the throughput limitations of the system. In addition, the value of N can be set adaptively depending on the processing lag in the system.

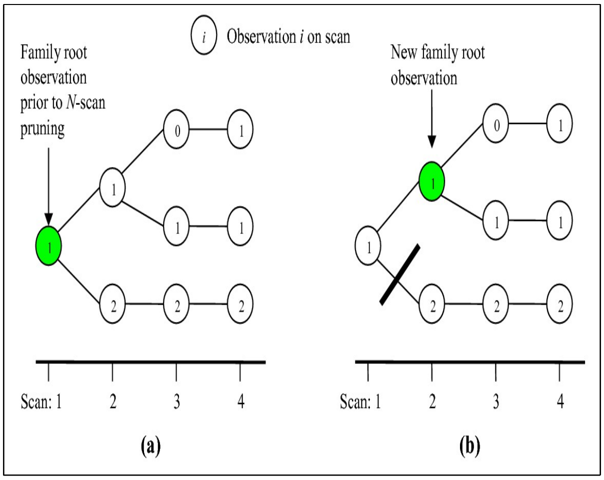

Observation *i* on scan
Family root observation prior to *N*-scan pruning
New family root observation
Scan: 1
2
3
4
(a)
(b)

Fig. 4. An example of a family's root observation and branches before and after N-scan pruning

Having established the basic framework of the algorithm, we now discuss the particular steps in the algorithm.

4.1 Formation and Maintenance of Tracks and Families

Let τ_k and $\left\{T_j^k\right\}_{j\ =1}^{\tau}$ be the number and set, respectively, of existing tracks just prior to processing observations from scan $k+1$ in step 3 of Figure 2, where the j_kth such track is $T_j^k = \left\{z_{i_1}^1, \ldots, z_i^k\right\}$. By "existing track," we mean a track that the algorithm has kept in memory for further processing. Similarly, let ϕ_k and $\left\{F_l^k\right\}_{l\ =1}^{\phi}$ be the number and set, respectively, of families at this stage of the algorithm. There are three types of tracks which emerge after a new scan of data is processed. First, all existing tracks are extrapolated without an observation to the time of the next scan to account for the possibility that no observation was collected for that track, i.e., T_j^k becomes $T_{j\ +1}^{k+1} = \left\{z_{i_1}^1, \ldots, z_i^k, z_0^{k+1}\right\}$. Second, for each existing track T_j^k at scan k, we create a new (child) track for each observation $z_{i\ +1}^{k+1}$ that "gates" with T_j^k, i.e., is reasonably close to where the predicted observation would fall. These tracks represent new branches in the family. Finally, each observation $z_{i\ +1}^{k+1}$ forms a newly initiated track and family, since it could potentially be a new target. In total, the set of tracks present immediately after processing $Z(k+1)$ is

$$\left\{T_{j\ +1}^{k+1}\right\} = \{\text{extrapolated tracks}\} \cup \{\text{updated tracks}\} \cup \{\text{newly initiated tracks}\}.$$

Thus, subsequent to "processing" observations from scan $k+1$ in step 3 of Figure 2, we could have as many as $\tau_{k+1} = \tau_k + \tau_k m_k + m_k$ tracks and $\phi_{k+1} = \phi_k + m_k$

families in memory. For practical implementations, the number of tracks and families maintained in storage must be kept below some threshold in order to meet computational time requirements. Methods to do so are discussed in [8].

For our purposes, two tracks are said to be *compatible* if they do not share any observations, i.e., if no non-dummy observation appears in more than one sequence. (In the event of possibly unresolved targets, that is, when two or more closely-spaced targets generate a single observation on a given scan, two tracks may be permitted to share an observation, but this issue lies outside the scope of this work. Daum [Dau94] describes the significance of this largely neglected issue.) Similarly, a *global (joint) hypothesis* is defined as a collection of tracks that do not share any actual observations.

From the above description, it should be clear that many tracks may share an observation and will, therefore, be incompatible with one another based on the typical assumption that one observation can be associated with at most one track. Consequently, unlike other MHT algorithms in the literature, we maintain an incompatibility list (ICL) for each track, which will later be used for clustering and for solving a MWISP to identify the most likely set of (compatible) tracks. An ICL can be either explicit or implicit. Whereas an explicit ICL contains all existing tracks with which a given track is incompatible, an implicit ICL contains a subset of these tracks with the understanding that a given track is incompatible with other tracks (e.g., by definition, a track is incompatible with every other track in its family). For sake of presentation, we assume an explicit ICL is maintained.

A naive but straightforward procedure for updating track ICLs after gating observations with tracks on scan k can be done in $O\left(m_k \tau_{k-1}^2\right)$ time, where τ_{k-1} is the number of existing tracks upon entering the "Track Formation" step in Figure 2. The procedure works as follows: For each observation z_i^k , for $i_k = 1, \ldots, m_k$, consider all pairwise comparisons of tracks that were updated with z_i^k , and update each track's ICL to reflect this incompatibility.

4.2 Track-Level Pruning and Confirmation

As mentioned in Section 3, the score associated with a track is the log likelihood ratio of the probability that the track was generated from true target returns to the probability that all observations are false alarms. This score represents the probability that a track is a valid target. The track-level pruning process simply compares this track-level probability to a suitably chosen deletion threshold. Additional application specific pruning, such as kinematic pruning of targets that are going too fast for the target type, can also be applied at this time. The tracks that fail this test are deleted and the surviving tracks are tested for confirmation. Once all tracks in a family have been pruned, the family is pruned.

The rules for confirming a track family must balance the requirement to minimize false track confirmation versus the requirement to minimize track reporting time. False track confirmation can cause confusion to the display operator, increased sensor loading, and added processor burden. Delayed track confirmation can cause engagement delay and, if the system needs to aim the sensor at confirmed targets in a timely manner, missed detections. Common rules for track confirmation are based on score, probability, and number of detection thresholds.

4.3 Track Clustering

Clustering is an important preprocessing step used for decomposing the set of families into smaller disjoint subsets to improve computational efficiency since the processing within each cluster can be done independently from other clusters. A cluster is composed of families whose tracks share at least one observation. A cluster can include tracks that do not share observations directly. For example, if track 1 is incompatible with track 2 and track 2 is incompatible with track 3, then all three tracks are in the same cluster. Clusters can be determined from scratch on each scan by applying a breadth-first or a depth-first search over the ICLs of all tracks. For real-time applications, clustering is not performed from scratch at every scan; rather, a more efficient procedure is used (see, e.g., [16]).

4.4 Global Hypothesis Formation

At this point in the algorithm, we have a set of families $\left\{F_l^k\right\}_{l=1}^{\phi}$ and a set of tracks $\left\{T_j^k\right\}_{j=1}^{\tau}$, many of which may be incompatible with one another, and we wish to determine the most likely global (joint) hypothesis, i.e., collection of tracks that do not share any observations. The motivation for computing the best global hypothesis is twofold. First, the best global hypothesis represents the most likely set of tracks given the data received, which is useful information for the user (e.g., a human operator, a higher-level entity in the system, or another algorithm). Second, the best global hypothesis is used when performing N-scan pruning. The global hypothesis formation step also serves another important purpose in our MHT framework, which is often not performed in other (e.g., Poore's [22]) MHT algorithms. Namely, we attempt to enumerate a sufficient subset of the global hypotheses with the aim of computing approximate global hypothesis probabilities and track probabilities as shown in Equations (1) and (2), which, in turn, will be used for further track pruning; tracks with an approximate probability below some threshold are pruned. This method of probability pruning has also been used in speech recognition [15].

The global hypothesis formation problem can be formulated as any one of the combinatorial optimization problems listed in the box on the right of Figure 2. Below, we describe how the MWISP and MAP formulations arise naturally as mathematical representations of this problem and then prove their equivalence.

Global Hypothesis Formation Formulated as a MWISP. We begin with the MWISP formulation. As noted above, we assume that a set of tracks $\left\{T_j^k\right\}_{j=1}^{\tau}$ exists on scan k just prior to entering the global hypothesis formation

Table 1. Pertinent data for tracks in Figure 1

Track ID	Observation Sequence	ICL	Score
1	$Z_{0,0,2,2}$	{4,6}	-2.3
2	$Z_{1,1,0,1}$	{3,4}	3.4
3	$Z_{1,1,1,1}$	{2,4,5}	9.1
4	$Z_{1,2,2,2}$	{1,2,3,5,6}	7.5
5	$Z_{2,2,1,0}$	{3,4,6}	4.8
6	$Z_{3,2,3,2}$	{1,4,5}	10.1

step in Figure 2 and that clusters have been formed. For each cluster, we create an undirected graph G with vertex set $V = \{1, \dots, \tau'_k\}$ and edge set E, where τ'_k is the number of tracks with a positive score in that cluster. Each vertex corresponds to a track and has a weight w_i equal to that track's score. An edge (i, j) is present in E if track/vertex i is incompatible with track/vertex j and note that these incompatibilities are available with a single lookup through a track's ICL. This is precisely the description of the MWISP given in Section 2. Solving this MWISP to optimality yields the best global hypothesis for this cluster, and will be used as output for the user and for N-scan pruning. As an example, let $N = 3$ and consider the six tracks shown in Figure 1. Table 1 includes the pertinent data required for creating an instance of the MWISP.

Using this data, the MWISP created to perform global hypothesis formation is shown in Figure 5. Although there are six tracks, only five are included in the MWISP because only positive score tracks will contribute to the best global hypothesis, which in this case consists of tracks (track IDs) 3 and 6.

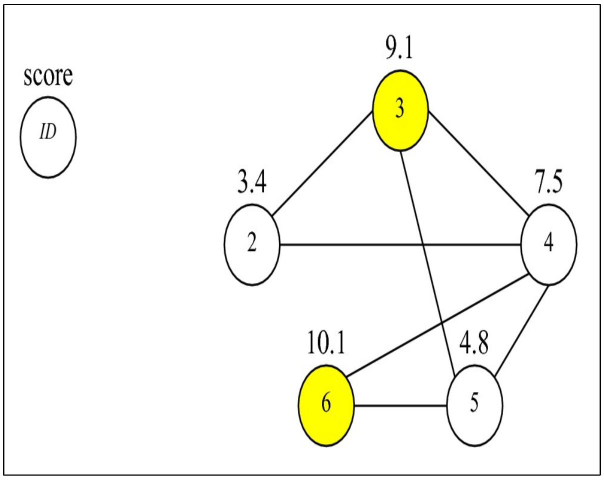


Fig. 5. The undirected graph associated with the MWISP for the tracks in Table 1

Note that all observations in the $(N+1)$-scan window not assigned to a track in a given global hypothesis are implicitly assumed to be false alarms in that hypothesis. In order to compute track probabilities, we also enumerate a sufficient subset of global hypotheses, which correspond to feasible solutions (i.e., suboptimal independent sets) for this instance of the MWISP. Let H denote the set of global hypotheses (independent sets) formed. Since we would like to compute track probabilities for all tracks in the cluster, we loop over all global hypotheses in H and attempt to create new hypotheses by including tracks with negative scores. This results in a larger set H' of global hypotheses/independent sets from which we will compute approximate global hypothesis probabilities and, subsequently, track probabilities.

The MWISP instances encountered in MTT, in general, do not possess any special structure for which polynomial-time solution methods exist. This fact should not be too surprising given the discussion and the proof to follow that shows the equivalence between the MWISP and the MAP, which is known to be $\mathcal{NP}$-hard.

Global Hypothesis Formation Formulated as a MAP. We now describe how the MAP can be used in lieu of the MWISP to find the best global hypothesis in our MHT framework. Note that the MAP formulation that arises in our MHT framework differs from the variants in Poore's framework [21] in which only a subset of the tracks are considered when performing track continuation and initiation. In addition, our description differs from that given in [8] as we emphasize the family data structure as one should use in implementation; conceptually, the descriptions given here and in [8] are the same.

Recall that clusters are composed of one or more families and families are composed of a set of incompatible tracks, all of which represent a track hypothesis for the same target. In order for the MAP formulation to be applicable, we must impose one further restriction than in the MWISP formulation: each family must have a root observation (dummy or actual) in some scan $s \in \{k-N,\ldots,k\}$ of the $(N+1)$-sliding window. A family whose root observation occurs on scan $k-N$ corresponds to a family that has existed for more than N scans, while families whose root observation occurs on some scan $s \in \{k-N+1,\ldots,k\}$ was initialized within the last N scans.

Our goal is to assign the actual observations, which have not yet been permanently assigned, over the $N+1$ scans $k-N,\ldots,k$ to existing tracks or otherwise designate them as false alarms. After which, we will perform N-scan pruning, fix the assignment of observations on scan $k-N$, and proceed to the next step of the algorithm. To avoid notational clutter, we re-index the $N+1$ scans $1,\ldots,N+1$. For a given cluster, suppose there are m_0 families with at least one positive score track. We index these families $i_0 = 1,\ldots,m_0$ so that the $(N+2)$-tuple $\left\{F^k_{i_0}, z^1_{i_1},\ldots,z^{N+1}_{i_{\;+1}}\right\}$ denotes a positive score track in family $F^k_{i_0}$ composed of the sequence of actual or dummy observations $\left\{z^1_{i_1},\ldots,z^{N+1}_{i_{\;+1}}\right\}$ in the $(N+1)$-scan window. We include a dummy family $i_0 = 0$ to account for false alarms that appear in this window. Notice that we allow two or more families to have the same root observation.[6] For each positive score track in family $F^k_{i_0}$, for $i_0 = 1, \ldots, m_0$, we define a binary decision variable $z_{i_0 i_1 \cdots i \ \ +1}$ such that

$$z_{i_0 i_1 \cdots i \ \ +1} = \begin{cases} 1 \text{ if track } \left\{F^k_{i_0}, z^1_{i_1}, \ldots, z^{N+1}_{i \ \ +1}\right\} \text{ is in best global hypothesis} \\ 0 \text{ otherwise} \end{cases}$$

and cost coefficients $c_{i_0 i_1 \cdots i \ \ +1}$ equal to the negative score of this track. Following standard practice, we set $c_{0 \cdots 0} = 0$. We can now state the integer linear programming formulation of an $(N+2)$-dimensional assignment problem, which yields an optimal partitioning of the observations in this $(N+1)$-scan window:

$$\min_z \sum_{i_0=0}^{m_0} \sum_{i_1=0}^{m_1} \cdots \sum_{i \ \ +1=0}^{m \ \ +1} c_{i_0 \cdots i \ \ +1} z_{i_0 \cdots i \ \ +1} \tag{4a}$$

$$\text{s.t.} \sum_{i_1=0}^{m_1} \cdots \sum_{i \ \ +1=0}^{m \ \ +1} z_{i_0 \cdots i \ \ +1} = 1, \text{ for } i_0 = 1, \ldots, m_0 \tag{4b}$$

$$\sum_{i_0=0}^{m_0} \cdots \sum_{i \ \ -1=0}^{m \ \ -1} \sum_{i \ \ +1=0}^{m \ \ +1} \cdots \sum_{i \ \ +1=0}^{m \ \ +1} z_{i_0 \cdots i \ \ +1} = 1,$$
$$\text{for } i_j = 1, \ldots, m_j \text{ and for } j = 1, \ldots, N \tag{4c}$$

$$\sum_{i_0=0}^{m_0} \cdots \sum_{i \ \ =0}^{m} z_{i_0 i_1 \cdots i \ \ +1} = 1, \text{ for } i_{N+1} = 1, \ldots, m_{N+1} \tag{4d}$$

$$z_{i_0 i_1 \cdots i \ \ +1} \in \{0, 1\}, \forall i_0, \ldots, i_{N+1} \tag{4e}$$

The constraints (4b) – (4d) can be interpreted as follows: Every actual observation must be assigned to exactly one track, where a track in an N-scan pruning framework is an $(N+2)$-tuple of the form $\left\{F^k_{i_0}, z^1_{i_1}, \ldots, z^{N+1}_{i \ \ +1}\right\}$. In order for the above MAP to be feasible, we must also include decision variables $z_{0 \cdots 0 i \ \ 0 \cdots 0}$ with cost coefficients $c_{0 \cdots 0 i \ \ 0 \cdots 0} = 0$, for $i_j = 1, \ldots, m_j$, for $j = 1, \ldots, N+1$, to correspond to observations that are designated as false alarms, as well as decision variables $z_{i_0 0 \cdots 0}$ with cost coefficients $c_{i_0 0 \cdots 0} = 0$, for $i_0 = 1, \ldots, m_0$, to correspond to families that are not updated in this window. As a final note, the summations need not include all m_k observations, for $k = 1, \ldots, N+1$, but rather only those observations used by positive score tracks in the cluster being processed.

For comparison, we now describe the MAP instance corresponding to the MWISP shown in Figure 5. Consider again the tracks in Figure 1 and the corresponding data in Table 1. As before, let $N = 3$ for N-scan pruning. We can

[6] Although rare, two families may have the same current root observation when N-scan pruning is applied. For example, when two targets cross or one target splits into two targets, we may only receive one (possibly unresolved!) observation per scan at the time when they cross or begin to split. Under our standard "one-to-one" assumption, this one observation can be assigned to at most one track.

Table 2. Data for the MAP instance using tracks in Figure 1

Track ID	Obs. Sequence	Family	Dec. Variable	Cost Coeff.
2	$Z_{1,1,0,1}$	F_1^4	$z_{1,1,1,0,1}$	-3.4
3	$Z_{1,1,1,1}$	F_1^4	$z_{1,1,1,1,1}$	-9.1
4	$Z_{1,2,2,2}$	F_1^4	$z_{1,1,2,2,2}$	-7.5
5	$Z_{2,2,1,0}$	F_2^4	$z_{2,2,2,1,0}$	-4.8
6	$Z_{3,2,3,2}$	F_3^4	$z_{3,3,2,3,2}$	-10.1

find the best global hypothesis by solving a 5-dimensional assignment problem. To do so, we assume that tracks (track IDs) in Table 1 are organized as follows: Tracks $Z_{1,1,0,1}$, $Z_{1,1,1,1}$, and $Z_{1,2,2,2}$ belong to family F_1^4, track $Z_{2,2,1,0}$ belongs to family F_2^4, track $Z_{3,2,3,2}$ belongs to family F_3^4, and track $Z_{0,0,2,2}$ belongs to family F_4^4. Since $Z_{0,0,2,2}$ has a negative score, family F_4^4 is not included in the MAP instance, so $m_0 = 3$. These tracks are shown in Table 2. As described above, in order for the MAP instance to have a feasible solution, we must also include decision variables for false alarms and for families that were not updated in the 4-scan window. All other decision variables, corresponding to tracks that were never formed or already pruned, are undefined or can be considered to have infinite cost. The optimal solution z^*_{MAP} to this MAP instance will be $z^*_{1,1,1,1,1}$ $= z^*_{3,3,2,3,2} = z^*_{2,0,0,0,0} = z^*_{0,2,0,0,0} = z^*_{0,0,0,2,0} = 1$ and all other decision variables equal to zero. This solution is equivalent to the best global hypothesis found with the MWISP formulation in Figure 5.

Equivalence of the MWISP and the MAP Formulations. We can prove that the MWISP and $(N+2)$-dimensional assignment problems for N-scan pruning are equivalent, i.e., a solution is an optimal solution to the MWISP if and only if it is an optimal solution to the MAP. Verifying that the two formulations are equivalent is important for the following reason. In general, it is not difficult to show that an instance of the MAP can be transformed into an instance of the MWISP [14]. However, the converse is not true. That is, given an arbitrary MWISP instance (i.e., an undirected graph of edges and weighted vertices), one cannot simply transform it into a MAP instance because the notion of dimensions does not exist for the MWISP.

As done above, we assume that a set of tracks $\{T_j^k\}_{j=1}^{\tau}$ and families $\{F_l^k\}_{l=1}^{\phi}$ exist on scan k just prior to entering the global hypothesis formation step in Figure 2 and that clusters have been formed. We also require every family to have a root observation in some scan $s \in \{k-N, \ldots, k\}$. This root may be an actual or dummy observation. The idea behind the proof is to first show that a feasible solution to the MWISP must also be feasible to the MAP and then to show that if this solution is also optimal to the MWISP it must also be optimal to the MAP. We only show one direction since one can use reverse logic to show the other direction.

Proposition 1. *The MWISP and MAP formulations presented above are equivalent.*

Proof. Let z'_{MWISP} be an optimal solution to the MWISP and let z^*_{MWISP} be the union of (the tracks in) z'_{MWISP} and all remaining observations in the cluster, which have been implicitly designated as false alarms. First, notice that z'_{MWISP} is an independent set, which, by definition, means that all tracks in z'_{MWISP} are compatible with one another, that at most one track per family is used, and that each observation per scan has been assigned to at most one track. Consequently, z^*_{MWISP} is a partition of the $(N+1)$ scans of observations and satisfies constraints (4c), (4d), and (4e). Furthermore, since at most one track per family is used, if we interpret all families $F^k_{i_0}$, for $i_0 = 1, \ldots, m_0$, that did not have a positive score track in this hypothesis to be of the form $\{F^k_{i_0}, z^1_0, \ldots, z^{N+1}_0\}$, then z^*_{MWISP} satisfies constraints (4b) as well and must be a feasible solution to the MAP. To show that z^*_{MWISP} is optimal to the MAP, suppose to the contrary that there exists some other solution z'_{MAP} such that $-c(z'_{MAP}) > w(z^*_{MWISP})$, where $c(z)$ is the sum in (4a) and $w(z)$ is sum of track scores in an independent set z. Immediately, we have a contradiction since the tracks in z'_{MAP}, by virtue of satisfying the MAP constraints, form an independent set. In this case, z^*_{MWISP} cannot be optimal contradicting our initial assumption. □

5 Qualitative Discussion Comparing the MWISP and the MAP Formulations

In this section, we offer a qualitative comparison of the MWISP and MAP formulations in the global hypothesis formation step of our MHT framework. The MWISP formulation has been employed in tactical MHT software whereas the MAP formulation has been demonstrated to work on test data with prototype software. However, to the best of our knowledge, no formal quantitative comparison between the two has been made.

Theoretically, we believe that the MAP formulation is more appealing than the MWISP formulation for two reasons. First, the MAP is a natural extension of the two-dimensional linear assignment problem formulation that is customarily used in single scan MTT algorithms. Second, numerous researchers have shown that for large problem instances found in MTT applications, various Lagrangian relaxation-based algorithms are capable of solving these instances in real-time within some quantifiable accuracy. Indeed, it is this quantifiable accuracy, which emerges through the approximate duality gap of a Lagrangian relaxation algorithm, that makes the MAP formulation theoretically attractive.

Algorithmically, both approaches have their fortes and foibles. In our opinion, the MWISP formulation has at least two advantages in our MHT framework. First, the MWISP and the data structures needed to implement an algorithm to solve it are conceptually simpler than the MAP. No two-dimensional linear assignment problem solvers, nonsmooth optimization techniques, subgradients, or Lagrangian relaxation algorithms are needed. (We recognize that these techniques are well studied and that they have been shown to be applicable in real-time systems.) Second, powerful local search methods for the MWISP make it far easier to enumerate tens of thousands of global hypotheses, including an optimal and many near optimal hypotheses, (in less than a second) to be used for computing approximate global hypothesis probabilities. The same cannot be said for local search techniques for the MAP formulation. The MAP instances encountered in MTT are typically very sparse because many observation-to-track associations are never considered through gating, which motivates the use of data structures that allow for sparse representation, e.g., a forward-star representation [18]. As reported in [23], performing local search with the same effectiveness as one can using the MWISP formulation can be relatively unwieldy. Similarly, "constructive" algorithms for the MAP, e.g., algorithms like a branch-and-bound or a greedy randomized adaptive search procedure (GRASP) which iteratively select a track from a neighborhood of tracks that preserve feasibility of the global hypothesis, require more time to generate this neighborhood. To understand this, realize that a MWISP local search method can determine with a single lookup whether or not another track is incompatible with it by looking at its incompatibility list. A branch-and-bound [19] or GRASP [25] method for the MAP must ensure that by introducing a new track into the global hypothesis, the global hypothesis remains feasible, which takes several lookups, at most $(N + 2)$ – the number of dimensions of the MAP, regardless of the representation.

On the other hand, the approximate duality gap furnished by Lagrangian relaxation algorithms for the MAP is not only a "nice" theoretical property, but it is equally as valuable algorithmically since it provides clear cut termination criteria, e.g., "stop trying to find a better partition once the duality gap is below x%." At the same time, its importance should not be overstated. In most MTT applications, an entire loop of the steps shown in Figure 2 must be performed in a matter of seconds or less. After completing the computationally intensive step of updating and propagating tracks, finding the best global hypothesis and performing track-level pruning must be done in as little as one-tenth of a second in some systems. Thus, while at first glance the number of lookups may seem irrelevant, real-time systems must perform these operations in far less time than is typically allotted in other applications in which the MWISP and the MAP arise.

6 Conclusions

We have shown that it is possible to formulate the observation-to-track association problem arising in multiple hypothesis tracking as a MWISP. Although this formulation has been known for some time, it has never been formally recognized. Blackman [8,6] only describes a "breadth-first ... and A* search method" as an algorithm for solving this formulation. Moreover, the survey by Pattipati et al. [20] never even mentions this formulation. We believe that this problem recognition is valuable as it provides an opportunity for state-of-the-art MWISP algorithms to be applied when current methods are inadequate. Finally, we have shown that this formulation is equivalent to the more popular MAP formulation in the context of our MHT framework.

Acknowledgements

The authors wish to thank Fred Daum and Tom Kurien of Raytheon as well as the anonymous referees for reviewing this chapter and providing insightful feedback.

References

1. Bar-Shalom, Y., Fortmann, T.E.: Tracking and Data Association. Academic Press, Boston (1988)
2. Bar-Shalom, Y.: Multitarget-Multisensor Tracking: Applications and Advances, vol. 1. Artech House, Norwood (1990)
3. Bar-Shalom, Y., Blackman, S.S., Fitzgerald, R.J.: Dimensionless Score Function for Multiple Hypothesis Tracking. IEEE Transactions on Aerospace and Electronic Systems 43(1), 392–400 (2007)
4. Bomze, I.M., Budinich, M., Pardalos, P.M., Pelillo, M.: The Maximum Clique Problem. In: Du, D.-Z., Pardalos, P.M. (eds.) Handbook of Combinatorial Optimization, vol. A (suppl.), pp. 1–74. Kluwer Academics, Boston (1999)
5. Blackman, S.S.: Association and Fusion of Multiple Sensor Data. In: Bar-Shalom, Y. (ed.) Multitarget-Multisensor Tracking: Applications and Advances, ch. 6, vol. I. Artech House, Norwood (1990)
6. Blackman, S.S.: Multiple Hypothesis Tracking for Multiple Target Tracking. IEEE Aerospace and Electronic Systems Magazine 19(1) (January 2004)
7. Blackman, S.S., Dempster, R.: Description of the Upgraded Early Warning Radar (UEWR) MHT tracker (revised), Technical Report, Raytheon Corporation (2002)
8. Blackman, S.S., Popoli, R.: Design and Analysis of Modern Tracking Systems. Artech House, Norwood (1999)
9. Borndorfer, R.: Aspects of Set Packing, Partitioning, and Covering, Ph.D. thesis, Technical University of Berlin (1998)
10. Capponi, A.: A Polynomial Time Algorithm for Data Association in Multitarget Tracking. IEEE Transactions on Aerospace and Electronic Systems 40(4), 1398–1410 (2004)
11. Daum, F.E.: The Importance of Resolution in Multiple Target Tracking. In: Drummond, O.E. (ed.) Signal and Data Processing of Small Targets 1994. SPIE Proceedings, vol. 2235, pp. 329–338 (1994)
12. Deb, S., Yeddanapudi, M., Pattipati, K.R., Bar-Shalom, Y.: A Generalized S-D Assignment Algorithm for Multisensor-Multitarget State Estimation. IEEE Transactions on Aerospace and Electronic Systems 33(2), 523–538 (1997)
13. DiPalma, L., Groves, G., Kiczuk, B., Kinney, B., Vahey, M.: Raytheon Processing Technology. Raytheon Technology Toda 2(1) (Spring 2003)
14. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of $\mathcal{NP}$-Completeness. Freeman, New York (1979)
15. Jevtic, N., Klautau, A., Orlitsky, A.: Estimated rank pruning and Java-based speech recognition. In: Automatic Speech Recognition and Understanding IEEE Workshop, pp. 401–404 (2001)
16. Kurien, T.: Issues in the Design of Practical Multitarget Tracking Algorithms. In: Bar-Shalom, Y. (ed.) Multitarget-Multisensor Tracking: Applications and Advances, ch. 3, vol. I. Artech House, Norwood (1990)

17. Morefield, C.L.: Application of 0-1 Integer Programming to Multitarget Tracking Problems. IEEE Transactions on Automatic Control AC-22(3), 302–312 (1977)
18. Pasiliao, E.L.: Algorithms for Multidimensional Assignment Problems. Ph.D. thesis, University of Florida at Gainesville (2003)
19. Pasiliao, E.L., Pardalos, P.M., Pitsoulis, L.S.: Branch and Bound Algorithms for the Multidimensional Assignment Problem. Optimization Methods and Software 20(1), 127–143 (2005)
20. Pattipati, K.R., Popp, R.L., Kirubarajan, T.: Survey of Assignment Techniques for Multitarget Tracking. In: Bar-Shalom, Y., Blair, W.D. (eds.) Multitarget-Multisensor Tracking: Applications and Advances, vol. III. Artech House, Norwood (2000)
21. Poore, A.B.: Multidimensional Assignment Formulation of Data Association Problems Arising from Multitarget Tracking and Multisensor Data Fusion. Computational Optimization and Applications 3, 27–57 (1994)
22. Poore, A.B., Drummond, O.E.: Track Initiation and Maintenance Using Multidimensional Assignment Problems. In: Pardalos, P.M., Hearn, D., Hager, W. (eds.) Network Optimization. Lecture Notes in Economics and Mathematical Systems, pp. 407–422. Springer, Heidelberg (1996)
23. Poore, A.B., Robertson III, A.J.: A New Lagrangian Relaxation Based Algorithm for a Class of Multidimensional Assignment Problems. Computational Optimization and Applications 8, 129–150 (1997)
24. Popp, R.L., Pattipati, K.R., Bar-Shalom, Y.: m-Best S-D Assignment Algorithm with Application to Multitarget Tracking. IEEE Transactions on Aerospace and Electronic Systems 37(1), 22–39 (2001)
25. Robertson, A.J.: A Set of Greedy Randomized Adaptive Search Procedure (GRASP) Implementations for the Multidimensional Assignment Problem. Computational Optimization and Applications 19, 145–164 (2001)
26. Reid, D.B.: An Algorithm for Tracking Multiple Targets. IEEE Transactions on Automatic Control 24(6), 843–854 (1979)
27. Sittler, R.W.: An Optimal Data Association Problem in Surveillance Theory. IEEE Transactions on Military Electronics 8, 125–139 (1964)

New Results in Stochastic Cooperative Games: Strategic Coordination for Multi-resolution Performance Robustness

Khanh D. Pham

Space Vehicles Directorate
Air Force Research Laboratory
Kirtland AFB, NM 87117 U.S.A.
khanh.pham@kirtland.af.mil

Abstract. This chapter provides a concise and up-to-date analysis of the foundations of performance robustness of a linear-quadratic class of cooperative decision-making with respect to variability in a stationary and stochastic environment. The dynamics of large-scale and interconnected dynamical systems corrupted by a standard stationary Wiener process include decision activities that are controlled by decentralized decision makers. Basic assumptions will be that cooperative decision makers form a coalition and have access to the current value of the states of the systems. The performance robustness and uncertainty of the coalition is now affected by other non-cooperative participants such as model deviations and environmental disturbances, named Nature. A decentralized model-following approach where interactions among coalitive decision makers are represented by reduced order models, is taken to derive various local decisions without intensive information interchanges. Decentralized decision strategies considered here collaboratively optimize an advanced multi-objective criterion over time where the optimization takes place with high regard for possible random sample realizations by Nature who may more likely not be acting in concert. The implementation of decision gain parameters in the finite horizon case is shown to be feasible. The inherent decision structure now has two degrees of freedom including: one, state feedback gains with state measurements that are robust against performance uncertainty; and two, interactive feedback gains that minimize differences between the actual and desirable interactions.

1 Introduction

The study of cooperative games is concerned with the behavior of systems constrained by differential equations with some number of independent activity levels determined by decision makers, each of whom has a separate objective functional he wishes to optimize. Since the objectives are possibly conflicting, decision makers therefore enter into binding agreements for achieving mutual benefits. Furthermore, every decision maker may have all the information on the state dynamics of the shared interaction model and objective functionals of other decision makers. All decision makers are able to implement their decisions without restrictions. Works in deterministic multi-person decision problems were initiated by [8] and [9] when a set of policies which is optimal for one decision maker, may have rather negative effects on the evolution of interaction from another decision maker's point of view. More recently, people have begun to examine the general problem in which more than two decision makers attempt to improve their performances with respect to all realizations of the environmental disturbances, named Nature. For a detailed discussion of these results, see [6] and [7]. Significant advances in these works involve: 1) whether it is possible to construct beliefs that reflect genuine uncertainty about Nature's mixed random realizations, yet are narrow enough to permit learning on performance robustness with respect to variability in the stochastic environment and 2) the support of beliefs should, on the one hand, encompass mixed random strategies, reflecting uncertainty about Nature and, on the other hand, needs to include strategies that actually are optimal for decision makers given the support of their beliefs.

The aforementioned developments and formulations are motivated by the need of uncertainty analysis which evaluates the impact of uncertainty caused by stochastic disturbances on system performance and has long been recognized as an important and indispensable consideration in reliability-based design [2] and incorporation of aversion to specification uncertainty [4]. The research investigation considered here includes a general discussion on the formulation of multi-person, and multi-objective criteria, coordination and control problems that involve uncertainty, performance robustness, information decentralization and potential conflicts of interest between cooperative decision makers and Nature. Possible solution concepts for such decision problems are also proposed as the extension of the account [6] in the areas of minimal information interchanges and balances of local objectives and global objectives. Having said that, all respective measures of performance of cooperative decision makers are viewed as random variables with Nature's mixed random realizations. The action space of Nature regarding all realizations of the underlying stochastic process is assumed to be common knowledge. Included with the present work are some novel answers to the following open research problems: i) Cooperative decision makers do not comply with the traditional average measure of performance; ii) An effective knowledge organization and construct that can be able to extract the knowledge of higher-order characteristics of the shared performance distributions; and iii) Collaborative decision strategies that guarantee performance robustness with multiple attributes beyond performance averaging as such variance, skewness, flatness, etc. of the probability density function of the common performance, just to name a few.

The chapter is organized as follows. In Section 2 the multi-person decision problem against Nature, together with the definitions of performance-measure statistics and their supporting equations associated with the Chi-squared random measure of performance is presented. Problem statements for the resulting Mayer problem in dynamic programming are given in Section 3. Construction of a candidate function for the value function and the calculation of decentralized efficient Pareto decision strategies accounting for multiple attributes of performance robustness are included in Section 4, while conclusions are drawn in Section 5.

2 Problem Formulation

Before going into a formal presentation, it is necessary to consider some conceptual notations. To be specific, for a given Hilbert space X with norm $||\cdot||_X$, $1 \le p \le \infty$ and $a, b \in \mathbb{R}$ such that $a \le b$, a Banach space is defined as follows

$$L^p_{\mathcal{F}}(a,b;X) \triangleq \Bigg\{ \phi(\cdot) = \{\phi(t,\omega) : a \le t \le b\} \ni \phi(\cdot) \, \text{is an } X\text{-valued } \mathcal{F}_t\text{-measurable} \\ \text{process on } [a,b] \text{ with } E\left\{ \int_a^b ||\phi(t,\omega)||_X^p dt \right\} < \infty \Bigg\} \tag{1}$$

with norm

$$||\phi(\cdot)||_{\mathcal{F},p} \triangleq \left(E\left\{ \int_a^b ||\phi(t,\omega)||_X^p \, dt \right\} \right)^{1/p} \tag{2}$$

where the elements ω of the filtered sigma field $\mathcal{F}_t$ of a sample description space Ω that is adapted for the time horizon $[a,b]$, are random outcomes or events. Also, the Banach space of X-valued continuous functionals on $[a,b]$ with the max-norm induced by $||\cdot||_X$ is denoted by $C(a,b;X)$. The deterministic version of (1) and its associated norm (2) is written as $L^p(a,b;X)$ and $||\cdot||_p$.

Now a class of stochastic multi-person decision problems with a set of decision makers denoted by $\overline{N} \triangleq \{1,2,\ldots,N\}$ and a typical element by i is considered. An underlying filtered probability space $(\Omega_i, \mathcal{F}_i, \{\mathcal{F}_i\}_{t \ge t_0 > 0}, \mathcal{P}_i)$ is defined with a stationary p_i-dimensional Wiener process $w_i(t) \triangleq w_i(t,\omega_i) : [t_0,t_f] \times \Omega_i \mapsto \mathbb{R}^{p_i}$ on a finite horizon $[t_0,t_f]$ and the correlation of independent increments

$$E\left\{ [w_i(\tau) - w_i(\xi)][w_i(\tau) - w_i(\xi)]^T \right\} = W_i|\tau - \xi|, \quad W_i > 0$$

for the environmental uncertainties, named Nature from now on which are beyond the control of decision maker i. A mathematical description that defines an information flow structure and some level of interactions of decision maker i within the i-th subsystem and with the uncertain behaviors $\omega_i \in \Omega_i$ of Nature

$$dx_i(t) = (A_i(t)x_i(t) + B_i(t)u_i(t) + C_i(t)z_i(t) + E_i(t)d_i(t))dt + G_i(t)dw_i(t) \tag{3} \\ x_i(t_0) = x_i^0$$

where all the coefficients $A_i \in C(t_0,t_f;\mathbb{R}^{n_i \times n_i})$, $B_i \in C(t_0,t_f;\mathbb{R}^{n_i \times m_i})$, $C_i \in C(t_0,t_f;\mathbb{R}^{n_i \times q_i})$, $E_i \in C(t_0,t_f;\mathbb{R}^{n_i \times r_i})$, and $G_i \in C(t_0,t_f;\mathbb{R}^{n_i \times p_i})$ are deterministic matrix-valued functions. Furthermore, it should be noted that

$x_i \in L^2_{\mathcal{F}}(t_0, t_f; \mathbb{R}^{n_i})$ is the n_i-dimensional state of the i-th subsystem with the initial state $x_i^0 \in \mathbb{R}^{n_i}$ fixed, $u_i \in L^2_{\mathcal{F}}(t_0, t_f; \mathbb{R}^{m_i})$ is the m_i-dimensional control decision, and $d_i \in L^2(t_0, t_f; \mathbb{R}^{r_i})$ is the r_i-dimensional known disturbance.

The interaction $z_i \in L^2_{\mathcal{F}}(t_0, t_f; \mathbb{R}^{q_i})$ is the q_i-dimensional process for decision maker i that is a function of the other subsystem state variables

$$z_i(t) = \sum_{j=1, j\neq i}^{N} L_{ij}(t) x_{ij}(t)\,, \quad i \in \overline{N}\,. \tag{4}$$

Therefore, the process z_i is not available at the subsystem level i. It is then desired to have decentralized decision making without intensive communications that subsequently involves the selection of a crude model of reduced order for the interactions among the decision makers. The actual interaction $z_i(\cdot)$ is now approximated by an explicit model-following of the type

$$\begin{aligned} dz_{mi}(t) &= (A_{zi}(t) z_{mi}(t) + E_{zi}(t) d_{mi}(t))dt + G_{zi}(t) dw_{mi}(t)\,, \\ z_{mi}(t_0) &= z^0_{mi} \end{aligned} \tag{5}$$

where $A_{zi} \in C(t_0, t_f; \mathbb{R}^{q_i \times q_i})$ is an arbitrary deterministic matrix-valued function which describes a crude model for the actual interaction $z_i(\cdot)$. For instance, A_{zi} can be chosen to be the off-diagonal block of the global matrix coefficient A corresponding the partition vector $z_i(\cdot)$. Other coefficients $E_{zi} \in C(t_0, t_f; \mathbb{R}^{q_i \times r_{mi}})$ and $G_{zi} \in C(t_0, t_f; \mathbb{R}^{q_i \times p_{mi}})$ are deterministic matrix-valued functions of the stochastic differential equation (5). The uncertainty and robustness of the interaction $z_{mi}(\cdot)$ produced by the model is affected by the known disturbance $d_{mi} \in L^2(t_0, t_f; \mathbb{R}^{r_{mi}})$ and Nature $w_{mi}(t) \triangleq w_{mi}(t, \omega_{mi}) : [t_0, t_f] \times \Omega_{mi} \mapsto \mathbb{R}^{p_{mi}}$ is an p_{mi}-dimensional stationary Wiener process with $\{\mathcal{F}_{mi}\}_{t \geq t_0 > 0}$ defined as its filtration on a complete filtered probability space $(\Omega_{mi}, \mathcal{F}_{mi}, \{\mathcal{F}_{mi}\}_{t \geq t_0 > 0}, \mathcal{P}_{mi})$ over $[t_0, t_f]$ with the correlation of independent increments

$$E\left\{[w_{mi}(\tau) - w_{mi}(\xi)][w_{mi}(\tau) - w_{mi}(\xi)]^T\right\} = W_{mi}|\tau - \xi|, \quad W_{mi} > 0\,.$$

In addition, the local models of dynamics (3) and interaction prediction (5) in the absence of known disturbances and stationary environments are assumed to be uniformly exponentially stable. For instance, there exist positive constants η_1 and η_2 such that the pointwise matrix norm of the closed-loop state transition matrix associated with the local dynamical model (3) satisfies the inequality

$$||\Phi_i(t, \tau)|| \leq \eta_1 e^{-\eta_2 (t - \tau)} \qquad \forall\; t \geq \tau \geq t_0\;\;.$$

The pair $(A_i(t), [B_i(t), C_i(t)])$ is pointwise stabilizable if there exist bounded matrix-valued functions $K_i(t)$ and $K_{zi}(t)$ so that the closed-loop system $dx(t) = (A_i(t) + B_i(t)K_i(t) + C_i(t)K_{zi}(t))\, x_i(t)dt$ is uniformly exponentially stable.

With the approach considered here, there is a need to treat the actual interaction $z_i(\cdot)$ as the control process that is supposed to follow the predicted interaction process $z_{mi}(\cdot)$. Thus, this requirement leads to the classes of admissible decisions associated with (3) to be denoted by $U_i \times Z_i \subset L^2_{\mathcal{F}}(t_0, t_f; \mathbb{R}^{m}) \times L^2_{\mathcal{F}}(t_0, t_f; \mathbb{R}^{q})$. Given $(u_i(\cdot), z_i(\cdot)) \in U_i \times Z_i$, the 3-tuple $(x_i(\cdot), u_i(\cdot), z_i(\cdot))$ shall be referred to as an admissible 3-tuple if $x_i(\cdot) \in L^2_{\mathcal{F}}(t_0, t_f; \mathbb{R}^{n})$ is a solution of the stochastic differential equation (3) associated with $u_i(\cdot) \in U_i$ and $z_i(\cdot) \in Z_i$.

Under the assumptions of $x_i^0 \in \mathbb{R}^{n}$, $z_{mi}^0 \in \mathbb{R}^{q}$, $u_i(\cdot) \in U_i$, and $z_i(\cdot) \in Z_i$, there are tradeoffs among the closeness of the local states from desired states, the size of the local decision levels, and the size of the locally mismatched interactions, and the decision maker i has to carefully balance the three in order to achieve his local performance. Mathematically, there assumes existence of an integral-quadratic form (IQF) performance-measure $J_i : \mathbb{R}^{n} \times \mathbb{R}^{q} \times U_i \times Z_i \mapsto \mathbb{R}$

$$J_i(x_i^0, z_{mi}^0; u_i(\cdot), z_i(\cdot)) = x_i^T(t_f) Q_i^f x_i(t_f) + \int_{t_0}^{t} \Big[x_i^T(\tau) Q_i(\tau) x_i(\tau) + u_i^T(\tau) R_i(\tau) u_i(\tau) + (z_i(\tau) - z_{mi}(\tau))^T R_{zi}(\tau)(z_i(\tau) - z_{mi}(\tau)) \Big] d\tau \quad (6)$$

where respective design parameters $Q_i^f \in \mathbb{R}^{n \times n}$, $Q_i \in C(t_0, t_f; \mathbb{R}^{n \times n})$, $R_i \in C(t_0, t_f; \mathbb{R}^{m \times m})$, and $R_{zi} \in C(t_0, t_f; \mathbb{R}^{q \times q})$ representing relative weightings for terminal states, transient states, decision levels and interaction mismatches are deterministic and positive semidefinite with $R_i(t)$ and $R_{zi}(t)$ invertible.

The aggregate autonomy-interaction arrangement that locally depicts the aspirations and interaction details of decision maker i, then requires the following augmented subsystem variables and parameters

$$x_{ai}(t) \triangleq \begin{bmatrix} x_i(t) \\ z_{mi}(t) \end{bmatrix}; \quad x_{ai}^0 \triangleq \begin{bmatrix} x_i^0 \\ z_{mi}^0 \end{bmatrix}; \quad w_{ai}(t) \triangleq \begin{bmatrix} w_i(t) \\ w_{mi}(t) \end{bmatrix}. \quad (7)$$

The respective tradeoff decision levels and preferences (6) for decision maker i is rewritten as follows

$$J_i(x_{ai}^0; u(\cdot), z_i(\cdot)) = x_{ai}^T(t_f) Q_{ai}^f x_{ai}(t_f) + \int_{t_0}^{t} \big[x_{ai}^T(\tau) Q_{ai}(\tau) x_{ai}(\tau) + u_i^T(\tau) R_i(\tau) u_i(\tau) + z_i^T(\tau) R_{zi}(\tau) z_i(\tau) - 2 x_{ai}^T(\tau) S_{ai}(\tau) z_i(\tau) \big] d\tau \quad (8)$$

where the quadratic weightings at local performance are given by

$$Q_{ai}^f \triangleq \begin{bmatrix} Q_i^f & 0 \\ 0 & 0 \end{bmatrix}; \quad Q_{ai}(\tau) \triangleq \begin{bmatrix} Q_i(\tau) & 0 \\ 0 & R_{zi}(\tau) \end{bmatrix}; \quad S_{ai}(\tau) \triangleq \begin{bmatrix} 0 \\ R_{zi}(\tau) \end{bmatrix} \quad (9)$$

subject to the local dynamics for strategic relations

$$dx_{ai}(t) = (A_{ai}(t) x_{ai}(t) + B_{ai}(t) u_i(t) + C_{ai}(t) z_i(t) + D_{ai}(t)) dt + G_{ai}(t) dw_{ai}(t)$$
$$x_{ai}(t_0) = x_{ai}^0 \quad (10)$$

with the corresponding system parameters

$$A_{ai}(t) \triangleq \begin{bmatrix} A_i(t) & 0 \\ 0 & A_{zi}(t) \end{bmatrix}; \quad B_{ai}(t) \triangleq \begin{bmatrix} B_i(t) \\ 0 \end{bmatrix}; \quad C_{ai}(t) \triangleq \begin{bmatrix} C_i(t) \\ 0 \end{bmatrix} \quad (11)$$

$$D_{ai}(t) \triangleq \begin{bmatrix} E_i(t) d_i(t) \\ E_{zi}(t) d_{mi}(t) \end{bmatrix}; \; G_{ai}(t) \triangleq \begin{bmatrix} G_i(t) & 0 \\ 0 & G_{zi}(t) \end{bmatrix}; \; W_{ai} \triangleq \begin{bmatrix} W_i & 0 \\ 0 & W_{mi} \end{bmatrix}. \quad (12)$$

Next, all decision makers decide to act cooperatively via an enforceable negotiation that comes from a coalitive game defined by

- A finite set of decision makers $\overline{S} \triangleq \{1, \ldots, S\} \subset \overline{N}$
- A collection of decision sets indexed on $\overline{S}$, i.e., $\{U_j\}_{j\in\overline{S}}$ and $\{Z_j\}_{j\in\overline{S}}$
- A measure of performance, $J_i : \mathbb{R}^{n\ +q} \times \varPi_{j\in\overline{S}} U_j \times \varPi_{j\in\overline{S}} Z_j \mapsto \mathbb{R}$.

By cooperation, it seems reasonable that all decision makers would prefer to have information exchanges for coalitive coordinations that result in some cooperative outcomes that no one improves his performance without negatively affecting that of another. Hence, only those outcomes motivate the concept of Pareto efficiency.

Definition 1. *Efficient and Collective Decisions.*
For any random realization $\omega_{ai} \in \varOmega_{ai}$ drawn by Nature, coalitive decisions $\hat{u}_S$ and $\hat{z}_S$ are Pareto efficient if

$$(\hat{u}_S, \hat{z}_S) \in \operatorname*{argmin}_{u\ \in U\ ,z\ \in Z} \left\{J_1(x_{a1}^0; u_S, z_S), \ldots, J_S(x_{aS}^0; u_S, z_S)\right\} \tag{13}$$

where $U_S \triangleq \varPi_{j\in\overline{S}} U_j$ and $Z_S \triangleq \varPi_{j\in\overline{S}} Z_j$. Furthermore, the corresponding point $(J_1(x_{a1}^0; \hat{u}_S, \hat{z}_S), \ldots, J_S(x_{aS}^0; \hat{u}_S, \hat{z}_S)) \in \mathbb{R}^S$ is called a Pareto solution. The set of all Pareto solutions is called the Pareto frontier.

In the subsequent analysis, the following set of parameters, $\varXi$ represents for cooperative profiles

$$\varXi \triangleq \left\{\xi = (\xi_1, \ldots, \xi_S) \in \mathbb{R}^S | \xi_i > 0 \,\text{and}\, \sum_{i=1}^{S} \xi_i = 1\right\}.$$

Proposition 1. *Efficient Pareto Parameterizations.*
Let $\xi \in \varXi$ and $\omega_{ai} \in \varOmega_{ai}$. If

$$(\hat{u}_S, \hat{z}_S) \in \operatorname*{argmin}_{u\ \in U\ ,z\ \in Z} \left\{\sum_{i=1}^{S} \xi_i J_i(x_{ai}^0; u_S, z_S)\right\} \tag{14}$$

then $(\hat{u}_S, \hat{z}_S)$ is Pareto efficient.

Since the performance-measures (8) are convex functions on a convex set $\mathcal{U}_S \times \mathcal{Z}_S$ with convex constraints (10), the problem of finding all Pareto efficient strategies for the linear quadratic game with a vector-valued objective criterion is equivalent to the problem of solving an $S-1$ parameter family of optimization problems with scalar-valued objective criteria [5].

Proposition 2. *Necessary and Sufficient Conditions.*
Suppose $U_S \times Z_S$ is convex and realized performance measure J_i is convex. $\hat{u}_S \in U_S$, $\hat{z}_S \in Z_S$ are efficient if and only if there exits $\xi \in \varXi$ such that $\hat{u}_S$ and $\hat{z}_S$ are the corresponding Pareto-efficient decision strategies obtained as

$$(\hat{u}_S, \hat{z}_S) = \operatorname*{argmin}_{u\ \in U\ ,z\ \in Z} \left\{J_\xi(x_a^0; u_S, z_S) \triangleq \sum_{i=1}^{S} \xi_i J_i(x_{ai}^0; u_S, z_S)\right\}. \tag{15}$$

However, there are, in general, many Pareto solutions which depend on parameters ξ and Nature's actions ω_{ai}. This observation raises the question as to which one is the best. To address this question, the present work suggests an emerging arena of what is called performance robustness for stochastic cooperative games against Nature based upon the cumulant-based control theory, in conjunction with the concept of Pareto efficiency.

From the aspect of interaction management within a cooperative environment, the coalitive-conscious formulation which takes into account of model uncertainties, coalitions and Nature actions, requires that

$$x_a(t) \triangleq \begin{bmatrix} x_{a1}(t) \\ \vdots \\ x_{aS}(t) \end{bmatrix}; \quad w_a \triangleq \begin{bmatrix} w_{a1}(t) \\ \vdots \\ w_{aS}(t) \end{bmatrix} \tag{16}$$

which leads to the quadratic measure of performance

$$J_\xi(x_a^0; \hat{u}_S(\cdot), \hat{z}_S(\cdot)) = x_a^T(t_f) Q_a^f x_a(t_f) + \int_{t_0}^{t} \big[x_a^T(\tau) Q_a(\tau) x_a(\tau) \\ + \hat{u}_S^T(\tau) R_a(\tau) \hat{u}_S(\tau) + \hat{z}_S^T(\tau) R_{az}(\tau) \hat{z}_S(\tau) - 2 x_a^T(\tau) S_a(\tau) \hat{z}_S(\tau) \big] d\tau \tag{17}$$

where the quadratic weightings for the coalitive measure of performance are

$$\begin{aligned} &S_a(\tau) \triangleq diag(\xi_1 S_{a1}(\tau), \ldots, \xi_S S_{aS}(\tau)) \\ &Q_a^f \triangleq diag(\xi_1 Q_{a1}^f, \ldots, \xi_S Q_{aS}^f); \quad Q_a(\tau) \triangleq diag(\xi_1 Q_{a1}(\tau), \ldots, \xi_S Q_{aS}(\tau)); \\ &R_a(\tau) \triangleq diag(\xi_1 R_1(\tau), \ldots, \xi_S R_S(\tau)); \quad R_{az}(\tau) \triangleq diag(\xi_1 R_{z1}(\tau), \ldots, \xi_S R_{zS}(\tau)) \end{aligned} \tag{18}$$

subject to the aggregate dynamics strategically coordinated for the coalition $\overline{S}$

$$\begin{aligned} &dx_a(t) = (A_a(t) x_a(t) + B_a(t) \hat{u}_S(t) + C_a(t) \hat{z}_S(t) + D_a(t)) dt + G_a(t) dw_a(t) \\ &x_a(t_0) = x_a^0 \end{aligned} \tag{19}$$

provided that

$$\begin{aligned} &A_a(t) \triangleq diag(A_{a1}(t), \ldots, A_{aS}(t)); \quad B_a(t) \triangleq diag(B_{a1}(t), \ldots, B_{aS}(t)); \\ &C_a(t) \triangleq diag(C_{a1}(t), \ldots, C_{aS}(t)); \quad D_a^T(t) \triangleq \left[D_{a1}^T(t) \ldots D_{aS}^T(t) \right]; \\ &G_a(t) \triangleq diag(G_{a1}(t), \ldots, G_{aS}(t)); \quad W_a \triangleq diag(W_{a1}, \ldots, W_{aS}). \end{aligned} \tag{20}$$

Since the significance of (17) and (19) is linear-quadratic in nature, it is often argued that the actions of cooperative decision makers should be functions of the states. The restriction of decision strategy spaces to the set of so-called Markov functions can be justified by the assumption that decision makers participate in the cooperative game where they only have access to the current time and state of the interaction. Therefore, it amounts to considering only those feedback Pareto strategies which permit linear feedback syntheses $\hat{\gamma}_{si} : [t_0, t_f] \times L^2_{\mathcal{F}}(t_0, t_f; \mathbb{R}^{n+q}) \mapsto L^2_{\mathcal{F}}(t_0, t_f; \mathbb{R}^{m})$ and $\hat{\gamma}^z_{si} : [t_0, t_f] \times L^2_{\mathcal{F}}(t_0, t_f; \mathbb{R}^{n+q}) \mapsto L^2_{\mathcal{F}}(t_0, t_f; \mathbb{R}^{q})$

$$\hat{u}_{si}(t) = \hat{\gamma}_{si}(t, x_{ai}(t)) \triangleq K_i(t)x_{ai}(t) + p_i(t) \tag{21}$$

$$\hat{z}_{si}(t) = \hat{\gamma}^z_{si}(t, x_{ai}(t)) \triangleq K_{zi}(t)x_{ai}(t) + p_{zi}(t) \tag{22}$$

where admissible matrix-valued decision gains $K_i \in C(t_0, t_f; \mathbb{R}^{m \times (n+q)})$ and $K_{zi} \in C(t_0, t_f; \mathbb{R}^{q \times (n+q)})$ as well as vector-valued decision affine functions $p_i \in C(t_0, t_f; \mathbb{R}^{m})$ and $p_{zi} \in C(t_0, t_f; \mathbb{R}^{q})$ will be appropriately defined, respectively.

Having local mechanisms (21)-(22) for information sampling in place, coalitive decision makers simultaneously implement Pareto efficient strategies so that the awareness of the corresponding interaction process is maintained up to date

$$\begin{aligned}\hat{u}_S(t) &= \begin{bmatrix} \hat{u}_{s1}(t) \\ \vdots \\ \hat{u}_{sS}(t) \end{bmatrix} = \begin{bmatrix} K_1(t) & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & K_S(t) \end{bmatrix} \begin{bmatrix} x_{a1}(t) \\ \vdots \\ x_{aS}(t) \end{bmatrix} + \begin{bmatrix} p_1(t) \\ \vdots \\ p_S(t) \end{bmatrix} \\ &\triangleq K(t)x_a(t) + p(t)\,, \end{aligned} \tag{23}$$

$$\begin{aligned}\hat{z}_S(t) &= \begin{bmatrix} \hat{z}_{s1}(t) \\ \vdots \\ \hat{z}_{sS}(t) \end{bmatrix} = \begin{bmatrix} K_{z1}(t) & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & K_{zS}(t) \end{bmatrix} \begin{bmatrix} x_{a1}(t) \\ \vdots \\ x_{aS}(t) \end{bmatrix} + \begin{bmatrix} p_{z1}(t) \\ \vdots \\ p_{zS}(t) \end{bmatrix} \\ &\triangleq K_z(t)x_a(t) + p_z(t)\,. \end{aligned} \tag{24}$$

For the given (t_0, x_a^0) and subject to the decision strategies (23)-(24), the coalition $\overline{S}$ forms a global awareness of the interaction dynamics (19) as follows

$$\begin{aligned} dx_a(t) &= \big((A_a(t) + B_a(t)K(t) + C_a(t)K_z(t))x_a(t) \\ &\quad + B_a(t)p(t) + C_a(t)p_z(t) + D_a(t)\big)dt + G_a(t)dw_a(t)\,, \quad x_a(t_0) = x_a^0 \end{aligned} \tag{25}$$

and the realized measure of coalitive performance (17) becomes

$$\begin{aligned} &J_\xi(x_a^0; K(\cdot), p(\cdot); K_z(\cdot), p_z(\cdot)) = x_a^T(t_f)Q_a^f x_a(t_f) \\ &+ \int_{t_0}^{t} \Big\{ x_a^T(\tau)\big[K^T(\tau)R_a(\tau)K(\tau) + K_z^T(\tau)R_{az}(\tau)K_z(\tau) - 2S_a(\tau)K_z(\tau)\big]x_a(\tau) \\ &\qquad + 2x_a^T(\tau)\big[K^T(\tau)R_a(\tau)p(\tau) + K_z^T(\tau)R_{az}(\tau)p_z(\tau) - S_a(\tau)p_z(\tau)\big] \\ &\qquad\qquad + p^T(\tau)R_a(\tau)p(\tau) + p_z^T(\tau)R_{az}(\tau)p_z(\tau)\Big\}d\tau\,. \end{aligned} \tag{26}$$

Within the view of the linear-quadratic structure of the decision problem, the performance-measure (26) for cooperative decision makers is clearly a random variable with Chi-squared type. Hence, the uncertainty of performance distribution must be assessed via a complete set of higher-order statistics beyond the statistical averaging. It is therefore necessary to develop a mathematical construct and support of the beliefs on performance uncertainty to extract the knowledge in definite terms of performance-measure statistics for the coalition $\overline{S}$. This is done by adapting the results in [6].

Theorem 1. *Performance-Measure Statistics.*
Let the pairs (A_a, B_a) and (A_a, C_a) be uniformly stabilizable on $[t_0, t_f]$ in the cooperative game governed by (25) and (26). Then, for any given $k \in \mathbb{Z}^+$, one obtains the k-th cumulant associated with the performance-measure (26) as

$$\kappa_k(t_0, x_a^0) = (x_a^0)^T H(t_0, k) x_a^0 + 2(x_a^0)^T \breve{D}(t_0, k) + D(t_0, k) \tag{27}$$

where the cumulant components $\{H(\alpha, r)\}_{r=1}^k$, $\{\breve{D}(\alpha, r)\}_{r=1}^k$ and $\{D(\alpha, r)\}_{r=1}^k$ evaluated at $\alpha = t_0$ satisfy the supporting matrix-valued differential equations (with the dependence of $H(\alpha, r)$, $\breve{D}(\alpha, r)$ and $D(\alpha, r)$ upon the admissible K, K_z, p and p_z suppressed)

$$\begin{aligned}\frac{d}{d\alpha}H(\alpha,1) = &- \left[A_a(\alpha) + B_a(\alpha)K(\alpha) + C_a(\alpha)K_z(\alpha)\right]^T H(\alpha,1)\\ &- H(\alpha,1)\left[A_a(\alpha) + B_a(\alpha)K(\alpha) + C_a(\alpha)K_z(\alpha)\right] - Q_a(\alpha)\\ &- K^T(\alpha)R_a(\alpha)K(\alpha) - K_z^T(\alpha)R_{az}(\alpha)K_z(\alpha) + 2S_a(\alpha)K_z(\alpha)\end{aligned} \tag{28}$$

$$\begin{aligned}\frac{d}{d\alpha}H(\alpha,r) = &- \left[A_a(\alpha) + B_a(\alpha)K(\alpha) + C_a(\alpha)K_z(\alpha)\right]^T H(\alpha,r)\\ &- H(\alpha,r)\left[A_a(\alpha) + B_a(\alpha)K(\alpha) + C_a(\alpha)K_z(\alpha)\right]\\ &- \sum_{s=1}^{r-1}\frac{2r!}{s!(r-s)!}H(\alpha,s)G_a(\alpha)W_aG_a^T(\alpha)H(\alpha,r-s)\,, \quad 2 \le r \le k\end{aligned} \tag{29}$$

$$\begin{aligned}\frac{d}{d\alpha}\breve{D}(\alpha,1) = &- \left[A_a(\alpha) + B_a(\alpha)K(\alpha) + C_a(\alpha)K_z(\alpha)\right]^T \breve{D}(\alpha,1)\\ &- H(\alpha,1)\left[B_a(\alpha)p(\alpha) + C_a(\alpha)p_z(\alpha) + D_a(\alpha)\right]\\ &- K^T(\alpha)R_a(\alpha)p(\alpha) - K_z^T(\alpha)R_{az}(\alpha)p_z(\alpha) + S_a(\alpha)p_z(\alpha)\end{aligned} \tag{30}$$

$$\begin{aligned}\frac{d}{d\alpha}\breve{D}(\alpha,r) = &- \left[A_a(\alpha) + B_a(\alpha)K(\alpha) + C_a(\alpha)K_z(\alpha)\right]^T \breve{D}(\alpha,r)\\ &- H(\alpha,r)\left[B_a(\alpha)p(\alpha) + C_a(\alpha)p_z(\alpha) + D_a(\alpha)\right]\,, \quad 2 \le r \le k\end{aligned} \tag{31}$$

$$\begin{aligned}\frac{d}{d\alpha}D(\alpha,1) = &-2\breve{D}^T(\alpha,1)\left[B_a(\alpha)p(\alpha) + C_a(\alpha)p_z(\alpha) + D_a(\alpha)\right]\\ &- \mathrm{Tr}\{H(\alpha,1)G_a(\alpha)W_aG_a^T(\alpha)\}\\ &- p^T(\alpha)R_a(\alpha)p(\alpha) - p_z^T(\alpha)R_{az}(\alpha)p_z(\alpha)\end{aligned} \tag{32}$$

$$\begin{aligned}\frac{d}{d\alpha}D(\alpha,r) = &-2\breve{D}^T(\alpha,r)\left[B_a(\alpha)p(\alpha) + C_a(\alpha)p_z(\alpha) + D_a(\alpha)\right]\\ &- \mathrm{Tr}\{H(\alpha,r)G_a(\alpha)W_aG_a^T(\alpha)\}\,, \quad 2 \le r \le k\end{aligned} \tag{33}$$

where the terminal-value conditions $H(t_f, 1) = Q_a^f$, $H(t_f, r) = 0$ for $2 \le r \le k$; $\breve{D}(t_f, r) = 0$; and $D(t_f, r) = 0$ for $1 \le r \le k$.

Clearly then, the compactness offered by logic from the state-space model description (25) has been successfully combined with the quantitativity from a-priori probabilistic knowledge of Nature's noise characteristics. Thus, the uncertainty of coalitive performance (26) can now be represented in a compact and robust way. Subsequently, the time-backward differential equations (28)-(33) not only offer a tractable procedure for the calculation of (27) but also allow the incorporation of a subclass of linear feedback syntheses so that coalitive stakeholders i, for all $i \in \overline{S}$ are actively mitigating the performance uncertainty. Such performance-measure statistics are therefore, referred as "information" statistics which are extremely valuable for shaping the coalitive performance distribution.

3 Problem Statements

Suffice it to say here that all the performance-measure statistics, or equivalently cumulants (27) depend in part of the known initial condition $x_a(t_0)$. Although different states $x_a(t)$ will result in different values for the traditional "performance-to-come", the cumulant values are however, functions of time-backward evolutions of the cumulant-generating variables $H(\alpha, r)$, $\breve{D}(\alpha, r)$ and $D(\alpha, r)$ that totally ignore all the values $x_a(t)$. This fact therefore makes the new optimization problem as being considered in cumulant-based control particularly unique as compared with the more traditional dynamic programming class of investigations. In other words, the time-backward trajectories (28)-(33) should be considered as the "new" dynamical equations from which the resulting Mayer optimization and associated value function in the framework of dynamic programming [3] thus depend on these "new" state variables $H(\alpha, r)$, $\breve{D}(\alpha, r)$ and $D(\alpha, r)$.

For notational simplicity, k-tuple variables $\mathcal{H}$, $\breve{\mathcal{D}}$ and $\mathcal{D}$ are introduced as the states of multi-attribute projection of future status of (26) with $\mathcal{H}(\cdot) \triangleq (\mathcal{H}_1(\cdot), \ldots, \mathcal{H}_k(\cdot))$, $\breve{\mathcal{D}}(\cdot) \triangleq (\breve{\mathcal{D}}_1(\cdot), \ldots, \breve{\mathcal{D}}_k(\cdot))$ and $\mathcal{D}(\cdot) \triangleq (\mathcal{D}_1(\cdot), \ldots, \mathcal{D}_k(\cdot))$ where each element $\mathcal{H}_r \in C^1(t_0, t_f; \mathbb{R}^{n \times n})$ of $\mathcal{H}$, each element $\breve{\mathcal{D}}_r \in C^1(t_0, t_f; \mathbb{R}^n)$ of $\breve{\mathcal{D}}$ and each element $\mathcal{D}_r \in C^1(t_0, t_f; \mathbb{R})$ of $\mathcal{D}$ have the representations of $\mathcal{H}_r(\cdot) = H(\cdot, r)$, $\breve{\mathcal{D}}_r(\cdot) = \breve{D}(\cdot, r)$ and $\mathcal{D}_r(\cdot) = D(\cdot, r)$ with the right members satisfying the dynamic equations (28)-(33) given $n \triangleq \sum_{i=1}^S (n_i + q_i)$, $m \triangleq \sum_{i=1}^S m_i$ and $m_z \triangleq \sum_{i=1}^S q_i$. Subsequently, the convenient mappings are defined by

$$\mathcal{F}_r : [t_0, t_f] \times (\mathbb{R}^{n \times n})^k \times \mathbb{R}^{m \times n} \times \mathbb{R}^{m\ \times n} \mapsto \mathbb{R}^{n \times n}$$

$$\breve{\mathcal{G}}_r : [t_0, t_f] \times (\mathbb{R}^{n \times n})^k \times (\mathbb{R}^n)^k \times \mathbb{R}^{m \times n} \times \mathbb{R}^{m\ \times n} \times \mathbb{R}^n \times \mathbb{R}^m \mapsto \mathbb{R}^n$$

$$\mathcal{G}_r : [t_0, t_f] \times (\mathbb{R}^{n \times n})^k \times (\mathbb{R}^n)^k \times \mathbb{R}^m \times \mathbb{R}^m \mapsto \mathbb{R}$$

with the rules of action

$$\begin{aligned}\mathcal{F}_1(\alpha; \mathcal{H}; K, K_z) \triangleq & -\left[A_a(\alpha) + B_a(\alpha)K(\alpha) + C_a(\alpha)K_z(\alpha)\right]^T \mathcal{H}_1(\alpha) \\ & - \mathcal{H}_1(\alpha)\left[A_a(\alpha) + B_a(\alpha)K(\alpha) + C_a(\alpha)K_z(\alpha)\right] \\ & - Q_a(\alpha) - K^T(\alpha)R_a(\alpha)K(\alpha) - K_z^T(\alpha)R_{az}(\alpha)K_z(\alpha) + 2S_a(\alpha)K_z(\alpha)\,,\end{aligned}$$

$$\begin{aligned}\mathcal{F}_r(\alpha;\mathcal{H};K,K_z) &\triangleq -\left[A_a(\alpha)+B_a(\alpha)K(\alpha)+C_a(\alpha)K_z(\alpha)\right]^T\mathcal{H}_r(\alpha)\\&\quad-\mathcal{H}_r(\alpha)\left[A_a(\alpha)+B_a(\alpha)K(\alpha)+C_a(\alpha)K_z(\alpha)\right]\\&\quad-\sum_{s=1}^{r-1}\frac{2r!}{s!(r-s)!}\mathcal{H}_s(\alpha)G_a(\alpha)W_aG_a^T(\alpha)\mathcal{H}_{r-s}(\alpha)\,,\quad 2\le r\le k\end{aligned}$$

$$\begin{aligned}\breve{\mathcal{G}}_1(\alpha;\mathcal{H},\breve{\mathcal{D}};K,K_z;p,p_z) &\triangleq -\left[A_a(\alpha)+B_a(\alpha)K(\alpha)+C_a(\alpha)K_z(\alpha)\right]^T\breve{\mathcal{D}}_1(\alpha)\\&\quad-\mathcal{H}_1(\alpha)\left[B_a(\alpha)p(\alpha)+C_a(\alpha)p_z(\alpha)+D_a(\alpha)\right]\\&\quad-K^T(\alpha)R_a(\alpha)p(\alpha)-K_z^T(\alpha)R_{az}(\alpha)p_z(\alpha)+S_a(\alpha)p_z(\alpha)\end{aligned}$$

$$\begin{aligned}\breve{\mathcal{G}}_r(\alpha;\mathcal{H},\breve{\mathcal{D}};K,K_z;p,p_z) &\triangleq -\left[A_a(\alpha)+B_a(\alpha)K(\alpha)+C_a(\alpha)K_z(\alpha)\right]^T\breve{\mathcal{D}}_r(\alpha)\\&\quad-\mathcal{H}_r(\alpha)\left[B_a(\alpha)p(\alpha)+C_a(\alpha)p_z(\alpha)+D_a(\alpha)\right]\,,\quad 2\le r\le k\end{aligned}$$

$$\begin{aligned}\mathcal{G}_1(\alpha;\mathcal{H},\breve{\mathcal{D}};p,p_z) &\triangleq -2\breve{\mathcal{D}}_1^T(\alpha)\left[B_a(\alpha)p(\alpha)+C_a(\alpha)p_z(\alpha)+D_a(\alpha)\right]\\&\quad-\mathrm{Tr}\{\mathcal{H}_1(\alpha)G_a(\alpha)W_aG_a^T(\alpha)\}-p^T(\alpha)R_a(\alpha)p(\alpha)-p_z^T(\alpha)R_{az}(\alpha)p_z(\alpha)\end{aligned}$$

$$\begin{aligned}\mathcal{G}_r(\alpha;\mathcal{H},\breve{\mathcal{D}};p,p_z) &\triangleq -2\breve{\mathcal{D}}_r^T(\alpha)\left[B_a(\alpha)p(\alpha)+C_a(\alpha)p_z(\alpha)+D_a(\alpha)\right]\\&\quad-\mathrm{Tr}\{\mathcal{H}_r(\alpha)G_a(\alpha)W_aG_a^T(\alpha)\}\,,\quad 2\le r\le k.\end{aligned}$$

Now it is straightforward to establish the product mappings

$$\mathcal{F}_1\times\cdots\times\mathcal{F}_k:[t_0,t_f]\times(\mathbb{R}^{n\times n})^k\times\mathbb{R}^{m\times n}\times\mathbb{R}^{m\ \times n}\mapsto(\mathbb{R}^{n\times n})^k$$
$$\breve{\mathcal{G}}_1\times\cdots\times\breve{\mathcal{G}}_k:[t_0,t_f]\times(\mathbb{R}^{n\times n})^k\times(\mathbb{R}^n)^k\times\mathbb{R}^{m\times n}\times\mathbb{R}^{m\ \times n}\times\mathbb{R}^n\times\mathbb{R}^m\ \mapsto(\mathbb{R}^n)^k$$
$$\mathcal{G}_1\times\cdots\times\mathcal{G}_k:[t_0,t_f]\times(\mathbb{R}^{n\times n})^k\times(\mathbb{R}^n)^k\times\mathbb{R}^m\times\mathbb{R}^m\ \mapsto(\mathbb{R})^k$$

along with the corresponding notations $\mathcal{F}\triangleq\mathcal{F}_1\times\cdots\times\mathcal{F}_k$, $\breve{\mathcal{G}}\triangleq\breve{\mathcal{G}}_1\times\cdots\times\breve{\mathcal{G}}_k$ and $\mathcal{G}\triangleq\mathcal{G}_1\times\cdots\times\mathcal{G}_k$. Thus, the dynamic equations (28)-(33) for performance uncertainty and robustness can be rewritten as follows

$$\frac{d}{d\alpha}\mathcal{H}(\alpha)=\mathcal{F}(\alpha;\mathcal{H}(\alpha);K(\alpha),K_z(\alpha))\,,\tag{34}$$
$$\frac{d}{d\alpha}\breve{\mathcal{D}}(\alpha)=\breve{\mathcal{G}}(\alpha;\mathcal{H}(\alpha),\breve{\mathcal{D}}(\alpha);K(\alpha),K_z(\alpha);p(\alpha),p_z(\alpha))\,,\tag{35}$$
$$\frac{d}{d\alpha}\mathcal{D}(\alpha)=\mathcal{G}(\alpha;\mathcal{H}(\alpha),\breve{\mathcal{D}}(\alpha);p(\alpha),p_z(\alpha))\tag{36}$$

where the terminal-value conditions $\mathcal{H}(t_f)\triangleq\mathcal{H}_f=(Q_a^f,0,\ldots,0)$, $\breve{\mathcal{D}}(t_f)\triangleq\breve{\mathcal{D}}_f=(0,\ldots,0)$ and $\mathcal{D}(t_f)\triangleq\mathcal{D}_f=(0,\ldots,0)$.

Note that the product system (34)-(36) uniquely determines $\mathcal{H}$, $\breve{\mathcal{D}}$, and $\mathcal{D}$ once the admissible 4-tuple (K,K_z,p,p_z) is specified. Hence, $\mathcal{H}$, $\breve{\mathcal{D}}$, and $\mathcal{D}$ are considered as $\mathcal{H}(\cdot,K,K_z,p,p_z)$, $\breve{\mathcal{D}}(\cdot,K,K_z,p,p_z)$, and $\mathcal{D}(\cdot,K,K_z,p,p_z)$, respectively. The performance index for the multi-person stochastic game can be formulated in terms of admissible K, K_z, p, and p_z.

Definition 2. *Performance Index.*
Let $k \in \mathbb{Z}^+$ and the sequence $\mu = \{\mu_r \geq 0\}_{r=1}^k$ with $\mu_1 > 0$. Then, for the given initial condition (t_0, x_a^0), the performance index for the coalition $\overline{S}$

$$\phi_0 : \{t_0\} \times (\mathbb{R}^{n \times n})^k \times (\mathbb{R}^n)^k \times \mathbb{R}^k \mapsto \mathbb{R}^+$$

over a finite optimization horizon is defined by

$$\begin{aligned}\phi_0 &\Big(t_0, \mathcal{H}(\cdot, K, K_z, p, p_z), \breve{\mathcal{D}}(\cdot, K, K_z, p, p_z), \mathcal{D}(\cdot, K, K_z, p, p_z)\Big) \\ &\triangleq \sum_{r=1}^k \mu_r \kappa_r(K, K_z, p, p_z) = \sum_{r=1}^k \mu_r \big[(x_a^0)^T \mathcal{H}_r(t_0, K, K_z, p, p_z) x_a^0 \\ &\qquad + 2(x_a^0)^T \breve{\mathcal{D}}_r(t_0, K, K_z, p, p_z) + \mathcal{D}_r(t_0, K, K_z, p, p_z)\big] \qquad (37)\end{aligned}$$

where all the parametric design freedom μ_r's considered here by the coalition $\overline{S}$, represent for strategic coordination and co-design towards the coalitive performance robustness. All solutions $\{\mathcal{H}_r(\alpha, K, K_z, p, p_z)\}_{r=1}^k$, $\{\breve{\mathcal{D}}_r(\alpha, K, K_z, p, p_z)\}_{r=1}^k$ and $\{\mathcal{D}_r(\alpha, K, K_z, p, p_z)\}_{r=1}^k$ when evaluated at $\alpha = t_0$ satisfy the time-backward differential equations (34)-(36) together with the terminal-value conditions $\mathcal{H}_f = (Q_a^f, 0, \ldots, 0)$, $\breve{\mathcal{D}}_f = (0, \ldots, 0)$, and $\mathcal{D}_f = (0, \ldots, 0)$.

Remark 1. *The performance index (37) associated with the coalition $\overline{S}$ is a weighted summation of some "information" statistics with μ_r representing multiple degrees of shaping the probability density function of the Chi-squared random measure of performance (26) given Nature's mixed random realizations. If all the cumulants of (26) remain bounded as the realized performance-measure (26) arbitrarily closes to 0, the first cumulant dominates the summation and the cumulant-based optimization problem reduces to the classical Linear-Quadratic-Gaussian (LQG) control problem for one decision maker.*

For the given $(t_f, \mathcal{H}_f, \breve{\mathcal{D}}_f, \mathcal{D}_f)$, the class $\mathcal{K}_{t\ ,\mathcal{H}\ ,\breve{\mathcal{D}}\ ,\mathcal{D}\ ;\mu}$ of admissible 4-tuple (K, K_z, p, p_z) is then defined.

Definition 3. *Admissible Feedback Gains and Affine Inputs.*
Let compact subsets $\overline{K} \subset \mathbb{R}^{n \times n}$, $\overline{K}_z$, $\overline{P} \subset \mathbb{R}^m$, and $\overline{P}_z \subset \mathbb{R}^m$ be the sets of allowable matrices and vectors. Then, for the given $k \in \mathbb{Z}^+$, $\mu = \{\mu_r \geq 0\}_{r=1}^k$ with $\mu_1 > 0$, the sets of matrix-valued feedback gains $\mathcal{K}_{t\ ,\mathcal{H}\ ,\breve{\mathcal{D}}\ ,\mathcal{D}\ ;\mu} \in C(t_0, t_f; \mathbb{R}^{m \times n})$ and $\mathcal{K}^z_{t\ ,\mathcal{H}\ ,\breve{\mathcal{D}}\ ,\mathcal{D}\ ;\mu} \in C(t_0, t_f; \mathbb{R}^{m\ \times n})$ in addition to the sets of vector-valued affine inputs $\mathcal{P}_{t\ ,\mathcal{H}\ ,\breve{\mathcal{D}}\ ,\mathcal{D}\ ;\mu} \in C(t_0, t_f; \mathbb{R}^m)$ and $\mathcal{P}^z_{t\ ,\mathcal{H}\ ,\breve{\mathcal{D}}\ ,\mathcal{D}\ ;\mu} \in C(t_0, t_f; \mathbb{R}^m\ )$ with respective values $K(\cdot) \in \overline{K}$, $K_z(\cdot) \in \overline{K}_z$, $p(\cdot) \in \overline{P}$, and $p_z(\cdot) \in \overline{P}_z$ are admissible if the resulting solutions to the time-backward differential equations (34)-(36) exist on the finite horizon $[t_0, t_f]$.

The optimization problem for the stochastic cooperative game with uncertainty and robustness where instantiations are aimed at reducing coalitive performance dispersion and constituent strategies are robust to Nature's stochastic variability, is stated.

Definition 4. *Optimization Problem.*
Suppose that $k \in \mathbb{Z}^+$ and the sequence $\mu = \{\mu_r \geq 0\}_{r=1}^k$ with $\mu_1 > 0$ are fixed. Then, the cumulant-based optimization problem over $[t_0, t_f]$ is given by the minimization of the coalitive performance index (37) over all $K(\cdot) \in \mathcal{K}_{t\ ,\mathcal{H}\ ,\breve{\mathcal{D}}\ ,\mathcal{D}\ ;\mu}$, $K_z(\cdot) \in \mathcal{K}^z_{t\ ,\mathcal{H}\ ,\breve{\mathcal{D}}\ ,\mathcal{D}\ ;\mu}$, $p(\cdot) \in \mathcal{P}_{t\ ,\mathcal{H}\ ,\breve{\mathcal{D}}\ ,\mathcal{D}\ ;\mu}$, and $p_z(\cdot) \in \mathcal{P}^z_{t\ ,\mathcal{H}\ ,\breve{\mathcal{D}}\ ,\mathcal{D}\ ;\mu}$ subject to the time-backward dynamical constraints (34)-(36) for $\alpha \in [t_0, t_f]$.

Since a direct dynamic programming approach is taken here, it is therefore necessary to introduce the value function $\mathcal{V}(\varepsilon, \mathcal{Y}, \breve{\mathcal{Z}}, \mathcal{Z})$ which is continuously differentiable for the stochastic cooperative game starting at $(\varepsilon, \mathcal{Y}, \breve{\mathcal{Z}}, \mathcal{Z})$.

Definition 5. *Value Function.*
The value function $\mathcal{V} : [t_0, t_f] \times (\mathbb{R}^{n \times n})^k \times (\mathbb{R}^n)^k \times \mathbb{R}^k \mapsto \mathbb{R}^+ \cup \{+\infty\}$ associated with the Mayer problem defined as $\mathcal{V}(\varepsilon, \mathcal{Y}, \breve{\mathcal{Z}}, \mathcal{Z})$ is the minimization of $\phi_0\left(t_0, \mathcal{H}(\cdot, K, K_z, p, p_z), \breve{\mathcal{D}}(\cdot, K, K_z, p, p_z), \mathcal{D}(\cdot, K, K_z, p, p_z)\right)$ over all admissible 4-tuple of $K(\cdot) \in \mathcal{K}_{t\ ,\mathcal{H}\ ,\breve{\mathcal{D}}\ ,\mathcal{D}\ ;\mu}$, $K_z(\cdot) \in \mathcal{K}^z_{t\ ,\mathcal{H}\ ,\breve{\mathcal{D}}\ ,\mathcal{D}\ ;\mu}$, $p(\cdot) \in \mathcal{P}_{t\ ,\mathcal{H}\ ,\breve{\mathcal{D}}\ ,\mathcal{D}\ ;\mu}$, and $p_z(\cdot) \in \mathcal{P}^z_{t\ ,\mathcal{H}\ ,\breve{\mathcal{D}}\ ,\mathcal{D}\ ;\mu}$ subjected to some $(\varepsilon, \mathcal{Y}, \breve{\mathcal{Z}}, \mathcal{Z}) \in [t_0, t_f] \times (\mathbb{R}^{n \times n})^k \times (\mathbb{R}^n)^k \times \mathbb{R}^k$.

The development illustrated here for cumulant-based control theory is motivated by the excellent treatment in [3], and is intended to follow it closely. Unless otherwise specified, the dependence of trajectory solutions $\mathcal{H}$, $\breve{\mathcal{D}}$, and $\mathcal{D}$ on admissible K, K_z, p, and p_z is now omitted for notational clarity.

Theorem 2. *Necessary Conditions.*
The value function associated with the coalition $\overline{S}$ evaluated along any time-backward trajectory corresponding to decentralized decision strategies feasible for its terminal states is an increasing function of time. Moreover, the value function evaluated along any optimal time-backward trajectory is constant.

It is important to note that the properties aforementioned are necessary conditions for optimality. The theorem in the sequel shows that these conditions are also sufficient for optimality and serve as a construction of potential candidates for the value function.

Theorem 3. *Sufficient Condition.*
Let $\mathcal{W}(\varepsilon, \mathcal{Y}, \breve{\mathcal{Z}}, \mathcal{Z})$ be a real-valued function defined on $[t_0, t_f] \times (\mathbb{R}^{n \times n})^k \times (\mathbb{R}^n)^k \times \mathbb{R}^k$ such that $\mathcal{W}(\varepsilon, \mathcal{Y}, \breve{\mathcal{Z}}, \mathcal{Z}) = \phi_0(\varepsilon, \mathcal{Y}, \breve{\mathcal{Z}}, \mathcal{Z})$.

Let t_f, $\mathcal{H}_f$, $\breve{\mathcal{D}}$, $\mathcal{D}_f$ be given terminal-value conditions and let, for each 3-tuple trajectory $(\mathcal{H}, \breve{\mathcal{D}}, \mathcal{D})$ corresponding to the 4-tuple coalitive strategy (K, K_z, p, p_z) in $\mathcal{K}_{t\ ,\mathcal{H}\ ,\breve{\mathcal{D}}\ ,\mathcal{D}\ ;\mu} \times \mathcal{K}^z_{t\ ,\mathcal{H}\ ,\breve{\mathcal{D}}\ ,\mathcal{D}\ ;\mu} \times \mathcal{P}_{t\ ,\mathcal{H}\ ,\breve{\mathcal{D}}\ ,\mathcal{D}\ ;\mu} \times \mathcal{P}^z_{t\ ,\mathcal{H}\ ,\breve{\mathcal{D}}\ ,\mathcal{D}\ ;\mu}$, the real function $\mathcal{W}(\alpha, \mathcal{H}(\alpha), \breve{\mathcal{D}}(\alpha), \mathcal{D}(\alpha))$ be finite and time-backward increasing on $[t_0, t_f]$.

If (K^*, K_z^*, p^*, p_z^*) is a coalitive choice in $\mathcal{K}_{t\ ,\mathcal{H}\ ,\breve{\mathcal{D}}\ ,\mathcal{D}\ ;\mu} \times \mathcal{K}^z_{t\ ,\mathcal{H}\ ,\breve{\mathcal{D}}\ ,\mathcal{D}\ ;\mu} \times \mathcal{P}_{t\ ,\mathcal{H}\ ,\breve{\mathcal{D}}\ ,\mathcal{D}\ ;\mu} \times \mathcal{P}^z_{t\ ,\mathcal{H}\ ,\breve{\mathcal{D}}\ ,\mathcal{D}\ ;\mu}$ such that for the corresponding 3-tuple trajectory $(\mathcal{H}^*, \breve{\mathcal{D}}^*, \mathcal{D}^*)$, $\mathcal{W}(\alpha, \mathcal{H}^*(\alpha), \breve{\mathcal{D}}^*(\alpha), \mathcal{D}^*(\alpha))$ is constant then this choice of (K^*, K_z^*, p^*, p_z^*) is the optimal and $\mathcal{W}(t_f, \mathcal{H}_f, \breve{\mathcal{D}}_f, \mathcal{D}_f) = \mathcal{V}(t_f, \mathcal{H}_f, \breve{\mathcal{D}}_f, \mathcal{D}_f)$.

Definition 6. *Reachable Set.*
Let the reachable set $\mathcal{Q}$ *be defined as follows*

$$\mathcal{Q} \triangleq \Big\{(\varepsilon, \mathcal{Y}, \breve{\mathcal{Z}}, \mathcal{Z}) \in [t_0, t_f] \times (\mathbb{R}^{n\times n})^k \times (\mathbb{R}^n)^k \times \mathbb{R}^k$$
$$\text{such that } \mathcal{K}_{t\ ,\mathcal{H}\ ,\breve{\mathcal{D}}\ ,\mathcal{D}\ ;\mu} \times \mathcal{K}^z_{t\ ,\mathcal{H}\ ,\breve{\mathcal{D}}\ ,\mathcal{D}\ ;\mu} \times \mathcal{P}_{t\ ,\mathcal{H}\ ,\breve{\mathcal{D}}\ ,\mathcal{D}\ ;\mu} \times \mathcal{P}^z_{t\ ,\mathcal{H}\ ,\breve{\mathcal{D}}\ ,\mathcal{D}\ ;\mu} \neq 0\Big\}.$$

Moreover, it can be shown that the value function is satisfying a partial differential equation at each interior point of $\mathcal{Q}$ at which it is differentiable.

Theorem 4. *HJB Equation-Mayer Problem.*
Let $(\varepsilon, \mathcal{Y}, \breve{\mathcal{Z}}, \mathcal{Z})$ be any interior point of the reachable set $\mathcal{Q}$ at which the value function $\mathcal{V}(\varepsilon, \mathcal{Y}, \breve{\mathcal{Z}}, \mathcal{Z})$ is differentiable. If there exists an optimal 4-tuple strategy $(K^*, K_z^*, p^*, p_z^*) \in \mathcal{K}_{t\ ,\mathcal{H}\ ,\breve{\mathcal{D}}\ ,\mathcal{D}\ ;\mu} \times \mathcal{K}^z_{t\ ,\mathcal{H}\ ,\breve{\mathcal{D}}\ ,\mathcal{D}\ ;\mu} \times \mathcal{P}_{t\ ,\mathcal{H}\ ,\breve{\mathcal{D}}\ ,\mathcal{D}\ ;\mu} \times \mathcal{P}^z_{t\ ,\mathcal{H}\ ,\breve{\mathcal{D}}\ ,\mathcal{D}\ ;\mu}$, then the partial differential equation of the cooperative game

$$\begin{aligned}
0 = \min_{K\in\overline{K}, K\ \in\overline{K}\ , p\in\overline{P}, p\ \in\overline{P}} \Big\{ & \frac{\partial}{\partial\varepsilon}\mathcal{V}(\varepsilon, \mathcal{Y}, \breve{\mathcal{Z}}, \mathcal{Z}) \\
& + \frac{\partial}{\partial\,\mathrm{vec}(\mathcal{Y})}\mathcal{V}(\varepsilon, \mathcal{Y}, \breve{\mathcal{Z}}, \mathcal{Z}) \cdot \mathrm{vec}(\mathcal{F}(\varepsilon; \mathcal{Y}; K, K_z)) \\
& + \frac{\partial}{\partial\,\mathrm{vec}(\breve{\mathcal{Z}})}\mathcal{V}(\varepsilon, \mathcal{Y}, \breve{\mathcal{Z}}, \mathcal{Z}) \cdot \mathrm{vec}(\breve{\mathcal{G}}(\varepsilon; \mathcal{Y}, \breve{\mathcal{Z}}; K, K_z; p, p_z)) \\
& + \frac{\partial}{\partial\,\mathrm{vec}(\mathcal{Z})}\mathcal{V}(\varepsilon, \mathcal{Y}, \breve{\mathcal{Z}}, \mathcal{Z}) \cdot \mathrm{vec}(\mathcal{G}(\varepsilon; \mathcal{Y}, \breve{\mathcal{Z}}; p, p_z))\Big\} \qquad (38)
\end{aligned}$$

is satisfied together with $\mathcal{V}(t_0, \mathcal{H}_0, \breve{\mathcal{D}}_0, \mathcal{D}_0) = \phi_0(t_0, \mathcal{H}_0, \breve{\mathcal{D}}_0, \mathcal{D}_0)$ and vec$(\cdot)$ the vectorizing operator of enclosed entities. The optimum in (38) is achieved by the 4-tuple $(K^*(\varepsilon), K_z^*(\varepsilon), p^*(\varepsilon), p_z^*(\varepsilon))$ of the optimal decision strategy at ε.

Finally, the following theorem gives the sufficient condition used to verify optimal decisions for the coalition $\overline{S}$.

Theorem 5. *Verification Theorem.*
Fix $k \in \mathbb{Z}^+$. Let $\mathcal{W}(\varepsilon, \mathcal{Y}, \breve{\mathcal{Z}}, \mathcal{Z})$ be a continuously differentiable solution of the HJB equation (38) and satisfy the boundary condition

$$\mathcal{W}(t_0, \mathcal{H}_0, \breve{\mathcal{D}}_0, \mathcal{D}_0) = \phi_0\left(t_0, \mathcal{H}_0, \breve{\mathcal{D}}_0, \mathcal{D}_0\right). \qquad (39)$$

Let $(t_f, \mathcal{H}_f, \breve{\mathcal{D}}_f, \mathcal{D}_f)$ be a point of $\mathcal{Q}$; 4-tuple (K, K_z, p, p_z) in $\mathcal{K}_{t\ ,\mathcal{H}\ ,\breve{\mathcal{D}}\ ,\mathcal{D}\ ;\mu} \times \mathcal{K}^z_{t\ ,\mathcal{H}\ ,\breve{\mathcal{D}}\ ,\mathcal{D}\ ;\mu} \times \mathcal{P}_{t\ ,\mathcal{H}\ ,\breve{\mathcal{D}}\ ,\mathcal{D}\ ;\mu} \times \mathcal{P}^z_{t\ ,\mathcal{H}\ ,\breve{\mathcal{D}}\ ,\mathcal{D}\ ;\mu}$; and $\mathcal{H}$, $\breve{\mathcal{D}}$, and $\mathcal{D}$ the corresponding solutions of the equations (34)-(36). Then, $\mathcal{W}(\alpha, \mathcal{H}(\alpha), \breve{\mathcal{D}}(\alpha), \mathcal{D}(\alpha))$ is *time-backward increasing in* α. *If* (K^*, K_z^*, p^*, p_z^*) *is a 4-tuple in* $\mathcal{K}_{t\ ,\mathcal{H}\ ,\breve{\mathcal{D}}\ ,\mathcal{D}\ ;\mu} \times \mathcal{K}^z_{t\ ,\mathcal{H}\ ,\breve{\mathcal{D}}\ ,\mathcal{D}\ ;\mu} \times \mathcal{P}_{t\ ,\mathcal{H}\ ,\breve{\mathcal{D}}\ ,\mathcal{D}\ ;\mu} \times \mathcal{P}^z_{t\ ,\mathcal{H}\ ,\breve{\mathcal{D}}\ ,\mathcal{D}\ ;\mu}$ *defined on* $[t_0, t_f]$ *with corresponding solutions,* $\mathcal{H}^*$, $\breve{\mathcal{D}}^*$ *and* $\mathcal{D}^*$ *of the dynamical equations such that*

$$\begin{aligned}0 = \frac{\partial}{\partial\varepsilon}\mathcal{W}(\alpha, \mathcal{H}^*(\alpha), \breve{\mathcal{D}}^*(\alpha), \mathcal{D}^*(\alpha)) + \frac{\partial}{\partial\,\mathrm{vec}(\mathcal{Y})}\mathcal{W}(\alpha; \mathcal{H}^*(\alpha), \breve{\mathcal{D}}^*(\alpha), \mathcal{D}^*(\alpha))\\ \cdot \mathrm{vec}(\mathcal{F}(\alpha; \mathcal{H}^*(\alpha); K^*(\alpha), K_z^*(\alpha))) + \frac{\partial}{\partial\,\mathrm{vec}(\breve{\mathcal{Z}})}\mathcal{W}(\alpha, \mathcal{H}^*(\alpha), \breve{\mathcal{D}}^*(\alpha), \mathcal{D}^*(\alpha))\\ \cdot \mathrm{vec}(\breve{\mathcal{G}}(\alpha; \mathcal{H}^*(\alpha)), \breve{\mathcal{Z}}^*(\alpha); K^*(\alpha), K_z^*(\alpha); p^*(\alpha), p_z^*(\alpha))\\ + \frac{\partial}{\partial \mathrm{vec}(\mathcal{Z})}\mathcal{W}(\alpha, \mathcal{H}^*(\alpha), \breve{\mathcal{D}}^*(\alpha), \mathcal{D}^*(\alpha))\mathrm{vec}(\mathcal{G}(\alpha; \mathcal{H}^*(\alpha), \breve{\mathcal{Z}}^*(\alpha); p^*(\alpha), p_z^*(\alpha))\end{aligned} \quad (40)$$

then, 4-tuple (K^*, K_z^*, p^*, p_z^*) *in* $\mathcal{K}_{t\ ,\mathcal{H}\ ,\breve{\mathcal{D}}\ ,\mathcal{D}\ ;\mu} \times \mathcal{K}^z_{t\ ,\mathcal{H}\ ,\breve{\mathcal{D}}\ ,\mathcal{D}\ ;\mu} \times \mathcal{P}_{t\ ,\mathcal{H}\ ,\breve{\mathcal{D}}\ ,\mathcal{D}\ ;\mu} \times \mathcal{P}^z_{t\ ,\mathcal{H}\ ,\breve{\mathcal{D}}\ ,\mathcal{D}\ ;\mu}$ *is optimal and*

$$\mathcal{W}(\varepsilon, \mathcal{Y}, \breve{\mathcal{Z}}, \mathcal{Z}) = \mathcal{V}(\varepsilon, \mathcal{Y}, \breve{\mathcal{Z}}, \mathcal{Z}) \quad (41)$$

where $\mathcal{V}(\varepsilon, \mathcal{Y}, \breve{\mathcal{Z}}, \mathcal{Z})$ *is the value function.*

Remark 2. *To have a cooperative solution from the choice* $(K^*, K_z^*, p^*, p_z^*) \in \mathcal{K}_{t\ ,\mathcal{H}\ ,\breve{\mathcal{D}}\ ,\mathcal{D}\ ;\mu} \times \mathcal{K}^z_{t\ ,\mathcal{H}\ ,\breve{\mathcal{D}}\ ,\mathcal{D}\ ;\mu} \times \mathcal{P}_{t\ ,\mathcal{H}\ ,\breve{\mathcal{D}}\ ,\mathcal{D}\ ;\mu} \times \mathcal{P}^z_{t\ ,\mathcal{H}\ ,\breve{\mathcal{D}}\ ,\mathcal{D}\ ;\mu}$ *well defined and continuous for all* $\alpha \in [t_0, t_f]$, *the trajectory solutions* $\mathcal{H}(\alpha)$, $\breve{\mathcal{D}}(\alpha)$ *and* $\mathcal{D}(\alpha)$ *to the dynamical equations (34)-(36) when evaluated at* $\alpha = t_0$ *must then exist. Therefore, it is necessary that* $\mathcal{H}(\alpha)$, $\breve{\mathcal{D}}(\alpha)$ *and* $\mathcal{D}(\alpha)$ *are finite for all* $\alpha \in [t_0, t_f)$. *Moreover, the solutions of the dynamical equations (34)-(36) exist and are continuously differentiable in a neighborhood of* t_f. *Applying the results from [1], these trajectory solutions can further be extended to the left of* t_f *as long as* $\mathcal{H}(\alpha)$, $\breve{\mathcal{D}}(\alpha)$ *and* $\mathcal{D}(\alpha)$ *remain finite. Hence, the existence of unique and continuously differentiable solutions to the equations (34) through (36) is certain if* $\mathcal{H}(\alpha)$, $\breve{\mathcal{D}}(\alpha)$ *and* $\mathcal{D}(\alpha)$ *are bounded for all* $\alpha \in [t_0, t_f)$. *As a result, the candidate value function* $\mathcal{W}(\alpha, \mathcal{H}, \breve{\mathcal{D}}, \mathcal{D})$ *is continuously differentiable as well.*

4 Coalitive Pareto Decision Strategies

Recall that the optimization problem being considered herein is in "Mayer form" and can be solved by applying an adaptation of the Mayer form verification theorem of dynamic programming given in [3]. In the framework of dynamic programming, it is often required to denote the terminal time and states of a family of optimization problems as $(\varepsilon, \mathcal{Y}, \breve{\mathcal{Z}}, \mathcal{Z})$ rather than $(t_f, \mathcal{H}_f, \breve{\mathcal{D}}_f, \mathcal{D}_f)$. Stating precisely, for $\varepsilon \in [t_0, t_f]$ and $1 \le r \le k$, the states of the performance robustness (34)-(36) defined on the interval $[t_0, \varepsilon]$ have the terminal values denoted by $\mathcal{H}(\varepsilon) \equiv \mathcal{Y}$, $\breve{\mathcal{D}}(\varepsilon) \equiv \breve{\mathcal{Z}}$ and $\mathcal{D}(\varepsilon) \equiv \mathcal{Z}$.

Since the performance index (37) is quadratic affine in terms of arbitrarily fixed x_a^0, this observation suggests a solution to the HJB equation (38) may be of the form as follows. It is assumed that $(\varepsilon, \mathcal{Y}, \breve{\mathcal{Z}}, \mathcal{Z})$ is any interior point of the reachable set $\mathcal{Q}$ at which the real-valued function

$$\mathcal{W}(\varepsilon, \mathcal{Y}, \breve{\mathcal{Z}}, \mathcal{Z}) = (x_a^0)^T \sum_{r=1}^{k} \mu_r (\mathcal{Y}_r + \mathcal{E}_r(\varepsilon))\, x_a^0 \\ + 2(x_a^0)^T \sum_{r=1}^{k} \mu_r (\breve{\mathcal{Z}}_r + \breve{\mathcal{T}}_r(\varepsilon)) + \sum_{r=1}^{k} \mu_r (\mathcal{Z}_r + \mathcal{T}_r(\varepsilon)) \quad (42)$$

is differentiable. The parametric functions of time $\mathcal{E}_r \in C^1(t_0, t_f; \mathbb{R}^{n\times n})$, $\breve{\mathcal{T}}_r \in C^1(t_0, t_f; \mathbb{R}^n)$ and $\mathcal{T}_r \in C^1(t_0, t_f; \mathbb{R})$ are yet to be determined. Furthermore, the time derivative of $\mathcal{W}(\varepsilon, \mathcal{Y}, \breve{\mathcal{Z}}, \mathcal{Z})$ can be shown to be

$$\frac{d}{d\varepsilon}\mathcal{W}(\varepsilon, \mathcal{Y}, \breve{\mathcal{Z}}, \mathcal{Z}) = (x_a^0)^T \sum_{r=1}^{k} \mu_r \left(\mathcal{F}_r(\varepsilon; \mathcal{Y}; K, K_z) + \frac{d}{d\varepsilon}\mathcal{E}_r(\varepsilon) \right) x_a^0 \\ + 2(x_a^0)^T \sum_{r=1}^{k} \mu_r \left(\breve{\mathcal{G}}_r(\varepsilon; \mathcal{Y}, \breve{\mathcal{Z}}; K, K_z; p, p_z) + \frac{d}{d\varepsilon}\breve{\mathcal{T}}_r(\varepsilon) \right) \\ + \sum_{r=1}^{k} \mu_r \left(\mathcal{G}_r(\varepsilon, \mathcal{Y}, \breve{\mathcal{Z}}; p, p_z) + \frac{d}{d\varepsilon}\mathcal{T}_r(\varepsilon) \right). \quad (43)$$

The substitution of this hypothesized solution (42) into the HJB equation (38) and making use of (43) results in

$$0 \equiv \min_{K\in\overline{K}, K\ \in\overline{K}\ , p\in\overline{P}, p\ \in\overline{P}} \left\{ (x_a^0)^T \left(\sum_{r=1}^{k} \mu_r \frac{d}{d\varepsilon}\mathcal{E}_r(\varepsilon) \right) x_a^0 + \sum_{r=1}^{k} \mu_r \frac{d}{d\varepsilon}\mathcal{T}_r(\varepsilon) \right. \\ + 2(x_a^0)^T \left(\sum_{r=1}^{k} \mu_r \frac{d}{d\varepsilon}\breve{\mathcal{T}}_r(\varepsilon) \right) + (x_a^0)^T \left(\sum_{r=1}^{k} \mu_r \mathcal{F}_r(\varepsilon; \mathcal{Y}; K, K_z) \right) x_a^0 \\ \left. + 2(x_a^0)^T \left(\sum_{r=1}^{k} \mu_r \breve{\mathcal{G}}_r(\varepsilon; \mathcal{Y}, \breve{\mathcal{Z}}; K, K_z; p, p_z) \right) + \sum_{r=1}^{k} \mu_r \mathcal{G}_r(\varepsilon; \mathcal{Y}, \breve{\mathcal{Z}}; p, p_z) \right\}. \quad (44)$$

Differentiating the expression within the bracket of (44) with respect to K, K_z, p, and p_z yields the necessary conditions for an extremum of (37) on $[t_0, \varepsilon]$,

$$0 \equiv \left(B_a^T(\varepsilon) \sum_{r=1}^{k} \mu_r \mathcal{Y}_r + \mu_1 R_a(\varepsilon) K \right) M_0 + \left(B_a^T(\varepsilon) \sum_{r=1}^{k} \mu_r \breve{\mathcal{Z}}_r + \mu_1 R_a(\varepsilon) p \right) (x_a^0)^T$$

$$0 \equiv \left(C_a^T(\varepsilon) \sum_{r=1}^{k} \mu_r \mathcal{Y}_r + \mu_1 R_{az}(\varepsilon) K_z - \mu_1 S_a^T(\varepsilon) \right) M_0 \\ + \left(C_a^T(\varepsilon) \sum_{r=1}^{k} \mu_r \breve{\mathcal{Z}}_r + \mu_1 R_{az}(\varepsilon) p_z \right) (x_a^0)^T$$

$$0 \equiv \left(B_a^T(\varepsilon) \sum_{r=1}^{k} \mu_r \mathcal{Y}_r + \mu_1 R_a(\varepsilon) K \right) x_a^0 + B_a^T(\varepsilon) \sum_{r=1}^{k} \mu_r \breve{\mathcal{Z}}_r + \mu_1 R_a(\varepsilon) p$$

$$0 \equiv \left(C_a^T(\varepsilon) \sum_{r=1}^{k} \mu_r \mathcal{Y}_r + \mu_1 R_{az}(\varepsilon) K_z - \mu_1 S_a^T(\varepsilon) \right) x_a^0 + C_a^T(\varepsilon) \sum_{r=1}^{k} \mu_r \breve{\mathcal{Z}}_r + \mu_1 R_{az}(\varepsilon) p_z$$

Because all the $M_0 \triangleq x_a^0 (x_a^0)^T$, $(x_a^0)^T$, and x_a^0 have arbitrary ranks of one , it must be true that

$$K = -(\mu_1 R_a(\varepsilon))^{-1} B_a^T(\varepsilon) \sum_{r=1}^{k} \mu_r \mathcal{Y}_r \,, \tag{45}$$

$$K_z = -(\mu_1 R_{az}(\varepsilon))^{-1} C_a^T(\varepsilon) \sum_{r=1}^{k} \mu_r \mathcal{Y}_r + R_{az}^{-1}(\varepsilon) S_a^T(\varepsilon) \,, \tag{46}$$

$$p = -(\mu_1 R_a(\varepsilon))^{-1} B_a^T(\varepsilon) \sum_{r=1}^{k} \mu_r \breve{\mathcal{Z}}_r \,, \tag{47}$$

$$p_z = -(\mu_1 R_{az}(\varepsilon))^{-1} C_a^T(\varepsilon) \sum_{r=1}^{k} \mu_r \breve{\mathcal{Z}}_r \,. \tag{48}$$

Replacing these results (45)-(48) into the right member of the HJB equation (38) yields the value of the minimum

$$(x_a^0)^T \left(\sum_{r=1}^{k} \mu_r \frac{d}{d\varepsilon} \mathcal{E}_r(\varepsilon) \right) x_a^0 + 2 (x_a^0)^T \left(\sum_{r=1}^{k} \mu_r \frac{d}{d\varepsilon} \breve{\mathcal{T}}_r(\varepsilon) \right) + \sum_{r=1}^{k} \mu_r \frac{d}{d\varepsilon} \mathcal{T}_r(\varepsilon)$$
$$+ (x_a^0)^T \Bigg\{ - \Bigg[A_a(\varepsilon) - B_a(\varepsilon)(\mu_1 R_a(\varepsilon))^{-1} B_a^T(\varepsilon) \sum_{s=1}^{k} \mu_s \mathcal{Y}_s$$
$$- C_a(\varepsilon)(\mu_1 R_{az}(\varepsilon))^{-1} C_a^T(\varepsilon) \sum_{s=1}^{k} \mu_s \mathcal{Y}_s + C_a(\varepsilon) R_{az}^{-1}(\varepsilon) S_a^T(\varepsilon) \Bigg]^T \sum_{r=1}^{k} \mu_r \mathcal{Y}_r$$
$$- \sum_{r=1}^{k} \mu_r \mathcal{Y}_r \Bigg[A_a(\varepsilon) - B_a(\varepsilon)(\mu_1 R_a(\varepsilon))^{-1} B_a^T(\varepsilon) \sum_{s=1}^{k} \mu_s \mathcal{Y}_s$$
$$- C_a(\varepsilon)(\mu_1 R_{az}(\varepsilon))^{-1} C_a^T(\varepsilon) \sum_{s=1}^{k} \mu_s \mathcal{Y}_s + C_a(\varepsilon) R_{az}^{-1}(\varepsilon) S_a^T(\varepsilon) \Bigg] - \mu_1 Q_a(\varepsilon)$$
$$- \mu_1 \sum_{s=1}^{k} \mu_s \mathcal{Y}_s B_a(\varepsilon)(\mu_1 R_a(\varepsilon))^{-1} R_a(\varepsilon)(\mu_1 R_a(\varepsilon))^{-1} B_a^T(\varepsilon) \sum_{q=1}^{k} \mu_q \mathcal{Y}_q$$
$$- \mu_1 \Bigg[- \sum_{s=1}^{k} \mu_s \mathcal{Y}_s C_a(\varepsilon)(\mu_1 R_{az}(\varepsilon))^{-1} + S_a(\varepsilon) R_{az}^{-1}(\varepsilon) \Bigg] R_{az}^{-1}(\varepsilon)$$

$$
\begin{aligned}
&\cdot \left[-(\mu_1 R_{az}(\varepsilon))^{-1} C_a^T(\varepsilon) \sum_{q=1}^{k} \mu_q \mathcal{Y}_q + R_{az}^{-1}(\varepsilon) S_a^T(\varepsilon) \right] \\
&\quad + 2\mu_1 S_a(\varepsilon) \left[-(\mu_1 R_{az}(\varepsilon))^{-1} C_a^T(\varepsilon) \sum_{s=1}^{k} \mu_s \mathcal{Y}_s + R_{az}^{-1}(\varepsilon) S_a^T(\varepsilon) \right] \\
&\quad - \sum_{r=2}^{k} \mu_r \sum_{v=1}^{r-1} \frac{2v!}{v!(r-v)!} \mathcal{Y}_v G_a(\varepsilon) W_a G_a^T(\varepsilon) \mathcal{Y}_{r-v} \Bigg\} x_a^0 \\
&\quad + 2(x_a^0)^T \Bigg\{ - \Bigg[A_a(\varepsilon) - B_a(\varepsilon)(\mu_1 R_a(\varepsilon))^{-1} B_a^T(\varepsilon) \sum_{s=1}^{k} \mu_s \mathcal{Y}_s \\
&\quad - C_a(\varepsilon)(\mu_1 R_{az}(\varepsilon))^{-1} C_a^T(\varepsilon) \sum_{s=1}^{k} \mu_s \mathcal{Y}_s + C_a(\varepsilon) R_{az}^{-1}(\varepsilon) S_a^T(\varepsilon) \Bigg]^T \sum_{r=1}^{k} \mu_r \breve{\mathcal{Z}}_r \\
&\quad - \sum_{r=1}^{k} \mu_r \mathcal{Y}_r \Bigg[- B_a(\varepsilon)(\mu_1 R_a(\varepsilon))^{-1} B_a^T(\varepsilon) \sum_{s=1}^{k} \mu_s \breve{\mathcal{Z}}_s \\
&\quad - C_a(\varepsilon)(\mu_1 R_{az}(\varepsilon))^{-1} C_a^T(\varepsilon) \sum_{s=1}^{k} \mu_s \breve{\mathcal{Z}}_s + D_a(\varepsilon) \Bigg] \\
&\quad - \mu_1 \sum_{s=1}^{k} \mu_s \mathcal{Y}_s B_a(\varepsilon)(\mu_1 R_a(\varepsilon))^{-1} R_a(\varepsilon)(\mu_1 R_a(\varepsilon))^{-1} B_a^T(\varepsilon) \sum_{q=1}^{k} \mu_q \breve{\mathcal{Z}}_q \\
&\quad + \mu_1 \left[- \sum_{s=1}^{k} \mu_s \mathcal{Y}_s C_a(\varepsilon)(\mu_1 R_{az}(\varepsilon))^{-1} + S_a(\varepsilon) R_{az}^{-1}(\varepsilon) \right] R_{az}(\varepsilon)(\mu_1 R_{az}(\varepsilon))^{-1} C_a^T(\varepsilon) \\
&\quad \cdot \sum_{q=1}^{k} \mu_q \breve{\mathcal{Z}}_q - \mu_1 S_a(\varepsilon)(\mu_1 R_{az}(\varepsilon))^{-1} C_a^T(\varepsilon) \sum_{s=1}^{k} \mu_s \breve{\mathcal{Z}}_s \Bigg\} \\
&\quad - 2 \sum_{r=1}^{k} \mu_r \breve{\mathcal{Z}}_r \Bigg[- B_a(\varepsilon)(\mu_1 R_a(\varepsilon))^{-1} B_a^T(\varepsilon) \sum_{s=1}^{k} \mu_s \breve{\mathcal{Z}}_s \\
&\quad - C_a(\varepsilon)(\mu_1 R_{az}(\varepsilon))^{-1} C_a^T(\varepsilon) \sum_{s=1}^{k} \mu_s \breve{\mathcal{Z}}_s + D_a(\varepsilon) \Bigg] \\
&\quad - \mathrm{Tr} \left\{ \sum_{r=1}^{k} \mu_r \mathcal{Y}_r G_a(\varepsilon) W_a G_a^T(\varepsilon) \right\} \\
&\quad - \mu_1 \sum_{s=1}^{k} \mu_s \breve{\mathcal{Z}}_s B_a(\varepsilon)(\mu_1 R_a(\varepsilon))^{-1} R_a(\varepsilon)(\mu_1 R_a(\varepsilon))^{-1} B_a^T(\varepsilon) \sum_{q=1}^{k} \mu_q \breve{\mathcal{Z}}_q \\
&\quad - \mu_1 \sum_{s=1}^{k} \mu_s \breve{\mathcal{Z}}_s C_a(\varepsilon)(\mu_1 R_{az}(\varepsilon))^{-1} R_{az}(\varepsilon)(\mu_1 R_{az}(\varepsilon))^{-1} C_a^T(\varepsilon) \sum_{q=1}^{k} \mu_q \breve{\mathcal{Z}}_q \,. \qquad (49)
\end{aligned}
$$

It is now necessary to exhibit $\{\mathcal{E}_r(\cdot)\}_{r=1}^k$, $\{\breve{\mathcal{T}}_r(\cdot)\}_{r=1}^k$, and $\{\mathcal{T}_r(\cdot)\}_{r=1}^k$ which render the left side of the HJB equation (38) equal to zero for $\varepsilon \in [t_0, t_f]$, when $\{\mathcal{Y}_r\}_{r=1}^k$, $\{\breve{\mathcal{Z}}_r\}_{r=1}^k$, and $\{\mathcal{Z}_r\}_{r=1}^k$ are evaluated along the solution trajectories of the dynamical equations (34) through (36). With a careful examination of the expression (49), it reveals that

$$
\begin{aligned}
\frac{d}{d\varepsilon}\mathcal{E}_1(\varepsilon) = \Bigg[& A_a(\varepsilon) - B_a(\varepsilon)(\mu_1 R_a(\varepsilon))^{-1} B_a^T(\varepsilon) \sum_{s=1}^k \mu_s \mathcal{H}_s(\varepsilon) \\
& - C_a(\varepsilon)(\mu_1 R_{az}(\varepsilon))^{-1} C_a^T(\varepsilon) \sum_{s=1}^k \mu_s \mathcal{H}_s(\varepsilon) + C_a(\varepsilon) R_{az}^{-1}(\varepsilon) S_a^T(\varepsilon)\Bigg]^T \mathcal{H}_1(\varepsilon) \\
& + \mathcal{H}_1(\varepsilon)\Bigg[A_a(\varepsilon) - B_a(\varepsilon)(\mu_1 R_a(\varepsilon))^{-1} B_a^T(\varepsilon) \sum_{s=1}^k \mu_s \mathcal{H}_s(\varepsilon) \\
& - C_a(\varepsilon)(\mu_1 R_{az}(\varepsilon))^{-1} C_a^T(\varepsilon) \sum_{s=1}^k \mu_s \mathcal{H}_s(\varepsilon) + C_a(\varepsilon) R_{az}^{-1}(\varepsilon) S_a^T(\varepsilon)\Bigg] + Q_a(\varepsilon) \\
& + \sum_{s=1}^k \mu_s \mathcal{H}_s(\varepsilon) B_a(\varepsilon)(\mu_1 R_a(\varepsilon))^{-1} R_a(\varepsilon)(\mu_1 R_a(\varepsilon))^{-1} B_a^T(\varepsilon) \sum_{q=1}^k \mu_q \mathcal{H}_q(\varepsilon) \\
& + \Bigg[-\sum_{s=1}^k \mu_s \mathcal{H}_s(\varepsilon) C_a(\varepsilon)(\mu_1 R_{az}(\varepsilon))^{-1} + S_a(\varepsilon) R_{az}^{-1}(\varepsilon)\Bigg] R_{az}^{-1}(\varepsilon) \\
& \cdot \Bigg[-(\mu_1 R_{az}(\varepsilon))^{-1} C_a^T(\varepsilon) \sum_{q=1}^k \mu_q \mathcal{H}_q(\varepsilon) + R_{az}^{-1}(\varepsilon) S_a^T(\varepsilon)\Bigg] \\
& - 2 S_a(\varepsilon)\Bigg[-(\mu_1 R_{az}(\varepsilon))^{-1} C_a^T(\varepsilon) \sum_{s=1}^k \mu_s \mathcal{H}_s(\varepsilon) + R_{az}^{-1}(\varepsilon) S_a^T(\varepsilon)\Bigg] \qquad (50)
\end{aligned}
$$

$$
\begin{aligned}
\frac{d}{d\varepsilon}\mathcal{E}_r(\varepsilon) = \Bigg[& A_a(\varepsilon) - B_a(\varepsilon)(\mu_1 R_a(\varepsilon))^{-1} B_a^T(\varepsilon) \sum_{s=1}^k \mu_s \mathcal{H}_s(\varepsilon) \\
& - C_a(\varepsilon)(\mu_1 R_{az}(\varepsilon))^{-1} C_a^T(\varepsilon) \sum_{s=1}^k \mu_s \mathcal{H}_s(\varepsilon) + C_a(\varepsilon) R_{az}^{-1}(\varepsilon) S_a^T(\varepsilon)\Bigg]^T \mathcal{H}_r(\varepsilon) \\
& + \mathcal{H}_r(\varepsilon)\Bigg[A_a(\varepsilon) - B_a(\varepsilon)(\mu_1 R_a(\varepsilon))^{-1} B_a^T(\varepsilon) \sum_{s=1}^k \mu_s \mathcal{H}_s(\varepsilon) \\
& - C_a(\varepsilon)(\mu_1 R_{az}(\varepsilon))^{-1} C_a^T(\varepsilon) \sum_{s=1}^k \mu_s \mathcal{H}_s(\varepsilon) + C_a(\varepsilon) R_{az}^{-1}(\varepsilon) S_a^T(\varepsilon)\Bigg] \\
& + \sum_{v=1}^{r-1} \frac{2v!}{v!(r-v)!} \mathcal{H}_v(\varepsilon) G_a(\varepsilon) W_a G_a^T(\varepsilon) \mathcal{H}_{r-v}(\varepsilon), \quad 2 \le r \le k \qquad (51)
\end{aligned}
$$

$$\frac{d}{d\varepsilon}\breve{\mathcal{T}}_1(\varepsilon) = \Bigg[A_a(\varepsilon) - B_a(\varepsilon)(\mu_1 R_a(\varepsilon))^{-1}B_a^T(\varepsilon)\sum_{s=1}^{k}\mu_s\mathcal{H}_s(\varepsilon)$$
$$- C_a(\varepsilon)(\mu_1 R_{az}(\varepsilon))^{-1}C_a^T(\varepsilon)\sum_{s=1}^{k}\mu_s\mathcal{H}_s(\varepsilon) + C_a(\varepsilon)R_{az}^{-1}(\varepsilon)S_a^T(\varepsilon)\Bigg]^T\breve{\mathcal{D}}_1(\varepsilon)$$
$$+ \mathcal{H}_1(\varepsilon)\Bigg[- B_a(\varepsilon)(\mu_1 R_a(\varepsilon))^{-1}B_a^T(\varepsilon)\sum_{s=1}^{k}\mu_s\breve{\mathcal{D}}_s(\varepsilon)$$
$$- C_a(\varepsilon)(\mu_1 R_{az}(\varepsilon))^{-1}C_a^T(\varepsilon)\sum_{s=1}^{k}\mu_s\breve{\mathcal{D}}_s(\varepsilon) + D_a(\varepsilon)\Bigg]$$
$$+\sum_{s=1}^{k}\mu_s\mathcal{H}_s(\varepsilon)B_a(\varepsilon)(\mu_1 R_a(\varepsilon))^{-1}R_a(\varepsilon)(\mu_1 R_a(\varepsilon))^{-1}B_a^T(\varepsilon)\sum_{q=1}^{k}\mu_q\breve{\mathcal{D}}_q(\varepsilon)$$
$$-\Bigg[-\sum_{s=1}^{k}\mu_s\mathcal{H}_s(\varepsilon)C_a(\varepsilon)(\mu_1 R_{az}(\varepsilon))^{-1} + S_a(\varepsilon)R_{az}^{-1}(\varepsilon)\Bigg]R_{az}(\varepsilon)(\mu_1 R_{az}(\varepsilon))^{-1}C_a^T(\varepsilon)$$
$$\cdot\sum_{q=1}^{k}\mu_q\breve{\mathcal{D}}_q(\varepsilon) + S_a(\varepsilon)(\mu_1 R_{az}(\varepsilon))^{-1}C_a^T(\varepsilon)\sum_{s=1}^{k}\mu_s\breve{\mathcal{D}}_s(\varepsilon) \quad (52)$$

$$\frac{d}{d\varepsilon}\breve{\mathcal{T}}_r(\varepsilon) = \Bigg[A_a(\varepsilon) - B_a(\varepsilon)(\mu_1 R_a(\varepsilon))^{-1}B_a^T(\varepsilon)\sum_{s=1}^{k}\mu_s\mathcal{H}_s(\varepsilon) + C_a(\varepsilon)R_{az}^{-1}(\varepsilon)S_a^T(\varepsilon)$$
$$-C_a(\varepsilon)(\mu_1 R_{az}(\varepsilon))^{-1}C_a^T(\varepsilon)\sum_{s=1}^{k}\mu_s\mathcal{H}_s(\varepsilon)\Bigg]^T\breve{\mathcal{D}}_r(\varepsilon) + \mathcal{H}_r(\varepsilon)\Bigg[-B_a(\varepsilon)(\mu_1 R_a(\varepsilon))^{-1}B_a^T(\varepsilon)$$
$$\cdot\sum_{s=1}^{k}\mu_s\breve{\mathcal{D}}_s(\varepsilon) - C_a(\varepsilon)(\mu_1 R_{az}(\varepsilon))^{-1}C_a^T(\varepsilon)\sum_{s=1}^{k}\mu_s\breve{\mathcal{D}}_s(\varepsilon) + D_a(\varepsilon)\Bigg],\ 2\le r\le k \quad (53)$$

$$\frac{d}{d\varepsilon}\mathcal{T}_1(\varepsilon) = 2\breve{\mathcal{D}}_1(\varepsilon)\Bigg[- B_a(\varepsilon)(\mu_1 R_a(\varepsilon))^{-1}B_a^T(\varepsilon)\sum_{s=1}^{k}\mu_s\breve{\mathcal{D}}_s(\varepsilon)$$
$$- C_a(\varepsilon)(\mu_1 R_{az}(\varepsilon))^{-1}C_a^T(\varepsilon)\sum_{s=1}^{k}\mu_s\breve{\mathcal{D}}_s(\varepsilon) + D_a(\varepsilon)\Bigg] + \mathrm{Tr}\left\{\mathcal{H}_1(\varepsilon)G_a(\varepsilon)W_aG_a^T(\varepsilon)\right\}$$
$$+\sum_{s=1}^{k}\mu_s\breve{\mathcal{D}}_s(\varepsilon)B_a(\varepsilon)(\mu_1 R_a(\varepsilon))^{-1}R_a(\varepsilon)(\mu_1 R_a(\varepsilon))^{-1}B_a^T(\varepsilon)\sum_{q=1}^{k}\mu_q\breve{\mathcal{D}}_q(\varepsilon)$$
$$+\sum_{s=1}^{k}\mu_s\breve{\mathcal{D}}_s(\varepsilon)C_a(\varepsilon)(\mu_1 R_{az}(\varepsilon))^{-1}R_{az}(\varepsilon)(\mu_1 R_{az}(\varepsilon))^{-1}C_a^T(\varepsilon)\sum_{q=1}^{k}\mu_q\breve{\mathcal{D}}_q(\varepsilon) \quad (54)$$

$$\frac{d}{d\varepsilon}\mathcal{T}_r(\varepsilon) = 2\breve{\mathcal{D}}_r(\varepsilon)\Bigg[- B_a(\varepsilon)(\mu_1 R_a(\varepsilon))^{-1}B_a^T(\varepsilon)\sum_{s=1}^{k}\mu_s\breve{\mathcal{D}}_s(\varepsilon) + D_a(\varepsilon) - C_a(\varepsilon)(\mu_1 R_{az}(\varepsilon))^{-1}C_a^T(\varepsilon)\sum_{s=1}^{k}\mu_s\breve{\mathcal{D}}_s(\varepsilon)\Bigg] + \mathrm{Tr}\left\{\mathcal{H}_r(\varepsilon)G_a(\varepsilon)W_aG_a^T(\varepsilon)\right\}, \quad 2 \le r \le k \quad (55)$$

will work. Furthermore, the boundary condition (39) requires that

$$(x_a^0)^T\sum_{r=1}^{k}\mu_r(\mathcal{H}_r^0+\mathcal{E}_r(t_0))x_a^0 + 2(x_a^0)^T\sum_{r=1}^{k}\mu_r(\breve{\mathcal{D}}_r^0+\breve{\mathcal{T}}_r(t_0)) + \sum_{r=1}^{k}\mu_r(\mathcal{D}_r^0+\mathcal{T}_r(t_0)) = (x_a^0)^T\sum_{r=1}^{k}\mu_r\mathcal{H}_r^0\,x_a^0 + 2(x_a^0)^T\sum_{r=1}^{k}\mu_r\breve{\mathcal{D}}_r^0 + \sum_{r=1}^{k}\mu_r\mathcal{D}_r^0\,.$$

Thus, matching the boundary condition yields the initial value conditions $\mathcal{E}_r(t_0)=0$, $\breve{\mathcal{T}}_r(t_0)=0$ and $\mathcal{T}_r(t_0)=0$ for the parametric time-forward differential equations (50)-(55).

Applying the 4-tuple decision strategy specified in (45)-(48) along the solution trajectories of the time-backward differential equations (34)-(36), these equations become the time-backward Riccati-type differential equations

$$\frac{d}{d\varepsilon}\mathcal{H}_1(\varepsilon) = -\Bigg[A_a(\varepsilon) - B_a(\varepsilon)(\mu_1 R_a(\varepsilon))^{-1}B_a^T(\varepsilon)\sum_{s=1}^{k}\mu_s\mathcal{H}_s(\varepsilon) - C_a(\varepsilon)(\mu_1 R_{az}(\varepsilon))^{-1}C_a^T(\varepsilon)\sum_{s=1}^{k}\mu_s\mathcal{H}_s(\varepsilon) + C_a(\varepsilon)R_{az}^{-1}(\varepsilon)S_a^T(\varepsilon)\Bigg]^T\mathcal{H}_1(\varepsilon)$$
$$- \mathcal{H}_1(\varepsilon)\Bigg[A_a(\varepsilon) - B_a(\varepsilon)(\mu_1 R_a(\varepsilon))^{-1}B_a^T(\varepsilon)\sum_{s=1}^{k}\mu_s\mathcal{H}_s(\varepsilon) - C_a(\varepsilon)(\mu_1 R_{az}(\varepsilon))^{-1}C_a^T(\varepsilon)\sum_{s=1}^{k}\mu_s\mathcal{H}_s(\varepsilon) + C_a(\varepsilon)R_{az}^{-1}(\varepsilon)S_a^T(\varepsilon)\Bigg] - Q_a(\varepsilon)$$
$$- \sum_{s=1}^{k}\mu_s\mathcal{H}_s(\varepsilon)B_a(\varepsilon)(\mu_1 R_a(\varepsilon))^{-1}R_a(\varepsilon)(\mu_1 R_a(\varepsilon))^{-1}B_a^T(\varepsilon)\sum_{q=1}^{k}\mu_q\mathcal{H}_q(\varepsilon)$$
$$- \Bigg[-\sum_{s=1}^{k}\mu_s\mathcal{H}_s(\varepsilon)C_a(\varepsilon)(\mu_1 R_{az}(\varepsilon))^{-1} + S_a(\varepsilon)R_{az}^{-1}(\varepsilon)\Bigg]R_{az}^{-1}(\varepsilon) \cdot \Bigg[-(\mu_1 R_{az}(\varepsilon))^{-1}C_a^T(\varepsilon)\sum_{q=1}^{k}\mu_q\mathcal{H}_q(\varepsilon) + R_{az}^{-1}(\varepsilon)S_a^T(\varepsilon)\Bigg]$$
$$+ 2S_a(\varepsilon)\Bigg[-(\mu_1 R_{az}(\varepsilon))^{-1}C_a^T(\varepsilon)\sum_{s=1}^{k}\mu_s\mathcal{H}_s(\varepsilon) + R_{az}^{-1}(\varepsilon)S_a^T(\varepsilon)\Bigg] \quad (56)$$

$$\frac{d}{d\varepsilon}\mathcal{H}_r(\varepsilon) = -\Bigg[A_a(\varepsilon) - B_a(\varepsilon)(\mu_1 R_a(\varepsilon))^{-1}B_a^T(\varepsilon)\sum_{s=1}^{k}\mu_s\mathcal{H}_s(\varepsilon)$$
$$- C_a(\varepsilon)(\mu_1 R_{az}(\varepsilon))^{-1}C_a^T(\varepsilon)\sum_{s=1}^{k}\mu_s\mathcal{H}_s(\varepsilon) + C_a(\varepsilon)R_{az}^{-1}(\varepsilon)S_a^T(\varepsilon)\Bigg]^T\mathcal{H}_r(\varepsilon)$$
$$- \mathcal{H}_r(\varepsilon)\Bigg[A_a(\varepsilon) - B_a(\varepsilon)(\mu_1 R_a(\varepsilon))^{-1}B_a^T(\varepsilon)\sum_{s=1}^{k}\mu_s\mathcal{H}_s(\varepsilon)$$
$$- C_a(\varepsilon)(\mu_1 R_{az}(\varepsilon))^{-1}C_a^T(\varepsilon)\sum_{s=1}^{k}\mu_s\mathcal{H}_s(\varepsilon) + C_a(\varepsilon)R_{az}^{-1}(\varepsilon)S_a^T(\varepsilon)\Bigg]$$
$$- \sum_{v=1}^{r-1}\frac{2v!}{v!(r-v)!}\mathcal{H}_v(\varepsilon)G_a(\varepsilon)W_aG_a^T(\varepsilon)\mathcal{H}_{r-v}(\varepsilon)\,, \quad 2 \le r \le k \quad (57)$$

$$\frac{d}{d\varepsilon}\breve{\mathcal{D}}_1(\varepsilon) = -\Bigg[A_a(\varepsilon) - B_a(\varepsilon)(\mu_1 R_a(\varepsilon))^{-1}B_a^T(\varepsilon)\sum_{s=1}^{k}\mu_s\mathcal{H}_s(\varepsilon)$$
$$- C_a(\varepsilon)(\mu_1 R_{az}(\varepsilon))^{-1}C_a^T(\varepsilon)\sum_{s=1}^{k}\mu_s\mathcal{H}_s(\varepsilon) + C_a(\varepsilon)R_{az}^{-1}(\varepsilon)S_a^T(\varepsilon)\Bigg]^T\breve{\mathcal{D}}_1(\varepsilon)$$
$$- \mathcal{H}_1(\varepsilon)\Bigg[-B_a(\varepsilon)(\mu_1 R_a(\varepsilon))^{-1}B_a^T(\varepsilon)\sum_{s=1}^{k}\mu_s\breve{\mathcal{D}}_s(\varepsilon)$$
$$- C_a(\varepsilon)(\mu_1 R_{az}(\varepsilon))^{-1}C_a^T(\varepsilon)\sum_{s=1}^{k}\mu_s\breve{\mathcal{D}}_s(\varepsilon) + D_a(\varepsilon)\Bigg]$$
$$- \sum_{s=1}^{k}\mu_s\mathcal{H}_s(\varepsilon)B_a(\varepsilon)(\mu_1 R_a(\varepsilon))^{-1}R_a(\varepsilon)(\mu_1 R_a(\varepsilon))^{-1}B_a^T(\varepsilon)\sum_{q=1}^{k}\mu_q\breve{\mathcal{D}}_q(\varepsilon)$$
$$+ \Bigg[-\sum_{s=1}^{k}\mu_s\mathcal{H}_s(\varepsilon)C_a(\varepsilon)(\mu_1 R_{az}(\varepsilon))^{-1} + S_a(\varepsilon)R_{az}^{-1}(\varepsilon)\Bigg]R_{az}(\varepsilon)(\mu_1 R_{az}(\varepsilon))^{-1}C_a^T(\varepsilon)$$
$$\cdot \sum_{q=1}^{k}\mu_q\breve{\mathcal{D}}_q(\varepsilon) - S_a(\varepsilon)(\mu_1 R_{az}(\varepsilon))^{-1}C_a^T(\varepsilon)\sum_{s=1}^{k}\mu_s\breve{\mathcal{D}}_s(\varepsilon) \quad (58)$$

$$\frac{d}{d\varepsilon}\breve{\mathcal{D}}_r(\varepsilon) = -\Bigg[A_a(\varepsilon) - B_a(\varepsilon)(\mu_1 R_a(\varepsilon))^{-1}B_a^T(\varepsilon)\sum_{s=1}^{k}\mu_s\mathcal{H}_s(\varepsilon)$$
$$- C_a(\varepsilon)(\mu_1 R_{az}(\varepsilon))^{-1}C_a^T(\varepsilon)\sum_{s=1}^{k}\mu_s\mathcal{H}_s(\varepsilon) + C_a(\varepsilon)R_{az}^{-1}(\varepsilon)S_a^T(\varepsilon)\Bigg]^T\breve{\mathcal{D}}_r(\varepsilon)$$
$$- \mathcal{H}_r(\varepsilon)\Bigg[-B_a(\varepsilon)(\mu_1 R_a(\varepsilon))^{-1}B_a^T(\varepsilon)\sum_{s=1}^{k}\mu_s\breve{\mathcal{D}}_s(\varepsilon)$$

$$-C_a(\varepsilon)(\mu_1 R_{az}(\varepsilon))^{-1}C_a^T(\varepsilon)\sum_{s=1}^{k}\mu_s\breve{\mathcal{D}}_s(\varepsilon)+D_a(\varepsilon)\Bigg], \quad 2\le r\le k \tag{59}$$

$$\frac{d}{d\varepsilon}\mathcal{D}_1(\varepsilon)=-2\breve{\mathcal{D}}_1(\varepsilon)\Bigg[-B_a(\varepsilon)(\mu_1 R_a(\varepsilon))^{-1}B_a^T(\varepsilon)\sum_{s=1}^{k}\mu_s\breve{\mathcal{D}}_s(\varepsilon)$$
$$-C_a(\varepsilon)(\mu_1 R_{az}(\varepsilon))^{-1}C_a^T(\varepsilon)\sum_{s=1}^{k}\mu_s\breve{\mathcal{D}}_s(\varepsilon)+D_a(\varepsilon)\Bigg]-\mathrm{Tr}\left\{\mathcal{H}_1(\varepsilon)G_a(\varepsilon)W_aG_a^T(\varepsilon)\right\}$$
$$-\sum_{s=1}^{k}\mu_s\breve{\mathcal{D}}_s(\varepsilon)B_a(\varepsilon)(\mu_1 R_a(\varepsilon))^{-1}R_a(\varepsilon)(\mu_1 R_a(\varepsilon))^{-1}B_a^T(\varepsilon)\sum_{q=1}^{k}\mu_q\breve{\mathcal{D}}_q(\varepsilon)$$
$$-\sum_{s=1}^{k}\mu_s\breve{\mathcal{D}}_s(\varepsilon)C_a(\varepsilon)(\mu_1 R_{az}(\varepsilon))^{-1}R_{az}(\varepsilon)(\mu_1 R_{az}(\varepsilon))^{-1}C_a^T(\varepsilon)\sum_{q=1}^{k}\mu_q\breve{\mathcal{D}}_q(\varepsilon) \tag{60}$$

and, for $2\le r\le k$

$$\frac{d}{d\varepsilon}\mathcal{D}_r(\varepsilon)=-2\breve{\mathcal{D}}_r(\varepsilon)\Bigg[-B_a(\varepsilon)(\mu_1 R_a(\varepsilon))^{-1}B_a^T(\varepsilon)\sum_{s=1}^{k}\mu_s\breve{\mathcal{D}}_s(\varepsilon)+D_a(\varepsilon)$$
$$-C_a(\varepsilon)(\mu_1 R_{az}(\varepsilon))^{-1}C_a^T(\varepsilon)\sum_{s=1}^{k}\mu_s\breve{\mathcal{D}}_s(\varepsilon)\Bigg]-\mathrm{Tr}\left\{\mathcal{H}_r(\varepsilon)G_a(\varepsilon)W_aG_a^T(\varepsilon)\right\} \tag{61}$$

where the terminal-value conditions $\mathcal{H}_1(t_f)=Q_a^f$, $\mathcal{H}_r(t_f)=0$ for $2\le r\le k$; $\breve{\mathcal{D}}_r(t_f)=0$ for $1\le r\le k$; and $\mathcal{D}_r(t_f)=0$ for $1\le r\le k$. Thus, whenever these equations (56)-(61) admit solutions $\{\mathcal{H}_r(\cdot)\}_{r=1}^k$, $\left\{\breve{\mathcal{D}}_r(\cdot)\right\}_{r=1}^k$, and $\{\mathcal{D}_r(\cdot)\}_{r=1}^k$, then the existence of $\{\mathcal{E}_r(\cdot)\}_{r=1}^k$, $\left\{\breve{\mathcal{T}}_r(\cdot)\right\}_{r=1}^k$, and $\{\mathcal{T}_r(\cdot)\}_{r=1}^k$ satisfying the equations (50)-(55) are assured. By comparing the time-forward differential equations (50)-(55) to those of time-backward equations (56)-(61), one may recognize that these sets of equations are related to one another by

$$\frac{d}{d\varepsilon}\mathcal{E}_r(\varepsilon)=-\frac{d}{d\varepsilon}\mathcal{H}_r(\varepsilon)\,,$$
$$\frac{d}{d\varepsilon}\breve{\mathcal{T}}_r(\varepsilon)=-\frac{d}{d\varepsilon}\breve{\mathcal{D}}_r(\varepsilon)\,,$$
$$\frac{d}{d\varepsilon}\mathcal{T}_r(\varepsilon)=-\frac{d}{d\varepsilon}\mathcal{D}_r(\varepsilon)\,,\quad 1\le r\le k\,.$$

Enforcing the initial value conditions of $\mathcal{E}_r(t_0)=0$, $\breve{\mathcal{T}}_r(t_0)=0$, and $\mathcal{T}_r(t_0)=0$ uniquely implies that

$$\mathcal{E}_r(\varepsilon)=\mathcal{H}_r(t_0)-\mathcal{H}_r(\varepsilon)\,,$$
$$\breve{\mathcal{T}}_r(\varepsilon)=\breve{\mathcal{D}}_r(t_0)-\breve{\mathcal{D}}_r(\varepsilon)\,,$$
$$\mathcal{T}_r(\varepsilon)=\mathcal{D}_r(t_0)-\mathcal{D}_r(\varepsilon)$$

for all $\varepsilon \in [t_0, t_f]$ and yields a value function

$$\mathcal{W}(\varepsilon, \mathcal{Y}, \breve{\mathcal{Z}}, \mathcal{Z}) = (x_a^0)^T \sum_{r=1}^{k} \mu_r \mathcal{H}_r(t_0)\, x_a^0 + 2(x_a^0)^T \sum_{r=1}^{k} \mu_r \breve{\mathcal{D}}_r(t_0) + \sum_{r=1}^{k} \mu_r \mathcal{D}_r(t_0)$$

for which the sufficient condition (40) of the verification theorem is satisfied. Therefore, the extremal decision laws (45)-(48) minimizing (37) become optimal

$$K^*(\varepsilon) = -(\mu_1 R_a(\varepsilon))^{-1} B_a^T(\varepsilon) \sum_{r=1}^{k} \mu_r \mathcal{H}_r^*(\varepsilon)\,, \tag{62}$$

$$K_z^*(\varepsilon) = -(\mu_1 R_{az}(\varepsilon))^{-1} C_a^T(\varepsilon) \sum_{r=1}^{k} \mu_r \mathcal{H}_r^*(\varepsilon) + R_{az}^{-1}(\varepsilon) S_a^T(\varepsilon)\,, \tag{63}$$

$$p^*(\varepsilon) = -(\mu_1 R_a(\varepsilon))^{-1} B_a^T(\varepsilon) \sum_{r=1}^{k} \mu_r \breve{\mathcal{D}}_r^*(\varepsilon)\,, \tag{64}$$

$$p_z^*(\varepsilon) = -(\mu_1 R_{az}(\varepsilon))^{-1} C_a^T(\varepsilon) \sum_{r=1}^{k} \mu_r \breve{\mathcal{D}}_r^*(\varepsilon)\,. \tag{65}$$

Theorem 6. *Cumulant-based Cooperative Strategies.*
Consider the multi-person cooperative game (17)-(19) whose pairs (A_a, B_a) and (A_a, C_a) are uniformly stabilizable on $[t_0, t_f]$. Let $k \in \mathbb{Z}^+$ and the sequence $\mu = \{\mu_i \geq 0\}_{i=1}^k$ with $\mu_1 > 0$. Then, the optimal coalitive decision laws are given by

$$K^*(\alpha) = -R_a(\alpha)^{-1} B_a^T(\alpha) \sum_{r=1}^{k} \hat{\mu}_r \mathcal{H}_r^*(\alpha)\,, \tag{66}$$

$$K_z^*(\alpha) = -R_{az}(\alpha)^{-1} C_a^T(\alpha) \sum_{r=1}^{k} \hat{\mu}_r \mathcal{H}_r^*(\alpha) + R_{az}^{-1}(\alpha) S_a^T(\alpha)\,, \tag{67}$$

$$p^*(\alpha) = -R_a(\alpha)^{-1} B_a^T(\alpha) \sum_{r=1}^{k} \hat{\mu}_r \breve{\mathcal{D}}_r^*(\alpha)\,, \tag{68}$$

$$p_z^*(\alpha) = -R_{az}(\alpha)^{-1} C_a^T(\alpha) \sum_{r=1}^{k} \hat{\mu}_r \breve{\mathcal{D}}_r^*(\alpha) \tag{69}$$

where normalized preferences $\hat{\mu}_r \triangleq \mu_r/\mu_1$'s are mutually chosen by $\overline{S}$ for strategic coordination towards co-design of the coalition performance robustness. Optimal supporting solutions $\{\mathcal{H}_r^*(\alpha)\}_{r=1}^k$, $\{\breve{\mathcal{D}}_r^*(\alpha)\}_{r=1}^k$, and $\{\mathcal{D}_r^*(\alpha)\}_{r=1}^k$ are satisfying the time-backward matrix-valued and vector-valued differential equations

$$\begin{aligned} \frac{d}{d\alpha}\mathcal{H}_1^*(\alpha) = &- \left[A_a(\alpha) + B_a(\alpha)K^*(\alpha) + C_a(\alpha)K_z^*(\alpha)\right]^T \mathcal{H}_1^*(\alpha) \\ &- \mathcal{H}_1^*(\alpha)\left[A_a(\alpha) + B_a(\alpha)K^*(\alpha) + C_a(\alpha)K_z^*(\alpha)\right] - Q_a(\alpha) \\ &- (K^*)^T(\alpha) R_a(\alpha) K^*(\alpha) - (K_z^*)^T(\alpha) R_{az}(\alpha) K_z^*(\alpha) + 2S_a(\alpha)K_z^*(\alpha) \end{aligned} \tag{70}$$

$$\begin{aligned}\frac{d}{d\alpha}\mathcal{H}_r^*(\alpha) = &-\left[A_a(\alpha)+B_a(\alpha)K^*(\alpha)+C_a(\alpha)K_z^*(\alpha)\right]^T\mathcal{H}_r^*(\alpha)\\ &-\mathcal{H}_r^*(\alpha)\left[A_a(\alpha)+B_a(\alpha)K^*(\alpha)+C_a(\alpha)K_z^*(\alpha)\right]\\ &-\sum_{s=1}^{r-1}\frac{2r!}{s!(r-s)!}\mathcal{H}_s^*(\alpha)G_a(\alpha)W_aG_a^T(\alpha)\mathcal{H}_{r-s}^*(\alpha)\,, \quad 2\le r\le k \qquad (71)\end{aligned}$$

$$\begin{aligned}\frac{d}{d\alpha}\breve{\mathcal{D}}_1^*(\alpha) = &-\left[A_a(\alpha)+B_a(\alpha)K^*(\alpha)+C_a(\alpha)K_z^*(\alpha)\right]^T\breve{\mathcal{D}}_1^*(\alpha)\\ &-\mathcal{H}_1^*(\alpha)\left[B_a(\alpha)p^*(\alpha)+C_a(\alpha)p_z^*(\alpha)+D_a(\alpha)\right]\\ &-(K^*)^T(\alpha)R_a(\alpha)p^*(\alpha)-(K_z^*)^T(\alpha)R_{az}(\alpha)p_z^*(\alpha)+S_a(\alpha)p_z^*(\alpha) \qquad (72)\end{aligned}$$

$$\begin{aligned}\frac{d}{d\alpha}\breve{\mathcal{D}}_r^*(\alpha) = &-\left[A_a(\alpha)+B_a(\alpha)K^*(\alpha)+C_a(\alpha)K_z^*(\alpha)\right]^T\breve{\mathcal{D}}_r^*(\alpha)\\ &-\mathcal{H}_r^*(\alpha)\left[B_a(\alpha)p^*(\alpha)+C_a(\alpha)p_z^*(\alpha)+D_a(\alpha)\right]\,, \quad 2\le r\le k \qquad (73)\end{aligned}$$

$$\begin{aligned}\frac{d}{d\alpha}\mathcal{D}_1^*(\alpha) = &-2(\breve{\mathcal{D}}_1^*)^T(\alpha)\left[B_a(\alpha)p^*(\alpha)+C_a(\alpha)p_z^*(\alpha)+D_a(\alpha)\right]\\ &-\mathrm{Tr}\{\mathcal{H}_1^*(\alpha)G_a(\alpha)W_aG_a^T(\alpha)\}-(p^*)^T(\alpha)R_a(\alpha)p^*(\alpha)-(p_z^*)^T(\alpha)R_{az}(\alpha)p_z^*(\alpha) \qquad (74)\end{aligned}$$

$$\begin{aligned}\frac{d}{d\alpha}\mathcal{D}_r^*(\alpha) = &-2(\breve{\mathcal{D}}_r^*)^T(\alpha)\left[B_a(\alpha)p^*(\alpha)+C_a(\alpha)p_z^*(\alpha)+D_a(\alpha)\right]\\ &-\mathrm{Tr}\{\mathcal{H}_r^*(\alpha)G_a(\alpha)W_aG_a^T(\alpha)\}\,, \quad 2\le r\le k \qquad (75)\end{aligned}$$

where the terminal-value conditions $\mathcal{H}_1^*(t_f)=Q_a^f$, $\mathcal{H}_r^*(t_f)=0$ for $2\le r\le k$; $\breve{\mathcal{D}}_r^*(t_f)=0$ for $1\le r\le k$; and $\mathcal{D}_r^*(t_f)=0$ for $1\le r\le k$.

Remark 3. *Note that the order of cooperative decision strategies (66)-(69) is too large for implementation at the outset. However, there are special properties that are used for the reduced-order decision strategies. In this case all the block diagonal properties of (18), (20) and (21)-(22) are reflected in the properties of the cumulant-supporting equations (70)-(75). With the terminal-value conditions remained diagonally dominant, the total solutions of (70)-(75) are therefore, in the block diagonal form and are summarized in the following results.*

Theorem 7. *Cumulant-based and Decentralized Decisions.*
Consider the stochastic cooperative game (17)-(19) with multiple decentralized decision makers $i\in\overline{S}$ whose pairs (A_{ai},B_{ai}) and (A_{ai},C_{ai}) are uniformly stabilizable on $[t_0,t_f]$. Let $k\in\mathbb{Z}^+$ and the sequence $\mu=\{\mu_i\ge 0\}_{i=1}^k$ with $\mu_1>0$. Then, the optimal decentralized decision laws are given by

$$K_i^*(\alpha) = -(\xi_i R_i)^{-1}(\alpha) B_{ai}^T(\alpha) \sum_{r=1}^{k} \hat{\mu}_r \mathcal{H}_{ir}^*(\alpha)\,, \tag{76}$$

$$K_{zi}^*(\alpha) = -(\xi_i R_{zi})^{-1}(\alpha) \left[C_{ai}^T(\alpha) \sum_{r=1}^{k} \hat{\mu}_r \mathcal{H}_{ir}^*(\alpha) + \xi_i S_{ai}^T(\alpha) \right] , \tag{77}$$

$$p_i^*(\alpha) = -(\xi_i R_i)^{-1}(\alpha) B_{ai}^T(\alpha) \sum_{r=1}^{k} \hat{\mu}_r \breve{\mathcal{D}}_{ir}^*(\alpha)\,, \tag{78}$$

$$p_{zi}^*(\alpha) = -(\xi_i R_{zi})^{-1}(\alpha) C_{ai}^T(\alpha) \sum_{r=1}^{k} \hat{\mu}_r \breve{\mathcal{D}}_{ir}^*(\alpha) \tag{79}$$

where normalized preferences $\hat{\mu}_r \triangleq \mu_r/\mu_1$'s are mutually chosen by $\overline{S}$ for strategic coordination towards co-design of the coalition performance robustness. At the local levels, optimal supporting solutions $\{\mathcal{H}_{ir}^*(\alpha)\}_{r=1}^k$, $\{\breve{\mathcal{D}}_{ir}^*(\alpha)\}_{r=1}^k$, $\{\mathcal{D}_{ir}^*(\alpha)\}_{r=1}^k$ satisfy the time-backward matrix-valued and vector-valued differential equations

$$\begin{aligned} \frac{d}{d\alpha}\mathcal{H}_{i1}^*(\alpha) &= -\left[A_{ai}(\alpha) + B_{ai}(\alpha)K_i^*(\alpha) + C_{ai}(\alpha)K_{zi}^*(\alpha)\right]^T \mathcal{H}_{i1}^*(\alpha) \\ &- \mathcal{H}_{i1}^*(\alpha)\left[A_{ai}(\alpha) + B_{ai}(\alpha)K_i^*(\alpha) + C_{ai}(\alpha)K_{zi}^*(\alpha)\right] - \xi_i Q_{ai}(\alpha) \\ &- (K_i^*)^T(\alpha)\xi_i R_i(\alpha)K_i^*(\alpha) - (K_{zi}^*)^T(\alpha)\xi_i R_{zi}(\alpha)K_{zi}^*(\alpha) + 2\xi_i S_{ai}(\alpha)K_{zi}^*(\alpha) \end{aligned} \tag{80}$$

$$\begin{aligned} \frac{d}{d\alpha}\mathcal{H}_{ir}^*(\alpha) &= -\left[A_{ai}(\alpha) + B_{ai}(\alpha)K_i^*(\alpha) + C_{ai}(\alpha)K_{zi}^*(\alpha)\right]^T \mathcal{H}_{ir}^*(\alpha) \\ &- \mathcal{H}_{ir}^*(\alpha)\left[A_{ai}(\alpha) + B_{ai}(\alpha)K_i^*(\alpha) + C_{ai}(\alpha)K_{zi}^*(\alpha)\right] \\ &- \sum_{s=1}^{r-1} \frac{2r!}{s!(r-s)!} \mathcal{H}_{is}^*(\alpha)G_{ai}(\alpha)W_{ai}G_{ai}^T(\alpha)\mathcal{H}_{i,r-s}^*(\alpha)\,, \quad 2 \le r \le k \end{aligned} \tag{81}$$

$$\begin{aligned} \frac{d}{d\alpha}\breve{\mathcal{D}}_{i1}^*(\alpha) &= -\left[A_{ai}(\alpha) + B_{ai}(\alpha)K_i^*(\alpha) + C_{ai}(\alpha)K_{zi}^*(\alpha)\right]^T \breve{\mathcal{D}}_{i1}^*(\alpha) \\ &- \mathcal{H}_{i1}^*(\alpha)\left[B_{ai}(\alpha)p_i^*(\alpha) + C_{ai}(\alpha)p_{zi}^*(\alpha) + D_{ai}(\alpha)\right] \\ &- (K_i^*)^T(\alpha)\xi_i R_i(\alpha)p_i^*(\alpha) - (K_{zi}^*)^T(\alpha)\xi_i R_{zi}(\alpha)p_{zi}^*(\alpha) + \xi_i S_{ai}(\alpha)p_{zi}^*(\alpha) \end{aligned} \tag{82}$$

$$\begin{aligned} \frac{d}{d\alpha}\breve{\mathcal{D}}_{ir}^*(\alpha) &= -\left[A_{ai}(\alpha) + B_{ai}(\alpha)K_i^*(\alpha) + C_{ai}(\alpha)K_{zi}^*(\alpha)\right]^T \breve{\mathcal{D}}_{ir}^*(\alpha) \\ &- \mathcal{H}_{ir}^*(\alpha)\left[B_{ai}(\alpha)p_i^*(\alpha) + C_{ai}(\alpha)p_{zi}^*(\alpha) + D_{ai}(\alpha)\right] , \quad 2 \le r \le k \end{aligned} \tag{83}$$

$$\begin{aligned} \frac{d}{d\alpha}\mathcal{D}_{i1}^*(\alpha) &= -2(\breve{\mathcal{D}}_{i1}^*)^T(\alpha)\left[B_{ai}(\alpha)p_i^*(\alpha) + C_{ai}(\alpha)p_{zi}^*(\alpha) + D_{ai}(\alpha)\right] \\ &- \mathrm{Tr}\{\mathcal{H}_{i1}^*(\alpha)G_{ai}(\alpha)W_{ai}G_{ai}^T(\alpha)\} - (p_i^*)^T(\alpha)\xi_i R_i(\alpha)p_i^*(\alpha) - (p_{zi}^*)^T(\alpha)\xi_i R_{zi}(\alpha)p_{zi}^*(\alpha) \end{aligned} \tag{84}$$

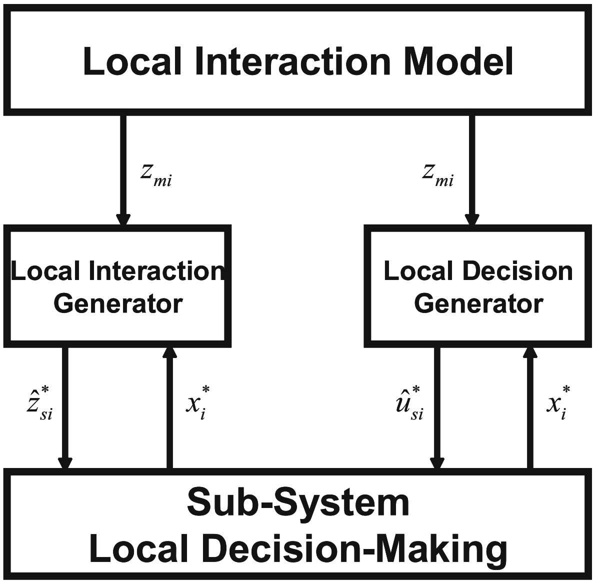


Fig. 1. Decentralized Decision Making: Off-line

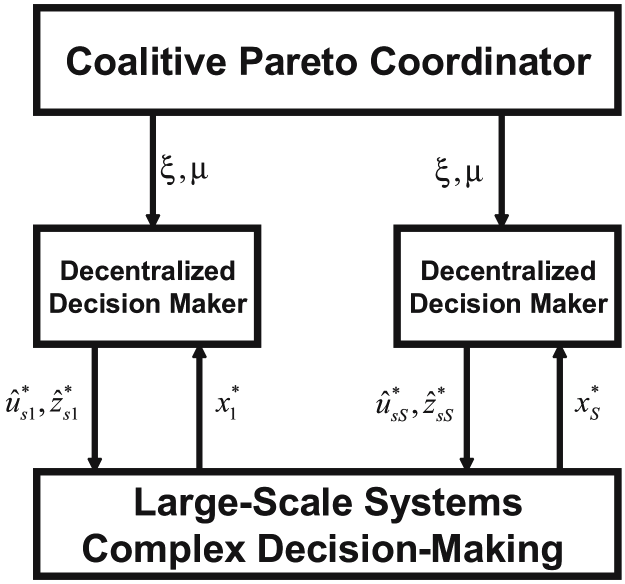

Coalitive Pareto Coordinator
ξ, μ
ξ, μ
Decentralized Decision Maker
Decentralized Decision Maker
$\hat{u}^*_{s1}, \hat{z}^*_{s1}$
x^*_1
$\hat{u}^*_{sS}, \hat{z}^*_{sS}$
x^*_S
Large-Scale Systems Complex Decision-Making

Fig. 2. Multi-Level Control and Coordination

$$\frac{d}{d\alpha}\mathcal{D}^{*}_{ir}(\alpha) = -2(\breve{\mathcal{D}}^{*}_{ir})^{T}(\alpha)\left[B_{ai}(\alpha)p^{*}_{i}(\alpha) + C_{ai}(\alpha)p^{*}_{zi}(\alpha) + D_{ai}(\alpha)\right] - \mathrm{Tr}\{\mathcal{H}^{*}_{ir}(\alpha)G_{ai}(\alpha)W_{ai}G^{T}_{ai}(\alpha)\}\,, \quad 2 \le r \le k \quad (85)$$

where the terminal-value conditions $\mathcal{H}^{*}_{i1}(t_f) = \xi_i Q^{f}_{ai}$, $\mathcal{H}^{*}_{ir}(t_f) = 0$ *for* $2 \le r \le k$*;* $\breve{\mathcal{D}}^{*}_{ir}(t_f) = 0$ *for* $1 \le r \le k$*; and* $\mathcal{D}^{*}_{ir}(t_f) = 0$ *for* $1 \le r \le k$*.*

Remark 4. *The results listed here illustrate the fact that all the higher-order statistics associated with the Chi-squared random measure of performance are now incorporated with a class of local feedback decision strategies. Such decentralized decision laws shown in Figure 1, making use of these information statistics, are therefore robust and adaptable to uncertain environments. Consequently, these cooperative decision makers take active roles to mitigate all variations in performance caused by Nature's mixed random realizations and the coalition is finally capable of shaping the performance distribution and robustness as illustrated in Figure 2.*

5 Conclusions

The present work presents a new innovative and generalized solution concept to address multiple attributes of performance robustness which is widely recognized as the unresolved issue in stochastic control and Linear-Quadratic (LQ) decision problems. The framework based upon performance-measure statistics has three distinguishing aspects in the learning process. First, the learning rule is more sophisticated because it involves a rule for decision makers to forecast the behaviors of the coalitive performance and a rule for responding to random actions by Nature. Second, the criterion of performance that cooperative decision makers use to evaluate how they are doing in maintaining preferable shapes of the coalitive performance, is selected with high regard for possible random sample path realizations by Nature. The third aspect involves a notion of regularity property of the learning process where best responses with no regret lead to sample-by-sample behaviors for cooperative decision makers against uncertain actions employed by Nature. For instance, these decentralized feedback decision gains operate dynamically on the time-backward histories of the cumulant-supporting equations from the final to the current time. In addition, it is noted that these cumulant-supporting equations also depend on Nature's a-priori probabilistic characteristics. Therefore, cooperative decision makers have traded the property of the certainty equivalence principle as they would obtain from the special case of LQ decision problems, for the adaptability to deal with uncertain environments. Future work will be another attainment of cumulant-based solutions in multi-person decision problems for complex situations: (i) performance robustness via a desired statistical description, (ii) cooperative decision selection via output feedback information patterns, and (iii) confrontations among coalitive decision makers and adversaries.

References

1. Dieudonne, J.: Foundations of Modern Analysis. Academic Press, New York (1960)
2. Field, R.V., Bergman, L.A.: Reliability based Approach to Linear Covariance Control Design. Journal of Engineering Mechanics 124, 193–199 (1998)
3. Fleming, W.H., Rishel, R.W.: Deterministic and Stochastic Optimal Control. Springer, New York (1975)
4. Hansen, L.P., Sargent, T.J., Tallarini Jr., T.D.: Robust Permanent Income and Pricing. Review of Economic Studies 66, 873–907 (1999)
5. Klinger, A.: Vector-Valued Performance Criteria. IEEE Transactions on Automatic Control 9, 117–118 (1964)
6. Pham, K.D.: Cooperative Solutions in Multi-Person Quadratic Decision Problems: Performance-Measure Statistics and Cost-Cumulant Control Paradigm. In: The 46th Conference on Decisions and Control, pp. 2484–2490 (2007)
7. Pham, K.D.: Non-Cooperative Outcomes for Stochastic Nash Games: Decision Strategies towards Multi-Attribute Performance Robustness. In: The 17th World Congress International Federation on Automatic Control (2008)
8. Starr, A.W., Ho, Y.C.: Nonzero-sum Differential Games. Journal of Optimization Theory and Applications 3, 184–206 (1969)
9. Starr, A.W., Ho, Y.C.: Further Properties of Nonzero-sum Differential Games. Journal of Optimization Theory and Applications 3, 207–209 (1969)

Switching among Solvers: Collaborative Algorithms with Parameter Control[⋆]

Carlos Castro and María Cristina Riff

Departamento de Informática
Universidad Técnica Federico Santa María
Valparaíso, Chile
{Carlos.Castro,Maria-Cristina.Riff}@inf.utfsm.cl

Abstract. In this work, we introduce a framework to design cooperation schemes among solvers, some of them done with a complete search and others with an incomplete search. We also include a parameter control criteria that guides the cooperation among solvers in order to improve the collaborative strategy. Our framework can be used to solve both Constraint Satisfaction Problems (CSP) and Constraint Satisfaction Optimisation Problems (CSOP). The tests are carried out on instances of the vehicle routing problem used to test a solvers cooperation for CSOP. The empirical results obtained show the substantial gain of collaborating.

1 Introduction

Constraint satisfaction problems (CSPs) occur widely in artificial intelligence. They involve finding values for problem variables which satisfy all the constraints. For simplicity we restrict our attention here to binary CSPs, where the constraints only involve two variables. Constraint Satisfaction Optimisation Problems (CSOP) consists in assigning values to variables in such a way that a set of constraints is satisfied and an objective function is optimized. Nowadays, complete and incomplete techniques are available to solve both kind of problems. Constraint Programming is an example of complete techniques where a sequence of Constraint Satisfaction Problems (CSP) is solved by adding constraints that impose better bounds on the objective function until an unsatisfiable problem is reached. Local search techniques are incomplete methods where an initial solution is improved considering neighbourg solutions. The advantages and drawbacks of each of these techniques are well-known: complete techniques allow, when possible, global optimality but when we deal with hard problems it takes a lot of time, or in the worst case, they do not give a solution. Incomplete methods give solutions very quickly but they remain local. Recently, the integration of these approaches has been studied and it has been recognized that hybrid approaches should give good results when none is able to solve a problem. Prestwich [5] proposes a hybrid approach that sacrifices completeness of backtracking methods to achieve the scalability of local search. His method outperforms the

⋆ Partially Supported by the Fondecyt Project 1080110.

best local search algorithms. Jussien and Lhomme [2] present an hybrid technique where local search performs over partial assignements instead of complete assignements, and it uses constraint propagation and conflict based heuristics to improve the search. They applied their approach on open-shop scheduling problems with encouraging results. In this work, we introduce a framework to design cooperation schemes among solvers, some of them performing a complete search while others an incomplete one. The biggest difference between our approach and the hybrid algorithms is that we use each solver as it is, designing a high level cooperation. This chapter is organised as follows: In Section 2 we introduce the general framework for cooperation. Section 3 briefly describes a cooperation scheme to solve CSP using min-conflicts [4] as an hill-climbing algorithm helped by forward checking. In Section 4 we apply a collaborative strategy to solve CSOP using forward checking helped by an external hill-climbing procedure. Parameter control heuristics are discussed in Section 5. In Section 6 we present the results of the tests using instances of the vehicle routing problem. Finally, in Section 7 we present conclusions and future research directions.

2 General Framework for Cooperation

Our idea for solvers to cooperate is to take advantage of the efficiency of both kind of techniques: complete and incomplete approaches. In the following, we will call the initial solver *i-solver* and the collaborative one *c-solver*. In our approach, *i-solver* begins solving a CSP, if this solver is stuck trying to find a solution, the collaborative algorithm detects this situation and gives this information to a *c-solver* method, in charge of quickly finding a new feasible solution.

The communication between *c-solver* and *i-solver* depends on both, the direction of the communication and the kind of cooperation. For instance, a cooperation algorithm to solve CSOP could begin with a complete algorithm as *i-solver*, then could pass the control to an incomplete *c-solver*. This incomplete algorithm could receive the variable values previously found by *i-solver* and try to find a new better solution applying some heuristics. As another example, the algorithm could give information about the local optima that it found, considering in this case *i-solver* as an incomplete approach. This information modifies the bound of the objective function constraint, and *c-solver* that uses a complete approach could work trying to find a solution for this new problem configuration. Roughly speaking, we expect that helping *i-solver* by *c-solver*, will reduce the algorithm search tree by discarding some branches or by focusing the search where it has a higher probability to obtain the optimal value.

In Figure 1, we ilustrate the general cooperation strategy.

3 Cooperation in Constraint Satisfaction Problems

A *Constraint Satisfaction Problem* (CSP) is a set of *variables* $V = \{X_1, \ldots, X_n\}$, their related *domains* $D_1, \ldots, D_n$, and θ, a set of η *constraints* between these variables. A variable must be instantiated from values within its domain. The domain

```
Procedure Cooperating Solvers ($i - solver, c - solver, P$)
Begin /* Procedure Cooperating Solvers */
While (Not OK-Solution)
   While ((Not max-stuck-i-solver) and (Not OK-Solution))
      pre-solution-from-i-solver = $i - solver(P)$
      If (Not OK-solution)
         While ((Not max-stuck-c-solver) and
            (Not OK-Solution))
            $P_{ri}$ = pre-solution-from-i-solver
            pre-solution-c-solver = c-solver($P_{ri}$)
            P = pre-solution-c-solver
         EndWhile
      Else solution = pre-solution-from-i-solver
   EndWhile
EndWhile
End /* Procedure */
```

Fig. 1. Cooperating Solvers Strategy

sizes are respectively $m_1, \ldots, m_n$, where **m** equal the maximum of all m_i. Each variable X_j is *relevant* (denoted by *"be relevant to"* $\rhd$), for a subset $C_{j_1}, \ldots, C_j$ of constraints where $\{j_1, \ldots, j_k\}$ is some subsequence of $\{1, 2, \ldots, \eta\}$. A binary constraint has exactly two relevant variables. A binary CSP can be represented by a constraint graph where nodes are the variables and arcs are the constraints. We talk about inconsistency or constraint violation when the relevant variables for a given constraint do not have values that can satisfy the constraints. An instantiation of the variables that satisfies all the constraints is a solution of the CSP.

Definition 1. *(Instantiation)*
An Instantiation **I** *is an assignment from a n-tuple of the variables* $(X_1, \ldots, X_n) \rightarrow D_1 \times \ldots \times D_n$*, such that it gives a value from its domain to each variable in V.*

Definition 2. *(Partial Instantiation)*
Given $V_p \subseteq V$*, a Partial Instantiation* $\mathbf{I_p}$ *is an assignment from a j-tuple of variables* $(X_{p_1}, \ldots, X_p\ ) \rightarrow D_{p_1} \times \ldots \times D_p$ *, such that it gives a value from its domain to each variable in* V_p*.*

Remark: We speak about satisfaction (or conflicts) of C_α *in* $\mathbf{I_p}$ *iff all the relevant variables of* C_α *are instantiated*

3.1 A Hill-Climbing Algorithm Helped by Forward Checking

In this Section we address how forward checking can help a Hill-climbing procedure when it is searching for a solution for a Constraint Satisfaction Problem. Forward Checking is a classical complete approach to solve CSP that instantiates the variables following some order and filters them during the search. We remark that this technique focuses on exploiting the search space, and it is a good algorithm to accomplish this kind of task. The most well-known incomplete algorithm to solve CSP is Min Conflicts proposed by Minton [4]. In this approach a local search procedure begins with a complete instantiation and it is modified by randomly choosing a variable and selecting a value for this variable from its domain, so that the number of constraint violations that are reduced. Using this algorithm many complex problems have been solved like scheduling and planning. However, this method has two drawbacks: It can quickly get stuck in a local optima, and it is not able to detect when a problem does not have a solution. Thus, we propose here to use forward checking, as *c-solver* to help the hill-climbing algorithm which is the *i-solver*, to tackle these problems. Figure 2 shows this collaboration. When the hill-climbing method gets stuck in a local optima it gives the best complete instantiation that it found to the collaborative algorithm. This algorithm constructs a partial instantiation by discarding from the complete instantiation the variables with conflicts. Then it gives the partial instantiations $(variable, value)$ to forward checking which constructs the associated branch and normally continues the execution filtering and searching the values of the rest of the variables not still instantiated on this branch. In this context hill-climbing gives to forward checking the "promising branch" to have a solution for the CSP problem.

```
Procedure Cooperating HC with FC for CSP ($C, i - solver, c - solver$)
Begin /* Procedure Cooperating HC with FC for CSP */
While (Not OK-Solution)
    While ((Not max-stuck-i-solver) and (Not OK-CSP-Solution))
        pre-solution-from-i-solver = $i - solver(C)$
        If (Not OK-CSP-solution)
            While ((Not max-stuck-c-solver) and (Not OK-CSP-Solution))
              branch-solution-c-solver = c-solver(pre-solution-from-i-solver)
            EndWhile
        Endif
    EndWhile
EndWhile
End /* Procedure */
```

Fig. 2. Cooperating HC with FC for CSP

4 Cooperation in Constraint Satisfaction Optimisation Problems

As in CSP, to solve a *Constraint Satisfaction Optimisation Problem* (CSOP) consists of assigning values to variables in such a way that a set of constraints is satisfied. But, a CSOP has also an objective function that must be optimized. Thus, the goal here is to find the best variable instantiation.

4.1 General Algorithm with Propagation

Constraint Programming deals with optimisation problems, CSOPs, using the same basic idea of verifying the satisfiability of a set of constraints that is used for solving CSPs. Asuming that one is dealing with a minimisation problem, the idea is to use an upper bound that represents the best possible solution obtained so far. Then we solve a sequence of CSPs each one giving a better solution with respect to the optimisation function. More precisely, we compute a solution s to the original set of constraints C and we add the constraint $f < f(s)$, where f represents the optimisation function and $f(s)$ represents the evaluation of f in the solution s. Adding this constraint restricts the set of possible solutions to those that give better values for the optimisation function, while still satisfying the original set of constraints. When, after adding such a constraint, the problem becomes unsatisfiable, the last feasible solution obtained so far represents the global optimal solution [1]. Very efficient hybrid techniques, such as Forward Checking, Full Lookahead [3] or even more specialised algorithms, are usually applied to solve the sequence of CSPs. Figure 3 presents this basic optimisation scheme.

```
Procedure Basic Optimisation in Constraint Programming
Begin
    $s$ = GenerateFeasibleSolution($C$)
    best-solution = $s$
    solution-value = $f(s)$
    While Problem has solution
        $s$ = GenerateFeasibleSolution($C$ & $f < solution - value$)
        best-solution = $s$
        solution-value = $f(s)$
    EndWhile
End /* Procedure */
```

Fig. 3. Basic Constraint Programming Algorithm for CSOP

In this approach, the goal is to find the solution named `Global-Solution` of a constrained optimisation problem with the objective function $Min \quad f$, and its constraints represented by C. The cooperating strategy is an iterative process that begins to try solving the CSP associated to the optimisation problem using an initial `i-solver`. This algorithm has an associated converge condition criteria, i.e., when it becomes enable to find an instantiation in a reasonable either time or number of iterations. The `pre-solution-from-i-solver` corresponds to the variables values instantiated until now. When `i-solver` is stopped because it has reached the converge condition, `c-solver`, continues taking as input the `pre-solution-from-i-solver`. `c-solver` uses it to find a near-optimal-solution for the optimisation problem until it reaches a local optimality. A new CSP is defined including the new constraint that indicates that the objective function value must be lower than the value found either by `i-solver` or by `c-solver`.

4.2 Forward Checking Helped by Hill-Climbing

In the last Section we refer the approach to help hill-climbing by forward checking when it must solve a Constraint Satisfaction Problem. In this Section we briefly describe a basic cooperation scheme between forward checking and hill-climbing to solve Constraint Satisfaction Optimisation Problems.

Forward Checking focuses on finding a solution on the branch coming from the first variable instantiated. When it gets stuck in its search on this branch, it may be useful to help it to do both actions: to abort this search and to go exploit another one. On the other hand, hill-climbing, a classical incomplete technique, beginning with a complete initial instantiation does exploration by changing the initial instantiation, and it continues the exploitation of the search space perturbing the complete instantiation by applying local moves. Because hill-climbing focus its search by exploiting the neighboor of the initial instantiation can converge very quickly to a local optima. The collaborative algorithm uses this local optima as a new bound for forward checking in order to identify what is the most promising branch to exploit after aborting.

```
Procedure Cooperating Solvers for CSOP ($f, C, i - solver, c - solver$)
Begin /* Procedure Cooperating Solvers for CSOP */
While (Not OK-Global-Solution)
   While ((Not max-stuck-i-solver) and (Not OK-CSP-Solution))
      pre-solution-from-i-solver = $i - solver(C)$
      If (Not OK-CSP-solution)
         While ((Not condition max-stuck-c-solver) and
            (Not OK-CSP-Solution))
            $P_{ri}$ = pre-solution-from-i-solver
            near-optimal-solution-c-solver = c-solver($P_{ri}$, f)
            Bound = near-optimal-solution-c-solver
         EndWhile
      Else Bound = pre-solution-from-i-solver
      $C = C \quad \& \quad f < Bound$
   EndWhile
EndWhile
End /* Procedure */
```

Fig. 4. Cooperating Solvers Strategy for CSOP

5 Parameter Control

The goal of implementing solvers that cooperate is to improve the efficiency of the algorithms compared to their performance alone. In the collaborative approaches that we presented in the last Sections we observe that the collaborative algorithms must take some actions during the search. However, to identify both when hill-climbing gets stuck in a local optima and when a branch that forward checking is exploiting is no a promising branch to continue searching are complex tasks. This problem is similar to find the good parameters for metaheuristic algorithms. It has been adressed in the metaheuristics community as parameter control. The parameter control problem itself is complex. There are many strategies to implement parameter control [7]. The simplest one is to allways change the parameter values, after a fixed number of iterations, by incrementing them with a fixed value. The most sophisticated one is the adaptive dynamic parameter control [7], where the algorithm itself changes its parameter values in order to do more, either exploration or exploitation depending on the execution. We can see that collaborative algorithms also have parameters. For instance, when hill-climbing helps forward checking, the collaborative algorithm must define any criteria that it considers to abort the exploitation of this branch (e.g., a number of iterations, elapsed time, value of the evaluation function). On the other hand, when forward checking helps hill-climbing, the collaborative algorithm must also define when it considers that hill-climbing finds a local optima, for example after some iterations, or after a given amount of applied moves. Our key idea here is to include the parameter control task into the collaborative algorithm using a dynamic and adaptive strategy according to the search. In this way we use in our approaches the following heuristics:

1. For CSOP:
   - If Forward Checking (FC) converges in a depth of the tree search and it cannot go ahead for a number of iterations, then it is time to give the control to Hill-Climbing (HC).
   - When HC is searching for the new bound and it cannot improve its best found pre-solution, FC takes control, searching for a solution of the new CSP
2. For CSP:
   - If HC obtains an instantiation with at least k variables without conflicts, then FC takes the control in order to find a value for the rest of the variables.
   - If FC cannot obtain a solution in this branch, that means for HC that it is not useful to continue searching here, and it must begin at another point.

6 Evaluation and Comparison

In this Section, we first explain the problems that will be used as benchmarks and then we present the results obtained using our cooperative approach.

6.1 Test Problems for CSOP

In order to test our schemes of cooperation, we use the classical Capacity Vehicle Routing Problem (CVRP). In the basic Vehicle Routing Problem (VRP), m identical vehicles initially located at a depot must deliver discrete quantities of goods to n customers, each one having a demand for goods. A vehicle has to make only one tour starting at the depot, visiting a subset of customers, and returning to the depot. In CVRP, each vehicle has a capacity, extending in this way the VRP. A solution to a CVRP is a set of tours for a subset of vehicles such that all customers are served only once and the capacity constraints are respected. Our objective is to minimise the total distance travelled by a fixed number of vehicles to satisfy all customers.

Our problems are based on instances $C101$, $R101$ and $RC101$, proposed by Solomon [6] belonging to classes $C1$, $RC1$ and $R1$, respectively. Each class defines a different topology. Thus, in $C1$ the location of customers are clustered. In $R1$, the location of customers are randomly generated. In $RC1$, instances are generated considering clustered groups of randomly generated locations of customers. These instances are modified to include capacity constraints. We named the so obtained problems as instances $c1$, $r1$, and $rc1$. These problems are hard to solve for a complete approach. The goal of our tests is to evaluate and to compare the search made by a complete algorithm, in contrast to its behaviour when another algorithm, which does an incomplete search, is incorporated into the solution process.

6.2 Evaluating Forward Checking with Iterative Improvement

The goal of our test is to evaluate and to compare the search made by a stand alone complete algorithm, with its behaviour when an incomplete algorithm is introduced to the search process. For the tests we have selected two well known techniques: Forward Checking (FC) from Constraint Programming and Hill Climbing or Iterative Improvement from Local Search. Forward Checking is a technique specially designed to solve CSP, based on a backtracking procedure, but it includes filtering to eliminate values that the variables cannot take in any solution to the set of constraints. Some heuristics have been proposed in the literature to improve the search of FC. For example, in our tests we include the minimum domain criteria to guide the instantiation of the variables. Thus, the first variable instantiated by FC has the smallest domain size.

On the other hand, local search works with complete instantiations. We select iterative improvement which is especially designed to solve CVRP. The characteristics of our iterative improvement algorithm are:

– The Initial Solution is obtained from FC.
– The moves are made by the 2-opt operator proposed from Kernighan.
– The acceptance criterium is "best improvement".
– It works only with feasible neighbourhoods.

The first step in our research was to verify the performance of applying a standard FC algorithm to solve problems $c1$, $r1$, and $rc1$ as defined above. Table 1 presents the obtained results. For each instance, we show the time at which the partial solution has been found and the corresponding evaluated objective function. All times one in milliseconds.

Thus, reading the last row of columns t and z for the $r1$ instance, we can see that the best value of $z = 375.66$ is obtained for the objective function after

Table 1. Application of Forward Checking

#	r1		rc1		c1	
	t	z	t	z	t	z
1	30	431.39	29803	383.32	37063	248.36
2	180	431.38	31545	367.00	37754	245.53
3	210	431.16	35200	364.61	40629	234.44
4	260	425.41	109537	359.83	40719	233.90
5	270	418.62	111180	357.78	40719	231.78
6	410	414.20				
7	560	404.92				
8	560	398.13				
9	1091	392.03				
10	38014	391.76				
11	38014	385.21				
12	38014	383.53				
13	51694	377.33				
14	51694	375.66				
15	106854	375.66				

15 iterations in 106854 seconds. In the same way, we can see that for instance $rc1$, the best value obtained by the application of FC is $z = 357.78$ after 5 iterations in 111180 seconds, and for instance $c1$, the value $z = 231.78$ is also obtained after 5 iterations in 40719 seconds. For all applications of FC in this work, we consider a limit of 100 minutes to find the optimal solution and carry out optimality proofs. This Table only shows the results of applying FC for solving each instance; we cannot deduce anything about these results because we are solving three differents problems.

Our idea to make these two solvers cooperate is to help FC when the problem becomes too hard for this algorithm, and take advantage of HC that could be able to find a new bound for the search of the optimal solution. In our approach, FC first finds a solution to a CSP. Then, this solution is used to compute a bound for the optimal value of the problem. If FC has problems trying to find a solution, the collaborative algorithm detects this situation and gives this information to a HC method that is charged with quickly finding a new feasible solution. The communication between FC and HC depends on the direction of communication. Thus, when the algorithm gives the control to HC from FC, HC receives the variable values previously found by FC, then the algorithm accepts the move, if and only if it improves the current solution.

When the control is from HC to FC, HC gives information about the local optima that it has found. This information modifies the bound of the objective function constraint, and FC tries to find a solution for this new problem configuration.

Roughly speaking, we expect that using HC, FC will reduce its search tree cutting some branches using the new bound for the objective function. On the other hand, HC focuses its search when it uses an instantiation previously found by Forward Checking on an area of the search space where it has a higher probability to obtain the optimal value.

The first scheme of cooperation that we have tried consists in:

1. We first try to apply FC looking for an initial solution.
2. Once a solution has been obtained, we try to apply HC until it cannot be applied any more, i.e., a local optimum has been reached.
3. Then, we try again both algorithms in the same order until the problem becomes unsatisfiable or a limit time is achieved.

The results of applying this scheme are presented in Table 2.

Table 2. Trying Hill-Climbing after Forward Checking

	r1			rc1			c1		
#	s	t	z	s	t	z	s	t	z
1	FC	30	431.39	FC	29803	383.32	FC	37063	248.36
2	HC	470	392.03	HC	30023	341.42	HC	37344	211.22
3	HC	820	379.62	HC	30214	294.99	HC	37654	197.30
4	HC	1090	358.99				HC	38090	194.07
5	HC	1360	353.66				HC	38330	191.05
6							HC	38580	189.05
7							HC	38810	187.44

In order to verify the effect of applying HC inmediately after the application of FC, we try the same cooperation scheme but we give the possibility to FC to be applied several times before trying HC. The idea is to analyse the possibility to improve bounds just by the application of FC. As we know that FC can need

Table 3. Trying Hill-Climbing two seconds after Forward Checking

	r1			rc1			c1		
#	s	t	z	s	t	z	s	t	z
1	FC	30	431.39	FC	29803	383.32	FC	37063	248.36
2	FC	180	431.38	FC	31545	367.00	FC	37754	245.53
3	FC	210	431.16	FC	33200	364.61	FC	39629	234.44
4	FC	260	425.41	HC	35491	294.99	FC	40719	233.90
5	FC	270	418.62				FC	40729	231.78
6	FC	410	414.20				HC	43165	197.89
7	FC	560	404.92				HC	43465	194.86
8	FC	560	398.13				HC	43795	191.45
9	FC	1091	392.03				HC	44025	189.45
10	HC	3803	379.62				HC	44265	187.85
11	HC	4083	358.99				HC	44505	187.44
12	HC	4374	353.66						

a lot time to get a new solution, we establish a limit of two seconds. If this limit is reached and FC has not yet return a solution, we try to apply HC. The results of this second scheme of cooperation are presented in Table 3.

We can make the following remarks concerning these results:

– Surprisingly, when solving each instance, both cooperation schemes found the same best value.
– The first scheme of cooperation (Table 2) always takes less time than the second one (Table 3). In fact, the total time is mainly due to the time expended by FC.
– In general, applying both cooperations schemes, the results are better, in terms of z, than applying FC alone.

7 Conclusions

The main contribution of this work is that we have presented a framework to design cooperative strategies integrating solvers for combinatorial constrained problems. The results reported show that Hill Climbing can help Forward Checking by adding bounds during the search procedure. This is based on the well-known idea that adding constraints, can usually improve the performance of Constraint Programming. We have also tested a collaboration strategy using randomly generated binary CSP. We remark that when HC is charged with solving Constraint Satisfaction Problems, FC can help it, taken only the variables without conflicts from the solution found by HC and to find quickly the rest of the variables values to complete a good instantiation. If FC cannot find a complete instantiation, it is able to decide that there is not a solution ahead and that it is not useful to continue searching moves on this branch. Thus, HC can continue its search from another point. Nowadays, considering that the research carried out by each community separately has produced good results, we strongly believe that future research will consider the integration of both approaches.

References

1. Bockmayr, A., Kasper, T.: Branch-and-Infer: A unifying framework for integer and finite domain constraint programming. Computing, INFORMS 10(3), 287–300 (1998); Also available as Technical Report MPI-I-97-2-008 of the Max Planck Institut für Informatik, Saarbrücken, Germany
2. Jussien, N., Lhomme, O.: Local search with constraint propagation and conflict-based heuristics. Artificial Intelligence 139, 21–45 (2002)
3. Kumar, V.: Algorithms for constraint satisfaction problems survey. AI Magazine 13(1), 32–44 (1992)
4. Minton, S.: Integrating heuristics for constraint satisfaction problems: A case study. In: Proceedings of the Eleventh National Conference on Artificial Intelligence, pp. 51–62 (1993)

5. Prestwich, S.: Combining the scalability of local search with the pruning techniques of systematic search. Annals of Operations Research 115, 51–72 (2002)
6. Solomon, M.: Algorithms for the vehicle routing and scheduling problem with time window constraints. In: Operations Research, pp. 254–265 (1987)
7. Hinterding, R., Michalewicz, Z., Eiben, A.: Parameter control on evolutionary algorithms. IEEE Transactions on Evolutionary Computation 3(2), 124–141 (2000)

An Event-Driven Software Architecture for Multiple Unmanned Aerial Vehicles to Cooperatively Locate Mobile Targets

Dimitri Zarzhitsky, Marc Schlegel, Andreas Decker, and Daniel Pack

Department of Electrical and Computer Engineering
US Air Force Academy, CO 80840, USA
{dimitri.zarzhitsky,daniel.pack}@usafa.edu

Abstract. Today's state-of-the-art, real-time distributed systems require sophisticated software engineering techniques to support a diverse set of mission requirements. This chapter discusses an event-driven approach to the design of such systems in the context of a fully-functional search and localization application intended for unmanned aerial vehicles. Here, the discussion focuses on the basic design principles that form the foundation of the distributed, scalable, and cross-platform implementation, and it provides an evaluation of the system's performance on a set of representative tasks. The main contribution of this effort consists of a modular software framework that is well-suited for solving problems in distributed intelligence and autonomous robotic systems. This framework is an enabling technology that integrates different functional components, such as sensing, analysis, and control into one cohesive platform. This solution employs off-the-shelf commercial technologies, resulting in a low-cost implementation that exhibits both robustness and flexibility.

Keywords: cooperative search and localization, distributed sensor networks, multi-threaded design, intelligent systems.

1 Introduction

The main goal of this research is to develop an autonomous, cooperative, and heterogeneous Multiple Unmanned Aerial Vehicles (MUAVs) system that uses heterogeneous on-board sensors to search, detect, and locate moving ground targets. The necessary technologies have matured to make an implementation on hardware UAVs possible. A non-trivial challenge for the implementation is the integration of the associated control, sensor, and communication technologies. The subject of this chapter is a high-performance software architecture that merges interdependent air-to-air-to-ground control, sensor, and communication network functionalities into a cohesive intelligence, surveillance, and reconnaissance (ISR) system [1]. A novel aspect of our approach is the event-driven software implementation. Instead of relying on an explicitly-programmed flow of information and control through the entire system, each individual UAV reacts autonomously and independently to several indeterministic mission events. This chapter explains and illustrates how a methodical application of parallel software engineering techniques resulted in a multi-threaded, cross-platform, distributed software operating environment that optimizes utilization of available processing resources to improve the MUAVs system performance on the ISR task. This novel approach relies on widely-available off-the-shelf hardware and software products, and places a significant emphasis on the practical aspects of developing and deploying a functional ISR system. The solution consists of three major components: surveillance aircraft, ground station infrastructure, and a suite of supporting technologies for archiving and processing mission data.

To detect mobile ground targets (e.g., vehicles) we developed a fully autonomous unmanned aerial vehicle (UAV) platform, equipped with an on-board PC/104+ computer running the Linux operating system. In order to do so, our research team identified and incorporated several different types of sensors into the UAVs: the planes search for targets using a camera, covering both visual (VR) and infrared (IR) ranges. We also developed a unique prototype radio frequency (RF) sensor, capable of locating targets based on their radio emissions; this new sensor was tested successfully in both a laboratory setting and in an actual field deployment. These and several other airborne components comprise the *heterogeneous on-board processing system* (HOPS). Mission objectives and search progress are specified and monitored via the *multiple UAVs ground station* (MUGS), which consists of a wireless communication relay and several Windows-based laptops running avionics and ISR-specific software. Figure 1 illustrates a typical ISR scenario, and shows how the different components of the MUAV system may be deployed in the field.

The MUAV ISR system also includes an archive and visualization station, called SUDS (short for *server unit for display and storage*), that uses the 3D Google Earth rendering engine and a Microsoft SQLServer database back-end to display real-time status of multiple UAVs as they perform search, detection, and localization of ground vehicles. This ISR platform is a complete functioning prototype, thoroughly tested in software simulation, hardware-in-the-loop configuration, and in actual flight testing.

The chapter begins with an overview of the MUAV software architecture, focusing on the multi-threaded and event-driven aspects of the implementation. The discussion then highlights the key aspects of the software modules responsible for communication, sensor processing, and mission control. It is followed by a performance analysis of the framework based on a broad range of statistics collected during flight testing, and then examines the fault-tolerance properties of the implementation. The last section in the chapter summarizes the design trade-offs and offers ideas for future work in this area.

2 Motivation

This software engineering effort originated in response to a problem of component integration. Research and development activities performed by the Sensor-based Intelligent Robotics Laboratory at US Air Force Academy produced a set of

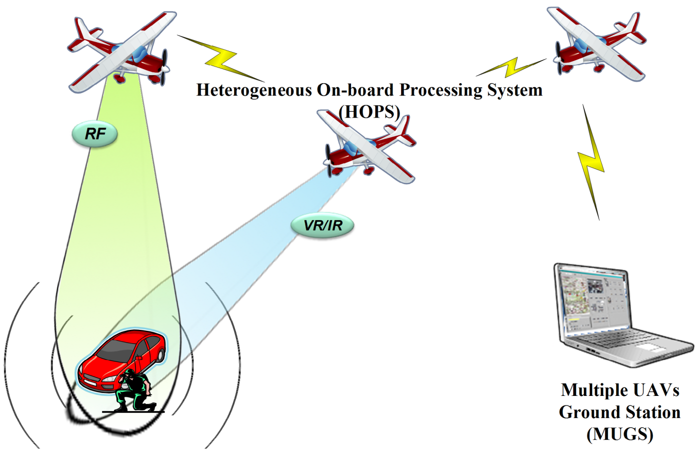


Fig. 1. An envisioned ISR scenario, in which multiple UAVs use heterogeneous sensors to locate mobile ground targets. The autonomous UAVs first form an airborne sensor and communication network, and then share the obtained sensor data to cooperatively search the area of interest. Note that the UAVs may form a multi-hop communication relay to report their findings to the ground station operator, although these individual navigational decisions are made separately by each UAV, without the need of explicit commands from a human observer.

requirements that a field-ready ISR implementation had to satisfy [2]. The system design called for multiple unmanned aircraft to carry a suite of different sensors, and the UAVs were tasked with locating several mobile targets on the ground. The aerodynamic control of the airframe was delegated to the Piccolo II autopilot, manufactured by Cloud Cap Technologies [5]. The autopilot on-board each UAV received periodic waypoint updates from the HOPS Control subsystem based on the UAV status, sensor observations, and the pertinent sensor and control data communicated by neighboring UAVs. When a target is detected, the target's location is estimated and updated with each additional sensor measurement obtained by an on-board sensor, or a target estimate communicated by a neighboring UAV. The sensor fusion process combines local and remote sensor measurements via a novel Kalman filter capable of processing out-of-order observations [3]. The functional components of this MUAV platform are shown in Figure 2. The sophistication of this design, its emphasis on the UAV autonomy by means of on-board processing of large amounts of data, and the resulting need to support multiple third-party hardware components necessitated an equally sophisticated software framework that unified the separate functionalities into a single real-time application. The software platform had to be agile enough to support an active research project, as requirements and specifications changed frequently. But at the same time, it also had to be robustly fault-tolerant in order to insure the safety of the airplanes.

Because the MUAV software framework unifies several different ISR functionalities into one application, it should be viewed as an enabling technology for

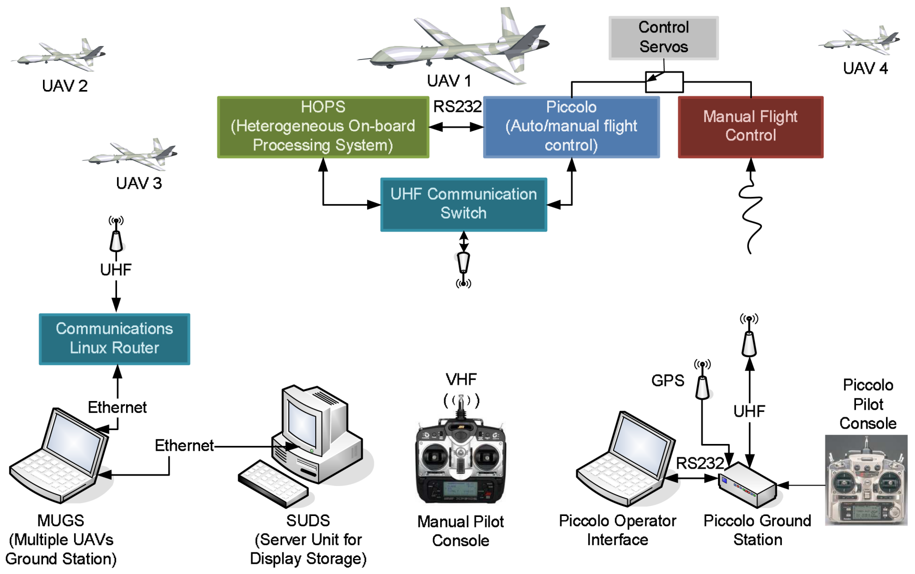

UAV 2
UAV 1
Control Servos
UAV 4
HOPS (Heterogeneous On-board Processing System)
RS232
Piccolo (Auto/manual flight control)
Manual Flight Control
UAV 3
UHF Communication Switch
UHF
Communications Linux Router
Ethernet
Ethernet
VHF
GPS
UHF
RS232
Piccolo Pilot Console
MUGS (Multiple UAVs Ground Station)
SUDS (Server Unit for Display Storage)
Manual Pilot Console
Piccolo Operator Interface
Piccolo Ground Station

Fig. 2. A functional component overview of the MUAV system, showing the combined requirements of autonomous flight and ISR operations

the algorithms and mission strategies that were developed in [1]. The software architecture owes much of its design to the reactive nature of a typical ISR mission, during which the aircraft monitors an area of interest and then alters its course in response to several events, such as the "target detected" signal from the on-board sensor fusion process, or the "help is needed" message received from another UAV. When *a priori* information about the target is limited, it is impractical to pre-program all of the UAV actions in advance. Furthermore, scripted surveillance missions become easy to subvert as soon as the search strategy is understood by the adversary. On the other hand, the use of fully autonomous, intelligent agents capable of real-time mission task planning based on the latest field observation data gives rise to a sophisticated and robust reconnaissance system.

Another noteworthy aspect of the MUAV design is the modularization of all sub-systems, such as the communication, sensor network, and navigation planning. Building and delivering a functioning ISR system requires a great deal of creative collaboration, subject to strict financial and scheduling constraints. An effective approach to managing the risks and uncertainties is to separate shared functionalities, such as communication and data archiving, so that concurrent development can take place on as many components as possible. This is why the software was architected using a multi-tiered hierarchical structure, with well-defined component interfaces that consist of a small set of decentralized command instructions. The remaining sections of this chapter provide more detail on this disciplined design.

3 MUAV Software Architecture

One of the most challenging software requirements that had to be addressed by the implementation is the need for cross-platform operation. The Linux operating system (OS) was selected for managing aircraft systems due to its open source model, which made it possible to perform modifications to the operating system's kernel in order to adapt to specific hardware, and to take advantage of useful network utilities, such as the MobileMesh dynamic routing protocol [7]. On the other hand, Microsoft Windows XP was chosen as the OS for the ground station components because of a larger Windows user base, and the high likelihood of Windows being a more familiar work environment for the MUAV system operators. Because of time and budget limitations, it was not practical to develop custom versions of both Linux and Windows components. To satisfy the multi-OS requirement, all of the MUAV software is built using the Qt cross-platform development framework [9], so that only very specific and isolated functionality, such as access to the universal serial bus (USB) devices or remote connections to the database are operating system dependent – the rest of the implementation will function "out of the box" on any platform that supports the Qt toolkit.

The Qt framework is a very powerful and feature-rich platform: it provides OS-independent modules for multi-threading and concurrent process execution, networking, structured query language (SQL) database access, and tools for graphical user interface (GUI) design. The Qt toolkit comes with comprehensive documentation, which greatly reduced the learning time for the MUAV development team. Furthermore, Qt is an *event-driven* architecture. This important aspect deserves a more in-depth discussion.

3.1 Event-Driven Design

Event-driven programming is a variation (or extension) of the more widely used object-oriented design methodology. Using the event-driven terminology, an *object* is an entity with an internal state, which can be observed by other objects. The internal state of an object is likely to alter as a result of changes in the state of the other objects in the system. Such state transitions are represented by events, and the logic of a program (e.g., what an event-driven program does) is determined by what events are produced, and how they are processed by the other objects that constitute the program. In other words, event-driven objects are *aware* of their environment, and they can *react* independently to changes in their environment. Thus, each event-driven object is a reactive automaton, programmed in advance with a set of behaviors that determine how this object affects and is affected by other components of the program.

An event-based program differs from a typical object-oriented implementation in its lack of explicit command directives between objects. For instance, consider the case of an aircraft losing a connection to the ground station due to a radio failure. With a "standard" object-oriented approach, the communication subsystem would contain a reference to the avionics control module, which it might then use to instruct the aircraft to abort the mission and return back to the base. In this case, the RETURN-TO-BASE directive is an explicit command issued by the communication module to the avionics controller, and the logic of what to do upon loss of the radio link is contained within the communication module. Now consider an event-driven implementation, where the communication module does not issue commands directly to the avionics sub-system. Instead, the communication object will first emit a LINK-DOWN signal, and after the avionics object detects this event, it will then execute the RETURN-TO-BASE behavior. Note that in the event-driven design, the logic of what the aircraft does in case of a hardware malfunction resides inside the avionics controller – arguably a more intuitive assignment than the communication sub-system in the object-oriented approach. In addition, there may be several other conditions that might trigger the RETURN-TO-BASE behavior, or it may also be possible that there are cases where the loss of a radio should not abort the rest of the mission. Hence, instead of having multiple sub-systems control the motion of the UAV, the avionics sub-system retains exclusive control of the aircraft throughout the mission, and it decides autonomously on the optimal action based on all of the events produced by other sub-systems. Thus, the event-driven approach helps to improve the program's *cohesion*, by reducing the number of separate places in the code where decisions are computed.

In addition to increasing the program's cohesion, the event-driven method can also reduce module *coupling* by eliminating the need for explicit references to other modules in the system. For instance, using the previous example, an event-based communication sub-system does not even need to know about the avionics module – all it needs to do is produce proper state transition signals, which then can be interpreted by any vehicle control module that may be active at the time the signal is emitted. In fact, this is exactly the approach used during ground-based test and validation of the MUAV software framework, for which autonomous ground vehicles (AGVs) are employed instead of airplanes. To support this advanced functionality, the software implements two separate event-driven hardware controllers, one for the UAVs and another for the ground robots. Thus, the system can be made to operate in two very different configuration modes by simply activating the appropriate controller. This allows the MUAV development team to perform extensive performance testing along with the validation of new code on the ground, without endangering the expensive UAVs. All that is required of the ground crew is to connect (in software) the navigation signals to the AGV robot controller module, and the rest of the system then behaves exactly as it did before, without any explicit changes to the software. The Qt framework ensures that such modifications are easy and fast to perform.

Readers who have prior experience with event-driven programming will find Qt's approach to inter-module communication straightforward. Every Qt entity can be described via a set of signals, which correspond to some observable property of the object (see Figure 3). For instance, when a mission-planning agent calculates a new waypoint, it emits the NEW-WAYPOINT signal. In turn, the aircraft navigation module listens for this signal (among many others), and

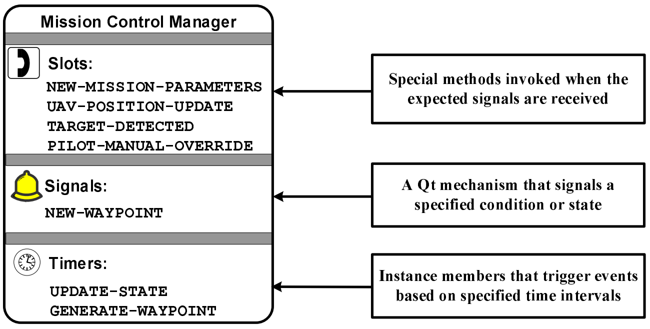

Mission Control Manager
Slots:
NEW-MISSION-PARAMETERS
UAV-POSITION-UPDATE
TARGET-DETECTED
PILOT-MANUAL-OVERRIDE
Signals:
NEW-WAYPOINT
Timers:
UPDATE-STATE
GENERATE-WAYPOINT
Special methods invoked when the expected signals are received
A Qt mechanism that signals a specified condition or state
Instance members that trigger events based on specified time intervals

Fig. 3. Qt's event-driven architecture is based around signals (e.g., notification events emitted as a result of state transitions) and slots (e.g., methods that execute in response to connected signals). Timer objects may be used to dispatch periodic events.

after detecting the signal's emission, it responds by invoking one of its *slots*, i.e., special methods that can be "connected" to signals. In other words, an object's slots determine the object's responses to changes observed in the system, while an object's signals are used to inform others of new changes. So in the previous waypoint example, one might connect the `NEW-WAYPOINT` signal to a `CHANGE-HEADING` slot of the UAV navigation module.

3.2 Event-Driven Implementation

The most intriguing aspect of an event-driven implementation is the unusual way the MUAV software components interact – there is no top-level control loop, and there is no predefined program execution path. Instead, every object within the system is controlled by events, because a component's response to an event is determined only by the slots that have been associated with that signal. A brief reflection on the practical ramifications of this design reveals that it is no longer possible to accurately predict *exactly* when any given function or slot will execute, and the precise order in which an event might be processed by different objects cannot be predicted. Instead, the system is best thought of as a large collective of independent agents that monitor and respond to a predetermined set of events. Thus the only reliable method of system analysis is to estimate the upper and lower bounds of when the execution of a given event handler might take place.

What is then the practical advantage of such an approach? For one, it achieves the important software engineering goal of decoupling the system into separate components that can be developed in parallel, thus reducing the total production time. As mentioned previously, another benefit is the increase in the cohesion of the modules: event-driven objects have few explicit dependencies on other components, so the individual modules can be activated and deactivated quickly, without disturbing other parts of the program. This provides an easy but safe mechanism to significantly alter the system's behavior in mid-flight (such as switching from autopilot to manual pilot control). This run-time dynamic system configuration feature proved to be a natural fit for the ISR problem, simplifying the MUAV software implementation.

Consider the event when the UAV first detects a target and several actions must be taken: the new sensor observation has to be fed into the on-board sensor fusion module, the control mode needs to be switched from search to tracking, information about the new target sighting has to be communicated to other UAVs, and the ground station operator must be notified of the detection. The sub-systems sensor fusion, control, and communication are affected by this event. Thus, if a procedural programming approach were to be followed, then the software designer would have to write an external condition statement to process the target detection event, directly invoking the necessary functions of each sub-system. Not only does this result in the unnecessary and undesired coupling of otherwise independent sub-systems, it moves the decision-making logic outside of the sub-system module, making component reuse more difficult. On the other hand, using the event-based paradigm, each sub-system manager can simply "listen" for the target detection event, and act on it independently of all the others. Exactly how each sub-system reacts to any given event is determined solely by the sub-system's programmer. And the decision algorithm remains encapsulated inside the module, making it easy to rearrange and interchange the modules to satisfy different mission requirements. This is exactly the mechanism that gives the ground crew the ability to replace the interface module for the UAV autopilot with one for AGV rovers, and safely perform initial validation of new untested features on the ground, all without making any changes to the main MUAV program.

The theme of modularization with heavy component reuse is very pronounced in the MUAV software implementation. The cross-platform support of the Qt framework allows for the same implementation of internal data structures and functional operators to be used on both the Linux-based aircraft and the Windows-operated ground station. Furthermore, because the focus of this research effort is on the collaborative aspects of the surveillance problem, communication between airborne UAVs plays a central role in the system. This fact must be reflected prominently in the software. In addition to information being transmitted from neighboring UAVs and the ground station, each vehicle also manages a large set of internal data. For instance, aerial images captured by the on-board camera are analyzed by Target Detection software; when a target is detected, the Target Detection module outputs a set of bearing angles, which are then passed into the UAV Sensor Fusion module (see Figure 4) for further processing. The Sensor Fusion module operates on the information received from local as well as remote sensors to provide the Mission Control module with a likely position of the ground target. In turn, the Mission Control module uses the information about the state and location of cooperating UAVs and mission objectives provided by a human observer to compute a desired flight path which is then sent to Vehicle Control for maneuvering the aircraft into a desired trajectory. The large volume of information, its irregularity, and the intermittent nature necessitate a regulation mechanism for the data flow, to ensure that sudden bursts or interruptions in the information stream do not jeopardize the mission. In the MUAV software architecture, the management and coordination of the various sub-systems (along with their input and output data) is performed via the sub-system manager objects, connected together as depicted in Figures 4 and 5, which show the functional block diagrams of the MUAV software for both the aircraft and the ground station components.

As can be observed in Figures 4 and 5, the UAV and the ground station software packages have a similar, symmetric structure. The main purpose of a sub-system manager, outlined as a hexagon in the diagrams, is to encapsulate the functionality of the given sub-system, and to isolate it as much as possible from all the other unrelated activities performed by the UAV. The shared module of the two MUAV components is the communication sub-system, which is used in both aircraft and ground station operations. During the mission, a variety of messages are constantly exchanged among the UAVs and the ground station – a process facilitated by the Communication Manager. The rest of the system data flows in a similar manner – exchanged by the managers of the internal sub-systems via the Qt signal-slot mechanism. For example, the Communication Manager enables other sub-system managers to send and receive their respective mission messages, obtain the network address associated with the UAV, and to intercept commands from the human supervisor. The On-board Hardware Manager on the other hand, is tasked with overseeing UAV hardware, such as the autopilot, and maintaining a real-time record of flight-related status information.

As was mentioned before, the hierarchical organization of the implementation makes it possible to alter the MUAV configuration at run-time by simply turning on and off the respective sub-system manager. The common *Dispatcher* object, shown in the center of the HOPS and MUGS diagrams (see Figures 4 and 5, respectively), provides a structured interface for connecting and disconnecting sub-system components. In addition, the Dispatcher provides shared memory storage and uniform access to system resources, such as the system clock, or UAV position and attitude information. The implementation supports both high-level interrupt handling and regular polling models for information processing. An example of an interrupt-driven process is the update of the HOPS Sensor Fusion module in response to target detection events produced by the Sensors sub-system. A more traditional polling method is used to read the serial port interface of the autopilot when updating the current UAV state information. Likewise, on the ground station, the GUI sub-system generates events when the human operator interacts with the MUAV user interface, while the Performance Monitor, a component inside the MUGS ISR Mission sub-system, uses an internal timer to produce a database record for a "global snapshot" of the system state at regular time intervals for post-mission analysis (see Figure 5). The next section contains a more detailed overview of the MUAV Communication and the HOPS Sensors sub-systems – two practical examples that illustrate how the ideas of event-driven design impact a real-world software application.

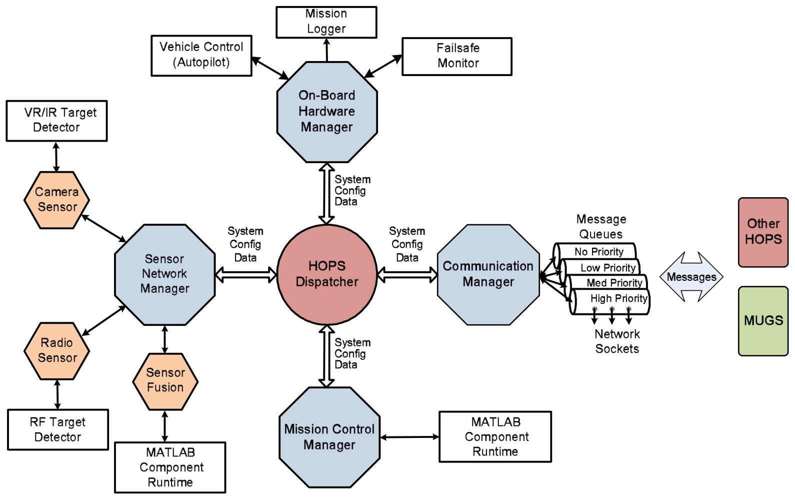


Fig. 4. Conceptual view of the internal organization of the heterogeneous on-board processing system (HOPS). Each distinct functionality of the aircraft (e.g., sensing, control, etc) is encapsulated inside an independent sub-system governed by a separate sub-system manager. The top-level sub-system managers facilitate data flow between the internal modules, and provide a generic representation of the functional capabilities of the MUAV system.

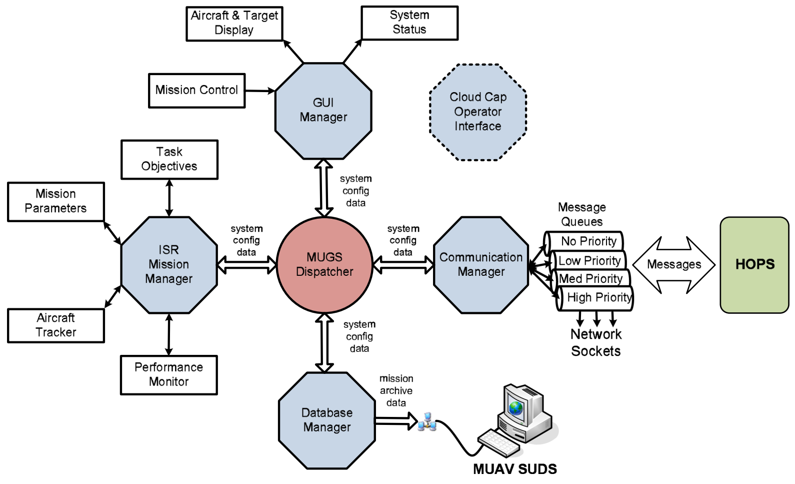


Fig. 5. Internal sub-system structure of the multiple UAVs ground station (MUGS) component. Similar to the HOPS software structure, the ground station software is a collection of independent, reactive sub-systems that collaborate via a generic interface that exposes the functional capabilities of each module, without requiring explicit command and data links between the individual objects that comprise the interacting sub-systems.

4 MUAV Software Implementation Details

Before presenting specific details on the implementation of the MUAV Communication and HOPS Sensors sub-systems, it is necessary to briefly list the overall software and hardware specifications of the MUAV system. Most of the MUAV software is written in C++ using the open source edition of Qt [9]. In addition, several third-party application programming interfaces (APIs) and libraries, also written in C/C++, provide support for the on-board UAV hardware, such as the Piccolo II autopilot [5], Axis 212 PTZ digital camera [4], and the Opal Kelly Field-Programmable Gate Array (FPGA) board [8]. On-board sensor processing and real-time mission planning are carried out using the Kontron PC/104+ computer with a 1.8 GHz Intel Pentium M processor and 1 GB of main system memory [6], running the Linux operating system. MUAV aircraft communicate with each other and with the ground station by using a commercial 802.11b extended-range radio, configured for an ad-hoc wireless network. Since the Qt framework features powerful multi-threading capabilities, each subsystem depicted in Figures 4 and 5 is implemented as a separate Qt execution thread. The concurrency of the independent modules allows the system to optimize resource utilization, something that will be addressed in more detail in Section 5.

4.1 MUAV Communication Sub-system

In order to locate targets quickly and accurately, the solution of the collaborative ISR problem requires a robust communication network. One of the goals of this project is to design a communication infrastructure capable of supporting arbitrary types and numbers of messages for both Windows and Linux operating environments. The MUAV communication system provides an acknowledgment of important messages (such as sensor configuration data), and also provides an application level time-to-live support by limiting the number of message retransmissions. The Communication sub-system manager is in charge of four message queues, each with a different priority; this allows the system to prioritize all network traffic (see Figure 4). The message queues in turn connect to persistent network sockets, allowing multiple UAVs to maintain active communication links with each other and with the ground station throughout the mission. In addition to regular message traffic, carried over the standard transmission control protocol and the Internet protocol (TCP/IP) network sockets, the MUAV software also supports streaming of live video from the UAV – a feature that makes use of the unreliable, but low-overhead user datagram protocol (UDP) client-server connections.

The preceding section described the event-oriented design philosophy that shapes the MUAV system. Many events within the MUAV software are implemented as communication messages (e.g., `TARGET-DETECTED`). In fact, because each message structure consists of just the payload data and a small header containing information about the source of the event and its time stamp, almost all of the internal data in the system is stored as a sub-type of a generic communication message. This approach reduces the number of expensive memory allocation operations needed during the system's operation, and simplifies the implementation with a common interface to all of the MUAV data. Since every MUAV data object derives from one common parent type, implementation of some extremely useful functions (e.g., creating archive logs) is trivial. The use of a common message type also allows for a straightforward and efficient flow of message data throughout the system. When a new message needs to be transmitted, the corresponding message object is serialized into a binary data stream. Each specific message type provides its own implementation for this functionality, but the first 16 bits always encode a unique type identifier. The MUAV Communication Manager implements a message handler registry that allows MUAV sub-system managers to request event notification when a message of particular type is detected. When the data stream is received on the destination host, the type identifier is used as an index into the message handler registry, and all sub-system managers who registered for messages of the same type are immediately notified. Of course the Communication Manager never needs to know the contents of the message, nor is it affected by how the other managers handle the message – it simply routes the data and waits for another message to arrive. A common ancestral type shared by all of the MUAV messages makes it easy to implement a generic processing routine for all possible MUAV events, without forcing the Communication Manager to know how to interpret each message type. This gives the Communication Manager the flexibility to handle arbitrary messages without being affected by future additions of new messages or changes in the version of the existing message types. A simplified example scenario of requesting a UAV video stream is depicted in Figure 6.

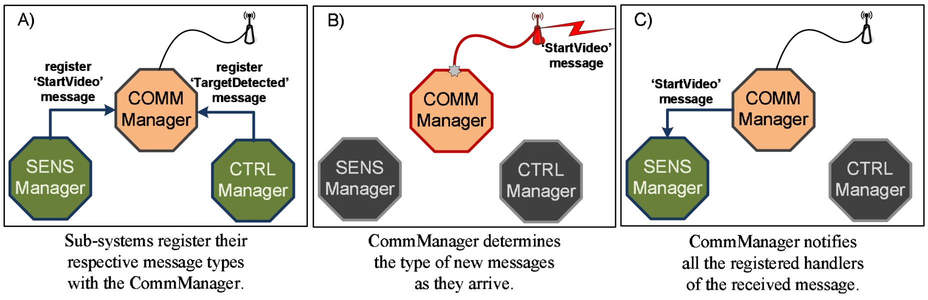

A)
register 'StartVideo' message
register 'TargetDetected' message
COMM Manager
SENS Manager
CTRL Manager
Sub-systems register their respective message types with the CommManager.
B)
'StartVideo' message
COMM Manager
SENS Manager
CTRL Manager
CommManager determines the type of new messages as they arrive.
C)
'StartVideo' message
COMM Manager
SENS Manager
CTRL Manager
CommManager notifies all the registered handlers of the received message.

Fig. 6. A sample execution sequence of MUAV message processing. In (A), sub-system managers inform the Communication Manager of the message types they would like to receive. This registration process can be carried out at any time, allowing the MUAV sub-systems to dynamically modify their event-handling capabilities. In (B), the arrival of new network messages causes the Communication Manager to create new MUAV communication message events, which are then routed in (C) to all the sub-system managers that are currently registered for this type of the event.

4.2 HOPS Sensors Sub-system

The ability of the UAVs to localize ground targets depends to a large extent on the Sensor Network component of the MUAV system. From the software standpoint, the two main challenges are the integration of different hardware interfaces and the coordination of data flow through the rest of the system. Figure 7 shows a diagram of the MUAV sensor network module, simplified to include only the camera and radio sensor interfaces. The other very important component of the MUAV sensor network is the Sensor Fusion module (see Figure 4). A detailed description of this module is omitted from this chapter because it is not relevant to the current discussion of software engineering methodology, but it is covered in detail by Plett, et al. [3].

As can be inferred from the structural symmetry depicted in Figure 7, the HOPS Sensor Network consists of similar data processing and interface modules, once again highlighting the theme of component reuse that is important in both the MUAV design and implementation. The Sensor Network Manager oversees two separate sensor-specific managers – the camera and the RF controllers. These controllers provide a convenient hardware abstraction, allowing for a clean separation and partition of hardware specific functionality into much smaller, and more streamlined Driver objects, which act as Qt adapters between

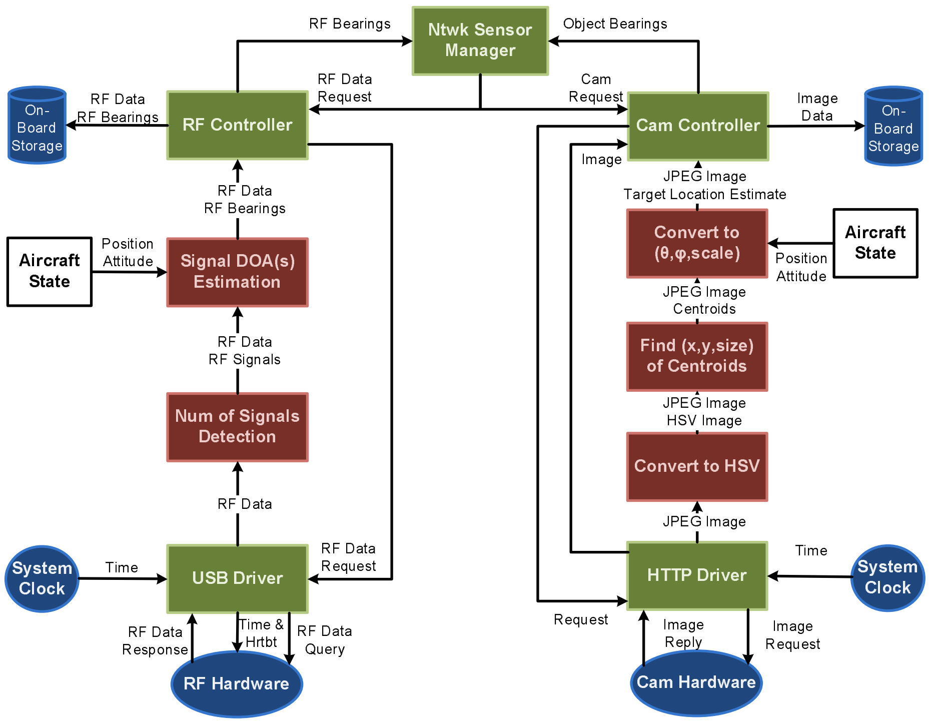


Fig. 7. Sensor Network

the high-level event-driven representations of the hardware and the low-level device drivers. As noted in Figure 7, the RF sensor uses a USB interface (provided by the OpalKelly FPGA board [8]), while the Axis camera is equipped with an embedded web server [4], which is accessed via the hypertext transfer protocol (HTTP). However, in spite of the different interfaces, both sensors output similar data, namely a bearing to the target (estimated from RF and visual sensor data), and an elevation to the target (camera data only). Because of the similarity in the sensor outputs, the Sensor Network Manager can relay sensor results to both the on-board Sensor Fusion module and the other UAVs participating in the search for the location of a target via a remote network link.

A noteworthy aspect of the MUAV Sensor Network implementation is the method of initiating and processing sensor readings. The MUGS user interface allows the MUAV system operator to turn on and off the individual sensor sub-systems to fit a variety of different missions. In most cases, a Qt timer object is used to periodically request and transmit data acquired by the sensor. In this way, the on-board camera manager can start a video streaming timer that initiates a downlink of a new surveillance image to the ground station every 125 milliseconds, while starting another timer to transmit the detected target's radio signature once every five seconds. Furthermore, these timers are dynamic: they can either be adjusted manually by the ground station operator or automatically by the Sensor Network Manager in response to different events.

In particular, the camera is a good example of a *shared resource* – it provides a single JPEG image for each HTTP request. Thus, when the operator is streaming UAV video to the ground station, and the UAV is using the camera to search for the target, two separate threads must use concurrency primitives, such as mutexes and read-write locks, to obtain the images. In addition, the requirements of the two processes are different – the real-time ISR video feed uses a stream of low-quality images with fast frame rates, while the target sensor needs high-quality, high-resolution data at a much slower refresh rate. Multi-threading and a combination of different timers allow the HOPS Sensor Network Manager to satisfy these two different goals in a structured, well-defined manner. Since both of these processes require camera images that differ only in their parameters (e.g., size, quality, etc), the same HTTP driver can concurrently accept requests for target imagery as well as the surveillance video stream. The empirical performance of the MUAV Sensor Network on real data collected during flight testing is the main topic of the next section.

5 MUAV Software Performance Analysis

As mentioned in the previous section, the HOPS camera sensor is a shared resource that provides image data needed for target detection, as well as the surveillance video for display on the MUGS user interface. Also, keep in mind that these two processes operate independently of each other. Therefore, it makes sense to examine how the system performance is affected by the interaction of these two functionalities. Figure 8 shows a plot of the surveillance video frame

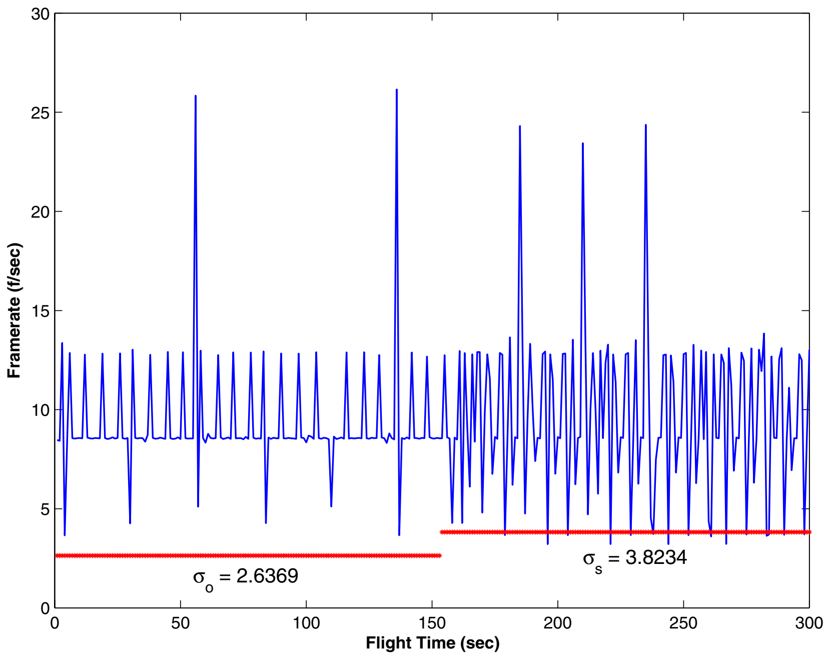


Fig. 8. Surveillance video throughput measured on the MUGS ISR display. The first 150-second period has less variance than the second sample due to the absence of competition for the camera sensor from the target detection process, which was turned on at approximately $t = 150$. Note that the average video frame rate remains nearly constant at the desired 8 fps rate.

rate prior to and after the initialization of the target VR sensor. For this test flight, the video timer was configured with a 125 millisecond timeout, yielding the desired video frame rate of 8 frames per second (fps). The plot in Figure 8 shows a time trace of the observed video frame rate on the MUGS ISR display. This data accounts for several factors, including network latency and concurrent processing of multiple tasks on both HOPS and MUGS. Note that the average frame rate is consistent with the expected 8 fps rate. However, after approximately 150 seconds, which coincided with the activation of the target image processing thread, the variability of the frame rate increased significantly – the timing period in the curve is less regular, and the variance of the measured video frame rate is higher. In fact, the standard deviation of the first 150-second period is 2.64, which then increases to 3.82 in the following 150-second sample. This increase in the variance demonstrates how the HOPS software is adapting to the increased demand for the camera – it attempts to maintain the specified 8 fps rate (as indicated by the nearly constant mean), but it does so via "catch up" video transmissions that compensate for delays incurred while waiting for the camera sensor to process the high-resolution target image data.

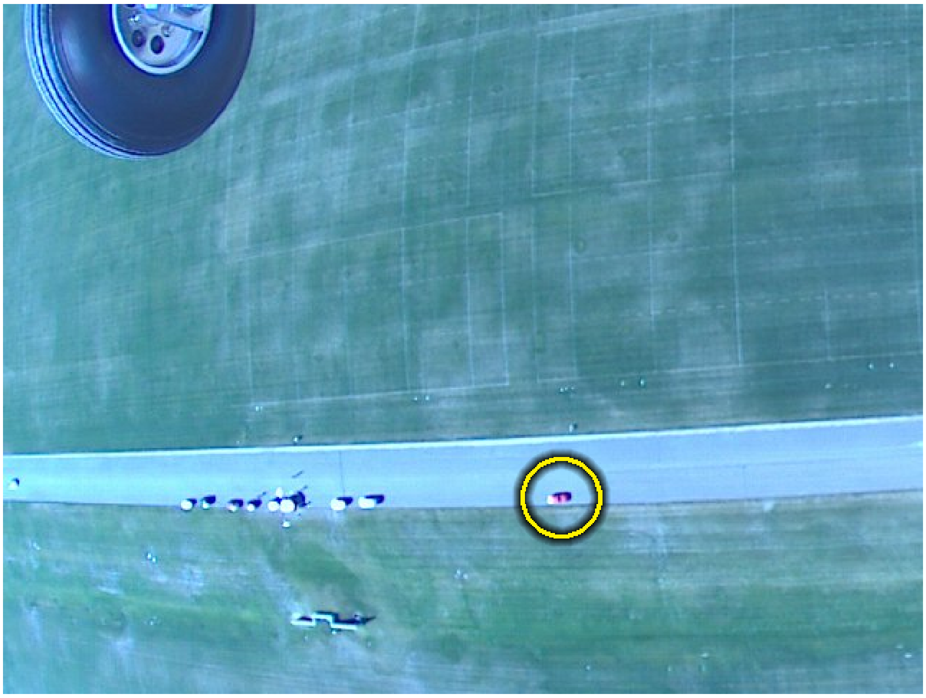


Fig. 9. Typical output of the airborne HOPS camera sensor (the target, a red car, is outlined with the yellow circle for clarity). After the target has been detected, the UAV will attempt to orbit around the estimated position of the moving target.

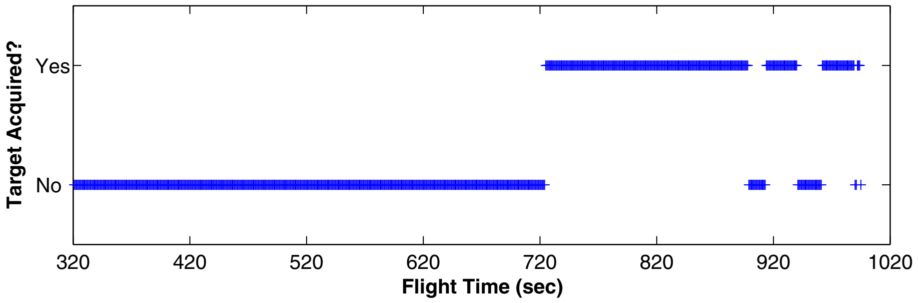


Fig. 10. Performance of the MUAV Sensor Network in localizing the target. During this flight, the target vehicle was identified approximately 12 min into the flight, and once localized, the UAV was able to maintain a lock on the target position for the majority of the remaining flight time.

Another very important performance metric for the ISR task is the UAVs' ability to locate the target. This includes finding the target through a given search process, and then maintaining an orbit around the moving vehicle. The

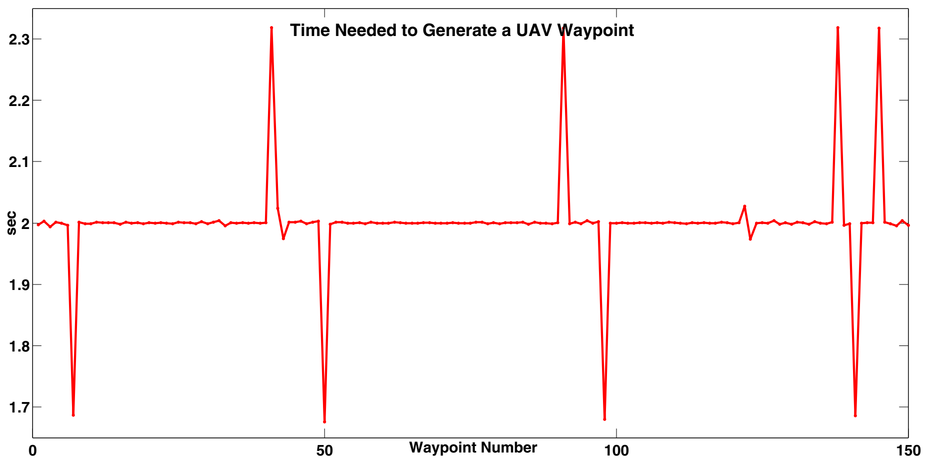


Fig. 11. The HOPS Control sub-system uses the output of the Sensor Network module to compute the next waypoint. For this flight, the ideal waypoint generation period is 2 sec, and the bound on the timing error was found to be ±0.3 sec, with the 8% total fault rate.

on-board image processing engine uses camera images, like the one shown in Figure 9, to identify features that could represent the target. As was discussed in Section 4.2, the output of both the RF and camera sensors is an estimated bearing to the target. This output from local on-board sensors is fused together inside the Sensor Fusion module with observations relayed by other UAVs to provide the HOPS Mission Control Manager (see Figure 4) with the estimate of the target's position in the global coordinate frame [3]. The stability of this estimate is shown in Figure 10, which contains a binary representation of the Sensor Fusion output (i.e., the target has been either localized or not). From the data in this plot, observe that it took the UAVs approximately 12 minutes to first detect and then localize a red vehicle, and that the UAVs maintained a lock on the target's position for almost the entire remainder of the mission. There were, however, a few isolated interruptions in the target localization status, which can be attributed to the periodic occlusions of the target by the aircraft's landing gear (see Figure 9), and an occasional wind gust, that blew a lightweight UAV away from its intended flight path, thus causing it to lose sight of the target.

Recall that the HOPS Control sub-system uses the output of the MUAV Sensor Network to control the UAV's flight path. Intuitively, issuing more frequent updates of the flight path, using the most recent sensor data, is likely to improve the speed and accuracy of the target localization task. Thus the ability of the HOPS software to support a fast waypoint update rate is important. Given the configuration of the avionics and aircraft actuator controls, the optimal waypoint update rate was calculated to be one new waypoint every two seconds. The plot in Figure 11 shows that the on-board Control Manager is able to maintain this level of performance well – the maximum deviation in the waypoint period never exceed 0.3 seconds, and deviations from the ideal 2 second period occurred only 8% of the time. Referencing the earlier analysis of the video frame rates (shown in Figure 8), a similar "catch-up" behavior in the waypoint timings is evident: a "lag" in the waypoint generation (denoted by upward peaks in Figure 11) is soon followed by a shorter waypoint generation period (represented by the downward peaks), meaning that the HOPS Control software strives to provide stable waypoint input to the autopilot in the face of a variable, event-driven load on the system.

6 Conclusions and Future Work

This chapter presented a complete implementation of an autonomous airborne sensor network tasked with locating ground targets. The event-driven software architecture enabled a transition from the previous work based on simulation and analytical experiments to a fully-functioning prototype, with real unmanned aerial vehicles performing search and localization of ground targets. Implementation of each software module as a stand-alone unit with a well-defined set of modification signals and an ability to react to changes in other parts of the system, made it possible to construct a scalable and robust framework for conducting collaborative intelligence gathering, surveillance, and reconnaissance missions. The scalability and versatility of this solution have been thoroughly verified by executing the ISR mission using autonomous ground vehicles prior to the testing on UAVs, and also through an evaluation of a variety of control strategies using real aircraft hardware and emulated sensor input provided by a sophisticated hardware-in-the-loop simulator engine. The ability to reuse these software components for other tasks is a definite advantage, and has already proved useful in the context of senior design projects being developed by US Air Force Academy cadets. In the end, the event-driven approach made it possible to meet the time requirements for the software deliverables and to satisfy the practical constraints of this project. The use of the sophisticated Qt programming framework is what enabled a small research and development team to write, test, and experimentally validate over 43,000 lines of cross-platform C, C++, and MATLAB code in less than six months.

This experience supports the conclusion that the event-driven software design approach, combined with the support of the Qt programming framework, is an extremely effective and practical way to design a distributed network of intelligent autonomous agents. Encapsulation of the decision algorithms inside isolated modules is a straightforward and effective approach to reducing system debugging time. Diligent and consistent separation of system functionality, along with various data sources, allows for frequent reuse of existing functional components, which is highly conducive to concurrent development of the software in a team-based environment.

However, in order to realize these advantages, one has to sacrifice some of the determinism in the system. It is possible that the lack of a linear execution algorithm for all of the sub-systems in this complex system may present a challenge when trying to analyze and document the system. Because of the practical restriction and consequent reliance on common off-the-shelf hardware and software tools, the system occasionally demonstrated sub-optimal performance in some of its multi-threaded components. In particular, the implementation proved ill-suited for precise time synchronization of parallel tasks, such as obtaining the exact location of the aircraft at the time a sensor reading is being taken, or enforcing strict bounds on communication delays. It is expected that some of these timing issues can be addressed through an improved infrastructure, such as the use of a real time operating system (RTOS). However, this design methodology does mandate strict isolation of all on-board software components, and this in turn prevents efficient simultaneous access to different hardware devices. So, in order to gain access to the needed information, some sub-system managers are forced to request the needed data from the other managers, and the extra time required to service such requests occasionally degrades performance of the system by increasing the uncertainty of a target position estimate. In practice, however, these small variations do not seem to have a significant impact. Therefore, it was determined that this kind of an error is an acceptable design trade off, consistent with the long-term goals and constraints of this project.

Another point that should be considered is the scalability of the event-driven approach. It is well known that concurrent execution of multi-threaded software requires a certain amount of computational overhead, such as the implementation and use of synchronization primitives, as well as an occasional duplication of shared data to allow for more efficient processing of large data sets. In addition, one should not ignore the overhead associated with the management of event queues and signal processing costs. While the extra time needed to service an event is typically much smaller than a standard memory allocation routine [9], whenever the system generates a very high number of events (e.g., more than a thousand timer events per second), this overhead becomes an important limiting factor for a distributed system. The cross-platform capability of the MUAV framework is extremely beneficial in reducing the development time; regretfully, it is also the cause of several inefficiencies that could otherwise be eliminated using a platform-specific implementation. But it is important to realize that these challenges are inherent in any large-scale distributed system. The event-driven paradigm remains a promising alternative to more monolithic, sequentially-oriented implementations. Both the laboratory and field experience with the MUAV framework has shown that it is ideal for supporting a large research project, especially when the final system specification may not be finalized until late in the development cycle. Due to its agile nature and its focus on creating independent, stand-alone functional modules without explicit interdependencies, the event-driven paradigm created a structured development environment in which all of the different modules could be integrated into a robust and efficient ISR application.

References

1. Pack, D., York, G.: Developing a control architecture for multiple unmanned aerial vehicles to search and localize rf time-varying mobile targets: part I. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 3965–3970 (April 2005)
2. Plett, G.L., De Lima, P., Pack, D.J.: Target localization using multiple uavs with sensor fusion via sigma-point kalman filtering. In: Proceedings of the 2007 AIAA (2007)
3. Plett, G.L., Zarzhitsky, D.V., Pack, D.J.: Out-of-order sigma-point kalman filtering for target localization using cooperating unmanned aerial vehicles. In: Hirsch, M.J., Pardalos, P.M., Murphey, R., Grundel, D. (eds.) Advances in Cooperative Control and Optimization. Lecture Notes in Control and Information Sciences, vol. 369, pp. 22–44 (2007)
4. Axis Communications. AXIS 212 PTZ Network Camera (accessed March 14, 2008), `http://www.axis.com/products/cam_212`
5. Cloud Cap Technology. Autopilots: Piccolo II, (accessed March 14, 2008), `http://cloudcaptech.com/piccolo_II.shtm`
6. Kontron. speedMOPSlcdPM (accessed March 14, 2008), `http://us.kontron.com/index.php?id=226\&cat=620\&productid=497`
7. MITRE MobileMesh. MITRE - Our Work - Technology Transfer Office - Downloadable Software - Mobile Mesh (accessed March 14, 2008), `http://www.mitre.org/work/tech_transfer/mobilemesh`
8. Opal Kelly. XEM3010 (accessed March 14, 2008), `http://opalkelly.com/products/xem3010`
9. Trolltech ASA. Qt: Cross Platform Rich Client Development Framework (accessed March 14, 2008), `http://trolltech.com/products/qt`

Robust Track Association and Fusion with Extended Feature Matching

Huimin Chen[1], Genshe Chen[2], Erik P. Blasch[3], and Tod M. Schuck[4]

[1] Dept. of Electrical Engineering, University of New Orleans, LA, USA
[2] DCM Research Resources, LLC, Germantown, MD, USA
[3] AFRL/SNAA, WPAFB, OH, USA
[4] Lockheed Martin, MS2, 199 Borton Landing Road, Moorestown, NJ, USA

Abstract. In this work, we propose a new data processing architecture as well as track association and fusion algorithms to improve target classification and tracking accuracy using distributed and, possibly, legacy-sensor platforms. We present a robust data fusion algorithm that can incorporate target classes/types at the fusion center when receiving sensor reports and/or local tracks. We aim to tackle the following technical challenges in feature aided tracking.

– **Unknown number of targets:** When the fusion center does not have any prior knowledge on the number of targets in the surveillance area, track fusion becomes extremely difficult especially when targets are closely spaced.
– **Measurement origin uncertainty:** The local tracker does not know which measurement comes from which target and each local tracker may provide false tracks or incorrect target types. Consequently, the fusion center does not know which local tracks are from the same target and fusion has to be made based on imperfect data association.
– **Tracks from legacy sensor systems:** Existing trackers often have very different filter designs. Some may be based on the state-of-the-art multiple model algorithm while some on the fixed gain Kalman filter. Thus some trackers can report both target state estimate and the associated covariance to the fusion center, but others may only provide target state estimate without the covariance information. Those legacy sensor systems require special treatment in the development of fusion algorithms.

Our track association framework can also incorporate tracks with extended feature points and kinematic constraints, which improves both data association and tracking accuracy.

1 Introduction

In multi-sensor multi-target tracking, each sensor can have its own target state estimate based on the local measurements. In order to form a global picture of the existing tracks, it is necessary to associate multiple local tracks and to obtain the global target state estimate [15]. Under this tracking configuration, the fusion center needs to associate and fuse local tracks on demand, which in general is less frequent than the measurement made at each local sensor [13]. The problem of track-to-track association — a prerequisite for the fusion of tracks — has been considered initially in the literature for tracks described by kinematic states [5]. Recently, it has been generalized to include additional (continuous valued) feature and (discrete valued) attribute variables which pertain to those tracks [8,10,47]. Examples of features are radar cross-section and target length. Examples of attributes are number of engines of an aircraft and type of emitter/waveform. Target classifications include platform (e.g., fighter, bomber, etc.), type (e.g., F-14, B-1, etc.), and class (e.g., F-14D, B-1B). A detailed discussion of target features, attributes and classification can be found in [16,17,18,19,20]. These approaches allow the search for the maximum likelihood (ML) or maximum a posteriori (MAP) association. Feature-aided track association was presented in [38,36,37,15]. A comprehensive procedure for incorporation of attributes and their possible dependence on the features was presented in [45,26,25] and shown to be amenable to obtain the MAP association of tracks from two sensors using linear programming. Feature aided tracking (FAT) has been studied using belief filter [9] and interacting multiple model approach [44]. Classification-aided tracking with measurement-to-track association via multidimensional assignment (MDA) was discussed in [4]. However, these approaches can not handle the case where the number of targets is unknown.

We will present the data processing architecture as well as track association and fusion algorithms for improved target classification and tracking accuracy using both kinematic and non-kinematic sensory data from distributed and, possibly, legacy-sensor platforms. In addition, our robust data fusion algorithm can incorporate target classes/types at the fusion center when receiving sensor reports and/or local tracks. Figure 1 shows the data processing flowchart of the track association and track fusion algorithms that can operate with different types of local trackers. Note that track fusion has to operate under imperfect track association where it is crucial to provide accurate assessment of the fused target state estimate.

We aim to address the following technical challenges in feature aided tracking. (i) Unknown number of targets: When the fusion center does not have any prior knowledge on the number of targets in the surveillance area, track fusion becomes extremely difficult especially when targets are closely spaced. (ii) Measurement origin uncertainty: The local tracker does not know which measurement comes from which target and each local tracker may provide false tracks or incorrect target types. Consequently, the fusion center does not know which local tracks are from the same target and fusion has to be made based on imperfect data association. It is generally believed that the data association is more challenging at the measurement level than at the track level. However, a systematic procedure to quantify the performance loss due to measurement origin uncertainty for feature aided track fusion is of great need. (iii) Tracks from legacy sensor systems: Existing trackers often have very different filter designs. Some may be

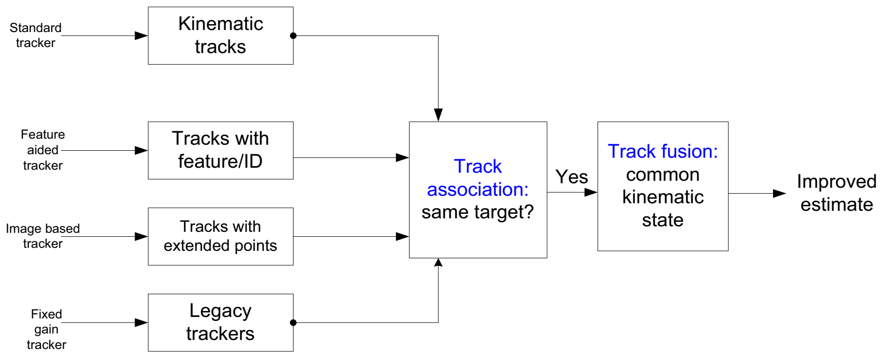


Fig. 1. Data processing flowchart of track association and fusion among different types of local tracks

based on state-of-the-art multiple model algorithms while others on the fixed gain Kalman filter. Thus some trackers can report both target state estimate and the associated covariance to the fusion center, but others may only provide target state estimate without the covariance information. Those legacy sensor systems require special treatment in the development of an estimation fusion algorithm [46].

The rest of the chapter is organized as follows. Section 2 formulates the track association problem of testing whether multiple tracks are from the same target using kinematic and feature/classification information. Section 3 presents the general track association problem and its efficient multiframe assignment solution. Section 4 extends the track association problem to the cases where extended feature points are available for each track. Junction matching via assignment and convex programming is presented to efficiently find the best feature correspondences. Section 5 presents the robust track fusion algorithm where the cross-covariance between local estimation errors can not be precisely known. Section 6 considers the track fusion problem where the legacy sensor systems can only provide the local tracks without error covariance information. Section 7 provides simulation study of the proposed track association and robust track fusion algorithms. Concluding summaries are given in Section 8.

2 Association of Multiple Tracks to a Single Target

Consider the problem of associating M local tracks from M local sensor systems. Each tracker provides the local target state estimate and possible target recognition or classification on the ID information. We are interested in testing whether these M local tracks are from the same target. To simplify the analysis, we first consider the case of testing two hypotheses, namely, all local tracks are from the same target vs. all local tracks are from different targets.

2.1 Track Association Using Multiple Local State Estimates

We first assume that only the target state information of M local trackers is available. Consider the hypothesis

H_0 : *all track estimates are from the same target.*
Traditional approaches reject H_0 only when the computed test statistic falls outside a predetermined confidence region, say an interval with 95% probability. However, there is no control of the misassociation probability since the alternative hypothesis is unspecified. Here we consider a typical alternative given by
H_1 : *all track estimates are from different targets.*
Clearly, it is a composite hypothesis depending on the target distribution in a surveillance region. We will use a representative hypothesis for H_1 which is simple and the likelihood function does not depend on the target state. Denote by x_i the true state of the i-th target. If we assume the surveillance region $\mathcal{V}$ is finite and its volume is V, without knowing the target density, we can assign uniform prior to each target state, i.e., $p(x_i) = 1/V$ for $i = 1, ..., M$. Note that perfect sensor registration is assumed at the fusion center. Otherwise, sensor bias has to be removed using the technique developed in [34].

Denote by $\hat{x}_i$ the local estimate from the i-th sensor. It is assumed that the estimation error is Gaussian with zero mean and covariance P_i. This is often the case when each local tracker uses a Kalman filter to recursively estimate the target state. Under hypothesis H_0 that M local state estimates are from the same target, i.e., $x_1 = x_2 = \cdots = x_M = x$, we have

$$\hat{\mathbf{x}} \overset{\Delta}{=} \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \\ \vdots \\ \hat{x}_M \end{bmatrix} = Hx + \mathbf{w} \overset{\Delta}{=} \begin{bmatrix} I \\ I \\ \vdots \\ I \end{bmatrix} x + \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_M \end{bmatrix} \tag{1}$$

where $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{P})$ and $\mathbf{P}$ is the estimation error covariance given by

$$\mathbf{P} = \begin{bmatrix} P_1 & P_{12} & \cdots & P_{1M} \\ P_{21} & P_2 & \cdots & P_{2M} \\ \vdots & & \ddots & \vdots \\ P_{M1} & P_{M2} & \cdots & P_M \end{bmatrix}. \tag{2}$$

The crosscovariance P_{ij} between the estimation errors from local trackers i and j is due to the common process noise used in the dynamic equation [5]. The true target state x is usually unknown. To simplify the analysis, we assume a Gaussian prior of the true target state, i.e., $x \sim \mathcal{N}(x_0, P_0)$. Note that the true state is correlated with the local estimation error. The covariance between x and the estimation error from local tracker i is obtained by

$$C_{xw_i} \overset{\Delta}{=} E[xw_i'] - E[x]E[w_i]' = E[(\hat{x}_i - w_i)w_i'] = -P_i \tag{3}$$

where the last equality follows the principle of orthogonality assuming the local estimates are optimal within the linear class [6]. Thus we have

$$C_{x\mathbf{w}} \overset{\Delta}{=} E[x\mathbf{w}'] - E[x]E[\mathbf{w}]' = [-P_1 \;\; -P_2 \;\; \cdots \;\; -P_M] \,. \tag{4}$$

The state prediction at the fusion center can also be used as the prior if the fusion center assumes a kinematic model to propagate the target state [32]. However, the prediction error is also correlated with the state estimation error of each local tracker. Finding the exact crosscovariance between the prediction error from the fusion center and the estimation error from a local tracker is a nontrivial issue. Hence throughout the chapter we assume that the fusion center only associates and fuses local track estimates without doing its own state prediction.

Note that the additive Gaussian noise assumption under H_0 implies a surveillance region of infinite volume which violates the assumption under H_1. However, a truncation of a normal density over $\mathcal{V}$ requires additional computation which makes the resulting likelihood function complicated. To simplify the analysis, we further assume that the volume of the surveillance region is large enough compared with the uncertainty on the target state so that the truncation probability is small and thus negligible. Note also that V^{-1}, while having a physical dimension in the likelihood function under H_1, is really a constant whose exact value will only scale the likelihood ratio. With the above assumptions, the likelihood ratio between the two hypotheses can be written as

$$\Lambda(\hat{\mathbf{x}}) = \frac{p(\hat{\mathbf{x}}|H_0)}{p(\hat{\mathbf{x}}|H_1)} = V^M \int_{\mathcal{V}} \mathcal{N}\left(\begin{bmatrix} \hat{\mathbf{x}} \\ x \end{bmatrix} ; \begin{bmatrix} Hx \\ x_0 \end{bmatrix} , \begin{bmatrix} \mathbf{P} & C'_{x\mathbf{w}} \\ C_{x\mathbf{w}} & P_0 \end{bmatrix} \right) dx. \tag{5}$$

If we assume that the integral is over the whole state space, then by completing the quadratic form, we have the log-likelihood ratio given by

$$\ln\Lambda(\hat{\mathbf{x}}) = -\frac{1}{2}(\hat{\mathbf{x}}'\mathbf{P}^{-1}\hat{\mathbf{x}} - \bar{x}'\bar{P}^{-1}\bar{x}) + c, \tag{6}$$

where

$$c = M\ln V - \frac{1}{2}x_0'P_0^{-1}x_0 - \frac{1}{2}\ln|2\pi\mathbf{P}| - \frac{1}{2}\ln|2\pi P_0| + \frac{1}{2}\ln|2\pi\bar{P}|, \tag{7}$$

$$\bar{x} = \mathbf{G}\hat{\mathbf{x}} + (I - \mathbf{G}H)x_0, \tag{8}$$

$$\bar{P} = P_0 - \mathbf{G}P_0\mathbf{G}', \tag{9}$$

$$\mathbf{G} = C_{x\hat{\mathbf{x}}}C_{\hat{\mathbf{x}}}^{-1}, \tag{10}$$

$$C_{x\hat{\mathbf{x}}} = [P_0 - P_1 \;\; P_0 - P_2 \;\; \cdots \;\; P_0 - P_M] \tag{11}$$

$$C_{\hat{\mathbf{x}}} = \begin{bmatrix} P_0 - P_1 & P_0 + P_{12} - P_1 - P_2 & \cdots & P_0 + P_{1M} - P_1 - P_M \\ P_0 + P_{21} - P_2 - P_1 & P_0 - P_2 & \cdots & P_0 + P_{2M} - P_2 - P_M \\ \vdots & & \ddots & \vdots \\ P_0 + P_{M1} - P_M - P_1 & P_0 + P_{M2} - P_M - P_2 & \cdots & P_0 - P_M \end{bmatrix}. \tag{12}$$

Note that if H_0 is accepted, then the fused target state estimate is (8) with the error covariance (9). Equations (8)–(9) are the linear minimum mean square error (LMMSE) estimator for the problem (1) [33]. The log-likelihood ratio test statistic is a quadratic function of $\hat{\mathbf{x}}$.

If prior the mean x_0 is unavailable, then we can not use (6) to test the two hypotheses. In this case, we assume that the target state has a noninformative prior, i.e., $p(x) = \frac{1}{V}$, $|V| \to \infty$, and thus $P_0^{-1} \to 0$. We further assume that the state and local estimation errors are uncorrelated, i.e., $C_{x\mathbf{w}} = \mathbf{0}$. With the above assumptions, the likelihood ratio between the two hypotheses can be written as

$$\Lambda^*(\hat{\mathbf{x}}) = \frac{p(\hat{\mathbf{x}}|H_0)}{p(\hat{\mathbf{x}}|H_1)} = V^{M-1}\int_{\mathcal{V}} \mathcal{N}(\hat{\mathbf{x}}; Hx, \mathbf{P})dx, \tag{13}$$

and by completing the quadratic form, the log-likelihood ratio becomes

$$\ln\Lambda^*(\hat{\mathbf{x}}) = -\frac{1}{2}\hat{\mathbf{x}}'[\mathbf{P}^{-1} - \mathbf{P}^{-1}H(H'\mathbf{P}^{-1}H)^{-1}H'\mathbf{P}^{-1}]\hat{\mathbf{x}} + c^*, \tag{14}$$

where

$$c^* = (M-1)\ln V - \frac{1}{2}\ln|2\pi\mathbf{P}| + \frac{1}{2}\ln|2\pi(H'\mathbf{P}^{-1}H)^{-1}|. \tag{15}$$

If H_0 is accepted, the fused state estimate is

$$\bar{x}^{\mathrm{LS}} = (H'\mathbf{P}^{-1}H)^{-1}H'\mathbf{P}^{-1}\hat{\mathbf{x}}, \tag{16}$$

with the covariance matrix given by

$$\bar{P}^{\mathrm{LS}} = (H'\mathbf{P}^{-1}H)^{-1}. \tag{17}$$

It is the weighted least squares solution to the problem

$$\min_x (\hat{\mathbf{x}} - Hx)'\mathbf{P}^{-1}(\hat{\mathbf{x}} - Hx). \tag{18}$$

The above fusion rule is also optimal in the maximum likelihood sense [12] and we will further discuss this in Section 5.

Another important point is that the log-likelihood ratio test statistic (13) can also be written as

$$\ln\Lambda^*(\hat{\mathbf{x}}) = -\frac{1}{2}(\hat{\mathbf{x}} - H\bar{x}^{\mathrm{LS}})'\mathbf{P}^{-1}(\hat{\mathbf{x}} - H\bar{x}^{\mathrm{LS}}) + c^*. \tag{19}$$

Thus the hypothesis testing becomes a goodness-of-fit term being compared with a threshold. Under H_0, the test statistic

$$T_{\mathbf{x}}(\hat{\mathbf{x}}) = (\hat{\mathbf{x}} - H\bar{x}^{\mathrm{LS}})'\mathbf{P}^{-1}(\hat{\mathbf{x}} - H\bar{x}^{\mathrm{LS}}) \tag{20}$$

has a chi-square distribution with $(M-1)n_x$ degrees of freedom where n_x is the dimension of target state x. Let $\mathbf{K}$ be an $(M-1)n_x \times Mn_x$ matrix with full row rank that satisfies $\mathbf{K}H = \mathbf{0}$. Denoting $\hat{\mathbf{y}} = \mathbf{K}\hat{\mathbf{x}}$, then, for example, we have the differences between the local track estimates given by

$$\hat{\mathbf{y}} \overset{\Delta}{=} \begin{bmatrix} \hat{x}_2 - \hat{x}_1 \\ \hat{x}_3 - \hat{x}_1 \\ \cdots \\ \hat{x}_M - \hat{x}_1 \end{bmatrix} = \mathbf{K}\hat{\mathbf{x}}, \tag{21}$$

when $\mathbf{K}$ is chosen as

$$\mathbf{K} = \begin{bmatrix} -I \; I \; 0 \cdots 0 \\ -I \; 0 \; I \cdots 0 \\ \vdots \qquad \cdots \vdots \\ -I \; 0 \; 0 \cdots I \end{bmatrix}. \tag{22}$$

Under H_1 we have $\hat{\mathbf{y}} \sim \mathcal{N}(\mathbf{0}, \mathbf{P}_{\hat{\mathbf{y}}})$ where

$$\mathbf{P}_{\hat{\mathbf{y}}} = \begin{bmatrix} P_1 + P_2 - P_{21} - P_{12} & P_1 + P_{23} - P_{21} - P_{13} & \cdots & P_1 + P_{2M} - P_{21} - P_{1M} \\ P_1 + P_{32} - P_{31} - P_{12} & P_1 + P_3 - P_{31} - P_{13} & \cdots & P_1 + P_{M3} - P_{31} - P_{1M} \\ \cdots & & \ddots & \cdots \\ P_1 + P_{M2} - P_{M1} - P_{12} & P_1 + P_{M3} - P_{M1} - P_{13} & \cdots & P_1 + P_M - P_{M1} - P_{1M} \end{bmatrix}. \tag{23}$$

The test statistic $T_{\mathbf{y}}(\hat{\mathbf{y}}) = \hat{\mathbf{y}}'(\mathbf{P}_{\hat{\mathbf{y}}})^{-1}\hat{\mathbf{y}}$ is also chi-square distributed with $(M-1)n_x$ degrees of freedom which was also rigorously derived in [3]. Next, we will show that all tests based on $\hat{\mathbf{y}}$ from such a linear transform are equivalent no matter which local track estimate is used to compute the differences.

Theorem: For any $(M-1)n_x \times Mn_x$ matrix $\mathbf{K}$ with full row rank that satisfies $\mathbf{K}H = \mathbf{0}$, the test statistic $T_{\mathbf{y}}(\hat{\mathbf{y}})$ is chi-square distributed with $(M-1)n_x$ degrees of freedom and $T_{\mathbf{y}}(\hat{\mathbf{y}}) = T_{\mathbf{x}}(\hat{\mathbf{x}})$.

Proof: By definition we have

$$T_{\mathbf{y}}(\hat{\mathbf{y}}) = \hat{\mathbf{y}}'(\mathbf{P}_{\hat{\mathbf{y}}})^{-1}\hat{\mathbf{y}} = \hat{\mathbf{x}}'\mathbf{K}'(\mathbf{K}\mathbf{P}\mathbf{K}')^{-1}\mathbf{K}\hat{\mathbf{x}}. \tag{24}$$

Denote by

$$\mathbf{Q} \overset{\Delta}{=} \mathbf{K}'(\mathbf{K}\mathbf{P}\mathbf{K}')^{-1}\mathbf{K}, \tag{25}$$

and

$$\mathbf{R} \overset{\Delta}{=} \mathbf{P}^{-1} - \mathbf{P}^{-1}H(H\mathbf{P}^{-1}H')^{-1}H'\mathbf{P}^{-1}. \tag{26}$$

It is easy to verify that

$$\mathbf{Q}H = \mathbf{R}H = \mathbf{0}, \tag{27}$$

and

$$\mathbf{K}\mathbf{P}\mathbf{Q} = \mathbf{K}\mathbf{P}\mathbf{R} = \mathbf{K}. \tag{28}$$

Thus

$$\mathbf{K}\mathbf{P}(\mathbf{Q} - \mathbf{R})\mathbf{P}\mathbf{K}' = \mathbf{0}. \tag{29}$$

Since $\mathbf{KP}$, $\mathbf{Q}$ and $\mathbf{R}$ have the same rank and both $\mathbf{Q}$ and $\mathbf{R}$ are in the null space of H, we have $\mathbf{Q} = \mathbf{R}$ and thus

$$T_{\mathbf{x}}(\hat{\mathbf{x}}) = \hat{\mathbf{x}}'\mathbf{R}\hat{\mathbf{x}} = \hat{\mathbf{x}}'\mathbf{Q}\hat{\mathbf{x}} = T_{\mathbf{y}}(\hat{\mathbf{y}}). \qquad \text{QED}$$

The above result tells us that the association of multiple local tracks amounts to a chi-square test where the uniform prior is assumed for the target state in the Bayesian formulation. The association of two local tracks in [5] is a special case of the above general test. When the noninformative prior assumption is violated, the popularly used chi-square test ([14]) will no longer be Bayesian optimal.

2.2 Track Association with Classification Information

This subsection discusses the track association of a single target with classification information which is assumed to be independent of the target state estimates. The target classification information, e.g., target identity, if available from the local tracker, should be used for the hypothesis testing to determine whether M local estimates are from the same target. It does not help in track fusion once a decision is made. We assume that each local tracker provides the probabilities of a target belonging to a finite number of classes which are independent of the target state estimate. Let L be the total number of classes and $\mathbf{p}_i = [p_{i1} \ ... \ p_{iL}]^T$ be the output of the classifier from the i-th local tracker. The output of the classifier is a vector of the posterior probabilities, which is described as the classification vector [43] — the probability that the target belongs to a particular class conditioned on all measurements up to the current time.

Given M local classification outputs $\mathbf{p}_1, ..., \mathbf{p}_M$, the two hypotheses being considered for possible track association are H_0 : *all track estimates are from the same target* and H_1 : *all track estimates are from different targets*. We are interested in obtaining the joint likelihood functions $\Lambda(\mathbf{p}_1, ..., \mathbf{p}_M|H_i)$ $(i = 0, 1)$ or, in this case, the probabilities $P(\mathbf{p}_1, ..., \mathbf{p}_M|H_i)$ $(i = 0, 1)$ which requires the knowledge on the dependency among local classifiers. Characterizing the dependency among local classifiers is very challenging especially when the classification algorithms at local sensors are different. Next, we highlight the solution in the simplest case and leave the general situation regarding the fusion of classifier outputs for future research. In the following it is assumed that (a) the classification errors are independent among the local classifiers and (b) the local estimates are from the same target only when they belong to the same class. With the above assumptions and equal prior probabilities for the two hypotheses, we have

$$P(\mathbf{p}_1, ..., \mathbf{p}_M|H_0) = \sum_{j=1}^{L} \prod_{i=1}^{M} p_{ij}, \tag{30}$$

$$P(\mathbf{p}_1, ..., \mathbf{p}_M|H_1) = 1 - P(\mathbf{p}_1, ..., \mathbf{p}_M|H_0). \tag{31}$$

An example of two-classifier and two-class case is given in Table 1 under the simplified assumptions. It is clear that accurate local classifiers can improve the discrimination capability between the two hypotheses.

Table 1. Common and different origin probabilities based on local classifiers

	Case 1	Case 2	Case 3	Case 4
$\mathbf{p}_1$	[0.5 0.5]	[0.8 0.2]	[0.2 0.8]	[0.9 0.1]
$\mathbf{p}_2$	[0.2 0.8]	[0.2 0.8]	[0.9 0.1]	[0.9 0.1]
$P(\mathbf{p}_1, \mathbf{p}_2 \| H_0)$	0.5	0.32	0.26	0.82
$P(\mathbf{p}_1, \mathbf{p}_2 \| H_1)$	0.5	0.68	0.74	0.18

Under the conditional independence assumption between the state and classification output, the log-likelihood ratio can be expressed as

$$\begin{aligned} \ln\Lambda^*(\hat{\mathbf{x}}, \mathbf{p}_1, ..., \mathbf{p}_M) &= \ln\left[\frac{p(\hat{\mathbf{x}}|H_0)}{p(\hat{\mathbf{x}}|H_1)} \frac{P(\mathbf{p}_1, ..., \mathbf{p} \quad |H_0)}{P(\mathbf{p}_1, ..., \mathbf{p} \quad |H_1)}\right] \\ &= -\frac{1}{2}\hat{\mathbf{x}}'[\mathbf{P}^{-1} - \mathbf{P}^{-1}H(H'\mathbf{P}^{-1}H)^{-1}H'\mathbf{P}^{-1}]\hat{\mathbf{x}} \\ &\quad +\ln\frac{P(\mathbf{p}_1, ..., \mathbf{p} \quad |H_0)}{P(\mathbf{p}_1, ..., \mathbf{p} \quad |H_1)} + c^* \end{aligned} \tag{32}$$

where, with the uniform prior of the target state, the constant term c^* is given in (15). Since $P(\mathbf{p}_1, ..., \mathbf{p}_M|H_0)$ and $P(\mathbf{p}_1, ..., \mathbf{p}_M|H_1)$ are in general data dependent, the test statistic is no longer chi-square distributed.

Remark 1: The classification results from each local classifier can be considered as "priors" in the test (32) without the target state information and would replace any existing priors or naive priors as classification information is received. The derivation of (30)–(31) relies on the deterministic relation between target class and its origin. It can be relaxed provided the fusion center has the knowledge of the confusion matrix $C^i = [c^i_{nm}]$ of local classifiers i where

$$c^i_{nm} \overset{\Delta}{=} P(\text{declare class } m|\text{true class is } n) \qquad n = 1, \ldots, L, \;\; m = 1, \ldots, L \tag{33}$$

are the elements of C^i $(i = 1, ..., M)$.

2.3 A Numerical Example

We want to study the effectiveness of the track association test for a different number of local sensors with various cross correlation coefficients. To make it simple, we assume that the local estimates are scalars with unknown means and unit variances. The cross correlation coefficients between two local estimates is denoted by ρ. We choose various values of ρ, namely, 0, 0.1, 0.3, 0.5, when the local tracks correspond to the same target. The hypothesis H_0 is that all local estimates correspond to the same target with its location uniformly distributed within the surveillance region of length $V = 10$. The hypothesis H_1 is that all local estimates correspond to different targets with their locations uniformly distributed within the surveillance region. In this case, the separation of two targets is random and it depends on the volume of the surveillance region. No prior knowledge is assumed in this case. The test based on (24) is used to compute the receiver operating characteristic (ROC) curves [31] for the cases of testing whether N local tracks correspond to the same target. In a ROC curve, the

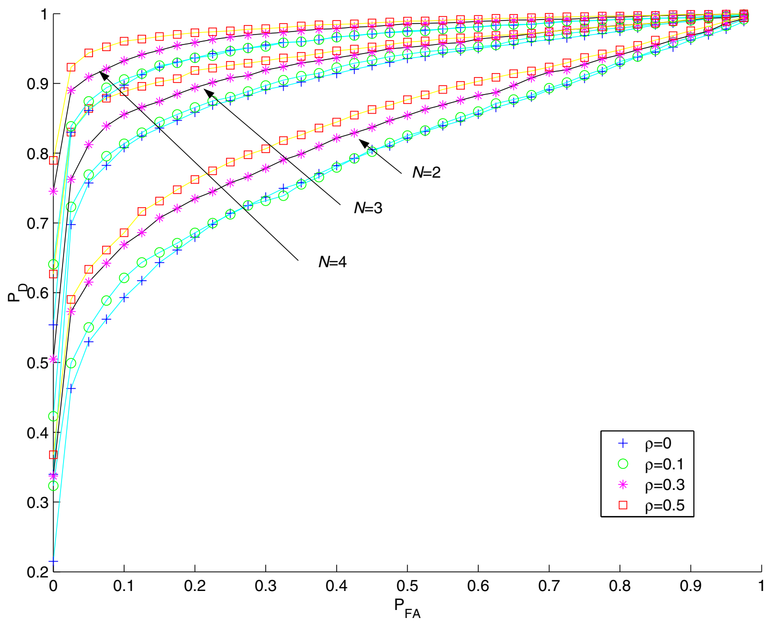


Fig. 2. ROC curves for the track association test with a different number of local estimates and various cross correlation coefficients

probability of false alarm P_{FA} refers to the probability of declaring H_1 when H_0 is true while the probability of detection P_D refers to the probability of declaring H_1 when H_1 is true. Figure 2 shows the ROC curves for the track association test with 2, 3, and 4 local track estimates and various cross correlation coefficients. 1000 random realizations are used for each hypothesis with fixed ρ and N to compute these curves. We can see that the test power increases as N increases for fixed V since the H_0 hypothesis becomes more distinguishable when more targets are uniformly distributed within the surveillance region. The cross correlation between the local track estimates is beneficial in terms of the test power under a given false alarm rate for all cases. As ρ increases, the alternative hypothesis becomes more distinguishable which improves the decision accuracy. However, once H_0 is declared, the variance of the fused track estimate becomes larger. We defer the simulation study using realistic tracking scenarios to Section 7.

3 General Track Association Via Multidimensional Assignment

We want to consider the association of local tracks to more than one target. The problem can be formulated as finding the best hypothesis regarding the track-to-target correspondence which has been studied extensively in the framework of multiple hypothesis tracking (MHT) [7]. A promising approach to solve the MHT problem is to treat it as a multidimensional assignment problem [2]. Here we first present the assignment formulation for track-to-track association from two sensors and then extend the formulation to the multiple sensor case. Assume sensor 1 has a list of N_1 tracks and sensor 2 has a list of N_2 tracks. Tracks from each local sensor are considered as coming from different targets. One wants to find all common origin tracks among the local sensors. Define the binary assignment variable χ_{ij} as

$$\chi_{ij} = \begin{cases} 1 \text{ track } i \text{ from sensor 1 and track } j \text{ from sensor 2 are from the same target,} \\ 0 \text{ otherwise.} \end{cases} \tag{34}$$

Denote by λ_{ij} the likelihood ratio of track i from sensor 1 and track j from sensor 2 being from the same target vs. from different targets which is defined as in (5) with M=2. Note that one can not directly compare two likelihood functions with different dimensions. In order to formulate the joint hypothesis for tracks corresponding to multiple targets, the log-likelihood ratio is used in the assignment cost function, which is a dimensionless quantity. If we assume that the track association events among different track pairs are independent, then the 2-D assignment formulation finds the most likely (joint) track-to-target association hypothesis by solving the following constrained optimization problem.

$$\min_{\chi} \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} c_{ij}\chi_{ij} \tag{35}$$

subject to

$$\sum_{i=1}^{N_1} \chi_{ij} \leq 1, \quad j = 1, ..., N_2 \tag{36}$$

i.e., each local track from sensor 2 can be associated with at most one target;

$$\sum_{j=1}^{N_2} \chi_{ij} \leq 1, \quad i = 1, ..., N_1 \tag{37}$$

i.e., each local track from sensor 1 can be associated with at most one target;

$$\chi_{ij} \in \{0, 1\}, \quad i = 1, ..., N_1, \quad j = 1, ..., N_2 \tag{38}$$

i.e., the association is binary; the cost is

$$c_{ij} = -\ln\lambda_{ij}. \tag{39}$$

As shown in [14] this problem can be solved optimally using linear programming by relaxing the integer constraint. Efficient algorithms such as the modified auction algorithm can also be applied to the above 2-D assignment problem [30,2].

When local tracks from multiple sensors need to be associated with multiple targets, a multidimensional assignment formulation is a natural extension of the 2-D assignment case. Assume there are S local sensors ($S \geq 3$) where sensor S_i

provides a list of N_i tracks to the fusion center. Define the binary assignment variable $\chi_{i_1 i_2 \dots i}$ as

$$\chi_{i_1 i_2 \dots i} = \begin{cases} 1 \text{ tracks } i_1, i_2, ..., i_S \text{ are from the same target,} \\ 0 \text{ otherwise.} \end{cases} \tag{40}$$

We allow a subset of the set of indices $\{i_1, i_2, ..., i_S\}$ to be zero in the assignment variable meaning that no local track from a particular list corresponds the target being considered. Denote by $\lambda_{i_1 i_2 \dots i}$ the likelihood ratio of the local tracks i_1, i_2, ..., i_S being from the same target vs. from different targets. If we assume that there are M nonzero indices corresponding to the local tracks from M sensors, then $\lambda_{i_1 i_2 \dots i}$ can be computed using (5). The S-D assignment finds the most likely hypothesis by solving the following integer program.

$$\min_{\chi_{1\ 2 \cdots}} \sum_{i_1=1}^{N_1} \sum_{i_2=1}^{N_2} \cdots \sum_{i\ =1}^{N} c_{i_1 i_2 \dots i}\ \chi_{i_1 i_2 \dots i} \tag{41}$$

subject to

$$\begin{array}{c} \sum_{i_2=0}^{N_2} \cdots \sum_{i\ =0}^{N} \chi_{j i_2 \dots i} \leq 1, \quad j = 1, ..., N_1 \\ \sum_{i_1=0}^{N_1} \sum_{i_3=0}^{N_3} \cdots \sum_{i\ =0}^{N} \chi_{i_1 j i_3 \dots i} \leq 1, \quad j = 1, ..., N_2 \\ \vdots \\ \sum_{i_1=0}^{N_1} \cdots \sum_{i\ -1=0}^{N\ -1} \chi_{i_1 i_2 \dots i\ -1 j} \leq 1, \quad j = 1, ..., N_S \end{array} \tag{42}$$

i.e., each local track can be assigned to at most one target;

$$\chi_{i_1 i_2 \dots i} \in \{0, 1\}, i_1 = 0, 1, ..., N_1, i_2 = 0, 1, ..., N_2, \cdots, i_S = 0, 1, ..., N_S \tag{43}$$

i.e., the track-to-target association is binary. In (41) the assignment cost is

$$c_{i_1 i_2 \dots i} = -\ln \lambda_{i_1 i_2 \dots i} \tag{44}$$

if there are at least two nonzero indices within the set $\{i_1, i_2, ..., i_S\}$. Otherwise, we let $c_{i_1 i_2 \dots i} = 0$. The above integer program is in general NP-hard [27] for $S \geq 3$. However, efficient algorithms exist to find a suboptimal solution via Lagrangian relaxation (see, e.g., [2,14]).

The solution of the above S-D assignment problem is the most likely joint hypothesis and it can be used to fuse the local track estimates being declared as from the same target. While this hypothesis is the most likely one, if the second most likely joint hypothesis has a cost close to the first one, the use of the best solution alone is potentially misleading. In such a case the next-to-best hypothesis should also be considered. In general, an algorithm that finds the top M solutions to the S-D assignment problem is needed for a composite display of the track estimates [1]. A practical approach to obtain the M-best solutions to the S-D assignment problem can be found in [41]. Its computational load is considerably heavier than finding only the single best solution of the

S-D assignment problem. Since this approach depends on solving a set of S-D assignment subproblems, it yields suboptimal solutions for $S \geq 3$. The number of hypotheses being considered depends on the ratio of the total cost of the M-th most likely and the most likely one. In practice, the hypotheses with individual cost less than 1% of the most likely hypothesis are dropped from the list of the solutions, although in some application domains these may want to be maintained [42,30].

Next, we present a way to estimate the posterior probability of each hypothesis from the top M most likely solutions. Denote by $\{\Theta^l\}_{l=1}^M$ the M-best (top-M) hypotheses corresponding to the solutions of the S-D assignment problem (41) and by $\{C^l\}_{l=1}^M$ the costs associated with these solutions, i.e.,

$$C^l = \sum_{i_1=0}^{N_1} \sum_{i_2=0}^{N_2} \cdots \sum_{i\ \ =0}^{N} c_{i_1 i_2 \ldots i} \ \ \chi^l_{i_1 i_2 \ldots i} \ \ , \tag{45}$$

where $\chi^l_{i_1 i_2 \ldots i}$ is determined by the joint hypothesis Θ^l. Assuming equal priors and the true track-to-target correspondence Θ belongs to one of the M-best hypotheses, then the probability that Θ^l is the true sequences of the joint track-to-target associations is

$$P(\Theta^l | \Theta \in \{C^l\}_{l=1}^M) = \frac{e^{-C}}{\sum_{n=1}^M e^{-C}} . \tag{46}$$

Note that those joint hypotheses with small probabilities are ignored due to the exponential growth of the total number of feasible associations.

3.1 A Numerical Example (Continued)

We continue using the example presented in Section 2.3 to evaluate the track association accuracy for the multiple target case. We assume that the local tracks are scalars with unknown means and unit variances. The correlation coefficients between two local track estimates is 0.1 when the local tracks correspond to the same target. When the local tracks correspond to different targets, the true target locations are uniformly distributed within the surveillance region $(-5, 5)$. We consider the association of two local tracks from each sensor and evaluate the probability of finding the correct hypothesis using the assignment algorithm. The hypotheses being considered are

H_1 : Each sensor has two tracks from the same two targets with their locations at -3 and 2, respectively.
H_2 : Each sensor has one track from the same target with its location at -3 and the other track from different targets.
H_3 : Each sensor has two tracks and all tracks correspond to different targets.

For each hypothesis with S local sensors, we generate 5000 random scenarios and estimate the probability of making a correct decision as shown in Table 2. We can see the performance degradation of hypotheses H_1 and H_3 compared with

Table 2. Probability of declaring the correct hypothesis for two target association problem

S	$P(``H_1"\|H_1)$	$P(``H_2"\|H_2)$	$P(``H_3"\|H_3)$
2	0.42	0.38	0.74
3	0.66	0.32	0.82
4	0.79	0.28	0.88

the ROC curves from a single target test in Figure 2. As the number of sensors increases, the performance improves for hypotheses H_1 and H_3 but degrades for hypothesis H_2. Note that H_2 is more difficult to declare than the other two hypotheses due to the the limited separation among different targets. The issue becomes more critical as the number of sensors increases. For the scenario being considered here, a single best solution is misleading since the probability of its correctness is fairly low especially under H_2.

4 Improved Track Association Via Extended Junction Matching

In principle, matching the observed feature points to the predicted ones can be done in a "pointwise" manner by using the well established data association techniques in the multiple target tracking of "point" targets, e.g., joint probability data association (JPDA), multiple hypothesis tracking (MHT) [7], or multiframe assignment algorithm [40]. However, when extended feature points are available for each track such as those from image based trackers, it is highly beneficial to solve the data association problem using junctions rather than individual feature points since by doing global pointwise association, the key information carried through the topology of a junction is largely ignored. In addition, the size of the association problem can be significantly reduced at the junction level compared with the pointwise assignment. In the following, we formulate the junction association algorithm by employing two efficient optimization frameworks, namely, assignment and convex programming.

4.1 Measuring Similarity between Junctions

Junction association requires a well defined similarity measure between two junctions (e.g., from a predicted and observed frames, respectively). One needs to construct a meaningful and appropriate *measure of closeness* between two junctions which utilizes the information carried through the topology and geometry to the full extent.

In computer vision, the closeness between two features, being characterized here as two sets of points, is usually measured by the Hausdorff set distance or its generalizations. Numerous similarity metrics based on Hausdorff distance have been proposed to handle outliers, occlusions, or other uncertainties arising in image matching [28], [21]. For two given junctions $\bar{J} = \{\bar{\eta}_0; \bar{\eta}_1, \ldots, \bar{\eta}_{\bar{n}}\}$ and $J = \{\eta_0; \eta_1, \ldots, \eta_n\}$ the Hausdorff distance $H(\bar{J}, J)$ would be defined as

$$H(\bar{J}, J) = \max(h(\bar{J}, J), h(J, \bar{J})) \tag{47}$$

where

$$h(\bar{J}, J) = \max_{0 \leq \bar{\imath} \leq \bar{n}} \min_{0 \leq i \leq n} ||\bar{\eta}_{\bar{\imath}} - \eta_i|| \tag{48}$$

and $|| \cdot ||$ is some vector norm. It measures the degree of mismatch between the *point-sets* $\bar{J}$ and J by the distance of the point in $\bar{J}$ that is farthest away from any point in J and vice versa. This implies that every point of J is within a distance $H(\bar{J}, J)$ from some point of $\bar{J}$ and vice versa, but no explicit one-to-one matching is obtained even if $\bar{n} = n$, e.g., many points of J can be close to the same point of $\bar{J}$. In addition, the Hausdorff distance also ignores important information carried in junction topology (i.e., the connections between the center location and the end-points) and junction geometry (e.g., the orientation of the junction segments). Apparently, direct usage of the Hausdorff distance is not the best choice for measuring the closeness of junctions.

As a measure of the closeness between two junctions $\bar{J}$ and J we define the following distance metric

$$d(\bar{J}, J) = \|\bar{\eta}_0 - \eta_0\| + \begin{cases} \frac{1}{\bar{n}} \min\limits_{\{j_1, \ldots, j_{\bar{}}\} \subseteq \{1,2,\ldots,n\}} \sum\limits_{i=1}^{\bar{n}} \phi_{ij} \; \|\bar{\eta}_i - \eta_j \| \text{ if} \bar{n} \leq n \\ \\ \frac{1}{n} \min\limits_{\{i_1, \ldots, i \;\} \subset \{1,2,\ldots,\bar{n}\}} \sum\limits_{j=1}^{n} \phi_{i \;\, i} \left\|\bar{\eta}_i \; - \eta_j\right\| \text{ if} \bar{n} > n \end{cases} \tag{49}$$

where $|| \cdot ||$ is a vector norm, and $\phi = \phi(\angle(\bar{\eta} - \bar{\eta}_0, \eta - \eta_0))$ is a weighting factor.

The proposed measure is tailored to the junction structure and inherently based on the idea of one-to-one point-wise assignment, and accounts for both the junction topology (links between center and end-points) and junction geometry (orientation of the junction segments). Next we discuss some details and provide a justification of the proposed junction closeness measure (49).

The vector norm $|| \cdot ||$ can be the Euclidean 2-norm. However, since the prediction and observation are subject to random errors, a better alternative is to measure the *statistical* distance between two points (vectors) $\bar{\eta}$ and η. If the prediction and observation covariances, $S_{\bar{\eta}}$ and S_{η}, respectively, are available, we use the Mahalanobis distance [35]

$$\|\bar{\eta} - \eta\| = (\bar{\eta} - \eta)' S_{\bar{\eta}\eta}^{-1} (\bar{\eta} - \eta) \tag{50}$$

where $S_{\bar{\eta}\eta} = S_{\bar{\eta}} + S_{\eta}$. Typically, as a result of the junction detection [29] the observation covariance S_{η} of an end-point η is much “larger” than that of the center-point η_0 (i.e., S_{η_0}). Using the statistical norm takes this effect into account by giving more “trust” to the distance between center points.

The weighting factor $\phi = \phi(\angle(\bar{\eta} - \bar{\eta}_0, \eta - \eta_0))$ of the distance $\|\bar{\eta} - \eta\|$ between two endpoints $\bar{\eta}$ and η is a design parameter. It is introduced to account for the orientation of junction segments $[\bar{\eta}_0, \bar{\eta}]$ and $[\eta_0, \eta]$ and is a function of the angle $\alpha = \angle(\bar{\eta} - \bar{\eta}_0, \eta - \eta_0)$ between vectors $\bar{\eta} - \bar{\eta}_0$ and $\eta - \eta_0$. The idea is to monotonically increase the point distance $\|\bar{\eta} - \eta\|$ as α increases and vise versa. This provides a highly desirable property of the modified distance $\phi|| \cdot ||$ – once the center η_0 of J is assigned to the center $\bar{\eta}_0$ of $\bar{J}$ (see (49)) then, the smaller the angular deviation α of a junction segment $[\eta_0, \eta]$ of J from $[\bar{\eta}_0, \bar{\eta}]$ is, the closer $[\eta_0, \eta]$ to $[\bar{\eta}_0, \bar{\eta}]$ is. In other words, matchings $\eta \longleftrightarrow \bar{\eta}$ with large angular deviations α are penalized by assigning a large distance $\phi(\alpha)||\bar{\eta} - \eta||$. If two points η' and η'' of J are at about the same distance from $\bar{\eta}$, i.e., $\|\bar{\eta} - \eta'\| \approx \|\bar{\eta} - \eta''\|$, then η' will be considered to be closer to $\bar{\eta}$ than η'' if and only if it is directionally closer to $\bar{\eta}$, i.e., if $\alpha' = \angle(\bar{\eta} - \bar{\eta}_0, \eta' - \eta_0) < \alpha'' = \angle(\bar{\eta} - \bar{\eta}_0, \eta'' - \eta_0)$. A linear penalty function in $|\alpha|$ with the desired properties has the form

$$\phi(\alpha) = c + b\frac{|\alpha|}{\pi}, \quad \alpha \in [-\pi, \pi] \tag{51}$$

where $b > 0$ and $c > 0$ are parameter to be specified by the designer. For example if $c = b = 1$ then $\phi(0)\, \|\bar{\eta} - \eta\| = \|\bar{\eta} - \eta\|$, $\phi(\pm\frac{\pi}{2})\, \|\bar{\eta} - \eta\| = 1.5\, \|\bar{\eta} - \eta\|$, and $\phi(\pm\pi)\, \|\bar{\eta} - \eta\| = 2\, \|\bar{\eta} - \eta\|$. Computing α explicitly could be computationally burdensome in some applications and can be avoided by using the easily computable $\cos(\alpha)$ instead. Since $\frac{|\alpha|}{\pi} \approx \frac{1}{2}\,[1 - \cos(\alpha)]$ another definition of weighting function which has a behavior quite similar to that of (51) is

$$\phi(\alpha) = c + \frac{b}{2}\,[1 - \cos(\alpha)]\,, \tag{52}$$

with

$$\cos(\alpha) = \frac{(\bar{\eta} - \bar{\eta}_0) \cdot (\eta' - \eta_0)}{||\bar{\eta} - \bar{\eta}_0||\; ||\eta' - \eta_0||},$$

where $\cdot$ is the dot product and $|| \cdot ||$ is the Euclidean norm.

The min-operation in (49) amounts to determination of the best *pointwise* assignment, with respect to the weighted Mahalanobis distances $\phi|| \cdot ||$ [35] used as the assignment cost between the set of end-points of $\bar{J}$ (or J) and a subset of end-points of J (or $\bar{J}$) with the same cardinality. Thus, the computation of the distance (49) is itself equivalent to solving a 2-D point assignment problem. Note that in practical applications each junction usually has a limited number of end-points (say a maximum of 5) and the min-operation in (49) can be feasibly computed by brute-force enumeration.

Note that the normalization factor $1/\bar{n}$ (resp. $1/n$) in (49) indicates the average (arithmetic mean) weighted distance $\phi|| \cdot ||$ of the optimal point assignment being used to form the distance $d(\bar{J}, J)$ instead of the sum of all weighted distances. This makes $d(\bar{J}, J)$ invariant of the rank (number of endpoints) of J. Otherwise, a junction with lower rank may have smaller distance in junction matching as compared to a junction with higher ranks even if the latter has better pointwise match than the former.

4.2 Junction Association Via 2-D Assignment

We first consider the junction association as the matching problem between two frames. After the junction detection module the observation of an object is represented in the two dimensional measurement space (camera frame) as a set of junctions with $J^{(j)} = \{\eta_0^{(j)}; \eta_1^{(j)}, \ldots, \eta_n^{(j)}\}$, $j = 0, 1, \ldots, M$ where $\eta_0^{(j)}$ and $\eta_1^{(j)}, \ldots, \eta_n^{(j)}$ denote respectively the *location* and *end-points* of the j-th junction. It is assumed that the state estimation module provides a set of *predicted* junctions with $\bar{J}^{(i)} = \{\bar{\eta}_0^{(i)}; \bar{\eta}_1^{(i)}, \ldots, \bar{\eta}_{\bar{n}}^{(i)}\}$, $i = 0, 1, \ldots, \bar{M}$ based on the state estimate at the previous time, the target motion model, and possibly occlusion and appearance models. The association of the observed and predicted junction sets is described as the one-to-one mapping through the assignment variables

$$\chi_{ij} = \begin{cases} 1 \text{ if} J^{(j)} \longleftrightarrow \bar{J}^{(i)}, \\ 0 \text{ otherwise}, \end{cases} \tag{53}$$

for $j = 0, 1, \ldots, M,\ i = 0, 1, \ldots, \bar{M}$. Note that χ_{0j} means that the observed junction $J^{(j)}$ is not associated with any of the predicted junctions $\bar{J}^{(i)}$, $i = 1, \ldots, \bar{M}$ (e.g., it is either a newborn junction or a false detection) and χ_{i0} means that the predicted junctions $\bar{J}^{(i)}$ is not associated with any of the observed junctions $J^{(j)}$, $j = 1, \ldots, M$ (as a result of occlusion or missed detection in the current frame).

Let c_{ij} be the cost of the association between $J^{(j)}$ and $\bar{J}^{(i)}$, $j = 0, 1, \ldots, M$, $i = 0, 1, \ldots, \bar{M}$ then the junction association problem can be cast into the following two dimensional (2-D) assignment problem

$$\min_{\chi} \sum_{i=0}^{\bar{M}} \sum_{j=0}^{M} c_{ij} \chi_{ij} \tag{54}$$

subject to

$$\sum\nolimits_{j=0}^{M} \chi_{ij} = 1,\ i = 1, \ldots, \bar{M}, \tag{55}$$

$$\sum\nolimits_{i=0}^{\bar{M}} \chi_{ij} = 1,\ j = 1, \ldots, M, \tag{56}$$

$$\chi_{ij} \in \{0, 1\} \tag{57}$$

The 2-D assignment problem (54)-(57) can be solved efficiently by a number of algorithms such as the modified auction algorithm, Jonker-Volgenant-Castanon (JVC) algorithm and the signature methods, which have been quite successful for multiple "point" target tracking [40]. Here we implemented the modified auction algorithm summarized as below.

The dual of the above 2-D assignment problem can be written as

$$\min_{\lambda ,\mu} \left\{ \sum_{i=1}^{\bar{M}} \lambda_i + \sum_{j=1}^{M} \mu_j \right\} \tag{58}$$

subject to

$$\lambda_i + \mu_j \geq -c_{ij}, \quad i = 1, ..., \bar{M}, \quad j = 1, ..., M \tag{59}$$

$$\lambda_i \geq -c_{i0}, \ i = 1, ..., \bar{M} \tag{60}$$

where λ_i and μ_j are Lagrangian multipliers. The dual problem is equivalent to

$$\min_{\mu} \left\{ \sum_{i=1}^{\bar{M}} \max \left\{ \max_j [-c_{ij} - \mu_j], -c_{i0} \right\} + \sum_{j=1}^{M} \mu_j \right\} \tag{61}$$

Clearly, we can define the cost for 2-D junction matching problem (54), (55), (56) as

$$c_{ij} = d(\bar{J}^{(i)}, J^{(j)}) \text{ for } i = 1, \ldots, \bar{M}, \ j = 1, \ldots, M. \tag{62}$$

Unfortunately, the costs c_{0j} and c_{i0} can not be expressed using a distance concept. Note that c_{0j} models the event that a new junction, namely, $J^{(j)}$, appears in the current data frame and c_{i0} models the event that an existing junction, namely, $\bar{J}^{(i)}$, disappears in the current data frame. The probabilistic models of these events depend on the sensor characteristics, junction detection algorithm (e.g., P_D and P_{fa} of a junction location/endpoint), possible scene occlusion, object geometry, which requires a lot of prior information.

4.3 Junction Association Via Convex Programming

Two-Frame Matching. As a better alternative to (54), we propose the overall cost for matching between two junction sets given by

$$\sum_{i=1}^{\bar{M}} \prod_{j=1}^{M} \frac{d(\bar{J}^{(i)}, J^{(j)})}{[\min_{k \neq i} d(\bar{J}^{(k)}, J^{(j)}) \min_{l \neq j} d(\bar{J}^{(i)}, J^{(l)})]^{\frac{1}{2}}} \cdot \chi_{ij}, \tag{63}$$

which needs to be minimized subject to the one-to-one matching constraints (55)–(56). This feature matching formulation not only considers the distance between matched junctions but also those junctions close to the matched one. In principle, a matched junction without strong competitors is preferred. Note that the above constrained optimization problem is no longer a 2-D assignment problem since the objective function is not the sum of the individual assignment costs.

Next, we treat the junction matching problem in a general convex programming framework which can handle an arbitrary twice differentiable objective function. Denote by $\mathbf{q} = \text{vec}(\{\chi_{ij}\})$. By relaxing the integer constraint, the objective function $f(\mathbf{q})$ becomes continuous with $\mathbf{q}$ which takes a value in a bounded set $Q = \{\mathbf{q} \in R^{\bar{M}M} : 0 \leq q_i \leq 1, \forall i\}$. Note that $f(\mathbf{q})$ may contain several local minima which makes it difficult for a gradient descent algorithm to find the global minimum. If we assume that $f(\mathbf{q})$ is twice differentiable and each entry of the Hessian matrix of $F(\mathbf{q})$ is continuous (denoted by $H^f_{ij}(\mathbf{q})$), then there exists a set of finite values d_i given by

$$d_i \geq \frac{1}{2}\left[\max_{\mathbf{q}\in Q}\sum_{j=1,j\neq i}^{\bar{M}M}|H^f_{ij}(\mathbf{q})| + \max_{\mathbf{q}\in Q}|H^f_{ii}(\mathbf{q})|\right]. \tag{64}$$

We use $\{d_i\}$ to construct a new objective function f_0 given by

$$f_0(\mathbf{q}) = f(\mathbf{q}) + \sum_{i=1}^{\bar{M}M} d_i q_i^2 - \sum_{i=1}^{\bar{M}M} d_i q_i. \tag{65}$$

The new objective function has the property that

$$H^{f_0}_{ii}(\mathbf{q}) > \sum_{j=1,j\neq i}^{\bar{M}M}|H^{f_0}_{ij}|, \quad \forall i, \tag{66}$$

which imposes a positive strictly dominant diagonal to the Hessian matrix of $f_0(\mathbf{q})$. A function with strictly positive diagonally dominant matrix is strictly positive definite. Hence $f_0(\mathbf{q})$ is strictly convex for $\mathbf{q} \in Q$. Note that for any integer solution $\mathbf{q}^*$, we have $f_0(\mathbf{q}^*) = f(\mathbf{q}^*)$. Therefore, the optimal solution to $f_0(\mathbf{q})$, if having all integer elements, is also the global minimum of $f(\mathbf{q}^*)$.

By changing the objective function to $f_0(\mathbf{q})$, we can write the new optimization problem as follows.

$$\mathbf{q}^* = \arg\min_{\mathbf{q}} f_0(\mathbf{q}) = \arg\min_{\mathbf{q}} f(\mathbf{q}) + \sum_{i=1}^{\bar{M}M} d_i q_i^2 - \sum_{i=1}^{\bar{M}M} d_i q_i \tag{67}$$

subject to

$$\mathbf{A}\mathbf{q} \leq \mathbf{b}, \quad \mathbf{0} \leq \mathbf{q} \leq \mathbf{1} \tag{68}$$

where

$$\mathbf{A} = \begin{bmatrix} \mathbf{1}'_{1\times M} \otimes \mathbf{I}_{\bar{M}\times\bar{M}} \\ \mathbf{1}'_{1\times\bar{M}} \otimes \mathbf{I}_{M\times M} \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \mathbf{1}_{\bar{M}\times 1} \\ \mathbf{1}_{M\times 1} \end{bmatrix}, \tag{69}$$

and $\otimes$ is the Kronecker product. It can be shown that $\mathbf{A}$ is totally unimodular and relaxation of integer constraint on $\mathbf{q}$ does not change the minimum of the original problem [11].

Extension to Multiframe Feature Matching. We want to extend the convex programming formulation to the case of multiframe junction association which can be solved in polynomial time. Assume that we have a set $\mathbf{X}$ containing N predicted junctions and measured junction sets $\mathbf{Y}_1$, ..., $\mathbf{Y}_S$ from S frames. Denote by $\mathcal{P} = [\mathbf{P}_1 \ldots \mathbf{P}_S]$ the association matrix where $\mathbf{P}_i$ denotes the correspondence matrix between $\mathbf{X}$ and $\mathbf{Y}_i$ as in the 2-D matching case. Let $\mathbf{q}_i = \text{vec}(\mathbf{P}_i)$ and $\mathbf{q} = [\mathbf{q}_1^T \ldots \mathbf{q}_S^T]^T$. For frame i, the constraints on $\mathbf{q}_i$ is $\mathbf{A}_i\mathbf{q}_i \leq \mathbf{b}_i$, i.e., any track in $\mathbf{X}$ can be assigned to at most one measurement in

$\mathbf{Y}_i$ and any measurement in $\mathbf{Y}_i$ can be assigned to at most one track in $\mathbf{X}$. Then the multiframe junction matching problem can be formulated as the following constrained optimization problem.

$$\mathbf{q}^* = \arg\min_{\mathbf{q}} f(\mathbf{X}, \mathbf{Y}_1, ..., \mathbf{Y}_S, \mathbf{q}) \tag{70}$$

subject to

$$\mathbf{A}\mathbf{q} \leq \mathbf{b}, \quad \mathbf{0} \leq \mathbf{q} \leq \mathbf{1}, \tag{71}$$

where

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 & & \\ & \ddots & \\ & & \mathbf{A}_S \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \mathbf{b}_1 \\ \cdots \\ \mathbf{b}_s \end{bmatrix}. \tag{72}$$

We assume the objective function is continuous in $\mathbf{q}$ and twice differentiable. Using similar technique as in the two frame case, we can construct a new objective function $f_0(\mathbf{q})$ which is strictly convex and has a unique global minimum $\mathbf{q}^*$ coincides with (70). We have shown that the constraint matrix $\mathbf{A}_i$ is totally unimodular for $i = 1, ..., S$. It can be shown that the block diagonal matrix $\mathbf{A}$ is totally unimodular since the row selection only applies to a submatrix of $\mathbf{A}$. Thus the optimal solution given by (70) will not change by relaxing the integer constraint on $\mathbf{q}$ [11].

5 Robust Track Fusion

5.1 Track Fusion as a Regularized Least Squares Problem

The local track estimates, if originating from the same target, can be fused by the global track estimation via equations (8)–(9) which is optimal within the linear estimators. When a noninformative prior is used for the target state at the fusion center, equations (8)–(9) become the least squares solution (16)–(17) which is optimal only within the class of linear unbiased estimators [33]. Note that the least squares solution minimizes the data fitting error rather than the mean square estimation error. This makes it possible to improve the estimation accuracy by exploring biased estimators. Various modifications of the least squares estimator have been proposed in [24,22,23]. The general guideline is that a biased estimator can reduce the mean square error when the variance being reduced is larger than the squared bias term. In the sequel, we present a scalar modification of the least squares estimator, which results in a regularized track fusion formula.

Recall the linear model for track fusion problem given in Section 2

$$\hat{\mathbf{x}} = Hx + \mathbf{w}, \tag{73}$$

where $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{P})$. The least squares (LS) solution to the problem $\min_x (\hat{\mathbf{x}} - Hx)'\mathbf{P}^{-1}(\hat{\mathbf{x}} - Hx)$ is given by (16)–(17). To find an estimator that can reduce the mean square estimation error of the LS solution, we introduce a multiplicative scalar bias term b and assume the regularized estimator (RE) has the form

$$\hat{x}^{\mathrm{RE}} = b\hat{x}^{\mathrm{LS}}. \tag{74}$$

The RE class includes shrunken estimators, an example of which is the well-known James-Stein estimator [24]. To find the optimal bias, we want to solve the following problem which directly minimizes the trace of the mean square error matrix.

$$\min_b E[(b\hat{x}^{\mathrm{LS}} - x)'(b\hat{x}^{\mathrm{LS}} - x)] \tag{75}$$

The solution is given by

$$b = \frac{E[(\hat{x}^{\mathrm{LS}})'x]}{E[(\hat{x}^{\mathrm{LS}})'\hat{x}^{\mathrm{LS}}]}, \tag{76}$$

where, after some algebraic manipulations, the two terms in the numerator and denominator are

$$E[(\hat{x}^{\mathrm{LS}})'x] = \mathrm{tr}[xx'] = ||x||^2, \tag{77}$$

$$E[(\hat{x}^{\mathrm{LS}})'\hat{x}^{\mathrm{LS}}] = \mathrm{tr}[xx' + (H'\mathbf{P}^{-1}H)^{-1}] = ||x||^2 + \mathrm{tr}[(H'\mathbf{P}^{-1}H)^{-1}]. \tag{78}$$

Thus the optimal bias is

$$b = \frac{||x||^2}{||x||^2 + \mathrm{tr}[(H'\mathbf{P}^{-1}H)^{-1}]}. \tag{79}$$

We can see that the optimal bias is always less than 1 which implies that regularization can always reduce the mean square error compared with the LS solution ($b = 1$). Note that the optimal bias depends on the unknown parameter x which can be replaced by the estimate $\hat{x}^{\mathrm{LS}}$. The resulting estimator is a nonlinear function of $\hat{\mathbf{x}}$. In practice, if we assume that x has a bounded norm, i.e., $||x||^2 \leq B$, then the bias can be chosen as

$$b = \frac{B}{B + \mathrm{tr}[(H'\mathbf{P}^{-1}H)^{-1}]}, \tag{80}$$

which makes the resulting regularized estimator linear. Note that the regularized estimator with bias given by (80) also minimizes the worst case regret [23]. It achieves the uniform Cramer-Rao lower bound under the same bounded bias gradient for each element of x [22]. The mean square error of the regularized estimator is

$$P^{\mathrm{RE}} = (1 - b)^2 xx' + b^2 (H'\mathbf{P}^{-1}H)^{-1}, \tag{81}$$

where the unknown state x can be replaced by its estimate. It is easy to verify that

$$\mathrm{tr}[P^{\mathrm{RE}}] = \frac{||x||^2 \mathrm{tr}[(H'\mathbf{P}^{-1}H)^{-1}]}{||x||^2 + \mathrm{tr}[(H'\mathbf{P}^{-1}H)^{-1}]} < \mathrm{tr}[P^{\mathrm{LS}}]. \tag{82}$$

Note that the regularized track fusion is useful only when the noninformative prior of the target state is used at the fusion center.

5.2 A Numerical Example (Continued)

We continue using the example presented in Section 2.3 to compare the mean square error of the track fusion algorithms. In this scalar track fusion problem, we assume the true location at -3 which is somewhere in the middle of the surveillance region with $|V| = 10$. We vary the correlation coefficient between the local estimation errors and plot the mean square error (MSE) of the fused estimate for the number of local trackers $N = 2, 3, 4$. Figure 3 shows the mean square errors of the least square fusion algorithm vs. the regularized fusion with optimal bias. We can see that the MSE gap between the LS and RE solution increases as the cross correlation coefficient ρ increases. The gap becomes smaller as the number of local trackers increases. For any fixed value x, the RE solution has a large performance gain when the variance of the LS solution is large.

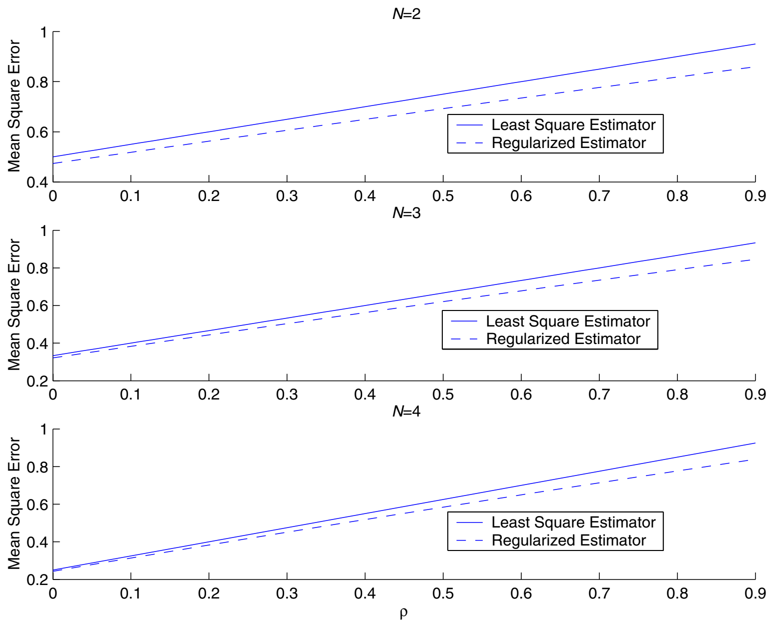


Fig. 3. Mean square errors of the fused estimates using least squares and regularization for different numbers of the local estimates for $x = 3$

6 Track Fusion with Legacy Track Sources

6.1 Problem Formulation

In this section the problems of track association and track fusion with legacy track sources are formulated assuming the trackers are Kalman filters. To simplify the discussion, we consider a one dimensional tracking example with target motion given by a discretized continuous time white noise acceleration (DCWNA) model [6]. For asynchronous sensors this model should be used in order to handle the white process noise for all values of the sampling interval consistently.

The state and measurement equations are[1] for sampling interval T

$$x(k+1) = Fx(k) + v(k) = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} x(k) + v(k), \tag{83}$$

$$z(k) = Hx(k) + w(k) = [1 \quad 0]x(k) + w(k), \tag{84}$$

where $v(k)$ is the zero mean white process noise sequence with covariance

$$E[v(k)v(k)'] \overset{\Delta}{=} Q(T) = Q(t_{k+1} - t_k) = \begin{bmatrix} \frac{T^3}{3} & \frac{T^2}{2} \\ \frac{T^2}{2} & T \end{bmatrix} \tilde{q}, \tag{85}$$

where $\tilde{q}$ is the (continuous time) process noise power spectral density (PSD)[2] and $w(k)$ is zero mean white measurement noise sequence, uncorrelated with the process noise, with variance

$$E[w(k)^2] = \sigma_w^2. \tag{86}$$

The target maneuvering index is defined as

$$\lambda_c = \sqrt{\frac{\tilde{q}T^3}{\sigma_w^2}}. \tag{87}$$

Then the steady state filter gain is [6]

$$W = \left[\alpha \quad \frac{\beta}{T}\right]', \tag{88}$$

where

$$\alpha = \beta\sqrt{u}, \tag{89}$$

$$\beta = \frac{12}{6(u + \sqrt{u}) + 1}, \tag{90}$$

$$u = \frac{1}{3} + \sqrt{\frac{1}{12} + \frac{4}{\lambda_c^2}}. \tag{91}$$

The state estimation covariance matrix is given by

$$P = \begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix} = \begin{bmatrix} \alpha & \frac{\beta}{T} \\ \frac{\beta}{T} & \frac{\beta(\alpha-\beta/2)}{(1-\alpha)T^2} \end{bmatrix} \sigma_w^2. \tag{92}$$

[1] This describes the motion along one coordinate. For motion in 2 or 3 dimensions, the model will consist of 2 or 3 such models with an appropriate stacked state vector.

[2] See [6] for a discussion on why it is incorrect to call this the variance of the process noise.

The above result is valid for the steady state of the DCWNA filter, but only with the optimal values of α and β as given in (89)–(90) [6].

A legacy tracker uses a fixed gain W, not necessarily the optimal one, in each of its α-β filter updates and sends the state estimates to the fusion center, typically, without covariance information. Since track association and track fusion algorithms require such information in order to combine local tracks from different sources, a procedure to obtain this missing information is discussed next.

6.2 Approximation of the Estimation Error Covariance

Because of the time-varying target-sensor geometry, an α-β filter, even though it uses fixed gains, is not necessarily in steady state. This is due to the nonstationarity of the measurement noises, which is accounted for next. Our model will assume that the tracking filter has a "slowly varying" (quasi-)steady state. The covariance of the target state estimate will be evaluated accounting for the fact that the sensor measurements (typically in polar or spherical coordinates for a radar), while having uncorrelated measurement noises between their components (range, azimuth/crossrange), have a coupling (correlation) between the track state estimation errors in different Cartesian coordinates.

Coupling Between Coordinates and Nonstationarity. For tracking in more than one dimension of the measurement space, the measurement covariance is converted from the sensor coordinates (typically polar or spherical) into the coordinates in which the state is defined (usually Cartesian). This will result in the correlation between the state estimation errors in the Cartesian coordinates. It is important to preserve the coupling between the coordinates when the uncertainty ellipse for position is elongated and slanted, e.g., a "cigar" with the main axis at 45° or 135°. Neglecting the correlation between the coordinates would yield a much larger uncertainty region.

To preserve the coupling between the state space coordinates due to the measurements, the fusion center should run the Joseph form of the covariance update at time k [6]

$$P(k) = [I - WH][FP(k-1)F' + Q][I - WH]' + WR(k)W', \tag{93}$$

with the appropriate sampling interval. The reason the Joseph form is needed is that the legacy filter gain is not optimal and only this equation is valid for the covariance (actually MSE matrix) update when arbitrary filter gains are used. The process noise covariance Q should be selected by the fusion center to model the target motion uncertainty to the extent possible. The filter gain W in (93) should be the same as in the legacy filter. The measurement noise covariance $R(k)$ in (93) is the covariance of the measurements converted from polar to Cartesian. The measurement conversion should be linearized at the latest measurement or the measurement prediction using the latest state. When $P(k-1)$ is unavailable at the fusion center, one can assume that $P(k-1) = P(k)$

in (93), resulting in an algebraic Riccati equation. This will yield a (slowly) time-varying covariance matrix that accounts for the nonstationarity of measurement noise.

Approximation of the Estimation Error Covariance of Legacy Trackers with Partial Information. When the communication network can provide a coarsely quantized 2-dimensional root mean square (RMS) position error, denoted as RMS_p, to the fusion center, the estimation error covariance can be obtained as follows.

We shall model RMS_p as the (steady state) error of two independent α-β filters, one in the range direction, the other in the cross-range direction. Denote the measurement noise RMS values in these directions as σ_r and $\sigma_{\times r}$, respectively. These are assumed to be known, based on the radar specifications and the radar-target geometry.

The position gains for these two filters are, according to (88),

$$\alpha_r = \alpha(\lambda_c\ ), \tag{94}$$

and

$$\alpha_{\times r} = \alpha(\lambda_{c_\times}\ ), \tag{95}$$

respectively, where the corresponding target maneuvering indices are, similar to (87),

$$\lambda_c\ = \sqrt{\frac{\tilde{q}T^3}{\sigma_r^2}}, \tag{96}$$

and

$$\lambda_{c_\times}\ = \sqrt{\frac{\tilde{q}T^3}{\sigma_{\times r}^2}}, \tag{97}$$

with $\tilde{q}$ the process noise PSD that models the motion uncertainty (in both the range and cross-range directions, uncorrelated between them) and T the sampling interval.

The RMS position error from the above filters is, based on (92), given by

$$\text{RMS}_p = \sqrt{\alpha_r\sigma_r^2 + \alpha_{\times r}\sigma_{\times r}^2}. \tag{98}$$

Assuming the value of RMS_p is available and the measurement noise variances are known, one can solve (98) (after substituting (96)–(97) into (94)–(95) and the result into (98)) to find $\tilde{q}$. Once this is obtained, one can use (92) or (93) to reconstruct (approximately) the covariance of the entire state estimate. Note, however, that while this is in a Cartesian coordinate system, this system is aligned with the line of sight from the radar to the target and it has to be rotated into the local common Cartesian system, which is, typically, East-North.

The above procedure allows to reconstruct (approximately) the track estimation error covariance from a coarsely quantized position RMS error, assumed to be conveyed by a communication network. A similar approach can be taken when RMS_p is a position prediction error, as well as for the 3-dimensional case.

Asynchronous Sensors. For asynchronous sensors, the state prediction (to the time for which fusion will be carried out) based on the legacy tracker's latest estimate should be used by the fusion center. Assume that the fusion is done at time k and the most recent estimate at the fusion center from the legacy tracker is $\hat{x}(\kappa)$ at time[3] κ, with $\kappa < k$. Then the fusion center needs to (i) approximate the estimation error covariance $P(\kappa)$ at time κ using (92) or (93) and (ii) apply the standard prediction equations given by

$$\hat{x}(k) = F(k,\kappa)\hat{x}(\kappa), \tag{99}$$

$$P(k) = F(k,\kappa)P(\kappa)F(k,\kappa)' + Q(k,\kappa), \tag{100}$$

to obtain the state prediction and the error covariance for time k. For the motion model (83), $Q(k,\kappa)$ is given by (85) with $T = t_k - t_\kappa$.

Thus what is needed to evaluate the covariance of the estimate from a legacy tracker are:

– the sampling interval,
– the process noise PSD,
– the measurement noise covariance.

It should be noted that the parameters based on which the legacy tracker has been designed are unlikely to be the same as listed above. Thus, what the fusion center should do is to replicate the performance of the legacy tracker to the extent possible.

6.3 Approximation of the Crosscovariance of the Estimation Errors

When two local tracks have correlated estimation errors, assuming they use the same target motion and measurement models, in the steady state, the crosscovariance matrix is given by [5]

$$P^{\times} = [I - WH][FP^{\times}F' + Q][I - WH]'. \tag{101}$$

The above Lyapunov matrix equation can be solved numerically for any given target maneuvering index. For a distributed tracking system, the calculation of the crosscovariance using (101) is not practical.

The following approximation is considered [12]. Denote by P^{ij} the approximate crosscovariance matrix between local tracks i and j. Each element of P^{ij}, which is a 2×2 matrix for the model considered in (83), is approximated by constant correlation coefficients as follows

$$P^{ij}_{lm} = \rho_{lm} \left[P^i_{ll} P^j_{mm}\right]^{\frac{1}{2}}, \qquad l, m = 1, 2, \tag{102}$$

where ρ_{11} is the position-position correlation coefficient, ρ_{12} is the position-velocity correlation coefficient and ρ_{22} is the velocity-velocity correlation coefficient.

[3] We use for simplicity the notations κ and k instead of t_κ and t_k.

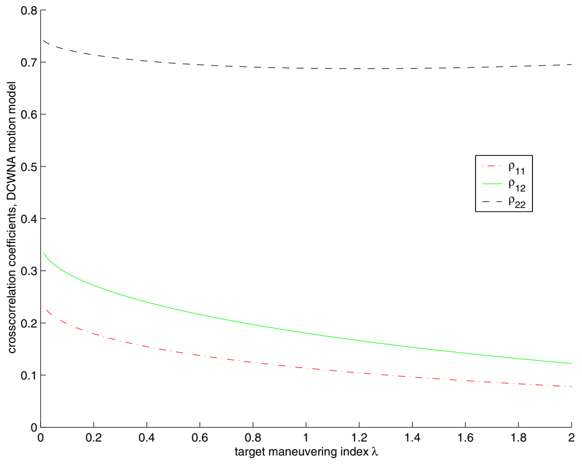


Fig. 4. Correlation coefficients vs. target maneuvering index for DCWNA model

Assuming equal variances of the measurement error for both sensors, we can solve the Lyapunov equation for the steady state DCWNA model. The resulting crosscorrelation coefficients between the estimation errors from the two local trackers, namely the position component ρ_{11}, the velocity component ρ_{22}, then position-velocity component ρ_{12}, are shown in Figure 4 for the target maneuvering index within (0.05, 2). The results are similar to those in [12] where the discrete time white noise acceleration model is used.

7 Simulation Study

We consider a ground target tracking scenario where three sensors are located at $(-50, 0)$km, $(0, 187)$km, and $(50, 0)$km, respectively. All three sensors measure the target range and bearing with the same standard deviations of the measurement error given by $\sigma_r = 50$m and $\sigma_b = 2$mrad. The sampling interval of sensors 1 and 2 is $T_1 = T_2 = 2$s while the sampling interval of sensor 3 is $T_3 = 5$s.

The two targets are initially at (0, 86.6)km and (0.4, 86.6)km, respectively. Both targets move in parallel with a speed of 300m/s. The motion of the two targets is characterized as follows. Both targets initially move toward south-east on a course of approximately $-135°$. Then at $t = 15$s both targets make a course change with a constant turn rate of $4°$/s (acceleration of about $2.1g$ over a duration T_{man} of about 11s) and heads toward east. Both targets make a second course change at $t = 35$s with a constant turn rate of $4°$/s and heads toward north-east. The trajectories of the two targets are shown in Figure 5 where the true target positions are indicated at the time instances at which a measurement is made by one of the three sensors. The total time for the two targets to complete the designated trajectories is 60s. Note that the target range is around 100km to each sensor at the beginning, where the standard

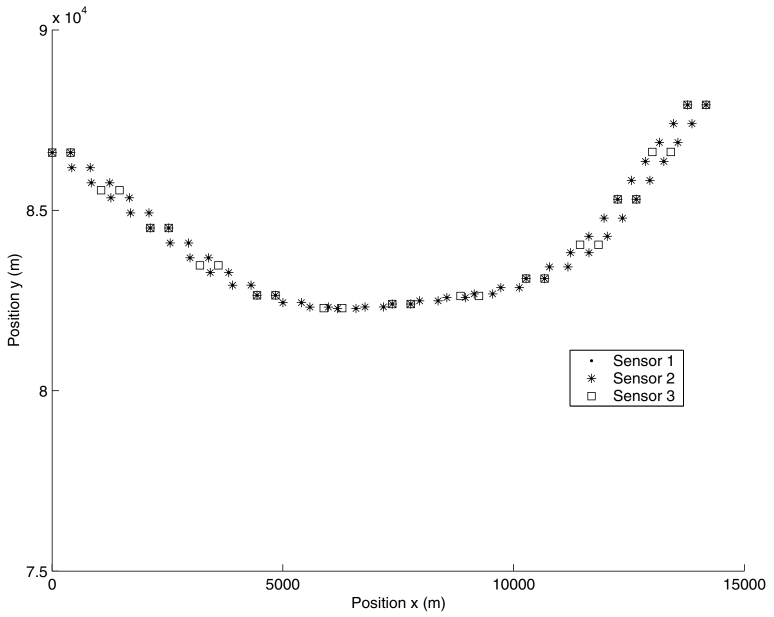


Fig. 5. Target trajectories with true positions at the times when measurements are made by the sensors

deviation of the crossrange measurement is around 200m. Thus the tracker has measurement origin uncertainty when updating the target state estimates. The true target motion has the random acceleration modeled by the white process noise spectrum $q = 1\text{m}^2/\text{s}^3$ in each realization. We assume that the two targets have unity detection probability by each sensor and no false measurements. This setting will make the track association error fairly small so that the estimation accuracy is mainly determined by the fusion algorithm.

We first evaluate the junction matching performance using assignment and convex programming algorithm based on synthetic images of 200 by 200 pixels. Ten junctions are generated from the the extended targets and each junction has a rank ranging from 1 to 4 with equal probability. The end points are generated with uniformly distributed angles and length from 5 to 30 pixels to the center. The matching frame has the same image size with all points of the junctions being perturbed with equal probability to one of the four directions (left, right, up and down) for up to k pixels. Figure 6 shows the percentage of correct track associations using extended feature matching versus that using the center of the track. As the distance between the two targets decreases, the percentage of correct track associations also decreases. Nevertheless, the track association accuracy using junction matching is superior to that with the best point-wise assignment.

To study the track fusion performance, two tracking configurations for performance comparison are implemented as follows.

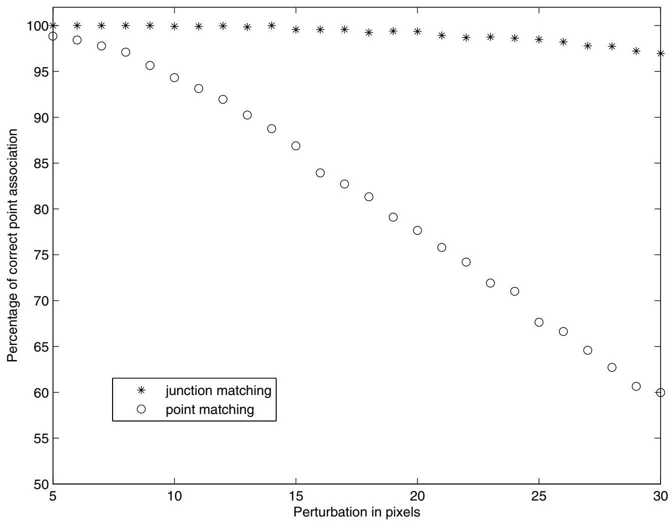


Fig. 6. Comparison of junction matching and point based track association, 500 Monte Carlo runs

(i) A centralized estimator which uses an interacting multiple model (IMM) filter [6] with two models and sequentially updates the target state with measurements from sensors 1–3. This IMM estimator has a discretized continuous white noise acceleration (DCWNA) model [6] with a low process noise PSD q_l to capture the uniform target motion and a DCWNA model with a high process noise PSD q_h to capture the two turns. We use $q_l = 1\text{m}^2/\text{s}^3$ and $q_h = 8000\text{m}^2/\text{s}^3$ which, for $T_{\text{man}} = 11\text{s}$, corresponds to a target average acceleration of $\sqrt{q_h/T_{\text{man}}} \approx 2.6g$. The process noise is the same in the east and north directions of the Cartesian coordinates and uncorrelated between these coordinates. The transition between the modes is modeled according to a continuous time Markov chain with the expected sojourn times ([6], Section 11.7.3) in these modes given by $1/\lambda_1$ and $1/\lambda_2$, respectively. These correspond to exponential sojourn time distributions with parameters λ_1 and λ_2, respectively. The transition probability matrix between the two models (generalized version of equation (11.6.7-1) in [6]) from any time t_1 to time t_2 is [39]

$$\Pi(t_2, t_1) = \frac{1}{\lambda_1 + \lambda_2} \begin{bmatrix} \lambda_2 + \lambda_1 e^{-(\lambda_1+\lambda_2)T} & \lambda_1 - \lambda_1 e^{-(\lambda_1+\lambda_2)T} \\ \lambda_2 - \lambda_2 e^{-(\lambda_1+\lambda_2)T} & \lambda_1 + \lambda_2 e^{-(\lambda_1+\lambda_2)T} \end{bmatrix}, \tag{103}$$

where $T = |t_2 - t_1|$. For the scenario used in simulation, we chose $\lambda_1 = 0.05\text{s}^{-1}$ and $\lambda_2 = 0.1\text{s}^{-1}$. For the centralized IMM estimator, 2-D assignment is used to solve the measurement-to-track association problem and the most likely hypothesis is chosen for the filter update.

(ii) In the decentralized tracking configuration each sensor uses an IMM estimator and the fusion center fuses the local estimates every $T_F = 10\text{s}$ using

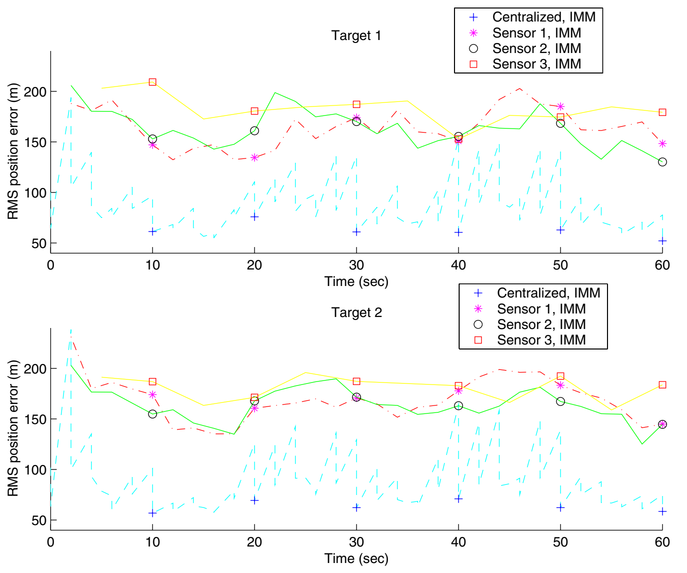


Fig. 7. Comparison of the RMS position errors for centralized IMM estimator (configuration (i)) and three local IMM estimators

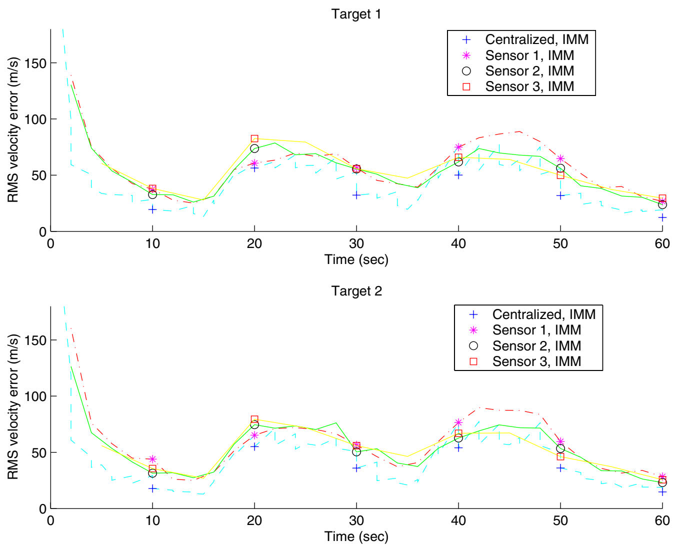


Fig. 8. Comparison of the RMS velocity errors for centralized IMM estimator (configuration (i)) and three local IMM estimators

all local state estimates and the corresponding covariances with approximate crosscovariances. For each local IMM estimator, 2-D assignment is used to solve the measurement-to-track association problem and the most likely hypothesis

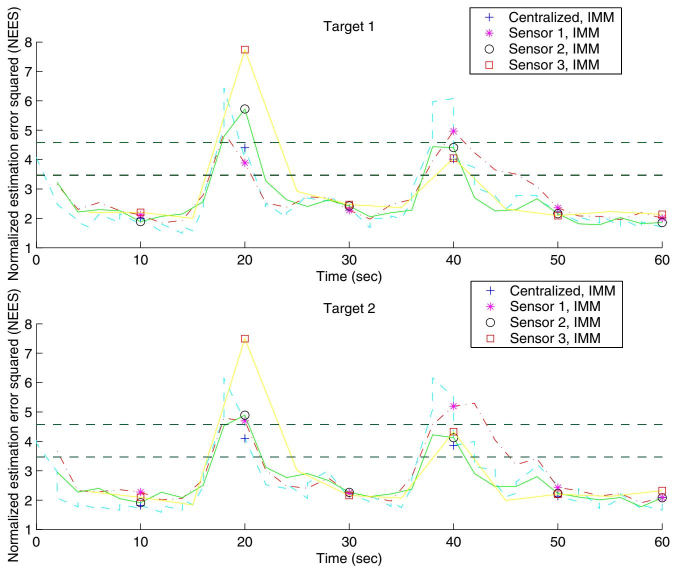


Fig. 9. Comparison of the NEES for centralized IMM estimator (configuration (i)) and three local IMM estimators

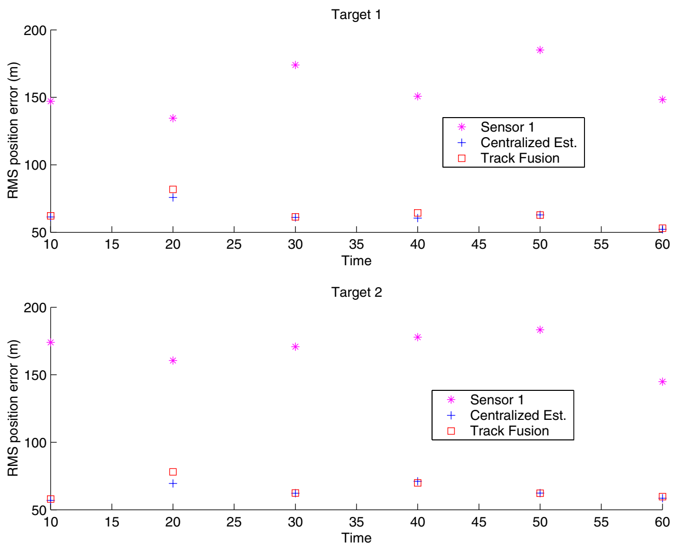


Fig. 10. Comparison of the RMS position errors for centralized IMM estimator (configuration (i)), track fusion from three IMM estimators (configuration (ii)); local IMM estimator from sensor 1 also shown

is chosen for the filter update. The track-to-track association is based on the most likely hypothesis obtained by solving the 3-D assignment among three local trackers. If the local tracks are declared as from the same target, then the

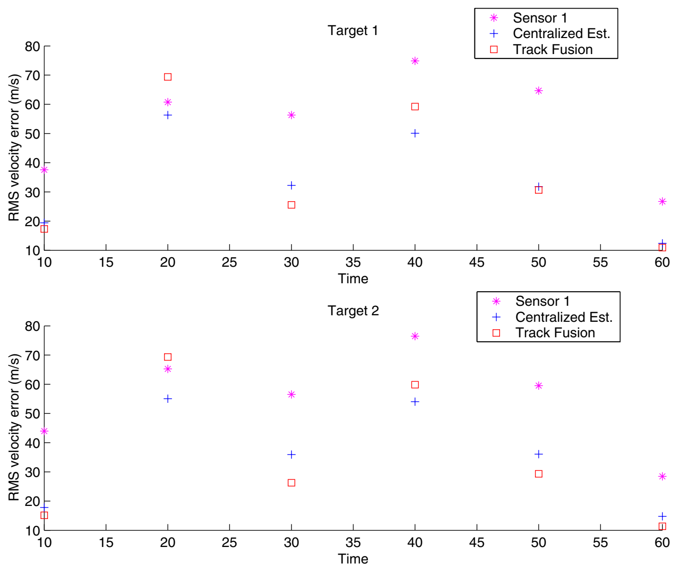


Fig. 11. Comparison of the RMS velocity errors for centralized IMM estimator (configuration (i)), track fusion from three IMM estimators (configuration (ii)); local IMM estimator from sensor 1 also shown

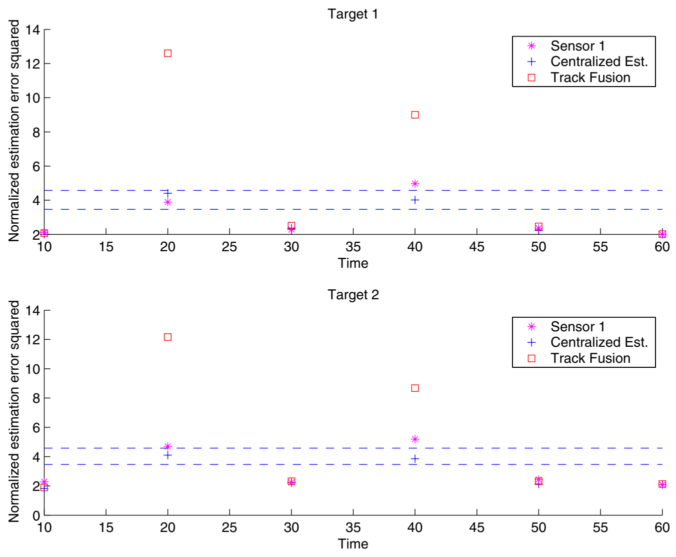


Fig. 12. Comparison of the NEES for centralized IMM estimator (configuration (i)), track fusion from three IMM estimators (configuration (ii)); local IMM estimator from sensor 1 also shown

track-to-track fusion is carried out using regularized track fusion with approximate cross covariance. The crosscovariance used at the fusion center is calculated with $\rho_{11} = 0.15$, $\rho_{12} = 0.25$ and $\rho_{22} = 0.7$ to obtain the crosscovariance matrix between local track pairs. Regularized track fusion is used with the bias b given by (79) with bounded norm of x in all cases. 100 Monte Carlo runs are used to compare the estimation accuracy and the filter credibility of the two tracking configurations.

Figure 7 shows the RMS position errors at the fusion center for the centralized IMM estimator and three local IMM estimators. The improvement of the tracking accuracy is fairly large for both targets and during both non-maneuver and maneuver segments. The position errors do not increase significantly during target maneuver. Figure 8 shows the corresponding RMS velocity errors for the centralized estimator and the local estimators. We can see that tracking accuracy does not improve too much because the sensors do not directly measure the velocity state. The velocity errors increase during the two maneuver segments with certain delay after the maneuver onset time. Figure 9 shows the normalized estimation error squared (NEES) [6] at the fusion center for the above four estimators. The NEES is an indication of the filter credibility for linear Gaussian dynamics. The lower and upper bound based on the 95% confidence interval of the chi-square distribution is also shown in the figure. We can see that the IMM estimator is conservative about its mean square estimation error when the target has a nearly constant velocity while it becomes optimistic during target maneuver. These plots provide the tracking performance of the ideal centralized tracker as well as the autonomous local trackers.

Figure 10 shows the RMS position errors at the fusion center for the above two tracking configurations as well as that by sensor 1 alone served as a baseline. We can see that the track fusion of the three local IMM estimates (configuration (ii)) yields nearly the same RMS position error as that of the centralized estimator (configuration (i)). Figure 11 compares the corresponding RMS velocity errors. We can see that the track fusion has smaller RMS velocity error when the targets have a nearly constant velocity but larger RMS velocity error after the targets start maneuver. Overall, the proposed assignment solution to the track-to-track association is very effective when the quality of the local tracks is good. Figure 12 shows the NEES at the fusion center for the above two tracking configurations. The lower and upper bound based on the 95% confidence interval of the chi-square distribution is also shown in the figure. We can see that the distributed track fusion yield larger NEES than the centralized estimator during the target turns. This indicates that the fused track is optimistic about its mean square estimation error. Thus caution has to be exercised when fusing the local estimates that are not credible on their own NEES statistics.

8 Conclusions

We used the Bayesian procedure to formulate the track association problem and provided algorithms that can solve a large scale distributed tracking problem where many sensors track many targets. When noninformative prior of the target state is assumed, the single target test becomes a chi-square test and it can be extended to the multiple target case by solving a multidimensional assignment problem. When a target has multiple feature points, we formulated the track association problem as junction matching and provided efficient algorithms to solve the corresponding problems via assignment and convex programming. With the noninformative prior assumption, the optimal track fusion algorithm can be a biased one where the regularized estimate has smaller mean square estimation error. A robust track fusion algorithm based on regularization was presented which modifies the optimal linear unbiased fusion rule by a less-than-unity scalar. When legacy sensor systems provide tracks without error covariance information, we provided a practical way of approximating the error covariance and cross-covariance between local estimates. Simulation results indicate the effectiveness of the proposed track association and fusion algorithm through a three-sensor two-target tracking scenario.

Acknowledgement

The authors would like to thank the anonymous referees for constructive comments that greatly improved the presentation of this chapter.

References

1. Bar-Shalom, Y.: Composite display and track management for the top m hypotheses tracking. ESPLab Technical Report TR-010522 (2001)
2. Bar-Shalom, Y., Blair, W.D. (eds.): Multitarget-multisensor tracking: applications and advances, vol. III. Artech House (2000)
3. Bar-Shalom, Y., Chen, H.: Multisensor track-to-track association for tracks with dependent errors. In: Proc. IEEE Conf. on Decision and Control, Atlantis, Bahamas, pp. 3498–3503 (2004)
4. Bar-Shalom, Y., Kirubarajan, T., Gokberk, C.: Tracking with classification-aided multiframe data association. IEEE Trans. Aerospace and Electronic Systems, 868–878 (2005)
5. Bar-Shalom, Y., Li, X.R.: Multitarget-multisensor tracking: principles and techniques. YBS Publishing (1995)
6. Bar-Shalom, Y., Li, X.R., Kirubarajan, T.: Estimation with applications to tracking and navigation: algorithms and software for information extraction. Wiley, Chichester (2001)
7. Blackman, S., Popoli, R.: Design and analysis of modern tracking systems. Artech House (1999)
8. Blasch, E.P.: Derivation of a belief filter for high range resolution radar simultaneous target tracking and identification. Ph.d. dissertation, Wright State University (1999)
9. Blasch, E.P., Hong, L.: Simultaneous identification and track fusion. In: Proc. IEEE Conf. on Decision and Control, Orlando, FL, USA, pp. 239–245 (1998)
10. Chang, K.C., Fung, R.: Target identification with bayesian networks in a multiple hypothesis tracking system. Opt. Eng., 684–691 (1999)

11. Chen, H., Kirubarajan, T.: A convex minimization approach to data association with prior constraints. In: Proc. SPIE on Signal and Data Processing of Small Targets, Orlando, FL, USA (2004)
12. Chen, H., Kirubarajan, T., Bar-Shalom, Y.: Performance limits of track-to-track fusion vs. centralized estimation: theory and application. IEEE Trans. Aerospace and Electronic Systems, 386–400 (2003)
13. Chen, H., Li, X.R.: On track fusion with communication constraints. In: Proc. of 10th International Conference on Information Fusion, Québec, Canada (2007)
14. Chen, H., Pattipati, K.R., Kirubarajan, T., Bar-Shalom, Y.: Data association with possibly unresolved measurements using linear programming. In: Proc. 5th ONR/GTRI Workshop on Target Tracking, Newport, RI, USA (2002)
15. Chong, C.-Y., Mori, S.: Metrics for feature-aided track association. In: Proc. 9th International Conference on Information Fusion, Florence, Italy (2006)
16. Drummond, O.E.: On features and attributes in multisensor multitarget tracking. In: Proc. 2nd International Conference on Information Fusion, Sunnyvale, CA, USA (1999)
17. Drummond, O.E.: Integration of features and attributes into target tracking. In: Proc. SPIE Conf. Signal and Data Processing of Small Targets, Orlando, FL, USA (2000)
18. Drummond, O.E.: Feature, attribute and classification aided target tracking. In: Proc. SPIE Conf. Signal and Data Processing of Small Targets, San Diego, CA, USA (2001)
19. Drummond, O.E.: On categorical feature aided target tracking. In: Proc. SPIE Conf. Signal and Data Processing of Small Targets, San Diego, CA, USA (2003)
20. Drummond, O.E.: Tracking and classification with attribute data from legacy sensors. In: Proc. ONR/GTRI Workshop on Target Tracking, Key West, FL, USA (2004)
21. Dubuisson, M.P., Jain, A.K.: A modified hausdorff distance for object matching. In: Proc. of Int. Conf. Pattern Recognition, pp. A: 566–568 (1994)
22. Eldar, Y.: Minimum variance in biased estimation: bounds and asymptotically optimal estimators. IEEE Trans. Signal Processing, 1915–1930 (2004)
23. Eldar, Y., Ben-Tal, A., Nemirovski, A.: Linear minimax regret estimation of deterministic parameters with bounded data uncertainties. IEEE Trans. Signal Processing, 2177–2188 (2004)
24. Eldar, Y., Oppenheim, A.V.: Covariance shaping least-square estimation. IEEE Trans. Signal Processing, 686–697 (2003)
25. Ferry, J.: Xmap: Track to track association with metric, feature, and target type data. In: Proc. 9th International Conference on Information Fusion, Florence, Italy (2006)
26. Ferry, J., Stone, L.D.: Track-to-track association with missing features. In: Proc. MSS Sensor and Data Fusion, Monterey, CA, USA (2005)
27. Horst, R., Pardalos, P.M., Thoai, N.V.: Introduction to global optimization, 2nd edn. Kluwer Academic Publishers, Dordrecht (2000)
28. Huttenlocher, D.P., Klanderman, G.A., Rucklidge, W.J.: Comparing images using the hausdorff distance. IEEE Trans. Pattern Analysis and Machine Intelligence, 850–863 (1993)
29. Jilkov, V.P., Chen, H., Li, X.R., Nguyen, T.: Feature association for object tracking. In: Proc. 9th International Conference on Information Fusion, Florence, Italy (2006)

30. Kaplan, L., Blair, W.D.: Assignment costs for multiple sensor track-to-track association. In: Proc. 7th International Conference on Information Fusion, Stockholm, Sweden (2004)
31. Kay, S.M.: Fundamentals of statistical signal processing, volume II: detection theory. Prentice-Hall, Englewood Cliffs (1998)
32. Li, X.R.: Optimal linear estimation fusion – part vii: dynamic systems. In: Proc. 2003 Int. Conf. Information Fusion, Cairns, Australia, pp. 455–462 (2003)
33. Li, X.R., Zhu, Y., Wang, J., Han, C.: Comparing images using the hausdorff distance. Optimal linear estimation fusion – part I: unified fusion rules, 2192–2208 (2003)
34. Lin, X.D., Bar-Shalom, Y., Kirubarajan, T.: Multisensor bias estimation using local tracks without a priori association. In: Proc. SPIE Conf. Signal and Data Processing of Small Targets, San Diego, CA, USA, pp. 236–246 (2003)
35. Mahalanobis, P.C.: On the generalized distance in statistics. In: Proceedings of the National Institute of Science in India, pp. 49–55 (1936)
36. Mori, S., Chong, C.-Y.: Effects of unpaired objects and sensor biases on track-to-track association: problems and solutions. In: Proc. MSS National Symp. on Sensor and Data Fusion, pp. 137–151 (2000)
37. Mori, S., Chong, C.-Y.: Evaluation of data association hypotheses: non-poisson iid cases. In: Proc. 7th International Conference on Information Fusion, Stockholm, Sweden (2004)
38. Mori, S., Chong, C.-Y., Tse, E., Wishner, E.P.: Tracking and classifying multiple targets without a priori identification. IEEE Trans. on Automatic Control (1986)
39. Papoulis, A., Pillai, S.U.: Probability, random variables and stochastic processes. McGraw-Hill, New York (2002)
40. Pattipati, K.R., Popp, R., Kirubarajan, T.: Survey of assignment techniques for multitarget tracking. In: Bar-Shalom, Y., Blair, W.D. (eds.) Multitarget-multisensor tracking: applications and advances, vol. III. Artech House (2000)
41. Popp, R., Pattipati, K.R., Bar-Shalom, Y.: An m-best multidimensional data association algorithm for multisensor multitarget tracking. IEEE Trans. Aerospace and Electronic Systems, 22–39 (2001)
42. Schuck, T., Friesel, M., Hunter, J.B.: Information properties as a means to define decision fusion methodologies in non-benign environments. In: Proc. Sixth International Conference on Information Fusion, Cairns, Australia (2003)
43. Schuck, T., Hunter, J.B.: Distributed classification in a multi-source environment. In: Proc. Sixth International Conference on Information Fusion, Cairns, Australia (2003)
44. Slocumb, B.J., Klusman, M.E.: A multiple model snr/rcs likelihood ratio score for radar-based feature-aided tracking. In: Proc. SPIE Conf. on Signal and Data Processing of Small Targets, San Diego, CA, USA (2005)
45. Stone, L.D., Williams, M.L., Tran, T.M.: Track to track association and bias removal. In: Proc. SPIE Conf. on Signal and Data Processing of Small Targets, Orlando, FL, USA (2002)
46. Sudano, J.: An algorithm that integrates dissimilar sensor tracks for today's platform configurations. In: Proc. IEEE National Aerospace and Electronics Conference, pp. 475–485 (1998)
47. Yang, C., Blasch, E.P.: Mutual-aided target tracking and identification. In: Proc. SPIE (2003)

Optimal Cooperative Thermalling of Unmanned Aerial Vehicles

Andrew T. Klesh, Pierre T. Kabamba, and Anouck R. Girard

University of Michigan, 1320 Beal Ave, Ann Arbor, MI 48105

Abstract. Motivated by cooperative exploration missions, this chapter considers the use of thermals to increase the altitudes of multiple unmanned aerial vehicles (UAVs). The mission of the UAVs is to travel through a given area and identify updrafts. The UAVs communicate to each other the location of each rise or fall in their altitude to form a map of the area. This imperfect map can then be used to identify areas of interest that may be potential thermals. The subsequent problem of utilizing these thermals is addressed from the viewpoint of information collection based on Shannon's channel capacity equation. This method yields paths that achieve the intended result, to elevate the aircraft to higher altitudes, while benefiting from cooperation. Several illustrations are given.

1 Introduction

This chapter is devoted to the problem of planning the paths of multiple vehicles thermalling in a given area. By thermalling we mean gaining altitude from updrafts.

The key idea of this work is to recognize and exploit the similarity between finding thermals and general exploration. Specifically, the identification of thermals can be viewed as an exploration process where the object of interest is the thermal, the sensor measures the aircraft's altitude and the collector of information is the UAV. Thus properties for time-optimal exploration paths are also properties of time-optimal thermalling paths. The identification of these similar properties is the main conceptual contribution of this chapter.

UAVs and other autonomous systems are increasingly used in long endurance missions[20]. While today's literature discusses UAVs as one of the most common types of autonomous systems, here we are interested solely in aircraft. The most common use of these systems is the collection of data for intelligence, surveillance and reconnaissance missions for which long endurance and high-altitude are often requirements. An example of such a mission is one of in situ weather reconnaissance[5], where the placement of UAVs around a forming storm is critical to the research. Moreover, it is often the case that gaining altitude quickly is of paramount importance (since inclement weather forms quickly). This motivates the time-optimal path planning problem formulated and solved in this chapter.

The literature on thermalling focuses on two aspects 1) the identification of thermals and 2) the utilization of thermals. For identification, much of the current literature uses a spiral path of a single UAV[3, 19] to discover thermal activity. This approach is similar to that of manned gliders [29]. The utilization of thermals with single autonomous aircraft has been accomplished at NASA Dryden [1, 2, 3] but to date there has been no work on cooperatively identifying thermals for later use.

A large body of research has been published in recent years about motion control of autonomous vehicles. Although an exhaustive overview of the state of the art is beyond the scope of this chapter, a brief review of the most relevant literature is as follows.

Many methods exist for solving the basic trajectory-planning problem [14]. However, not all of them solve the problem in its full generality. For instance, some methods require the workspace to be two-dimensional and the obstacles, if any, to be polygonal. Despite many external differences, the methods are based on a few different general approaches: roadmap [14, 16, 28], cell decomposition [17, 22, 23, 24, 28], potential field [4, 13, 15] and probabilistic [9, 27]. Optimal control approaches have also been studied in [7] and [25]. A survey of this work reveals that no study has been completed for finding time-optimal trajectories for thermalling.

Based on an integrated systems model, the problem of cooperative exploration for autonomous vehicles is formulated as an optimal path planning problem where the states are the Cartesian coordinates of the vehicles and the altitude gained from thermals in the area, the objective function is the total mission time, and the boundary conditions are subject to inequality constraints that reflect the requirements of altitude gain. The present chapter studies this optimization problem and provides the following original contributions:

– An integrated model of aircraft and thermals is presented.
– The necessary conditions for optimality are derived using this model.
– Similarities are shown between time-optimal thermalling and time-optimal exploration.
– The effect of uniform noise on the thermal model is shown to be negligible.

The remainder of the chapter is as follows. In Section II, the integrated model is presented. In Section III, the problem is formulated as an optimal control problem. Section IV describes cooperative methods to identify thermals. The necessary conditions for optimality are derived in Section V and the similarities between exploration and thermalling are described in Section VI. Section VII presents several examples and Section VIII describes conclusions and future work. Appendix A provides the necessary conditions for optimality of the exploration problem.

2 Modeling

In this section the model used throughout the chapter is presented. The model consists of two parts: the aircraft model and the thermal model.

2.1 Aircraft Model

The aircraft kinematic model is based upon a modified unicycle vehicle model [21]:

$$\dot{x_i} = v\cos(\psi_i), 1 \leq i \leq n, \tag{1}$$

$$\dot{y_i} = v\sin(\psi_i), 1 \leq i \leq n, \tag{2}$$

$$\dot{z_{ij}} = W_z \quad , \tag{3}$$

$$\dot{z_i} = \sum_{j=1}^{m} \dot{z_{ij}}, 1 \leq i \leq n. \tag{4}$$

where x_i, y_i and z_i are the Cartesian coordinates of the ith aircraft, v is the speed of the aircraft, ψ_i is the heading of the ith aircraft, W_z is the vertical component of jth thermal acting upon the ith UAV, z_{ij} is the amount of vertical altitude gained by the ith UAV from the jth thermal, m is the number of thermals and n is the number of aircraft. For simplicity, v is assumed to be the same for all aircraft. The aircraft are assumed to be powered and have the ability to maintain constant altitude flight.

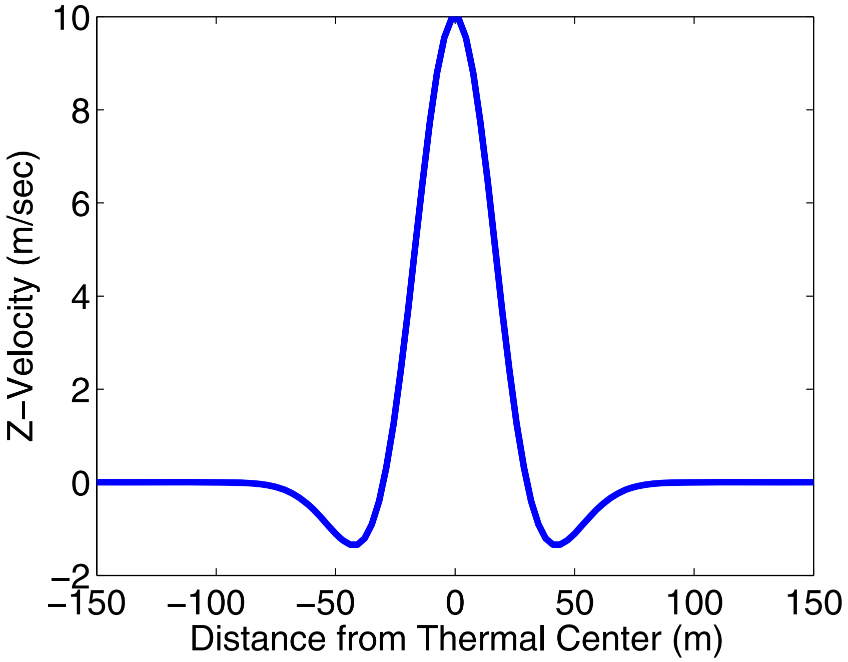


Fig. 1. 2-D Thermal Model

2.2 Thermal Model

A thermal is formed from differences in local air temperature. A typical thermal has a high velocity center and negative low velocity segments on either side of the center. A typical thermal model is shown in Figure 1.

The equation representing this model is:

$$W_z\ (r_{ij}) = W_{z,max} e^{-\left(\frac{}{}\right)^2} \left[1-\left(\frac{r_{ij}}{R_j}\right)^2\right], \tag{5}$$

where,

$$r_{ij} = \sqrt{(x_i-a_j)^2+(y_i-b_j)^2}, \tag{6}$$

and where r_{ij} is the range from the ith UAV to the center of the jth thermal, $W_{z,max}$ is the maximum velocity of the thermal, R_j is a scaling factor acting upon the jth thermal, and (a_j, b_j) are the Cartesian coordinates of the jth thermal. Typical thermals have R values between 100 and 1000 m [19].

To represent this model in 3-D space, this model is rotated about the z-axis producing Figure 2.

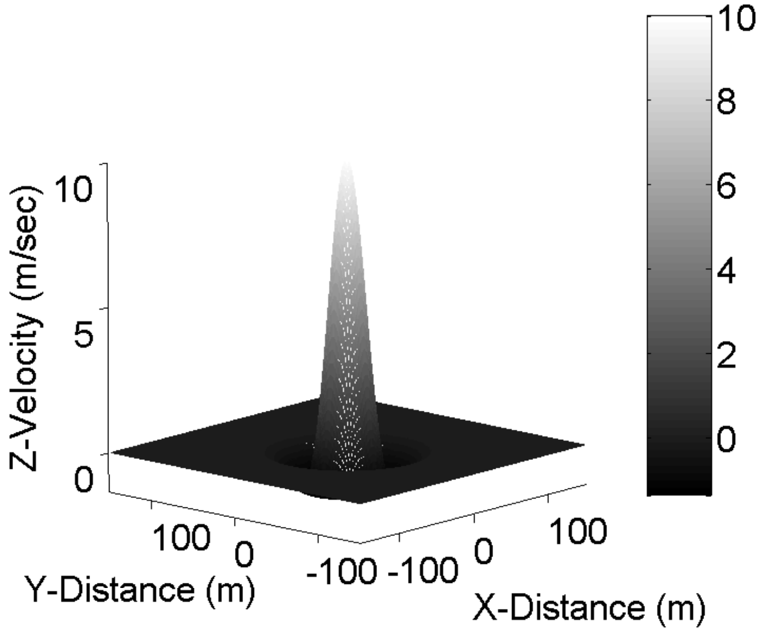


Fig. 2. 3-D Thermal Model

2.3 Model Summary

In summary, the integrated model is as follows. The heading of the aircraft and vertical velocity determine the position of the aircraft through (1) - (4). The vertical velocity of the aircraft is only changed due to external disturbances as shown in (4) and (5) respectively.

3 Problem Formulation

The problem treated in this chapter is motivated by the goal of minimizing the total mission time required for n UAVs to collectively identify some of the m thermals in a given area and use them to rise at least a specified distance. By rising a specified distance from each thermal (rather than an overall gain), the UAVs are able to sample the thermals and better form their map. The UAVs have no unique thermal sensing capability, but rely instead on mapping their vertical velocity over time and sharing this map. Here, we do not consider the bandwidth or communication issues involved in sharing this map. Hence the problem in this chapter is two fold:

1. Collectively identify the location of thermals in a given area.
2. Gain at least a specified amount of altitude from each identified thermal.

4 Thermal Identification

To increase the altitude of the UAVs through the use of thermals, the vertical updrafts must first be identified and targeted. The location of the thermals is not known a priori to the UAVs and no thermal sensing device is onboard the UAV. Instead, the UAVs are assumed to fly in constant altitude flight and any vertical disturbance is mapped. The recorded vertical velocities provide a picture-estimate of the thermal activity in the area.

The UAVs first fly in a spiral covering pattern to identify vertical disturbances as shown in Figure 3. It is assumed that each UAV will only gain or lose altitude due to external wind and will do so without changing bank angle. As the UAVs slowly spiral out, they keep store of an internal map of their position and the vertical disturbances at that location. Each internal map is shared to form a large common map. The more UAVs present, the more detailed and accurate the shared map will be.

A threshold filter is applied to the map to identify the most promising peaks of vertical velocity. As the approximate width, R, of the thermal is known, we use the collected information to form an estimated center of the thermal activity as shown in Figure 4.

Similar methods have been successfully used by single UAVs [1, 2, 3, 19]. There is a great deal of future work still to be accomplished in the area of cooperative identification of thermals in the presence of less than ideal conditions. Some work has been completed on this topic in the area of communication issues [8, 18, 26]. More work is needed to generalize these results.

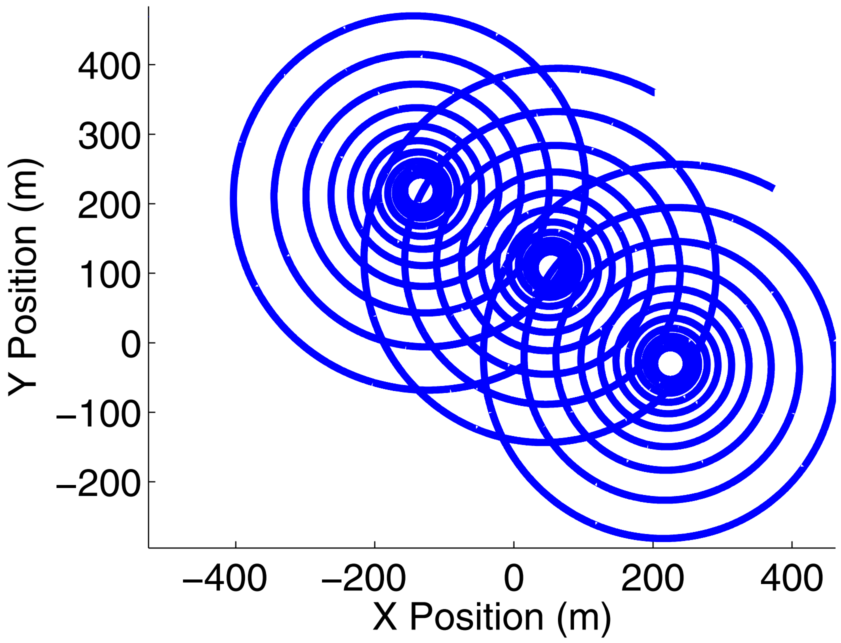


Fig. 3. Exploration Flight Paths

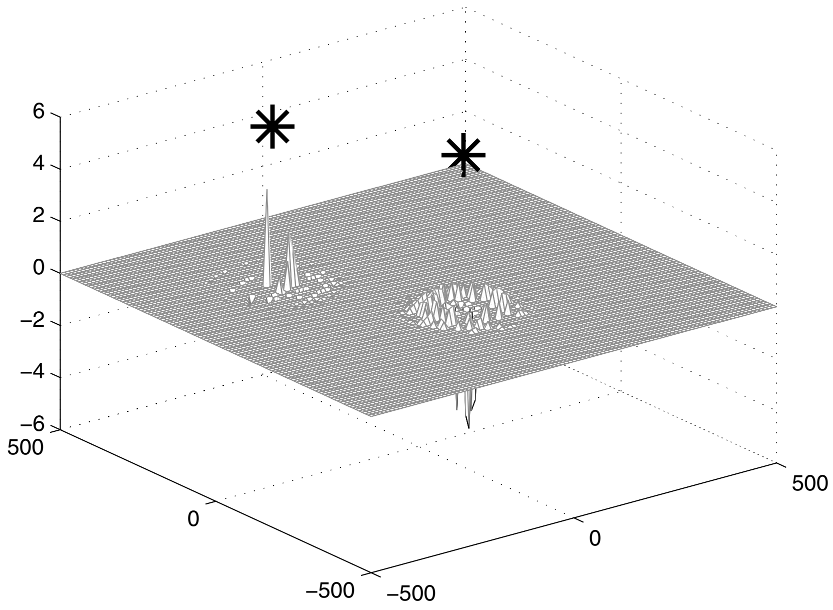


Fig. 4. Identification of the Thermal Activity

5 Time-Optimal Thermalling

The method of cooperatively identifying thermals in Section 4 uses a shared map of vertical disturbances. This allows the use of any aircraft to improve the map without specialized thermal sensors. Once the thermals have been identified cooperatively, each UAV can individually gain at least the specified height from each thermal.

Analysis of this problem reveals several similarities to the time-optimal information collection problem discussed in [10, 11, 12]. We will highlight these similarities in this section and from them draw useful conclusions about time-optimal thermalling.

5.1 Optimal Thermalling

Once the thermals are identified, the dynamic optimization problem is motivated by the requirement to minimize, with respect to the time-history of the heading angles, the total mission time, i.e.,

$$\min_{\psi_{\cdot}(\cdot)} t_f, \tag{7}$$

subject to (1)-(5) and boundary conditions:

$$x_i(0) = x_i^0, 1 \leq i \leq n, \tag{8}$$
$$y_i(0) = y_i^0, 1 \leq i \leq n, \tag{9}$$
$$z_{ij}(0) = z_{ij}^0, 1 \leq i \leq n, 1 \leq j \leq m \tag{10}$$
$$z_i(0) = z_i^0, 1 \leq i \leq n, \tag{11}$$
$$z_{ij}(t_f) \geq 1, 1 \leq j \leq m. \tag{12}$$

5.2 Optimal Path Planning

In this subsection, we derive the necessary conditions for optimality, adapted from [6]. With states $[x_i, y_i, z_{ij}, z_i]^T, 1 \leq i \leq n, 1 \leq j \leq m$ and control inputs $\psi_i, 1 \leq i \leq n$, the Hamiltonian is:

$$H = \sum_{i=1}^{n}\sum_{j=1}^{m} \lambda_{z_{\cdot}\,j} W_{z,max} e^{-(\frac{}{})^2}[1 - (\frac{r_{ij}}{R_j})^2] + \sum_{i=1}^{n} \lambda_{z_{\cdot}} (\sum_{j=1}^{m} \dot{z}_{ij})$$
$$+ \sum_{i=1}^{n} \lambda_{x_{\cdot}}\, v\cos(\psi_i) + \sum_{i=1}^{n} \lambda_{y_{\cdot}}\, v\sin(\psi_i) + 1, \tag{13}$$

where $\lambda_{x_{\cdot}}$, $\lambda_{y_{\cdot}}$, $1 \leq i \leq n$ $\lambda_{z_{\cdot}}$, $1 \leq i \leq n$ and $\lambda_{z_{\cdot}}$, $1 \leq i \leq n, 1 \leq j \leq m$ are costate variables.

In this problem formulation, there are no control constraints.

The state equations, derived from (13), are:

$$\dot{z_{ij}} = W_{z,max} e^{-(\frac{}{})^2}[1-(\frac{r_{ij}}{R_j})^2], 1 \leq i \leq n, 1 \leq j \leq m, \tag{14}$$

$$\dot{x_i} = v\cos(\psi_i), 1 \leq i \leq n, \tag{15}$$

$$\dot{y_i} = v\sin(\psi_i), 1 \leq i \leq n, \tag{16}$$

$$\dot{z}_i = \sum_{j=1}^{m} \dot{z}_{ij}, 1 \leq i \leq n. \tag{17}$$

The costate equations are:

$$\dot{\lambda_{z}} = 0, 1 \leq i \leq n, 1 \leq j \leq m, \tag{18}$$

$$\dot{\lambda} = \sum_{=1} -\frac{2W_{,} \; e^{-(\frac{}{})^2}[1-(\frac{}{})^2]\lambda \; (-a+x)(-1+\frac{(1.5+\;)(-2.5\;+\;)}{2})(-1.5-r\;)}{r}, \quad 1 \leq i \leq n, \tag{19}$$

$$\dot{\lambda} = \sum_{=1} -\frac{2W_{,} \; e^{-(\frac{}{})^2}[1-(\frac{}{})^2]\lambda \; (-b+y)(-1+\frac{(1.5+\;)(-2.5\;+\;)}{2})(-1.5-r\;)}{r}, \quad 1 \leq i \leq n, \tag{20}$$

$$\dot{\lambda_{z}} = 0, 1 \leq i \leq n. \tag{21}$$

The first-order optimality conditions are:

$$0 = v\lambda_{y}\cos(\psi_i) - v\lambda_{x}\sin(\psi_i), 1 \leq i \leq n. \tag{22}$$

The boundary conditions for this problem are:

$$x_i(0) = x_i^0, 1 \leq i \leq n, \tag{23}$$

$$y_i(0) = y_i^0, 1 \leq i \leq n, \tag{24}$$

$$z_{ij}(0) = 0, 1 \leq i \leq n, \tag{25}$$

$$z_i(0) = z_i^0, 1 \leq i \leq n, \tag{26}$$

$$z_{ij}(t_f) \geq 1, 1 \leq j \leq m, \tag{27}$$

$$\lambda_{x}(t_f) = 0, 1 \leq i \leq n, \tag{28}$$

$$\lambda_{y}(t_f) = 0, 1 \leq i \leq n, \tag{29}$$

$$\lambda_{z}(t_f) \begin{cases} = 0 \text{ if } z_{ij}(t_f) > 1, 1 \leq i \leq n, 1 \leq j \leq m, \\ \text{free, otherwise}, 1 \leq i \leq n, 1 \leq j \leq m, \end{cases} \tag{30}$$

$$\lambda_{z}(t_f) = 0, 1 \leq i \leq n. \tag{31}$$

The flight paths that satisfy the first order necessary conditions (14)-(22) will be referred to as *extremal paths*.

5.3 Discretization Procedure

To obtain numerical approximations of optimal paths, we discretize the problem as follows. For a chosen integer $p \geq 1$, we subdivide the interval $[t_o, t_f]$ into p subintervals $[t_o, t_1], [t_1, t_2], ..., [t_{p-1}, t_f]$ of equal duration. In each subinterval we assume that the control inputs are constant, i.e., $(\psi_i(t)) = (\psi_i \ ), t \in [t_g, t_{g+1}]$, where the parameters $\psi_i \ , 0 \leq g \leq p-1$, are initially unknown.

We treat the parameters $\psi_i \ , 0 \leq g \leq p-1$, and t_f as inputs to a nonlinear optimization problem. As an initial choice, in all subintervals we choose $\psi_i \ = 0$ and $t_f = t_o + T_M$ where T_M is a maximum duration allowed. Constraints upon this problem are imposed from the boundary conditions (23)-(31). From (7), the objective function is the total mission time. We then numerically solve for optimal flight paths using the MATLAB® Optimization Toolbox function *fmincon* and the ordinary differential equation solver *ode45*. We will call this strategy the discretization method.

6 Properties from General Exploration

The problem of time-optimal thermalling is very similar to another problem: time-optimal exploration. We can use the characteristics and properties of time-optimal exploration to infer similar characteristics for time-optimal thermalling. Specifically, the identification of thermals can be viewed as an exploration process where the object of interest is the thermal, the sensor measures the aircraft's altitude and the collector of information is the UAV. Thus properties for time-optimal exploration paths are also properties of time-optimal thermalling paths.

In the generalized exploration problem, the objective is to minimize the total mission time required for n autonomous vehicles to collect a specified amount of information about m objects of interest in a given area. The vehicles are assumed to have onboard sensors. Each sensor has a channel of limited bandwidth over which information is collected. In the neighborhood of each object of interest there exists a visibility disk, within which information can be collected. Outside of the visibility disk for a particular object, no information can be collected about that object. The visibility disk is assumed isotropic, i.e., the rate at which information is collected depends only upon the range from the object to the explorer, not on the azimuth.

Several differences exist between exploration and thermalling. In exploration, the ith UAV cooperatively collects information to increase the information state of the jth object of interest. In thermalling, the ith UAV gains altitude from all of the m thermals to increase its own altitude. Second, cooperative missions in exploration seek to collectively gain information while in thermalling, cooperation is used to identify potential thermals. Although these differences exist, many properties are still shared between the two problems.

6.1 Modeling

The model in the exploration consists of two parts: the vehicle kinematics model and the informatics model.

Kinematics Model. The kinematics model for the autonomous vehicles is given by:

$$\dot{\boldsymbol{\chi}}_{\boldsymbol{i}} = \boldsymbol{f}_{\boldsymbol{i}}(u_i), 1 \leq i \leq n, \tag{32}$$

where $\boldsymbol{\chi}_{\boldsymbol{i}}$ is the position vector, $\boldsymbol{f}_{\boldsymbol{i}}$ describes the kinematics and u_i is the control, of the ith vehicle.

Informatics Model. Let $I_j, 1 \leq j \leq m$ denote the information collected about the jth object of interest. Its rate of change is given by:

$$\dot{I}_j = \rho_j(\boldsymbol{\chi}_{\boldsymbol{1}}, ..., \boldsymbol{\chi}_{\boldsymbol{n}}, w_1, ..., w_n), 1 \leq j \leq m, \tag{33}$$

where the functions $\rho_j, 1 \leq j \leq m$ satisfy:

$$\rho_j(\boldsymbol{\chi}_{\boldsymbol{1}}, ..., \boldsymbol{\chi}_{\boldsymbol{n}}, w_1, ..., w_n) \begin{cases} = 0, \boldsymbol{\chi}_{\boldsymbol{1}}, ..., \boldsymbol{\chi}_{\boldsymbol{n}} \notin D_j, \\ > 0, \text{ otherwise,} \end{cases} \tag{34}$$

w_i is the bandwidth of the sensor on the ith vehicle, and D_j is the isotropic visibility disk centered on the jth object of interest. Without loss of generality, assume that the required amount of information is one bit for each object. When the ith vehicle is within the visibility disk of the jth object, i.e., $\chi_i \in D_j$, the object is considered visible. Otherwise the object is invisible to the vehicle. Define D to be the union of all visibility disks $D_j, 1 \leq j \leq m$, i.e., $D = \cup_{j=1}^m D_j$.

6.2 Properties of General Exploration

Here we emphasize relevant properties of time-optimal exploratory paths.

Proposition 1: *If the objects of interest are isolated, then the optimal flight paths consist of sequences of straight lines (far from the objects of interest) connected by short turns (near the objects of interest).*

A proof is provided in [11] for the general exploration problem.

Furthermore, from the optimality condition (42) the magnitudes of the time-rate of change of the control are:

$$\dot{u}_i = \frac{\displaystyle\sum_{j=1}^{n} \rho_j(\boldsymbol{\chi}_{\boldsymbol{1}}, ..., \boldsymbol{\chi}_{\boldsymbol{n}}, w_1, ..., w_n) \frac{\partial \boldsymbol{f}(u_i)}{\partial u_i}}{\displaystyle\sum_{i=1}^{n} \boldsymbol{\lambda}_\chi \frac{d}{dt}\left(\frac{\partial \boldsymbol{f}(u_i)}{\partial u_i}\right)}. \tag{35}$$

When $\chi_i \notin D$, (34) and (35) indicate that u_i is constant. Otherwise, the rate of change of u_i depends on the rates at which information is collected about visible objects, confirming Proposition 1.

We seek to exploit the similarity between information collection and altitude collection. Here we can view the object of interest as the thermal, the rate of information collection as a rate of vertical climb and D_j as R_j. From the similarity in the problems, we expect the properties presented to be evident in time-optimal thermalling.

6.3 Sensor Noise

The aircraft in the problem do not have the same limitations as other autonomous systems. Since they are outside, GPS provides a good estimate of their position. The thermal model, however, can be inaccurate. Consider (5), which is used to provide sensor information.

A modified model with a uniform random noise component is:

$$W_{z} \ (r_{ij}) = W_{z,max} e^{-\left(\frac{}{}\right)^2} \left[1 - \left(\frac{r_{ij}}{R_j}\right)^2\right] + W_{\text{Noise}}, \qquad (36)$$

where W_{Noise} is the term representing the uniform random noise.

We now seek to find time-optimal flights subject to the expected value of the altitude gained, or $E[z_{ij}] \geq 1$.

A Monte-Carlo simulation, with uniform noise added in the manner of (36), was run with an arbitrary 314 iterations. Noise was added with a fixed amplitude for 20 iterations, then the amplitude was increased. Analysis of the resulting flight paths show that the average flight path was very near the original noise free path. Furthermore, the RMS error of the flight path has no correlation with the amplitude of the simulated noise. While work needs to be completed,

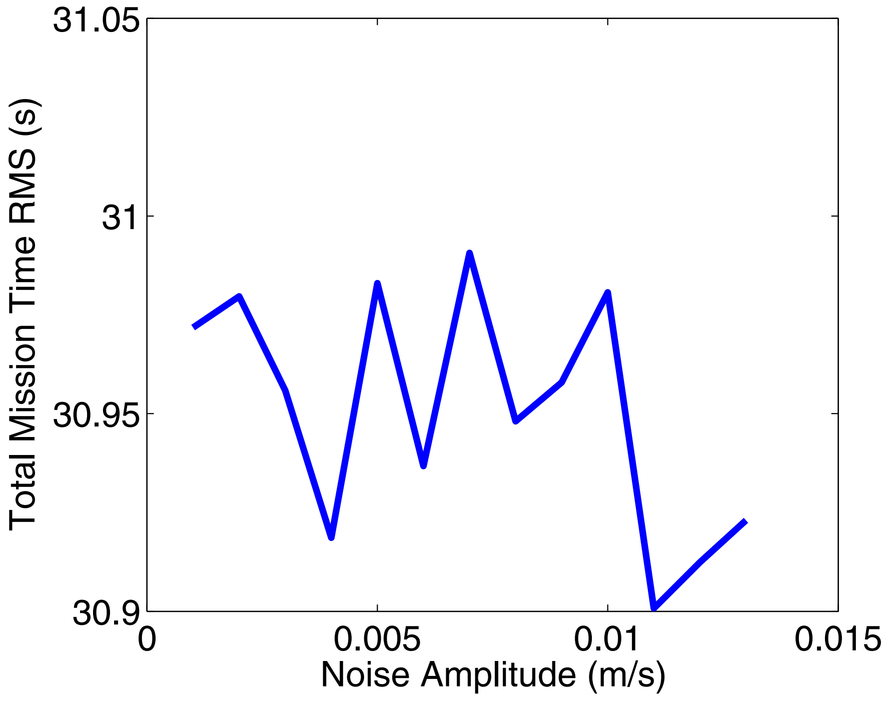


Fig. 5. Effect of noise in the thermal model on total mission time

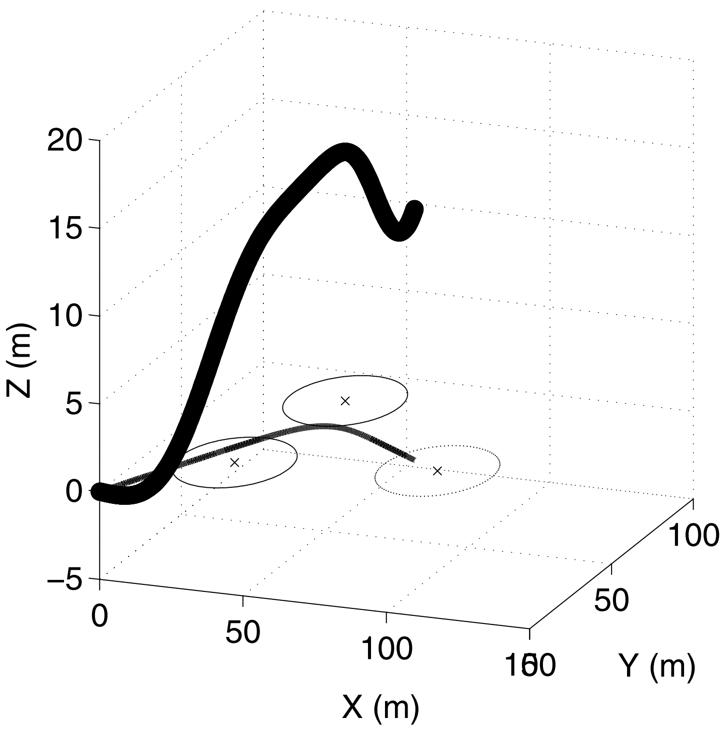


Fig. 6. Flight path and ground track of a UAV flying amongst three thermals

indications are that model noise will not increase the constraints on this problem. The result of this study is presented in Figure 5.

7 Thermalling Examples

Figure 6 demonstrates an example flight path of a single UAV among three thermals. The straight lines and short curves, predicted from the properties of general exploration, are evident in the ground track of the aircraft. Though the UAV drops in altitude between the thermals, it satisfies its boundary conditions by at least gaining a specified height by each thermal.

8 Conclusions and Future Work

This chapter has presented techniques for the identification and utilization of thermals. The problem of time-optimal thermalling is phrased in terms of altitude gain and vehicle kinematics. This formulation exploits the similarity between thermalling and exploration. We have shown that the identification of thermals can be completed cooperatively and using the same techniques as single UAVs. General properties for the utilization of thermals have been drawn from exploration problems. Furthermore, the presence of uniformly random inaccurate models has been shown to be negligible.

In future work, larger numbers of aircraft and thermals will be considered. Communication error will be introduced and the work may be extended to a time-varying environment.

References

1. Allen, M.J.: Updraft model for development of autonomous soaring uninhabited air vehicles. In: Proceedings of the ICAE 2003 Conference, ICAE (1996)
2. Allen, M.J.: Autonomous soaring for improved endurance of a small uninhabited air vehicle. In: Proceedings of the 43rd Aerospace Sciences Meeting, AIAA (2005)
3. Allen, M.J.: Guidance and control of an autonomous soaring UAV. Technical Report 214611 (February 2007)
4. Barraquand, J., Latombe, J.C.: Robot motion planning: A distributed representation approach. International Journal of Robotics Research 10, 628–649 (1991)
5. Blakeslee, R.J., Croskey, C.L., Desch, M.D., Farrell, W.M., Goldberg, R.A., Houser, J.G., Kim, H.S., Mach, D.M., Mitchell, J.D., Stoneburner, J.C.: The altus cumulus electrification study (aces): A UAV-based science demonstration. In: Proceedings of the ICAE 2003 Conference, ICAE (1996)
6. Bryson, B., Ho, Y.: Applied Optimal Control. Hemisphere Publishing Corporation (1975)
7. Dubins, L.E.: On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. American Journal of Mathematics 79, 497–517 (1957)
8. Julier, S.J., Uhlmann, J.K.: Real time distributed map building in large environments. In: International Society for Optical Engineering Proceedings (2000)
9. Kavraki, L.E., Latombe, P., Overmars, M.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Transactions on Robotics and Automation 12, 566–580 (1996)
10. Klesh, A., Girard, A., Kabamba, P.: Real-time path planning for time-optimal exploration. In: Proceedings of the 2008 AIAA GNC Conference, AIAA (accepted, August 2008)
11. Klesh, A., Kabamba, P., Girard, A.: Path planning for cooperative time-optimal information collection. In: Proceedings of the 2008 IEEE American Control Conference, ACC (accepted, June 2008)
12. Klesh, A., Kabamba, P., Girard, A.: A path planning framework for autonomous exploration. In: Proceedings of the 2008 IEEE Conference on Decision and Control, CDC (submitted, December 2008)
13. Koditschek, D.E.: Exact robot navigation by means of potential functions: Some topological considerations. In: Proceedings of the 1987 International Conference on Robotics and Automation. IEEE, Los Alamitos (1987)
14. Latombe, J.C.: Robot Motion Planning. Kluwer Academic Publishers, Dordrecht (1991)
15. LaValle, S.M., Kuffner, J.J.: Randomized kinodynamic planning. In: Proceedings of the IEEE International Conference on Robotics and Automation. IEEE, Los Alamitos (1999)
16. Lozano-Perez, T.: Automatic planning of manipulator transfer movements. IEEE Transactions on Systems, Man and Cybernetics 11, 681–698 (1981)

17. Lozano-Perez, T., Wesley, M.A.: An algorithm for planning collision-free paths among polyhedral obstacles. Communications of the ACM 22, 681–698 (1979)
18. Mahon, I., Williams, S.: Three-dimensional robotic mapping (2003)
19. Metzger, D.E., Hedrick, J.K.: Optimal flight paths for soaring flight. In: Proceedings of the 2nd International Symposium on the Technology and Science of Low Speed and Motorless Flight, AIAA (1974)
20. Office of the Secretary of Defense. Unmanned aircraft systems roadmap (2005-2030)
21. Savla, K., Bullo, F., Frazzoli, E.: On Traveling Salesperson Problem for Dubins' Vehicle: Stochastic and Dynamic Environments. In: Proceedings of the 44th IEEE Conference on Decision and Control and the European Control Conference 2005. IEEE, Los Alamitos (2005)
22. Schwartz, J.T., Sharir, M.: On the piano mover's problem: I. The case of a two-dimensional rigid polygonal body moving admist polygonal barriers. Communications on Pure and Applied Mathematics 36, 345–398 (1983)
23. Schwartz, J.T., Sharir, M.: On the Piano Mover's Problem: II. General Techniques for Computing Topological Properties of Real Algebraic Manifolds. Advances in Applied Mathematics 4, 51–96 (1983)
24. Schwartz, J.T., Sharir, M.: On the piano mover's problem: III. Coordinating the motion of several. independent bodies: The special case of circular bodies. Planning, Geometry, and Complexity of Robot Motion (1987)
25. Soueres, P., Laumond, J.P.: Shortest paths synthesis for a car-like robot. IEEE Transactions on Automatic Control, 672–688 (1996)
26. Speranzon, A., Fischione, C., Johansson, K.H.: Distributed and collaborative estimation over wireles sensor networks. In: 45th IEEE Conference on Decision and Control (2006)
27. Svestka, P., Overmars, M.: Coordinated motion planning for multiple car-like robots using probabilistic roadmaps. In: Proceedings of the IEEE International Conference on Robotics and Automation. IEEE, Los Alamitos (1995)
28. Udupa, S.: Collision detection and avoidance in computer controlled manipulators (1977)
29. Zhao, Y.J.: Energy-Efficient Trajectories of Unmanned Aerial Vehicles Flying through Thermals. Journal of Aerospace Engineering 18, 84 (2005)

Appendix A: Necessary Conditions for the Generalized Exploration Problem

In this section, we derive the necessary conditions for optimality adapted from [6]. With states $[I_j, \boldsymbol{\chi_i}]^T, 1 \leq i \leq n, 1 \leq j \leq m$ and control inputs $u_i, 1 \leq i \leq n$, the Hamiltonian is:

$$H = \sum_{j=1}^{m} \lambda_I \ \rho_j(\boldsymbol{\chi_1}, ..., \boldsymbol{\chi_n}, w_1, ..., w_n) + \sum_{i=1}^{n} \boldsymbol{\lambda}_\chi \ \boldsymbol{f}(u_i) + 1, \tag{37}$$

where $\boldsymbol{\lambda}_\chi$, $1 \leq i \leq n$ and λ_I , $1 \leq j \leq m$ are costate variables.

In this problem formulation, we have no user-imposed control constraints.

The state equations, derived from (37), are:

$$\dot{I}_j = \rho_j(\boldsymbol{\chi_1}, ..., \boldsymbol{\chi_n}, w_1, ..., w_n), 1 \leq j \leq m, \tag{38}$$
$$\dot{\boldsymbol{\chi}}_{\boldsymbol{i}} = \boldsymbol{f}(u_i), 1 \leq i \leq n. \tag{39}$$

The costate equations are:

$$\dot{\lambda_I} = 0, 1 \leq j \leq m, \tag{40}$$
$$\dot{\lambda_{x_{}}} = -\sum_{j=1}^{m} \lambda_I \ \frac{\partial \rho(\boldsymbol{\chi_1}, ..., \boldsymbol{\chi_n}, w_1, ..., w_n)}{\partial \boldsymbol{\chi_i}}, 1 \leq i \leq n. \tag{41}$$

The first-order optimality conditions are:

$$0 = \boldsymbol{\lambda}_{\boldsymbol{\chi_i}} \frac{\partial \boldsymbol{f}(u_i)}{\partial u_i}, 1 \leq i \leq n. \tag{42}$$

The boundary conditions for this problem are:

$$\boldsymbol{\chi_i}(0)\chi_i^0, 1 \leq i \leq n, \tag{43}$$
$$I_j(0) = 0, 1 \leq j \leq m, \tag{44}$$
$$I_j(t_f) \geq 1, 1 \leq j \leq m, \tag{45}$$
$$\boldsymbol{\lambda}_{\chi} \ (t_f) = 0, 1 \leq i \leq n, \tag{46}$$
$$\lambda_I \ (t_f) \begin{cases} \text{free if } I_j = 1, 1 \leq j \leq m, \\ = 0 \text{ otherwise.} \end{cases} \tag{47}$$

Model Predictive Control of Vehicle Formations*

Fernando A.C.C. Fontes,[1] Dalila B.M.M. Fontes[2], and Amélia C.D. Caldeira[3]

[1] Depart. de Matemática para a Ciência e Tecnologia and ISR-Porto, Universidade do Minho, 4800-058 Guimarães, Portugal
ffontes@mct.uminho.pt

[2] Faculdade de Economia and LIAAD-INESC Porto L.A., Universidade do Porto
Rua Dr. Roberto Frias, 4200-464 Porto, Portugal
fontes@fep.up.pt

[3] Departamento de Matemática, Instituto Superior de Engenharia do Porto, R. Dr. Ant. Bernardino de Almeida, 431, 4200-072 Porto, Portugal
acd@isep.ipp.pt

Abstract. We propose a two-layer scheme to control a set of vehicles moving in a formation.

The first layer, the trajectory controller, is a nonlinear controller since most vehicles are nonholonomic systems and require a nonlinear, even discontinuous, feedback to stabilize them. The trajectory controller, a model predictive controller, computes centrally a bang-bang control law and only a small set of parameters need to be transmitted to each vehicle at each iteration.

The second layer, the formation controller, aims to compensate for small changes around a nominal trajectory maintaining the relative positions between vehicles. We argue that the formation control can be, in most cases, adequately carried out by a linear model predictive controller accommodating input and state constraints. This has the advantage that the control laws for each vehicle are simple piecewise affine feedback laws that can be pre-computed off-line and implemented in a distributed way in each vehicle.

Although several optimization problems have to be solved, the control strategy proposed results in a simple and efficient implementation where no optimization problem needs to be solved in real-time at each vehicle.

1 Introduction

In this chapter we propose a control scheme for a set of vehicles moving in a formation. Vehicle formations are used in several applications, both in military and civilian operations, such as surveillance, forest fire detection, search missions, automated highways, exploration robots, among many others (see e.g., [20]).

* Research supported by FCT/POCI 2010/FEDER through Project POCTI/MAT/61842/2004.

The control methodology selected is a two-layer control scheme where each layer is based on model predictive control (MPC). Control of multi-vehicle formations and/or distributed MPC schemes have been proposed in the literature. See the recent works [5,22] and references therein.

The reason why two-layers are used in the control scheme is because there are two intrinsically different control problems:

- the trajectory control problem: to devise a trajectory, and corresponding actuator signals, for the formation as a whole.
- maintain the formation: change the actuator signals in each vehicle to compensate for small changes around a nominal trajectory and maintain the relative position between vehicles.

These control problems are intrinsically different because, on the one hand, most vehicles (cars, planes, submarines, wheeled vehicles) are nonholonomic (cannot move in all directions). On the other hand, while the vehicles are in motion, the relative position between them in a formation can be changed in all directions (as if they were holonomic).

As an example consider a vehicle whose dynamics we are familiar with: a car. Consider the car performing a parking maneuver or performing an overtaking maneuver. See figures 1 and 2.

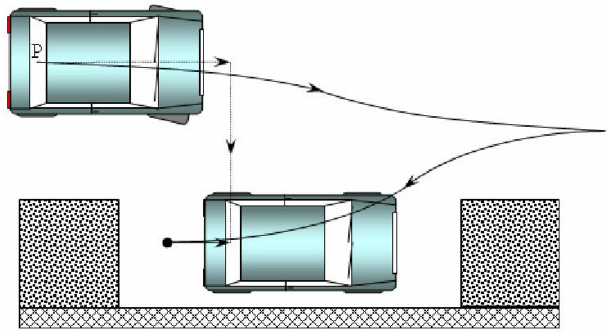


Fig. 1. Car in a parking maneuver: cannot move sideways

In the first situation, we are limited by the fact that the car cannot move sideways: it is nonholonomic. It is a known result that we need a nonlinear controller, allowing discontinuous feedback laws, to stabilize this system in this situation [23,3].

In the second, the vehicle is in motion and small changes around the nominal trajectory can be carried out in all directions of the space. In an overtaking maneuver we can move in all directions relative to the other vehicle. This fact simplifies considerably the controller design. In fact, a linear controller is an appropriate controller to deal with small changes in every spatial direction around a determined operating point.

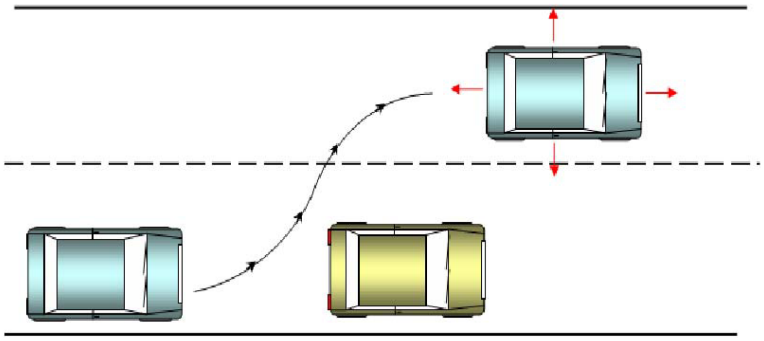


Fig. 2. Car in an overtaking maneuver: can move in all directions relative to the other car

For a different approach that also decouples the path following from the inter-vehicle coordination problem see [11].

The other option here is to design and use controllers based on the Model Predictive Control technique. Several reasons support this option.

MPC is known to be a technique that deals appropriately and explicitly with constraints. In fact, many researchers argue that the capacity of dealing naturally and effectively with constraints is the main reason for the industrial success of MPC; see e.g., [18,15,21].

In the problem of controlling a vehicle formation, the constraints on the trajectory are an important problem characteristic:

– to avoid collisions between the vehicles in the formation;
– to avoid obstacles in the path; and,
– for other reasons of safety or reliability of operation (e.g., minimum altitude or velocity in a plane).

Therefore, constraints should be appropriately dealt with. A possible way to deal with constraints is to use unconstrained design methods combined with a "cautious" approach of operating far from the constraints (e.g., imposing large distances between the vehicles in the formations). But, such procedures would, in general, reduce the performance that would be achievable by operating closer to the limits [12].

Another possible approach is to push the trajectory away from the constraints by penalizing a vehicle close to an obstacle or other vehicle (e.g., potential field approaches, or other optimization-based approaches that do not impose explicitly the constraints), but these methods do not guarantee that the constraints will be satisfied.

Here, we use MPC imposing explicit constraints both on the vehicle inputs and on the vehicle trajectory.

Another advantage of MPC is that, because MPC is an optimization-based method, it has desirable performance properties according to the performance criteria defined. In addition, it has intrinsic robustness properties [17].

The major concern in using such a technique is often the need to solve optimization problems in real-time which might be difficult for fast systems. However, there are recent results on particular parameterizations for the optimal control problems involved that allow MPC to address fast systems (see e.g., [1]). We shall see that the control strategy proposed appropriately deals with this problem.

2 The MPC Framework

Consider a nonlinear plant with input and state constraints, where the evolution of the state after time t_0 is predicted by the following model.

$$\begin{aligned} &\dot{x}(s) = f(s, x(s), u(s)) && \text{a.e. } s \geq t_0, && \text{(1a)}\\ &x(t_0) = x_{t_0} \in X_0, && && \text{(1b)}\\ &x(s) \in X \subset \mathbb{R}^n && \text{for all } s \geq t_0, && \text{(1c)}\\ &u(s) \in U && \text{a.e. } s \geq t_0. && \text{(1d)} \end{aligned}$$

The data of this model comprise a set $X_0 \subset \mathbb{R}^n$ containing all possible initial states at the initial time t_0, a vector x_{t_0} that is the state of the plant measured at time t_0, a given function $f : \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$, and a set $U \subset \mathbb{R}^m$ of possible control values.

We assume this system to be asymptotically controllable on X_0 and that for all $t \geq 0$ $f(t, 0, 0) = 0$. We further assume that the function f is continuous and locally Lipschitz with respect to the second argument.

The construction of the feedback law can be accomplished by using a sampled-data MPC strategy [10]. Consider a sequence of sampling instants $\pi := \{t_i\}_{i\geq 0}$, with a constant inter-sampling time $\delta > 0$ such that $t_{i+1} = t_i + \delta$ for all $i \geq 0$. Consider also the control horizon and predictive horizon, T_c and T_p, with $T_p \geq T_c > \delta$, and an auxiliary control law $k^{aux} : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^m$. The feedback control is obtained by repeatedly solving online open-loop optimal control problems $\mathcal{P}(t_i, x_{t_i}, T_c, T_p)$ at each sampling instant $t_i \in \pi$, every time using the current measure of the state of the plant x_{t_i}.

$\mathcal{P}(t, x_t, T_c, T_p)$: Minimize

$$\int_t^{t+T} L(s, x(s), u(s))ds + W(t + T_p, x(t + T_p)), \tag{2}$$

subject to:

$$\begin{aligned} &\dot{x}(s) = f(s, x(s), u(s)) && \text{a.e. } s \in [t, t + T_p], && \text{(3)}\\ &x(t) = x_t, && &&\\ &x(s) \in X && \text{for all } s \in [t, t + T_p], &&\\ &u(s) \in U && \text{a.e. } s \in [t, t + T_c], &&\\ &u(s) = k^{aux}(s, x(s)) && \text{a.e. } s \in [t + T_c, t + T_p], &&\\ &x(t + T_p) \in S. && && \text{(4)} \end{aligned}$$

The domain of this optimization problem is the set of admissible processes, namely pairs (x, u) comprising a control function u and the corresponding state trajectory x which satisfy the constraints of $\mathcal{P}(t, x_t, T_c, T_p)$. A process $(\bar{x}, \bar{u})$ is said to solve $\mathcal{P}(t, x_t, T_c, T_p)$ if it minimizes (2) among all admissible processes.

Note that in the interval $[t + T_c, t + T_p]$ the control value is determined by k^{aux} and therefore the optimization decisions are all carried out in the interval $[t, t + T_c]$.

We call *design parameters* those variables present in the open-loop optimal control problem that are not from the system model (i.e., variables we are able to choose); these comprise the control horizon T_c, the prediction horizon T_p, the running cost function L, the terminal cost function W, the auxiliary control law k^{aux}, and the terminal constraint set $S \subset \mathbb{R}^n$. The choice of these variables is important to obtain certain properties for the MPC strategy, such as stability, robustness, or performance (see e.g., [19,7,9] for a discussion on to choose the design parameters).

The MPC algorithm performs according to a receding horizon strategy, as follows.

1. Measure the current state of the plant $x^*(t_i)$.
2. Compute the open-loop optimal control $\bar{u} : [t_i, t_i + T_c] \to \mathbb{R}^n$ solution to problem $\mathcal{P}(t_i, x^*(t_i), T_c, T_p)$.
3. Apply to the plant the control $u^*(t) := \bar{u}(t; t_i, x^*(t_i))$ in the interval $[t_i, t_i + \delta)$ (the remaining control $\bar{u}(t), t \geq t_i + \delta$ is discarded).
4. Repeat the procedure from (1.) for the next sampling instant t_{i+1} (the index i is incremented by one unit).

The resultant control law u^* is a "sampling-feedback" control since during each sampling interval, the control u^* is dependent on the state $x^*(t_i)$. More precisely, the resulting trajectory is given by

$$x^*(t_0) = x_{t_0}, \quad \dot{x}^*(t) = f(t, x^*(t), u^*(t)) \quad t \geq t_0, \tag{5}$$

where

$$u^*(t) := \bar{u}(t; \lfloor t \rfloor_\pi, x^*(\lfloor t \rfloor_\pi)) \quad t \geq t_0, \tag{6}$$

and the function $t \mapsto \lfloor t \rfloor_\pi$ gives the last sampling instant before t, that is

$$\lfloor t \rfloor_\pi := \max_i \{t_i \in \pi : t_i \leq t\}. \tag{7}$$

Similar sampled-data frameworks using continuous-time models and sampling the state of the plant at discrete instants of time were adopted in [4,7,8,6,16] and are becoming the accepted framework for continuous-time MPC. It can be shown that with this framework it is possible to address — and guarantee stability, and robustness of the resultant closed-loop system — a very large class of systems, which are possibly nonlinear, time-varying, and nonholonomic.

3 The Two-Layer Control Scheme

As discussed, we propose a two-layer control scheme. The top layer, **the trajectory controller**, is applied to the group of vehicles as a whole. The trajectory controller is a nonlinear controller since most vehicles are nonholonomic systems and require a nonlinear, even discontinuous, feedback to stabilize them. The main ideas for the controller used in this section follow closely the results from [9].

The bottom layer, **the formation controller**, is computed and applied to each vehicle individually. The formation controller aims to compensate for small changes around a nominal trajectory maintaining the relative positions between vehicles. We argue that the formation control can be adequately carried out by a linear model predictive controller accommodating input and state constraints. This has the advantage that the control laws for each vehicle are simple piecewise affine feedback laws that can be pre-computed off-line and implemented in a distributed way in each vehicle. The main reference for this controller is [2].

4 The Vehicle Formation Models

4.1 Nonholonomic Vehicle Model for the Trajectory Controller

The framework developed here could easily be adapted to vehicles moving in 2D or 3D and having various dynamics. Nevertheless, we will explore a simple case of a a differential-drive mobile robot moving on a plane, Figure 3, and represented by the following kinematic model:

$$\dot{x}(t) = (u_1(t) + u_2(t)) \cdot \cos\theta(t) \tag{8}$$
$$\dot{y}(t) = (u_1(t) + u_2(t)) \cdot \sin\theta(t) \tag{9}$$
$$\dot{\theta}(t) = (u_1(t) - u_2(t)), \tag{10}$$

with $\theta(t) \in [-\pi, \pi]$, and the controls $u_1(t), u_2(t) \in [-1, 1]$.

The coordinates (x, y) are the position in the plane of the midpoint of the axle connecting the rear wheels, and θ denotes the heading angle measured from the x-axis. The controls u_1 and u_2 are the angular velocity of the right and left wheels, respectively. If the same velocity is applied to both wheels, the robot moves along a straight line (maximum forward velocity when $u_1 = u_2 = u_{max} = 1$). The robot can turn by choosing $u_1 \neq u_2$ (when $u_1 = -u_2 = 1$ the robot turns counter-clockwise around the midpoint of the axle).

The velocity vector is always orthogonal to the wheel axis. This is the non-slip or nonholonomic constraint

$$(\dot{x}, \dot{y})^T (\sin\theta, -\cos\theta) = 0. \tag{11}$$

Therefore, the vehicle model is a nonholonomic system. It is completely controllable but instantaneously it cannot move in certain directions. It is known that

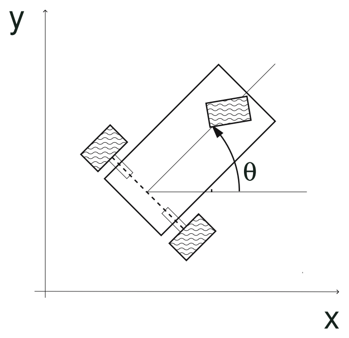


Fig. 3. A differential-drive mobile robot

this class of systems requires a nonlinear controller to stabilize it. Furthermore, the controller must allow discontinuous (or alternatively time-varying) feedback control laws. See [13] for a discussion on the control of nonholonomic systems, [3,23] for the need to use discontinuous feedbacks, and [8] for MPC of nonholonomic systems.

4.2 Linearized Vehicle Model for the Formation Controller

For the reasons explained above, to control the relative position between the vehicles in the formation, compensating for small deviations around a nominal trajectory, we might use linear control methods.

Consider that one of the vehicles is the reference vehicle. It might be the formation leader, or any of the vehicles in the formation, or we might even want to consider an additional nonexistent vehicle for modeling purposes. Consider that the formation moves at the nominal linear velocity $v_n = (u_{1n} + u_{2n})/2$, with $u_{1n} = u_{2n}$ being the nominal velocities of the wheels. Let v_l be the linear velocity added to the nominal linear velocity, and v_w the angular velocity. Assume that θ is small, so that $\cos\theta \simeq 1$ and $\sin\theta \simeq \theta$. We thus have the simplified model

$$\dot{x}(t) = (v_n(t) + v_l(t)) \tag{12}$$

$$\dot{y}(t) = (v_n(t) + v_l(t))\theta(t) \tag{13}$$

$$\dot{\theta}(t) = v_w(t). \tag{14}$$

We consider a linearized model for each of the differential drive mobile robots with the z_1 axis aligned with the velocity of the *reference vehicle* .

$$\begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \\ \dot{z}_3 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & v_n \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_l \\ v_w \end{bmatrix} \tag{15}$$

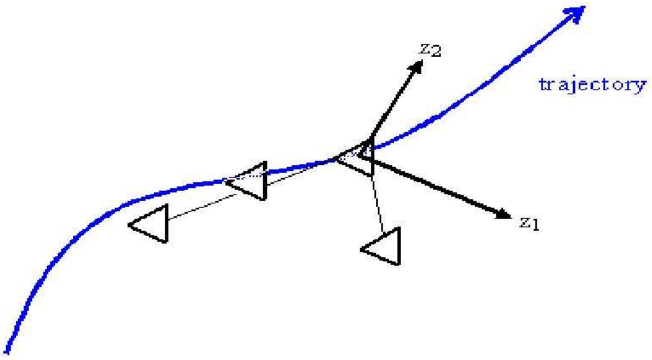


Fig. 4. Reference frame for the Linearized vehicle model aligned with the reference trajectory

For computation purposes, it is convenient to use a discrete-time model here. The discrete-time model, converted using zero order hold with sample time h, is

$$\begin{bmatrix} z_1(t+h) \\ z_2(t+h) \\ z_3(t+h) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & v_n h \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} z_1(t) \\ z_2(t) \\ z_3(t) \end{bmatrix} + \begin{bmatrix} h & 0 \\ 0 & 0 \\ 0 & h \end{bmatrix} \begin{bmatrix} v_l(t) \\ v_w(t) \end{bmatrix} \tag{16}$$

which we will simply denote as

$$\dot{z}(t+h) = Az(t) + Bv(t). \tag{17}$$

4.3 Formation Connections Model

Consider M vehicles and a *reference vehicle* (that might not exist) that follows exactly the trajectory predicted by the trajectory controller.

Consider that the $M+1$ vehicles are nodes of a directed graph $G = (V, E)$, which is a directed tree rooted at the reference vehicle. That is, all vertices are connected, there are no cycles, and all edges are directed in a path away from the root. Associate to each vertex i a triplet $z^i(t) = (z_1^i(t), z_2^i(t), z_3^i(t))$ with the relative position with respect to the reference vehicle at time t. Also, associate with each edge $(i,j) \in E$ a pair $\tilde{z}^{ij} = (\tilde{z}_1^{ij}, \tilde{z}_2^{ij})$ with the desired position in the plane of vehicle j with respect to vehicle i. The desired relative positions are defined a priori and define the geometry of the formation that we want to achieve. For node i we define its parent node $\mathcal{P}_i = j$ such that $(i,j) \in E$. See Figure 5.

The objective of the formation controller is to maintain the geometry of the formation, that is, the relative positions between the vehicles should be kept as close as possible to the desired relative positions. We define for each vehicle i with parent j a performance index that penalizes the difference between the desired and actual relative positions of the vehicles.

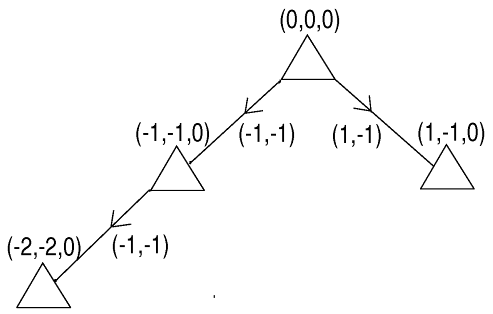


Fig. 5. A tree modelling the formation connections

$$\begin{aligned} J^i =& \|(\tilde{z}_1^{ij}, \tilde{z}_2^{ij}, 0) - (z^j(N) - z^i(N))\|_P^2 + \\ &+ \sum_{t=1}^{N-1} \|(\tilde{z}_1^{ij}, \tilde{z}_2^{ij}, 0) - (z^j(t) - z^i(t))\|_Q^2 + \|(v(t), w(t))\|_R^2 . \end{aligned} \tag{18}$$

5 The Controllers

5.1 Trajectory Controller

The controller used here follows closely the results in [9]. We use bang-bang feedback controls. That is, for each state, the corresponding control must be at one of the extreme values of its range. The exception is the target set Θ (here a small ball centered at the origin) where the control is chosen to be zero.

The feedbacks, when outside the target set, are defined by a switching surface $\sigma(x) = 0$. The control will attain its maximum or minimum value depending on which side of the surface the state is. More precisely, for each component $j = 1, \ldots, m$ of the control vector, define

$$k_j(x) = \begin{cases} 0 & \text{if } x \in \Theta, \\ u_j^{max} & \text{if } \sigma_j(x) \geq 0, \\ u_j^{min} & \text{if } \sigma_j(x) < 0. \end{cases} \tag{19}$$

The function σ_j is a component of the vector function σ, and is associated with the switching surface $\sigma_j(x) = 0$ which divides the state-space in two.

These surfaces must be parameterized in some way to be chosen in an optimization problem. Therefore, we define them to have a fixed part σ^{aux}, possibly nonlinear, and a variable part σ^{Λ} which is affine and defined by the parameter matrix Λ.

$$\sigma(x) = \sigma^{aux}(x) + \sigma^{\Lambda}(x). \tag{20}$$

For each component $j = 1, 2, \ldots, m$, the equation $\sigma_j^{\Lambda} = 0$ is the equation of a hyperplane which is defined by $n + 1$ parameters as

$$\sigma_j^{\Lambda}(x) := \lambda_{j,0} + \lambda_{j,1}\, x_1 + \ldots + \lambda_{j,n}\, x_n . \tag{21}$$

The half-spaces $\sigma_j^\Lambda(x) \geq 0$ and $\sigma_j^\Lambda(x) < 0$ are not affected by multiplying all parameters by a positive scalar, therefore we can fix one parameter, say $\lambda_{j,0}$, to be in $\{-1, 0, 1\}$. In total, for all components of the control vector, there will be $m \times (n+1)$ parameters to choose from. By selecting the parameter matrix

$$\Lambda := \begin{bmatrix} \lambda_{1,0} \cdots \lambda_{1,n} \\ \cdots \\ \lambda_{m,0} \cdots \lambda_{m,n} \end{bmatrix}, \tag{22}$$

we define the function

$$\sigma^\Lambda(x) = \Lambda \begin{bmatrix} 1 \\ x \end{bmatrix}, \tag{23}$$

and therefore we define the switching function σ by (20) and we define the feedback law k^Λ by (19). Each component of the feedback law can be described as

$$k_j^\Lambda(x) = \begin{cases} 0 & \text{if } x \in \Theta, \\ u_j^{max} & \text{if } \left[\sigma^{aux}(x) + \Lambda \begin{bmatrix} 1 \\ x \end{bmatrix}\right]_j \geq 0, \\ u_j^{min} & \text{if } \left[\sigma^{aux}(x) + \Lambda \begin{bmatrix} 1 \\ x \end{bmatrix}\right]_j < 0. \end{cases} \tag{24}$$

In our example the switching surfaces, for each control component, are planes in the state-space $\mathbb{R}^3$ and can be described by 4 real parameters. The surface, and therefore the feedback, are defined by a parameter matrix $\Lambda \in \mathbb{R}^{2\times 4}$.

The feedbacks are obtained by solving, in a receding horizon strategy, the following optimal control problems with respect to matrices $\Lambda \in \mathbb{R}^{2\times 4}$.

Minimize$_{\Lambda_1,\ldots,\Lambda \quad \in \mathbb{R}^{2\times 4}}$

$$\int_t^{t+T} L(x(s), u(s))ds + W(x(t+T_p)) \tag{25}$$

subject to (26)

$$\begin{aligned} & x(t) = x_t, \\ & \dot{x}(s) = f(x(s), u(s)) && \text{a.e. } s \in [t, t+T_p], && (27) \\ & x(s) \in X && \text{for all } s \in [t, t+T_p], \\ & x(t+T_p) \in S, && && (28) \end{aligned}$$

where

$$\begin{aligned} u(s) &= k^\Lambda\ (x(\lfloor s \rfloor_\pi)) && s \in [t+(i-1)\delta, t+i\delta), \quad i = 1, \ldots, N_c, && (29) \\ u(s) &= k^{aux}(x(\lfloor s \rfloor_\pi)) && s \in [t+(i-1)\delta, t+i\delta), \quad i = N_c+1, \ldots, N_p. && (30) \end{aligned}$$

In this optimal control problem, the control horizon T_c and prediction horizon T_p satisfy $T_c = N_c\delta$ and $T_p = N_p\delta$ with $N_c, N_P \in \mathbb{N}$ and $N_c \leq N_p$.

The guarantee of stability of the resulting closed loop system can be given by a choice of design parameters satisfying a sufficient stability condition. The following set of design parameters [9] guarantees stability:

$$L(x,y,\theta) = x^2 + y^2 + \theta^2, \tag{31}$$

$$W(x,y,\theta) = \frac{1}{3}(r^3 + |\theta|^3) + r\theta^2 \quad \text{with } r = \sqrt{x^2+y^2}, \tag{32}$$

$$T_p = \frac{2\pi}{3}, \tag{33}$$

$$\begin{aligned} S := \{(x,y,\theta) \in \mathbb{R}^2 \times [-\pi,\pi] : \phi_m(x,y) \le \theta \le \phi_M(x,y) \vee \\ \vee\ (x,y,\theta) \in \Theta\ \vee\ (x,y) = (0,0)\}. \end{aligned} \tag{34}$$

5.2 Formation Controller

For each vehicle i, with parent j, compute the control solving the constrained linear quadratic optimal control problem:

Minimize over sequences $\{v_1, \ldots, v_{N-1}\}$

$$\begin{aligned} J =& \|(\tilde{z}_1^{ij}, \tilde{z}_2^{ij}, 0) - (z^j(N) - z(N))\|_P^2 + \\ &+ \sum_{t=1}^{N-1} \|(\tilde{z}_1^{ij}, \tilde{z}_2^{ij}, 0) - (z^j(t) - z(t))\|_Q^2 + \|v(t)\|_R^2 \end{aligned} \tag{35}$$

subject to

$$\begin{aligned} & z(t+h) = Az(t) + Bv(t) && t = 0, \ldots, N-1, \\ & z(0) = z_0, \\ & v_{min} \le v(t) \le v_{max} && t = 0, \ldots, N-1, && (36) \\ & (\underline{z}_1^{ij}, \underline{z}_2^{ij}, 0) \le z(t) - z^j(t) \le (\bar{z}_1^{ij}, \bar{z}_2^{ij}, 0) && t = 1, \ldots, N, && (37) \\ & Dz(t) \le d && t = 1, \ldots, N. && (38) \end{aligned}$$

Here, constraints (36) are limits on the inputs. Constraints (37) set a maximum and minimum distance to the parent vehicle, and inequalities (38) are general constraints to accommodate, for example, forbidden zones of the state-space.

The matrices involved in the performance index are chosen to satisfy $Q = Q' \succeq 0, R = R' \succ 0$, and P solving the Lyapunov equation $P = A'PA + Q$. This strategy has, for each vehicle and in the conditions stated, guaranteed exponential stability [2,19].

One of the major advantages of this controller is the fact that the feedback laws obtained from solving the linear quadratic regulator with constraints are

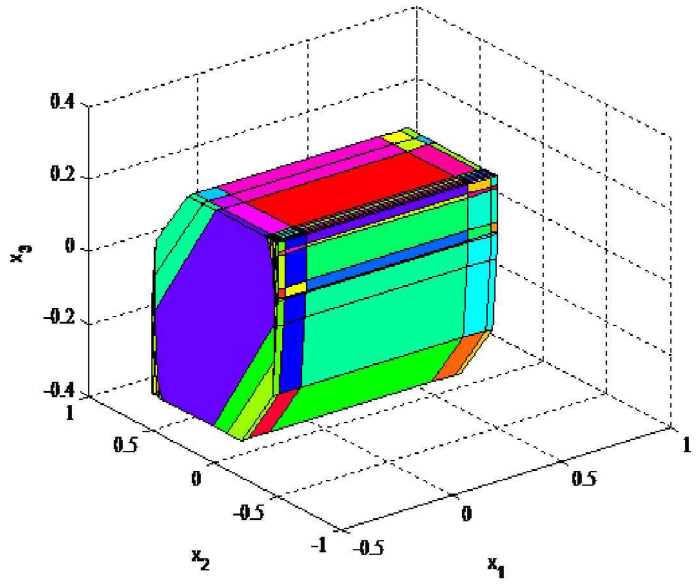


Fig. 6. State-space regions for the reference vehicle PWA control law

continuous piecewise affine (PWA) functions [2], i.e. , for a certain region of the state-space $\mathcal{R}_k$ (which is a polytope) the control law is affine

$$u(t) = F_k z(t) + G_k \qquad\qquad \text{if } z(t) \in \mathcal{R}_k. \tag{39}$$

In Figure 6 we can see the different state-space regions (195 polytopes) corresponding to different affine control laws for the reference vehicle.

This way, the parameters of the PWA feedback (matrices F_k and G_k for each region $\mathcal{R}_k$) can be determined explicitly a priori, off-line, by multi-parametric programming (e.g., using MPT - Multi Parametric Matlab Toolbox [14]). Each vehicle just has to store the parameters of the PWA feedback function. No optimization nor other complex computations are involved in real-time. Only lookup table operations are needed for each vehicle.

The stability of the whole formation is easy to establish. This is because, with the strategy above, the trajectory of each vehicle is exponentially stable with respect the desired relative position to its parent vehicle. As there is exactly one path from each vehicle to the reference vehicle, stability of any vehicle easily follows recursively.

If more general graphs are allowed, comprising not only trees but also admitting loops, the stability analysis is considerably more complex. For results on stability considering more general graphs see [5].

5.3 Integration of the Two Control Layers

In each vehicle, the control given by the Trajectory Controller *plus* the control given by the Formation Controller are applied. Therefore, we have to consider control limits in each layer in such a way that the physical limits are respected. The Trajectory Controller (bang-bang feedback) can only use part of the limit (70% in our simulations) leaving the rest to the Formation Controller.

For the Trajectory Controller, the bang-bang feedback is computed centrally and then the switching surface parameters (a $2{\times}4$ matrix per sampling period) are communicated to each vehicle. The frequency of update might be relatively low, and is dictated by the information from the outside (the formation) world of new obstacles in the trajectory, possibly identified by the sensors, that might alter the main path.

For the Formation Controller, the PWA feedback law is computed a priori, off-line, and implemented in each vehicle.

6 Conclusions

Model Predictive Control was shown to be an adequate tool for control of vehicles in a formation: it deals explicitly and effectively with the constraints that are an important problem feature; recent results on parameterized approaches to the optimal control problems allow addressing fast systems as well as allowing efficient implementations.

Although several optimization problems have to be solved, none need to be solved in real time by each of the vehicles. Part of the problem can be solved a priori, off-line, and an explicit control law can be implemented in each vehicle as a lookup table; the other part can be solved centrally and only a few parameters (a $2{\times}4$ matrix) need to be transmitted to all vehicles in each sampling period.

References

1. Alamir, M.: Stabilization of Nonlinear Systems Using Receding-Horizon Control Schemes: A Parameterized Approach for Fast System. Lecture Notes in Control and Information Sciences. Springer, London (2006)
2. Bemporad, A., Morari, M., Dua, V., Pistikopoulos, E.N.: The explicit linear quadratic regulator for constrained systems. Automatica 38, 3–20 (2002)
3. Brockett, R.W.: Asymptotic stability and feedback stabilization. In: Brockett, R.W., Millman, R.S., Sussmann, H.S. (eds.) Differential Geometric Control Theory, pp. 181–191. Birkhouser, Boston (1983)
4. Chen, H., Allgöwer, F.: Nonlinear model predictive control schemes with guaranteed stability. In: Berber, R., Kravaris, C. (eds.) Nonlinear Model Based Process Control. Kluwer, Dordrecht (1998)
5. Dunbar, W.B., Murray, R.M.: Distributed receding horizon control for multi-vehicle formation stabilization. Automatica 42, 549–558 (2006)
6. Findeisen, R., Imsland, L., Allgöwer, F., Foss, B.: State and output feedback nonlinear model predictive control: an overview. European Journal of Control 9, 190–206 (2003)
7. Fontes, F.A.C.C.: A general framework to design stabilizing nonlinear model predictive controllers. Systems & Control Letters 42, 127–143 (2001)

8. Fontes, F.A.C.C.: Discontinuous feedbacks, discontinuous optimal controls, and continuous-time model predictive control. International Journal of Robust and Nonlinear Control 13(3–4), 191–209 (2003)
9. Fontes, F.A.C.C., Magni, L.: Min-max model predictive control of nonlinear systems using discontinuous feedbacks. IEEE Transactions on Automatic Control 48, 1750–1755 (2003)
10. Fontes, F.A.C.C., Magni, L., Gyurkovics, E.: Sampled-data model predictive control for nonlinear time-varying systems: Stability and robustness. In: Allgower, F., Findeisen, R., Biegler, L. (eds.) Assessment and Future Directions of Nonlinear Model Predictive Control. Lecture Notes in Control and Information Systems, vol. 358, pp. 115–129. Springer, Heidelberg (2007)
11. Ghabcheloo, R., Pascoal, A., Silvestre, C., Kaminer, I.: Non-linear co-ordinated path following control of multiple wheeled robots with bidirectional communication constraints. International Journal of Adaptive Control and Signal Processing 21, 133–157 (2007)
12. Goodwin, G.C., Seron, M.M., De Doná, J.A.: Constrained Control and Estimation: An Optimisation Approach. Springer, London (2004)
13. Kolmanovsky, I., McClamroch, N.H.: Developments in nonholonomic control problems. IEEE Control Systems, 20–36 (December 1995)
14. Kvasnica, M., Grieder, P., Baotic, M., Christophersen, F.J.: MPT - Multiparametric Toolbox. Technical report, ETH - Swiss Federal Institute of Technology, Zurich (2006)
15. Maciejowski, J.M.: Predictive Control with Constraints. Prentice Hall, Harlow (2002)
16. Magni, L., Scattolini, R.: Model predictive control of continuous-time nonlinear systems with piecewise constant control. IEEE Transactions on Automatic Control 49, 900–906 (2004)
17. Magni, L., Sepulchre, R.: Stability margins of nonlinear receding horizon control via inverse optimality. Systems and Control Letters 32, 241–245 (1997)
18. Mayne, D.Q.: Control of constrained dynamic systems. European Journal of Control; Special issue: Fundamental Issues in Control 7, 87–99 (2001)
19. Mayne, D.Q., Rawlings, J.B., Rao, C.V., Scokaert, P.O.M.: Constrained model predictive control: Stability and optimality. Automatica 36, 789–814 (2000)
20. Murray, R.M.: Recent research in cooperative control of multi-vehicle systems. Journal of Dynamic Systems, Measurement and Control 129, 571–583 (2007)
21. Qin, S., Badgwell, T.: An overview of industrial model predictive control technology. Control Engineering Practice 11, 733–764 (2003)
22. Raimondo, D.M., Magni, L., Scattolini, R.: Decentralized MPC of nonlinear systems: An input-to-state stability approach. International Journal Of Robust And Nonlinear Control 17, 1651–1667 (2007)
23. Sontag, E.D., Sussman, H.J.: Remarks on continuous feedback. In: IEEE Conference on Decision and Control, Albuquerque, pp. 916–921 (1980)

Application of Monkey Search Meta-heuristic to Solving Instances of the Multidimensional Assignment Problem

Alla R. Kammerdiner, Antonio Mucherino, and Panos M. Pardalos

University of Florida, Gainesville, FL

Abstract. This study applies a novel metaheuristic approach called Monkey Search for solving instances of the Multidimensional Assignment Problem (MAP). Monkey Search is a global optimization technique inspired by the behavior of a monkey climbing trees. The combinatorial formulation of the MAP in order to incorporate the matrix representation of feasible solutions is considered. The developed software procedure is tested on randomly generated instances of the MAP, and the results of the numerical experiments are presented in this study.

1 Introduction

Many practical problems can be formally described as an instance of the Multidimensional Assignment problem (MAP). In particular, the MAP arises from data association problems arising in multiple target tracking and sensor fusion [20]. It can also be found in air traffic control, satellite launching, surveillance, dynamic facility location, capital investment, and even in tracking the motion of elementary particles [2,18,21].

Multidimensional Assignment problems are characterized by a very large number of local minima [9]. Furthermore, the number of feasible solutions grows exponentially with an increase in problem parameters [9,11]. This inherent complexity of the MAP creates serious difficulties in solving MAP instances of high dimensionality. Most solution methods applied to the MAP are enumerative in nature, and rely on some kind of local neighborhood search. Therefore, due to the large sizes of the problems with even moderate dimensionality and cardinality parameters, exact approaches are rather time consuming. This suggests investigating suboptimal approaches, which could allow one to efficiently explore the structure of the problem surfaces in order to find feasible solutions whose value is sufficiently close to optimal.

The Greedy Randomized Adaptive Search Procedure (GRASP) is one of the few heuristic approaches applied to the MAP [1, 13, 16]. Other heuristic approaches employed to solve the MAP includes Simulated Annealing [4] and Tabu Search [14].

In this study, we apply a heuristic called *Monkey Search*, which is a novel metaheuristic method for global optimization recently proposed by Seref et al. [15,23]. Similar to other metaheuristic algorithms such as ant colony optimization, this method is motivated by animal behavior. Loosely speaking, Monkey Search imitates the behavior of a monkey climbing trees in its search for food [15]. The procedure builds trees of solutions in such a way that two new branches stem from the current feasible solution to the neighboring solutions. When the monkey discovers a better solution, it remembers it. Later, on its way down, the monkey marks the corresponding branches, and then uses these marks for deciding what branches to climb up again. This marking strategy reflects the monkey's intentions to focus on a part of a tree where it has already found some good solutions.

The Monkey Search procedure was tested on the global optimization problem of finding stable conformations of clusters of atoms regulated by the Lennard Jones potential energy and by the Morse potential energy. The experiments showed Monkey Search to be more effective than some other metaheuristic methods such as Simulated Annealing and Harmony Search [15,24].

One of the advantages of Monkey Search is its flexibility, which arises naturally from the definition of the transformations creating the neighboring solutions. More specifically, the solution transformations builds on the ideas and strategies borrowed from other heuristics and optimization approaches. Monkey Search can incorporate these diverse techniques in a unique way.

This paper is organized in the following manner. Section 2 presents a formulation of the MAP as a combinatorial optimization problem, and discusses some relevant previous results. Section 3 introduces the Monkey Search metaheuristic. We utilized the Monkey Search method to solve randomly generated MAP instances. The results of our experiments are reported in Section 4. Finally, the last section provides some conclusions and future research directions.

2 The Multidimensional Assignment Problem

The Multidimensional Assignment Problem (MAP) represents a special type of assignment problem, and can be viewed as a generalization of the well-known Linear Assignment Problem (LAP). In fact, the LAP investigates a question of finding a one-to-one correspondence between *two* sets of n items, so that the total cost of such an assignment is minimized. Although the MAP is also about finding an exact correspondence with minimum total cost, the correspondence in the MAP case must be found among *three or more* different sets of the same cardinality.

The MAP was first introduced by Pierskalla as a three-dimensional extension of the LAP [17], and generalized to the assignment in n-dimensions [19]. Although Pierskalla formulated the MAP as a zero-one integer programming problem [19], several alternative formulations have been given to the MAP. One of such formulations is based on graph theory, where the problem is described from the standpoint of finding a partition of the vertex set of a d-partite graph into n pairwise disjoint cliques of minimum cost [3]. Another formulation of MAP is as a combinatorial optimization problem [12].

Here we consider the MAP formulation from the combinatorial optimization point of view as follows:

$$\text{minimize} \sum_{1 \le i \le n} c_{i\pi_1(i)\dots\pi_{\ -1}(i)}, \text{ subject to } \pi_1, \dots \pi_{d-1} \in \Pi^n, \tag{1}$$

where d is a dimensionality parameter of the MAP, n is a MAP parameter signifying the cardinality of each set, $c_{j_1 j_2 \dots j}$ (with $1 \le j_k \le n$, and $1 \le k \le d$) denote the assignment cost coefficients, and Π^n is the set of all possible permutations of elements in the set $\{1, 2, \dots, n\}$.

Simply put, a given MAP is specified by a d-dimensional cubic matrix of cost coefficients of the size n^d. Thus, solving the MAP means finding such a permutation of the rows and columns of the costs matrix that minimizes the sum of the diagonal elements.

One of the advantages of using the combinatorial formulation (1) of the MAP is that it admits a clear and convenient representation of each feasible solution of the MAP by the following $n \times d$ matrix:

$$\begin{pmatrix} \pi_1(1) & \pi_2(1) & \dots & \pi_d(1) \\ \pi_1(2) & \pi_2(2) & \dots & \pi_d(2) \\ \vdots & \vdots & \ddots & \vdots \\ \pi_1(n) & \pi_2(n) & \dots & \pi_d(n) \end{pmatrix} = (\pi_1 \ \pi_2 \ \dots \ \pi_d)\,, \tag{2}$$

where the d columns $\pi_i = (\pi_i(1) \ \pi_i(2) \ \dots \ \pi_i(n))^\top$, $1 \le i \le d$, of the matrix are permutations from Π^n.

For a given matrix representation (2) of a feasible solution, the associated solution cost is

$$z = c_{\pi_1(1)\pi_2(1)\dots\pi\ \ (1)} + c_{\pi_1(2)\pi_2(2)\dots\pi\ \ (2)} + \dots + c_{\pi_1(n)\pi_2(n)\dots\pi\ \ (n)}. \tag{3}$$

It is worth noting that the solution is invariant under any permutation of rows in the solution matrix. This follows from the fact the individual summands $c_{\pi_1(i)\pi_2(i)\dots\pi\ \ (i)}$ and $c_{\pi_1(j)\pi_2(j)\dots\pi\ \ (j)}$, $i \ne j$, in the expression (3) can be interchanged without affecting the solution value z.

However, we must point out that the matrix representation (2) is not unique. One of the ways to guarantee a one-to-one correspondence between the feasible solutions and their respective matrix representation is by setting the first column to be an identity permutation $\imath$. As a matter of fact, it is sufficient to specify a permutation $\pi_j \in \Pi^n$ of arbitrary fixed column j of the matrix representation of feasible solutions in order to obtain a one-to-one correspondence between all feasible solutions of the MAP and their representations as matrices of permutations [11].

For additional information about various formulations, solution methods, special cases and applications of the MAP, the interested reader is referred to [12].

3 The Monkey Search

Monkey Search is a meta-heuristic approach for global optimization. Its basic idea was originally proposed in [23], and successively elaborated and implemented in [15]. Monkey Search resembles the behavior of a monkey climbing trees in its search for food. The main assumption in this approach is that a monkey is able to survive in a jungle of trees because it is able to focus its food searches where it previously found good food. When the monkey climbs up a tree for the first time, it can only choose the branches of the tree in a random way, because it does not have any previous experience on that tree. However, when the monkey climbs up the tree again, it tries to follow the paths that led it to good food, allowing the monkey to discover a set of connected branches of the tree in which there are good food resources.

In Monkey Search, food is represented by suitable solutions of a global optimization problem to be solved. These solutions lie on the top of the branches of virtual trees that a virtual monkey climbs. The quality of these solutions is evaluated through the value of the objective function which has to be optimized. A branch represents a perturbation that creates the solution at the top of the branch when it is applied to the solution at the root of the branch. In Monkey Search, the trees are not pre-existing, instead they are built as the monkey climbs. At the start, the monkey is positioned at the root of the tree, and it has a solution associated to the root. Since this is the first time the monkey climbs this tree, the tree actually does not exist yet, and therefore its branches have to be created. Here we assume that all trees have a binary structure, i.e., unless we are at a tree leaf, only two branches are attached to a given solution. Therefore, two new branches are randomly built: two possible perturbations are applied to the current solution, and two new solutions are generated and placed at the top of these two branches. At this point, the monkey has to choose between which of the two branches to climb next. Since it does not have any previous experience on this tree, it chooses the branch in a random way. Once the chosen branch is climbed, the monkey is situated at the top of that branch. As before, the monkey needs to climb up new branches, and hence two other branches are created that have as root the solution in which the monkey is currently climbing. This procedure continues until the monkey reaches the top of the tree. Every time the monkey climbs a branch up, it updates its current solution. Moreover, it keeps in mind the best solution found on its way up.

When the monkey has climbed the tree all the way up to its top, it needs to climb down. The ability of the monkey to remember previously discovered branches is simulated in the Monkey Search procedure as follows. Every time the monkey climbs down a branch, it marks that branch with the best solution which can be found climbing that branch up. This is done so that the monkey can exploit the information later, when it climbs up the tree again. On its way down, the monkey can either return to the root of the tree, and restart climbing up from there, or it can decide to restart climbing up from a certain branch. This is controlled by a random mechanism so that the respective parameter is adjusted in order to force the monkey to come back toward the root of the tree with an appropriate probability.

When the monkey decides to restart climbing up, it encounters some previously visited branches on its way up. It then climbs these branches again. The monkey chooses between one of the two tree branches based on the marks it left before. Naturally, the monkey has greater probability of choosing a branch leading to better solutions, and this probability increases with the quality of the solution the branch leads to. Since this mechanism is based on probabilities, the monkey can choose to climb a branch that was rejected previously. When this happens, the monkey finds new previously unknown branches on its way up. From this point on, the monkey starts climbing up the tree as it did at the start: two new branches are created by applying two random perturbations to the current solution, and one of those is chosen in a totally random fashion. A flowchart describing the general behavior of the monkey climbing a tree is given in Figure 1.

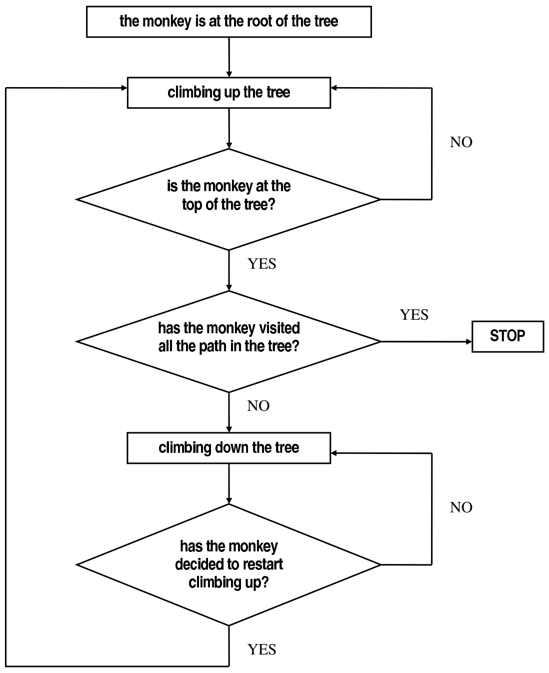

the monkey is at the root of the tree
climbing up the tree
is the monkey at the top of the tree?
NO
YES
has the monkey visited all the path in the tree?
YES
STOP
NO
climbing down the tree
has the monkey decided to restart climbing up?
NO
YES

Fig. 1. The behavior of a monkey climbing a tree

Figure 2 shows the possible behavior for a monkey during the Monkey Search. In the tree in Figure 2(a), the monkey has climbed from the root of the tree to its top. The path in bold represents the set of branches that have been chosen by the monkey. Whereas, all the other branches have not been climbed. When the monkey reached the top of the tree, it started to come back branch by branch,

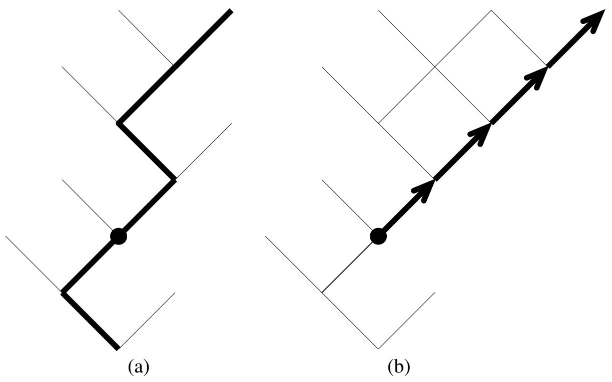


Fig. 2. Two phases of the behavior of a monkey climbing a tree: (a) the monkey climbs the tree from the root to the top, then it climbs back and stops in the location marked with the bold circle; (b) the monkey restarts climbing up: in this case, the monkey climbs some previously explored branches and also some newly generated branches

marking all the branches it passed. In this example, we suppose that the monkey decided to restart climbing up from the solution marked with a bold circle in the figure. The new path the monkey follows is marked in bold in Figure 2(b). The first branch the monkey climbs up has been previously climbed. The second branch is still unexplored: from that branch on, the monkey needs to climb new branches of the tree. In the process of climbing up and down, the best solution ever discovered is kept updated and saved. When all the paths in the tree have been explored, the monkey stops climbing the tree, and the obtained result corresponds to the best solution found during the whole search.

The Monkey Search procedure is based on a set of parameters that influence the convergence of the algorithm. The height of the trees is the total number of branches that the monkey can climb from the root to the top. The number of paths the tree contains is represented by the number of times the monkey starts climbing up the same tree. Two other parameters deal with the memory of the Monkey Search heuristic. In order to avoid local minima, a predetermined number of best solutions found on each individual tree is kept in memory. In fact, every time the monkey stops climbing a tree because it has reached the allowed total number of paths, it starts climbing a new tree from a different root solution. The best solutions are kept in memory, so that the monkey can select either one of the best solutions or a combination of them as a root of a new tree. The memory size is, therefore, a parameter of the method, as is the number of trees that the monkey starts climbing from a random solution. This is done at the beginning ("warm-up") stage of the algorithm in order to spread the search in different areas of the domain of the objective function.

The Monkey Search procedure stops when all the solutions in memory are sufficiently close to one another. For this reason, the cardinality of the best solutions set is one of the most important Monkey Search parameters. A small cardinality may, indeed, cause the method to stop in a local optima. On the other hand, although keeping too many solutions in memory allows the possibility of the search finding the global optimum, it may be too expensive computationally. As often is the case when dealing with meta-heuristic methods, a trade-off needs to be found, and this parameter can be tuned by classic trial and error procedure on the particular problem to be solved.

The perturbations in the Monkey Search play an important role. They can be totally random, as is done in the standard Simulated Annealing algorithms [10], but they can also be customized and tailored to the problem to solve. The idea is to exploit strategies borrowed from other meta-heuristic searches, such as Genetic Algorithms [8], Harmony Search [6], Tabu Search [7], etc., and/or develop perturbations which are inspired by the problem itself. If a certain set of perturbations is defined, then the perturbation to be applied for generating a new branch can be chosen from the set with a uniform probability. Another way of selecting the transformation for generating a new solution is by assigning higher probabilities of being selected to those perturbations that, so far, have been the most successful. Monkey Search has been shown more efficient than other meta-heuristic searches on difficult problems. Details about the Monkey Search and the comparisons to other meta-heuristics can be found in [15,24].

4 Experimental Results

This section discusses some technical aspects of solving MAP instances via the Monkey Search algorithm, including the perturbations applied to feasible solutions.

Since each feasible solution of the MAP admits a matrix representation (2), we utilized this representation for constructing the perturbations. In order to obtain new feasible solutions from the current solution, we employed basic random transformations, as in the standard Simulated Annealing algorithm [10]. Particularly, a matrix representation (2) of the current solution is transformed in the following fashion. First, one of the columns, say j, is selected at random according to a discrete uniform distribution. Next, two elements of the j-th column, say $\pi_j(i_1)$ and $\pi_j(i_2)$, $i_1 < i_2$, are selected at random, and interchanged. In other words, by applying this perturbation, the j-th column

$$(\pi_j(1) \dots \pi_j(i_1) \dots \pi_j(i_2) \dots \pi_j(n))^\top$$

in the matrix representation of the current solution is replaced by the corresponding permuted column

$$(\pi_j(1) \dots \pi_j(i_2) \dots \pi_j(i_1) \dots \pi_j(n))^\top$$

in the new solution. In addition, we considered an extension of the transformation by allowing several interchanges to be performed at once. In such transformations, we randomly selected multiple columns whose elements were interchanged. It is noteworthy to state that the transformations inspired by other heuristic approaches (e.g., Genetic Algorithm [8], and Tabu Search [7]) may also be implemented.

We ran some numerical experiments to test the effectiveness of Monkey Search in solving MAP instances by using the above perturbations for generating new feasible solutions.

The experiments are performed on a computer with an Intel CPU T2500 at 2 GHz, and 1 GB RAM. The algorithm was implemented in the C++ programming language using *CodeBlocks* compiler.

As noted in [22], randomly generated assignment problems has been used extensively in prior studies to assess the effectiveness of heuristic methods. In our study, the Monkey Search approach is tested on randomly generated instances of the MAP with parameters $n = 4$ and $d = 5$. Each MAP instance is obtained by generating a set of independent random assignment cost coefficients according to a continuous uniform distribution on $[0, 1]$. Despite the moderate values of n and d, the cardinality of the feasible solutions set for such MAP instances is 331,776, since the number of feasible solutions is given by $N = (n!)^{d-1}$. Moreover, the number of local optima is also significant, as can be seen from Table 1.

Table 1. Number of local minima for the MAP instances with $n = 4$ and $d = 5$

statistic	value	normalized w.r.t. N
minimum	958	0.29%
mode	1,207	0.36%
average	1,221.084	0.37%
maximum	1,460	0.44%

The parameters for the Monkey Search are set as follows.

- The maximum number of trees allowed equals 40;
- The maximum tree height is 70;
- The maximum number of best solutions kept in the memory is limited to 10; and
- The probability of the monkey to restart climbing up the tree when it is currently climbing down the tree is set to 0.94.

To investigate the role that is played by the number of interchanged pairs in the Monkey Search procedure, we tested random perturbations with different number of the pairs of interchanged elements. The number of interchanges performed in each transformation varied between 1 and 4. 500 MAP instances (with $n = 4$ and $d = 5$ and independent uniformly distributed cost coefficients) were generated using different random seeds. Four different versions of the Monkey Search procedure based on the perturbations with a fixed number of interchanged pairs ($s = 1, 2, 3, 4$) were tested. Each MAP instance was solved by four modifications of the Monkey Search procedure, and by enumeration. The latter approach was used to find the global optimum solution for each random instance. The execution times for the enumeration and the Monkey Search procedures, as well as the objective function values of the MAP global optimum and the solutions found by each Monkey Search version were recorded.

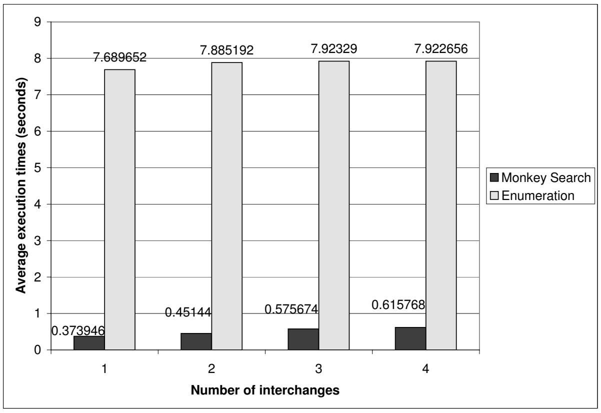


Fig. 3. The average execution times for the Monkey Search versions based on 1, 2, 3, and 4 interchanges in one perturbation

The execution times for the enumeration algorithm are compared to the average run times for the modifications of the Monkey Search algorithm based on the perturbations with one, two, three and four pairs of interchanged indices. The results are illustrated in Figure 3.

Notice that the fastest average run time of 0.373946 seconds was obtained by solving the MAP instances using the Monkey Search version, where a new feasible solution was generated by only a single interchange of two elements in each perturbation. The Monkey Search implementations that required additional interchanges were slower, with four-interchange version being the slowest (0.615768 sec). On the other hand, all the Monkey Search modifications were shown to run significantly faster (over 12 times) than the enumeration procedure, which took approximately 7.8551975 seconds.

The average objective values of the solutions found by all four versions of the Monkey Search procedure are compared to the average global optimum obtained via enumeration. The results are presented in Figure 4. The figure shows that the Monkey Search implementation, which incorporates the perturbation with a single interchange, as well as the implementation that transforms a current feasible solution by interchanging a pair three times, are able to find a solution that is on average closer to the MAP global optimum, than the solutions obtained by the two-interchanges and the four-interchanges Monkey Search procedures.

The last observation is also supported by the histograms of the normalized differences between the global optimum of a random MAP instance and the corresponding solution found by each Monkey Search version. Each difference between the Monkey Search solution value and the respective global value is normalized with respect to the global value of a given MAP. Hence, each normalized difference of the objective values is a percentage of the global optimum. The corresponding histograms are derived for all four Monkey Search implementations, and presented in Figure 5. The frequency for the differences between

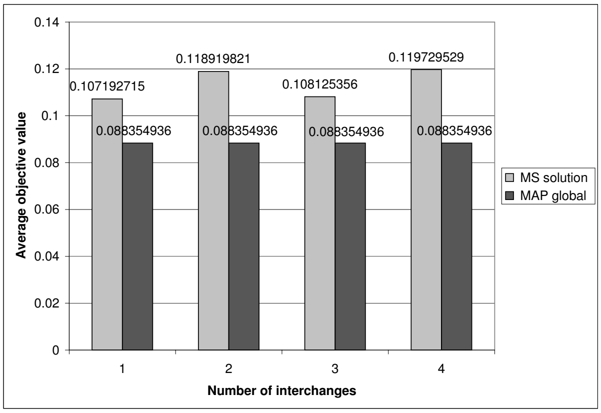


Fig. 4. The average objective values for the MAP global and the Monkey Search solution

the objective values of Monkey Search solutions and the global optimum normalized with respect to the global for the procedures with one-, two-, three-, and four-interchanges are presented in Table 2. The frequency values are given as the percentage of the total number of MAP instances. The column of bins lists

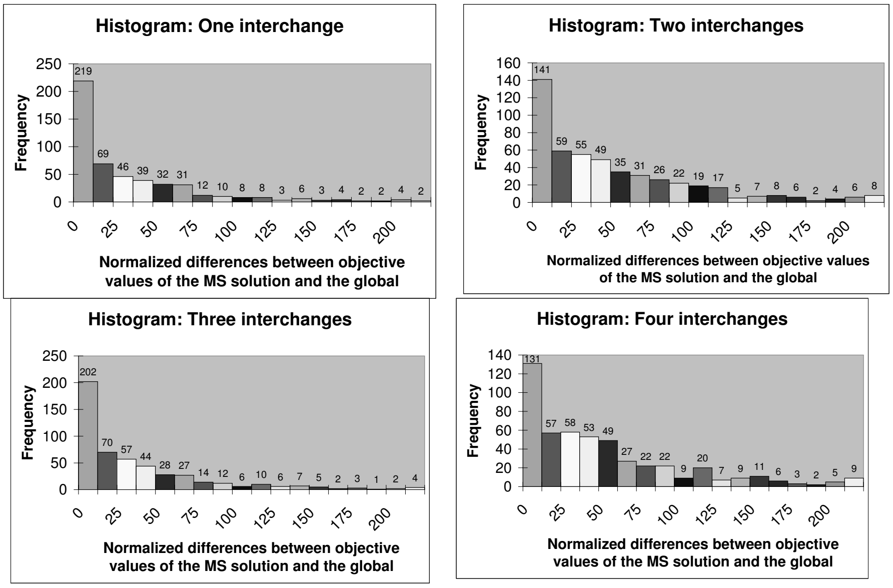


Fig. 5. Histograms of the normalized differences between the Monkey Search solution and the global optimum (one interchange, two interchanges, three interchanges, and four interchanges)

Table 2. Frequency of the normalized differences between the objective values of Monkey Search solutions and the global optimum for the procedures with one-, two-, three-, and four-interchanges

Bin of Norm.	Frequency			
Differences	1-interchange	2-interchanges	3-interchanges	4-interchanges
0	43.80%	28.20%	40.40%	26.20%
12.5	13.80%	11.80%	14.00%	11.40%
25	9.20%	11.00%	11.40%	11.60%
37.5	7.80%	9.80%	8.80%	10.60%
50	6.40%	7.00%	5.60%	9.80%
62.5	6.20%	6.20%	5.40%	5.40%
75	2.40%	5.20%	2.80%	4.40%
87.5	2.00%	4.40%	2.40%	4.40%
100	1.60%	3.80%	1.20%	1.80%
112.5	1.60%	3.40%	2.00%	4.00%
125	0.60%	1.00%	1.20%	1.40%
137.5	1.20%	1.40%	1.40%	1.80%
150	0.60%	1.60%	1.00%	2.20%
162.5	0.80%	1.20%	0.40%	1.20%
175	0.40%	0.40%	0.60%	0.60%
187.5	0.40%	0.80%	0.20%	0.40%
200	0.80%	1.20%	0.40%	1.00%
More	0.40%	1.60%	0.80%	1.80%

the upper bound values in the range of the differences normalized with respect to the MAP global values. From Table 2, it is clear that the single interchange Monkey Search procedure is able to find the best solutions of the MAP in highest percentage of instances.

5 Conclusion

In this chapter, a novel metaheuristic approach called the Monkey Search was applied to solving instances of general Multidimensional Assignment Problem. The constructed transformation of feasible solutions are based on the convenient matrix representation of the MAP, which follows from the MAP formulation as a combinatorial optimization problem. Transformations with several interchanged pairs of elements in the column were considered. Our numerical experiments indicate that for solving the MAP instances, the Monkey Search procedure appears to be significantly faster than the total enumeration approach. Furthermore, our numerical analysis of the Monkey Search algorithms, which incorporate the perturbations characterized by different numbers of interchanged pairs, has shown that the Monkey Search approach based on the single-interchange transformations is not only the fastest among all four implementations, but it is also more likely to find solutions close to the global optimum. Although the procedure based on the one-interchange transformations and the one based on three-interchanges perturbation have comparable likelihood of terminating with solutions that are closer to the global optimum of a given MAP instance. The execution time of the three-interchanges version is significantly longer than the single-interchange Monkey Search.

We continue our investigation of the Monkey Search approach for solving the MAP, by analyzing the performance of this new heuristic on different MAP instances. In particular, this includes considering additional randomly generated instances, and using different MAP parameters n and d. Also we are working on implementing other types of transformations of the MAP solutions and comparing their performance. For instance, one of the further steps is incorporating the perturbation, which interchanges the elements in such a way as to give the higher probability to an interchange that improves the solution value. Finally, we plan to experiment with Monkey Search using a set of different types of transformations of the feasible solutions in the same run.

References

1. Aiex, R.M., Resende, M.G.C., Pardalos, P.M., Toraldo, G.: GRASP with path relinking for the three-index assignment problem. INFORMS J. on Computing 17(2), 224–247 (2005)
2. Balas, E., Landweer, P.: Traffic Assignment in Communication Satelites. Operations Research Letters 2, 141–147 (1983)
3. Burkard, R.E., Cela, E.: Linear Assignment Problems and Extensions. In: Pardalos, P.M., Du, D.Z. (eds.) The Handbook of Combinatorial Optimization, pp. 75–149 (1999)
4. Clemons, W., Grundel, D., Jeffcoat, D.: Applying simulated annealing on the multidimensional assignment problem. In: Butenko, S., Murphey, R., Pardalos, P.M. (eds.) Recent Developments in Cooperative Control and Optimization: Cooperative Systems. Springer, Heidelberg (2004)
5. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-completeness. W.H. Freeman and Company, New York (1979)
6. Geem, Z.W., Kim, J.H., Loganathan, G.V.: A New Heuristic Optimization Algorithm: Harmony Search. Simulations 76(2), 60–68 (2001)
7. Glover, F., Laguna, F.: Tabu Search. Kluwer Academic Publishers, Dordrecht (1997)
8. Goldberg, D.E.: Genetic Algorithms in Search, Optimization & Machine Learning. Addison-Wesley, Reading (1989)
9. Grundel, D., Krokhmal, P., Oliveira, C., Pardalos, P.: On the Number of Local Minima for the Multidimensional Assignment Problem. Journal of Combinatorial Optimization 13(1), 1–18 (2007)
10. Kirkpatrick, S., Gelatt Jr., C.D., Vecchi, M.P.: Optimization by Simulated Annealing. Science 220(4598), 671–680 (1983)
11. Kammerdiner, A., Krokhmal, P., Pardalos, P.: Distribution of Hamming distances between Solutions of Multidimensional Assignment Problem. In: Hirsch, M.J., Pardalos, P.M., Murphey, R., Grundel, D. (eds.) Advances in Cooperative Control and Optimization, vol. 369. Springer, Heidelberg (2007)
12. Kammerdiner, A.: Multidimensional Assignment Problem. In: Floudas, C.A., Pardalos, P.M. (eds.) Encyclopedia of Optimization, 2nd edn. (to appear, 2008)

13. Lidstrom, N., Pardalos, P., Pistoulis, L., Toraldo, G.: An approximation algorithm for the three-index assignment problem, Technical Report, INFORMS San Diego-1997, Combinatorial Optimization Cluster, SA06 (1997)
14. Magos, D.: Tabu search for the planar three-dimensional assignment problem. Journal of Global Optimization 8, 35–48 (1996)
15. Mucherino, A., Seref, O.: Monkey search: a novel metaheuristic search for global optimization. In: Seref, O., Kundakcioglu, O.E., Pardalos, P.M. (eds.) DataMining, Systems Analysis, and Optimization in Biomedicine, pp. 162–173. American Institute of Physics (2007)
16. Murphey, R., Pardalos, P., Pitsoulis, L.: A greedy randomized adaptive search procedure for the multitarget multisensor tracking problem. DIMACS Series, vol. 40, pp. 277–302. American Mathematical Society (1998)
17. Pierskalla, W.P.: The Tri-Substitution Method for Obtaining Near-Optimal Solutions to the Three-Dimensional Assignment Problem, Tech. Memo. No. 71, Operations Research Group, Case Institute of Technology, Cleveland, Ohio (October 1966)
18. Pierskalla, W.P.: The tri-substitution method for the three-dimensional assignment problem. Journal of Canadian Operation Research Society 5, 71–81 (1967)
19. Pierskalla, W.P.: The multidimensional assignment problem. Operations Research 16, 422–431 (1968)
20. Poore, A.B.: Multidimensional assignment formulation of data association problems arising from multitarget and multisensor tracking. Computation Optimization and Applications 3, 27–54 (1994)
21. Pusztaszeri, J.F., Rensing, P.E., Liebling, T.M.: Tracking Elementary Particles Near Their Primary Vertex: A Combinatorial Approach. Journal of Global Optimization 9(1), 41–64 (1996)
22. Robertson, A.J.: A set of greedy randomized adaptive local search procedure (GRASP) implementations for the multidimensional assignment problem. Computational Optimization and Applications 19(2), 145–164 (2001)
23. Seref, O., Akcali, E.: Monkey Search: A New Meta-Heuristic Approach. In: INFORMS Annual Meeting, San Jose, CA (2002)
24. Seref, O., Mucherino, A., Pardalos, P.M.: The Novel Meta-Heuristic Monkey Search, working paper

Robust Wireless Network Jamming Problems

Clayton W. Commander[1], Panos M. Pardalos[2], Valeriy Ryabchenko[2], Sergey Sarykalin[2], Timofey Turko[2], and Stan Uryasev[2]

[1] Air Force Research Laboratory
Eglin Air Force Base
clayton.commander@eglin.af.mil
[2] Department of Industrial and Systems Engineering
University of Florida
{pardalos,valeriy,saryk,jk13,uryasev}@ufl.edu

Abstract. We extend the results presented in a previous paper which focused on deterministic formulations of the WIRELESS NETWORK JAMMING PROBLEM. This problem seeks to determine the optimal number and placement locations for a set of wireless jamming devices to sufficiently suppress a communication network according to some specified criterion. We now introduce robust variants of those formulations which account for the fact that the exact topology of the network to be jammed may not be known entirely. Particularly, we consider instances in which several topologies are considered likely, and develop robust scenarios for placing jamming devices which are able to suppress the network regardless of which candidate topology is realized. We derive several formulations and include percentile constraints to account for a variety of scenarios. Case studies are presented and the results are analyzed. We conclude with directions of future research.

1 Introduction

Research on suppressing and eavesdropping communication networks has seen a surge recently in the optimization community. Two recent papers by Commander et al. [5,6] represent the current state-of-the-art. These problems have several important military applications and represent a critical area of research as optimization of telecommunication systems improve technological capabilities [12]. In [5], the authors develop lower and upper bounds for the optimal number of wireless jamming devices required to suppress a network contained in a given area such as a map grid. In this work, there were no a priori assumptions made about the topology of the network to be jammed other than the geographical region in which it was contained. This problem is particularly important in the global war on terrorism as improvised explosive devices (IEDs) continue to plague the coalition forces. In fact, IEDs account for approximately 65% of all combat injuries in Iraq [11]. These homemade bombs are almost always detonated by some form of radio frequency device such as cellular telephones, pagers, and garage door openers. The ability to suppress radio waves in a given region will help prevent casualties resulting from IEDs [4].

In [6] the WIRELESS NETWORK JAMMING PROBLEM (WNJP) was introduced and several formulations derived. In the WNJP, the topology of the network is assumed and various objectives can be considered, from jamming all the communication nodes to constraining the connectivity index of the nodes. In this chapter, we introduce robust variants of those formulations which account for the fact that the exact topology of the network to be jammed may not be known entirely. Particularly, we consider instances in which several topologies are considered likely, and develop robust scenarios for placing jamming devices which are able to suppress the network regardless of which candidate topology is realized. The overarching goal is to develop robust formulations with respect to the uncertainties in the information about the network. These models will provide a more realistic interpretation of combat scenarios in urban and dynamic environments.

The organization of the chapter is as follows. In Section 2, we derive several formulations of the ROBUST WIRELESS NETWORK JAMMING PROBLEM (R-WNJP). In Section 3, we review several percentile measures and incorporate percentile constraints into the models in Section 4. The results of several case studies are presented in Section 5 and the results are analyzed. We conclude with directions of future research.

2 Problem Formulations

Denote a graph $G = (V, E)$ as a pair consisting of a set of vertices V, and a set of edges E. All graphs in this chapter are assumed to be undirected and unweighted. We use the symbol "$b := a$" to mean "the expression a defines the (new) symbol b" in the sense of King [9]. Finally, we will use *italics* for emphasis and SMALL CAPS for problem names.

We assume that the communication network to be jammed comprises a set $\mathcal{M} = \{1, 2, \ldots, m\}$ of radio devices which are outfitted with omnidirectional antennas and function as both transmitters and receivers [6]. Further we assume that the coordinates of the nodes and various parameters such as the frequency range are given by probability distributions. For example, we can assume that a Kalman filter provides some estimate of the locations of the nodes. In a deterministic setup, the topology, which represents the communication pattern, could be represented by a graph in which an edge connects two nodes if they are within a certain communication threshold.

As for the set of jamming devices, we assume that they too are outfitted with omnidirectional antennas with the effectiveness of a jamming device on a communication node being inversely proportional to their squared distance. Suppose that the set of jamming devices is giving by $\mathcal{N} = \{1, 2, \ldots, n\}$, and we are given a set potential locations in which to place them. Figure 1 provides an example of the communication network and the potential jamming device locations. Moreover, each potential location j has an associated cost $c_j, j = 1, 2, \ldots, n$. We can describe the jamming power received by network node

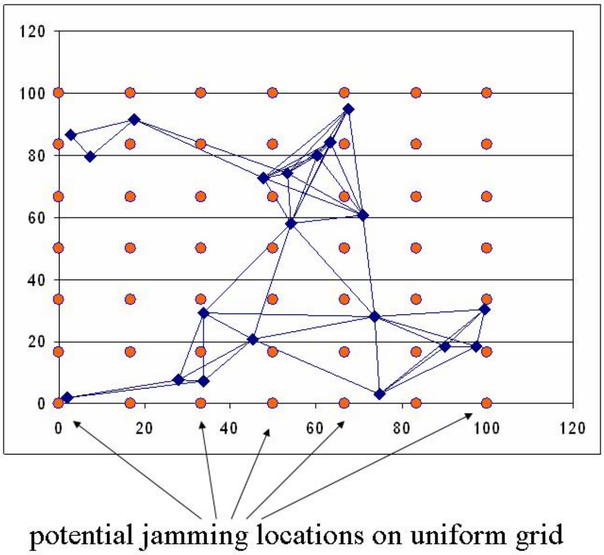


Fig. 1. Example network shown with potential jamming device locations

i located at a point $(\xi_i, \eta_i) \in \mathbb{R} \times \mathbb{R}$, from jamming device $j \in \mathcal{N}$ located at (x, y) as

$$d_j(\xi_i, \eta_i) \equiv d_{ij} := \frac{\lambda}{(x_j - \xi_i)^2 + (y_j - \eta_i)^2}, \tag{1}$$

where $\lambda \in \mathbb{R}$ is a constant. Without the loss of generality, we can let $\lambda = 1$. We say that node $i \in \mathcal{M}$ located at (ξ_i, η_i) is jammed if the total energy received at this point from all jamming devices exceeds some threshold value C_i. That is, node i is jammed if

$$\sum_{j=1}^{n} d_j(\xi_i, \eta_i) \geq C_i. \tag{2}$$

As mentioned above, we are considering robust formulations of the WNJP. Since the exact locations of the network nodes are unknown, we assume that a set of intelligence data has been collected and from that a set $\mathcal{S}$ of the most likely scenarios have been compiled. We assume that for scenario $s \in \mathcal{S}$ both the node locations $\{(\xi_1^s, \eta_1^s), (\xi_2^s, \eta_2^s), \ldots, (\xi_m^s, \eta_m^s)\}$ and the set of jamming thresholds $\{C_1^s, C_2^s, \ldots, C_m^s\}$ are modeled. We make no assumption on the equality of the number of devises to be jammed in the different scenarios. Therefore, we define for each scenario $s \in \mathcal{S}$, the set $\mathcal{M}_s = \{1, 2, \ldots, m_s\}$ which represents the set of nodes to be jammed, where m_s represents the number of nodes in scenario s. For example, the networks shown in Figure 2 represent a set of possible topologies for the network to be jammed.

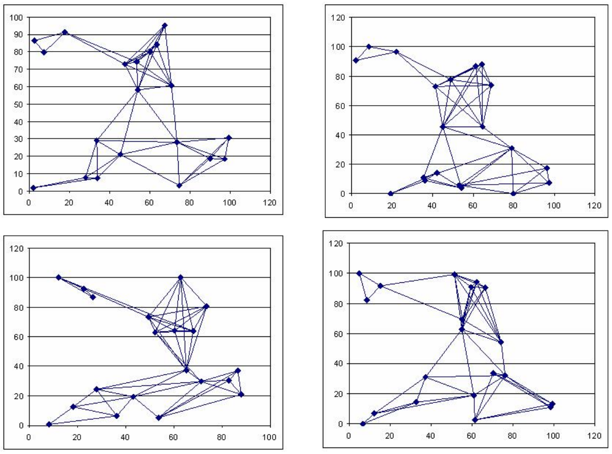


Fig. 2. Four example scenarios are shown

2.1 The Robust Connectivity Index Problem

Given a graph $G = (V, E)$, the connectivity index of a node is defined as the number of nodes reachable from that vertex, i.e., if there exists a path between two nodes. The first formulation of the WNJP we consider imposes constraints on the connectivity indices of the network nodes. The degree to which the connectivity index of a given node is constrained may be determined by its relative importance or how crucial it is for maintaining connectivity among many components. It is at the discretion of the analyst whether to assign arbitrary values to each node or use some heuristic for determining a relative importance. One way to determine the connectivity indices is to identify the so-called *critical nodes* of the graph and impose relatively tighter constraints on these nodes. Critical nodes are those vertices whose removal from the graph induces a set of disconnected components whose sizes are minimally variant [1]. Critical node detection has been recently applied to interdicting wired communication networks [4], to network security applications [2], and most recently to the analysis of protein-protein interaction networks in the context of computational drug design [3].

Suppose for example that for the scenarios shown in Figure 2 the maximum allowable connectivity index is set to 3 for each node. Then the objective of the ROBUST CONNECTIVITY INDEX PROBLEM (RCIP) is to determine the minimum number and locations for the jamming devices so that each node has no more than 3 neighbors in each of the four scenarios presented. Figure 3 provides an example solution for this case.

Suppose that communication between two nodes in the communication graph is said to be severed if at least one of the nodes is jammed. Then the objective of the ROBUST CONNECTIVITY INDEX PROBLEM (RCIP) is to determine the minimum required jamming devices such that the connectivity index of each node i in each scenario s does not exceed some predefined values L_i^s. In order to define

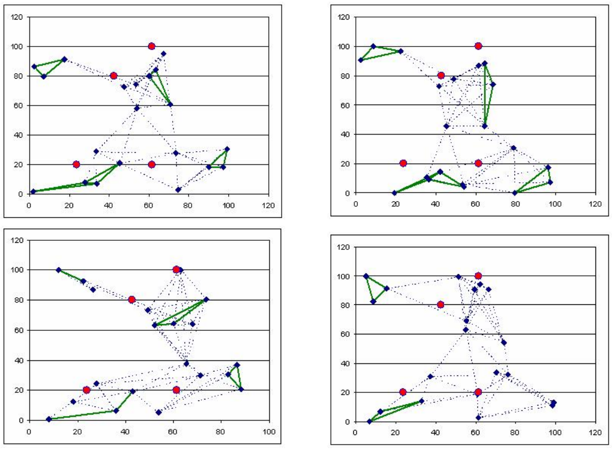


Fig. 3. The optimal solution for the networks in Figure 2 is given for the case when the maximum connectivity index is 3 for all nodes

the corresponding mathematical formulation we must define the following functions. First let $y^s : \mathcal{M}_s \times \mathcal{M}_s \mapsto \{0,1\}$ be a surjection where $y^s_{ij} := 1$ if there exists a path from node i to node j in the jammed network according to scenario $s \in \mathcal{S}$. Next let the function $z^s : \mathcal{M}_s \mapsto \{0,1\}$ be a surjective function where z^s_i returns 1 if node i is not jammed in scenario $s \in \mathcal{S}$. Finally, let $x_i, i = 1, \ldots, n$ be a set of decision variables where $x_i := 1$ if a jamming device location i is utilized. If c_k and d_{ij} are as defined in Equation 1, then we can formulate the RCIP as the following optimization problem.

$$\textbf{(RCIP)} \quad \min \sum_{k=1}^{n} c_k x_k \tag{3}$$

s.t.

$$\sum_{\substack{j=1 \\ j \neq i}}^{m} y^s_{ij} \leq L^s_i, \ \forall\, i \in \mathcal{M}_s, \ \forall\, s \in \mathcal{S}, \tag{4}$$

$$M(1 - z^s_i) > \sum_{k=1}^{n} d^s_{ik} x_k - C^s_i \geq -M z^s_i, \ \forall\, i \in \mathcal{M}_s, \quad \forall\, s \in \mathcal{S}, \tag{5}$$

$$x_j \in \{0,1\}, \ \forall\, j \in \mathcal{N}, \tag{6}$$

$$z^s_i \in \{0,1\}, \ \forall\, i \in \mathcal{M}_s, \ \forall\, s \in \mathcal{S}, \tag{7}$$

$$y^s_{ik} \in \{0,1\}, \ \forall\, i,k \in \mathcal{M}_s, \ \forall\, s \in \mathcal{S}, \tag{8}$$

where $M \in \mathbb{R}$ is some large constant.

In a manner similar to that shown in [6], we can formulate an equivalent integer programming formulation as follows. First let $v^s : \mathcal{M}_s \times \mathcal{M}_s \mapsto \{0,1\}$ and $v^{s'} : \mathcal{M}_s \times \mathcal{M}_s \mapsto \{0,1\}$ be respectively defined as

$$v_{ij}^s := \begin{cases} 1, \text{ if } (i,j) \in E^s, \\ 0, \text{ otherwise,} \end{cases} \tag{9}$$

and

$$v_{ij}^{s'} := \begin{cases} 1, \text{ if } (i,j) \text{ exists in the jammed network of scenario } s, \\ 0, \text{ otherwise.} \end{cases} \tag{10}$$

An equivalent integer program is then given by

$$\textbf{(RCIP-1)} \quad \min \sum_{k=1}^{n} c_k x_k, \tag{11}$$

s.t.

$$y_{ij}^s \geq v_{ij}^{s'}, \ \forall \ i,j \in \mathcal{M}_s, \ \forall \ s \in \mathcal{S}, \tag{12}$$

$$y_{ij}^s \geq y_{ik}^s y_{kj}^s, \ k \neq i,j; \ \forall \ i,j \in \mathcal{M}_s, \ \forall \ s \in \mathcal{S}, \tag{13}$$

$$v_{ij}^{s'} \geq v_{ij}^s z_j^s z_i^s, \ i \neq j; \ \forall \ i,j \in \mathcal{M}_s, \ \forall \ s \in \mathcal{S}, \tag{14}$$

$$\sum_{\substack{j=1 \\ j \neq i}}^{m} y_{ij}^s \leq L_i^s, \ \forall \ i \in \mathcal{M}_s, \ \forall \ s \in \mathcal{S}, \tag{15}$$

$$M(1 - z_i^s) > \sum_{k=1}^{n} d_{ik}^s x_k - C_i^s \geq -M z_i^s, \ \forall \ i \in \mathcal{M}_s, \ \forall \ s \in \mathcal{S}, \tag{16}$$

$$x_j \in \{0,1\}, \ \forall \ j \in \mathcal{N}, \tag{17}$$

$$z_i^s \in \{0,1\}, \ \forall \ i \in \mathcal{M}_s, \ \forall \ s \in \mathcal{S}, \tag{18}$$

$$y_{ij}^s \in \{0,1\} \ \forall \ i,j \in \mathcal{M}_s, \ \forall \ s \in \mathcal{S}, \tag{19}$$

$$v_{ij}^s \in \{0,1\}, \ \forall \ i,j \in \mathcal{M}_s, \ \forall \ s \in \mathcal{S}, \tag{20}$$

$$v_{ij}^{s'} \in \{0,1\}, \ \forall \ i,j \in \mathcal{M}_s, \ \forall \ s \in \mathcal{S}. \tag{21}$$

We establish the equivalence of formulations RCIP and RCIP-1 in the following theorem. The proof follows similarly to a result for the single scenario problem in [6].

Theorem 1. *If* RCIP *has an optimal solution, then* RCIP-1 *has an optimal solution. Furthermore, any optimal solution x^* of the integer programming problem* RCIP-1 *is an optimal solution of the optimization problem* RCIP.

Proof. It is easy to see that if communication nodes i and j are reachable in the jammed network of a given scenario $s \in \mathcal{S}$, then $y_{ij}^s = 1$ in RCIP-1. Indeed if i and j are reachable, then there exists a sequence of pairwise adjacent vertices

$$\{(i_0, i_1), \ldots, (i_{m-1}, i_m)\}, \tag{22}$$

where $i_0 = i$ and $i_m = j$. By inducting along the vertices, we can establish the fact that $y^s_{i_0,i\ \ +1} = 1$ for all $k = 1, \ldots, m$. To do this, first note that from (12) we have that $y^s_{i\ \ ,i\ \ +1} = 1$. Then if $y^s_{i_0,i\ \ } = 1$, then by (13) we have that

$$y^s_{i_0,i\ \ +1} \geq y^s_{i_0,i\ \ } \, y^s_{i\ \ ,i\ \ +1} = 1. \tag{23}$$

This completes the induction step. Thus far we have shown that

$$\sum_{\substack{j=1 \\ j \neq i}}^{m} y^s_{ij} \geq \text{ connectivity index of node } i.$$

Let F be the objective function in RCIP-1 and RCIP. Furthermore, suppose (x^*, y^*) and $(\hat{x}^*, \hat{y}^*)$ represent optimal solutions for each formulation respectively. Then so far, we have confirmed that

$$F(x^*) \geq F(\hat{x}^*). \tag{24}$$

It is easy to verify that $(\hat{x}^*, \hat{y}^*)$ is feasible in RCIP-1. This follows from the definition of y^s_{ij} in RCIP and the fact that $(\hat{x}^*, \hat{y}^*)$ satisfies the feasibility constraints in RCIP. This proves the first statement of the theorem. Hence from RCIP-1, we have that

$$F(x^*) \leq F(\hat{x}^*). \tag{25}$$

Therefore using (24) and (25), we have

$$F(x^*) = F(\hat{x}^*). \tag{26}$$

Now define y^s such that

$$y^s_{ij} := 1 \Leftrightarrow j \text{ is reachable from } i \text{ when scenario } s \text{ is jammed by } x^*. \tag{27}$$

Using the above results, we know that (x^*, y^s) is feasible in RCIP-1, and hence optimal. Also from the construction of y^s it follows that (x^*, y^s) is feasible in RCIP. According to (26), we can conclude that x^* is also optimal for RCIP. Thus the theorem is proved.

With the previous theorem, we have established a one-to-one correspondence between the two formulations. By using some standard techniques, we can now reformulate RCIP-1 into the following integer linear program

$$\textbf{(RCIP-2)} \quad \min \sum_{k=1}^{n} c_k x_k \tag{28}$$

s.t.

$$y^s_{ij} \geq v^{s'}_{ij}, \ \forall \ i, j = 1, \ldots, \mathcal{M}_s, \ \forall \ s \in \mathcal{S}, \tag{29}$$

$$y^s_{ij} \geq y^s_{ik} + y^s_{kj} - 1, \ k \neq i, j; \ \forall \ i, j \in \mathcal{M}_s, \ \forall \ s \in \mathcal{S}, \tag{30}$$

$$v^{s'}_{ij} \geq v^s_{ij} + z^s_j + z^s_i - 2, \ i \neq j; \ \forall \ i, j \in \mathcal{M}_s, \ \forall \ s \in \mathcal{S}, \tag{31}$$

$$\sum_{\substack{j=1\\j\neq i}}^{m} y_{ij}^s \leq L_i^s,\ \forall\ i \in \mathcal{M}_s,\ \forall\ s \in \mathcal{S}, \tag{32}$$

$$M(1-z_i^s) > \sum_{k=1}^{n} d_{ik}^s x_k - C_i^s \geq -Mz_i^s,\ \forall\ i \in \mathcal{M}_s,\ \forall\ s \in \mathcal{S}, \tag{33}$$

$$x_j \in \{0,1\},\ \forall\ j \in \mathcal{N}, \tag{34}$$

$$z_i^s \in \{0,1\},\ \forall\ i \in \mathcal{M}_s,\ \forall\ s \in \mathcal{S}, \tag{35}$$

$$y_{ij}^s \in \{0,1\}\ \forall\ i,j \in \mathcal{M}_s,\ \forall\ s \in \mathcal{S}, \tag{36}$$

$$v_{ij}^s \in \{0,1\},\ \forall\ i,j \in \mathcal{M}_s,\ \forall\ s \in \mathcal{S}, \tag{37}$$

$$v_{ij}^{s'} \in \{0,1\},\ \forall\ i,j \in \mathcal{M}_s,\ \forall\ s \in \mathcal{S}. \tag{38}$$

Theorem 2. *If* RCIP-1 *has an optimal solution, then* RCIP-2 *has an optimal solution. Further, any optimal solution x^* of* RCIP-2 *is an optimal solution of* RCIP-1*.*

Proof. For binary variables, notice that the following equivalence holds

$$y_{ij}^s \geq y_{ik}^s y_{kj}^s \Leftrightarrow y_{ij}^s \geq y_{ik}^s + y_{kj}^s - 1. \tag{39}$$

Then the only other difference between the formulations are the two constraints:

$$v_{ij}^{s'} = v_{ij}^s z_j^s z_i^s \tag{40}$$

$$v_{ij}^{s'} \geq v_{ij}^s + z_i^s + z_j^s - 2 \tag{41}$$

In a manner similar to (39) above, it is easy to verify that (40) implies (41). Therefore the feasible region of RCIP-2 includes the feasible region of RCIP-1. With this we have the first statement of the theorem.

As in the previous proof, let F represent the objective function in RCIP-1 and RCIP-2 and let x^* and $\hat{x}^*$ represent respective optimal solutions. Then it follows that

$$F(x^*) \geq F(\hat{x}^*). \tag{42}$$

Let $(x^*, y^{s*}, v^{'s*}, z^{s*})$ be an optimal solution for formulation RCIP-2. Now, define $v^{''s*}$ as follows:

$$v_{ij}^{''s*} := \begin{cases} 1, & \text{if } v_{ij}^s + z_i^{s*} + z_j^{s*} - 2 = 1, \\ 0, & \text{otherwise.} \end{cases} \tag{43}$$

Notice that if $v_{ij}^{'s*} \geq v_{ij}^{''s*}$ then $(x^*, y^{s*}, v^{''s*}, z^{s*})$ is feasible in RCIP-2 according to constraint (29) (i.e., $y_{ij}^s \geq v_{ij}^{''s*}$). Furthermore, $(x^*, y^{s*}, v^{''s*}, z^{s*})$ is optimal in RCIP-2 as the the objective value is $F(x^*)$, which is optimal by definition. Using (43), $(v^{''s*}, z^{s*})$ satisfies:

$$v_{ij}^{''s*} = v_{ij}^s z_j^{s*} z_i^{s*}. \tag{44}$$

Using this we have that $(x^*, y^{s*}, v^{''s*}, z^{s*})$ is feasible for RCIP-1. If $\hat{x}^*$ is an optimal solution of RCIP-1 then it follows that

$$F(\hat{x}^*) \leq F(x^*) \tag{45}$$

On the other hand, we have shown in (42) above, that

$$F(x^*) \leq F(\hat{x}^*). \tag{46}$$

(45) and (46) together imply $F(x_1) = F(x^*)$. The last equality proves that x^* is an optimal solution of RCIP-1. Thus, the theorem is proved.

Finally, we have the following theorem which establishes the equivalence between the optimization problem RCIP and the integer linear programming formulation RCIP-2 [6].

Theorem 3. *If* RCIP *has an optimal solution, then* RCIP-2 *has an optimal solution. Moreover, any optimal solution of* RCIP-2 *is an optimal solution of* RCIP.

Proof. The proof follows directly from Theorem 1 and Theorem 2.

2.2 Robust Node Covering Problem

What follows is a robust formulation of the OPTIMAL NODE COVERING problem presented in WNJP. As before, we are given $\mathcal{M}_s$, the set of nodes to be jammed. We are also given the set of potential locations for the jamming devices, $\mathcal{N}$. The objective of the ROBUST NETWORK COVERING PROBLEM (RNCP) is to minimize the number of jamming devices required to suppress communication on all of the nodes for each of the scenarios. Recall from Equation (2) that a node in a given scenario is said to be suppressed if the cumulative amount of energy received by that node from all jamming devices exceeds some threshold level. Let c_k, d_{ik}^s, and C_i^s be as defined previously. Also, recall that the decision variable $x_k := 1$ if a jamming device is installed at location $k \in \mathcal{N}$. With this, we can formulate the RNCP as follows.

$$\textbf{(RNCP)} \quad \min \sum_{k=1}^{n} c_k x_k, \tag{47}$$

s.t.

$$\sum_{k=1}^{n} d_{ik}^s x_k \geq C_i^s, \; i = 1, 2, \dots, m_s, s = 1, 2, \dots, S, \tag{48}$$

$$x_k \in \{0, 1\}, \; k = 1, 2, \dots, n, \tag{49}$$

The RNCP is $\mathcal{NP}$-hard which can be easily shown by a reduction from the MULTIDIMENSIONAL KNAPSACK PROBLEM [7]. With this in mind, we recognize that solving large-scale instances is unreasonable, thus we must seek alternative solution methods. One possible way of doing this is accepting the fact that jamming a sufficient percentage of the network nodes will suffice given the intractability of the problem. We now examine the R-WNJP with the inclusion of percentile constraints.

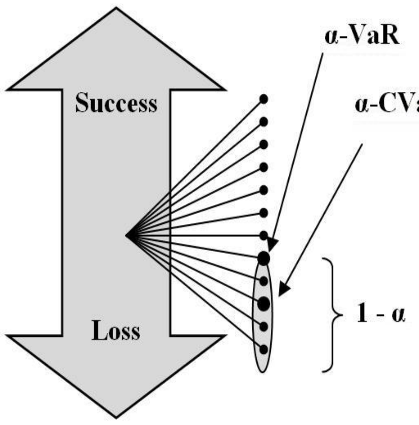


Fig. 4. A graphical depiction of VaR and CVaR

3 Percentile Constraints

As demonstrated in the seminal paper on deterministic network jamming problems [6], it is often the case that a network can be sufficiently neutralized by suppressing communication on a fraction of the total nodes. This can be accomplished by the inclusion of percentile constraints into a mathematical model. In this section, we review two commonly used risk measures and derive formulations of the RCIP and a RNCP.

3.1 Review of Value at Risk (VaR) and Conditional Value at Risk (CVaR)

The first percentile constraint we examine is the simplest risk measure used in optimization of robust systems and is known as the Value at Risk (VaR) measure [8]. VaR provides an upper bound, or percentile on a given loss distribution. For example, consider an application in which a constraint must be satisfied within a specific confidence level $\alpha \in [0,1]$. Then the corresponding α-VaR value is the lowest value ζ such that with probability α, the loss does not exceed ζ [10]. In economic terms, VaR is simply the maximum amount at risk to be lost from an investment. VaR is the most widely applied risk measure in stochastic settings primarily because it is conceptually simple and easy to incorporate into a mathematical model [6]. However with this ease of use come several complicating factors as we will soon see. Some disadvantages of VaR are that the inclusion of VaR constraints adds to the number of discrete variables in a problem. Also, VaR is not a so-called *coherent* risk measure, implying among other things that it is not sub-additive.

Another commonly applied risk measure is the so-called Conditional Value-at-Risk (CVaR) developed by Rockafellar and Uryasev [13]. CVaR is a more conservative measure of risk, defined as the expected loss under the condition that VaR is exceeded. A graphical representation of the relationship between CVaR and VaR is shown in Figure 4. In order to define CVaR and Var we need to determine the cumulative distribution function for a given decision vector subject to some uncertainties. Suppose $f(x, y)$ is a performance (or loss) function associated with a decision vector $x \in X \subseteq \mathbb{R}^n$, and a random vector $y \in \mathbb{R}^m$ which is the uncertainties that may affect the performance. Assume that y is governed by a probability measure P on a Borel set, say Y [6]. The loss $f(x, y)$ for each $x \in X$ is a random variable having a distribution in $\mathbb{R}$ induced by y. Then the probability of $f(x, y)$ not exceeding some value ζ is defined as

$$\psi(x, \zeta) := P\{y | f(x, y) \leq \zeta\}. \tag{50}$$

By fixing x, the cumulative distribution function of the loss associated with the decision x is thus given by $\psi(x, \zeta)$ [15].

Given the loss random variable $f(x, y)$ and any $\alpha \in (0, 1)$, we can use equation (50) to define α-VaR as

$$\zeta_\alpha(x) := \min\{\zeta \in \mathbb{R} : \psi(x, \zeta) \geq \alpha\}. \tag{51}$$

From this we see the probability that the loss $f(x, y)$ exceeds $\zeta_\alpha(x)$ is $1 - \alpha$. Using the definition above, CVaR is the conditional expectation that the loss according to the decision vector x dominates $\zeta_\alpha(x)$ [13]. Thus we have α-CVaR denoted as $\phi_\alpha(x)$ defined as

$$\phi_\alpha(x) := E\{f(x, y) | f(x, y) \geq \zeta_\alpha(x)\}. \tag{52}$$

In order to include CVaR and VaR constraints in optimization models we must characterize $\zeta_\alpha(x)$ and $\phi_\alpha(x)$ in terms of a function $F_\alpha : X \times \mathbb{R} \mapsto \mathbb{R}$ defined by

$$F_\alpha(x, \zeta) := \zeta + \frac{1}{(1 - \alpha)} E\{\max\{f(x, y) - \zeta, 0\}\}. \tag{53}$$

In the seminal paper on CVaR [13], Rockafellar and Uryasev prove that as a function of ζ, $F_\alpha(x, \zeta)$ is convex and continuously differentiable. Moreover, they show that α-CVaR of the loss associated with any $x \in X$, i.e., $\phi_\alpha(x)$, is equal to the global minimum of $F_\alpha(x, \zeta)$, over all $\zeta \in \mathbb{R}$. Further, if $A_\alpha(x) = \operatorname{argmin}_{\zeta \in \mathbb{R}} F_\alpha(x, \zeta)$ is the set consisting of the values of ζ for which F is minimized, then $A_\alpha(x)$ is a non-empty, closed and bounded interval and $\zeta_\alpha(x)$ is the left endpoint of $A_\alpha(x)$. In particular, it is always the case that $\zeta_\alpha(x) \in \operatorname{argmin}_{\zeta \in \mathbb{R}} F_\alpha(x, \zeta)$ and $\psi_\alpha(x) = F_\alpha(x, \zeta_\alpha(x))$ [13].

This result gives a linear optimization algorithm for computing α-CVaR. It is a result of the convexity of $F_\alpha(x, \zeta)$, that we are able to minimize CVaR for $x \in X$ without having to numerically calculate $\phi_\alpha(x)$ for every x. This has been shown by Rockafellar and Uryasev in [14]. Further, it has been shown in [14] that for any probability threshold α and loss tolerance ω, that constraining $\phi_\alpha(x) \leq \omega$ is equivalent to constraining $F_\alpha(x, \zeta) \leq \omega$.

3.2 Robust Jamming with Percentile Constraints

Now that we have theoretical groundwork for the VaR and CVaR percentile measures, we develop formulations of the robust jamming problems incorporating these risk constraints. We begin with the ROBUST NODE COVERING PROBLEM. Since we are given a set of possible network scenarios, we would like to develop a formulation for the RNCP in which the optimal solution will be guaranteed to cover some predetermined fraction $\alpha \in (0,1)$ of the network nodes, regardless of the scenario realized. To do this, we can include α-VaR constraints in the RNCP as follows. First we define the surjection $\rho^s : \mathcal{M}_s \mapsto \{0,1\}$ by

$$\rho_i^s := \begin{cases} 1, & \text{if node } i \text{ is jammed in scenario } s, \\ 0, & \text{otherwise.} \end{cases} \tag{54}$$

Then we can formulate the ROBUST NODE COVERING PROBLEM with Value-at-Risk constraints as

$$\textbf{(RNCP-VaR)} \quad \min \sum_{k=1}^{n} c_k x_k, \tag{55}$$

s.t.

$$\sum_{k=1}^{n} d_{ik}^s x_k \geq C_i^s \rho_i^s, \ \forall\ s \in \mathcal{S}, \forall\ i \in \mathcal{M}_s, \tag{56}$$

$$\sum_{i=1}^{m} \rho_i^s \geq \alpha m_s, \ \forall\ s \in \mathcal{S}, \tag{57}$$

$$x_k \in \{0,1\}, \ \forall\ k \in \mathcal{N}, \tag{58}$$

$$\rho_i^s \in \{0,1\}, \ \forall\ s \in \mathcal{S}, \forall\ i \in \mathcal{M}_s, \tag{59}$$

Notice that to include the VaR constraints an additional m_s binary variables are required for each scenario $s \in \mathcal{S}$.

In a similar manner, we can incorporate VaR constraints into the RCIP by introducing $\omega^s : \mathcal{M}_s \mapsto \{0,1\}$ as

$$\omega_i^s := \begin{cases} 1, & \text{connectivity index of node } i \text{ is constrained on scenario } s, \\ 0, & \text{no requirement on connectivity index of node } i. \end{cases} \tag{60}$$

Using this we have

$$\textbf{(RCIP-VaR)} \quad \min \sum_{k=1}^{n} c_k x_k \tag{61}$$

s.t.

$$M(1 - z_i^s) > \sum_{k=1}^{n} d_{ik}^s x_k - C_i^s \geq -M z_i^s, \ \forall\ s \in \mathcal{S}, \ \forall\ i \in \mathcal{M}_s, \tag{62}$$

$$\sum_{\substack{j=1\\j\neq i}}^{m} y_{ij}^s \leq L_i^s \omega_i^s + M(1-\omega_i^s),\ \forall\ s \in \mathcal{S},\ \forall\ i \in \mathcal{M}_s, \tag{63}$$

$$\sum_{j=1}^{m} \omega_j^s \geq \alpha m_s,\ \forall\ s \in \mathcal{S}, \tag{64}$$

$$x_k \in \{0,1\},\ \forall\ k \in \mathcal{N}, \tag{65}$$

$$z_i^s, \omega_i^s \in \{0,1\},\ \forall\ s \in \mathcal{S},\ \forall\ i \in \mathcal{M}_s, \tag{66}$$

$$y_{ik}^s \in \{0,1\},\ \forall\ s \in \mathcal{S},\ \forall\ i,k \in \mathcal{M}_s, \tag{67}$$

where $M \in \mathbb{R}$ is some large constant. As with the **RNCP-VaR** formulation, an additional m_s binary variables are required for each scenario $s \in \mathcal{S}$. We will see in the following section the dramatic effect that the inclusion of VaR constraints has on the computability of optimal solutions.

In order to develop formulations incorporating CVaR constraints, we must derive an appropriate loss function for each problem. We will begin with the RNCP. As in [6], we introduce the function $f^s : \{0,1\}^n \times \mathcal{M}_s \mapsto \mathbb{R}$ defined by

$$f^s(x,i) := C_i^s - \sum_{j=1}^{n} x_j d_{ij}^s. \tag{68}$$

Given a decision vector x representing the placement of the jamming devices, the loss function is defined as the difference between the energy required to jam network node i, namely C_i^s, and the cumulative amount of energy received at node i due to x over each scenario [6]. With this we can formulate the RNCP with CVaR constraints as follows.

$$\textbf{(RNCP-CVaR)} \quad \min \sum_{k=1}^{n} c_k x_k, \tag{69}$$

s.t.

$$\zeta^s + \frac{1}{(1-\alpha)m_s} \sum_{i=1}^{m} \max\left\{ C_{min}^s - \sum_{k=1}^{n} d_{ik}^s x_k - \zeta^s, 0 \right\} \leq 0,\ \forall\ s \in \mathcal{S}, \tag{70}$$

$$x_k \in \{0,1\},\ \forall\ k \in \mathcal{N}, \tag{71}$$

$$\zeta^s \in \mathbb{R},\ \forall\ s \in \mathcal{S}. \tag{72}$$

The CVaR constraint (70) implies that for the $(1-\alpha)\cdot 100\%$ of the worst (least) covered nodes, the average value of $f(x)$ is less than or equal to 0.

In a similar manner, we can formulate the ROBUST CONNECTIVITY INDEX PROBLEM with the addition of CVaR constraints. As before, we need to define an appropriate loss function. We define the loss function f_s' for a network node i in scenario s as the difference between the connectivity index of i and the maximum allowable connectivity index L_i^s which occurs as a result of the placement of the jamming devices according to x. That is, let $f' : \{0,1\}^n \times \mathcal{M}_s \mapsto \mathbb{Z}$ be defined by

$$f'_s(x,i) := \sum_{j=1, j\neq i}^{m} y^s_{ij} - L^s_i. \tag{73}$$

With this, the CIP-CVaR formulation is given as follows.

$$\textbf{(RCIP-CVaR)} \quad \min \sum_{k=1}^{n} c_k x_k, \tag{74}$$

s.t.

$$\zeta^s + \frac{1}{(1-\alpha)m_s} \sum_{i=1}^{m} \max \left\{ \sum_{\substack{j=1 \\ j\neq i}}^{m} y^s_{ij} - L^s_{max} - \zeta^s, 0 \right\} \leq 0, \ \forall\ s \in \mathcal{S}, \tag{75}$$

$$x_k \in \{0,1\}, \ \forall\ k \in \mathcal{N}, \tag{76}$$

$$y^s_{ik} \in \{0,1\}, \ \forall\ s \in \mathcal{S}, \ \forall\ i,k \in \mathcal{M}_s, \tag{77}$$

$$\zeta^s \in \mathbb{R}, \ \forall\ s \in \mathcal{S}, \tag{78}$$

where L^s_{max} is a maximum allowable connectivity index under scenario s which occurs as a result of the placement of the jamming devices. The constraint on CVaR provides that for the $(1-\alpha)\cdot 100\%$ of the worst cases, the average connectivity index will not exceed $L_{\max}$. Notice that to include the CVaR constraint, we only add real variables to the problem. The continuous nature of CVaR variables versus the discrete nature of the VaR variables will explain the vast difference in the computation times in the case studies presented in the following section.

4 Case Studies

In this section, we present some preliminary numerical results comparing the performance and solution quality of the proposed formulations. The experiments were performed on a PC equipped with a 1.4MHz Intel Pentium R 4 processor with 1GB of RAM, working under the Microsoft Windows R XP SP2 operating system. The optimal solutions for the case studies were calculated using CPLEX 9.0.

The problem set considered is relatively small, but provides some insights into the solutions obtained using VaR and CVaR constraints. The network consists of 20 nodes which must be jammed. For this problem, we consider five network scenarios. We note here that the jamming thresholds of the nodes do not depend upon the scenarios. As for the placement of the jamming devices, we use the same approach as in [6], which consists of 36 potential locations located on the vertices of a uniform grid placed over the region containing the network. One network scenario showing the potential locations of the jamming devices is shown in Figure 1. The remaining scenarios are depicted in Figure 2.

4.1 Node Covering Problems

We begin by examining the ROBUST NODE COVERING PROBLEM. For the first case study, we consider the RNCP with Value-at-Risk constraints. The loss threshold is .9 which implies that the covering constraints must be satisfied for at least 90% of the network nodes. The optimal solution requires 9 jamming devices. CPLEX computed the solution for this problem in 18 seconds. The results of this instance are provided in Table 1. The table shows the total jamming level as a percentage of the jamming threshold received by each node in each scenario. Notice that all but 7 (over all scenarios) were totally jammed; however, for those nodes not totally jammed VaR constraints provide no guarantee that they will receive any jamming energy whatsoever. Though not an important factor in this case, this fact could potentially lead to problems in large-scale instances of the problem.

Next, we examine the same problem only replacing the VaR constraints with Conditional Value-at-Risk constraints. As before, the loss threshold is .90, implying that the maximum allowable losses (uncovered nodes) exceeding VaR must be no greater than 10%. Interestingly, the optimal solution in this case also requires 9 jamming devices. However this solution was computed in 0.922 seconds. The results from this study are shown in Table 2. Notice in this case that with the same number of jamming devices all but 2 nodes (across all scenarios) were totally jammed. We see that not only is this solution better in terms of the total number of jammed nodes, but it was also computed in an order of magnitude less time.

Table 1. Network coverage with VaR constraints. The total jamming level (%) for each scenario is shown.

Node	Scenario 1	Scenario 2	Scenario 3	Scenario 4	Scenario 5
1	100	100	100	100	100
2	100	100	100	100	100
3	100	100	100	100	100
4	100	100	100	100	100
5	100	100	100	100	100
6	100	100	100	100	100
7	100	100	100	100	100
8	100	100	100	100	100
9	100	100	100	100	100
10	100	100	100	100	100
11	100	100	100	100	100
12	100	100	100	100	100
13	100	100	100	100	79.31
14	100	100	100	100	100
15	100	75.74	100	100	100
16	86.45	81.86	100	57.84	100
17	100	100	100	100	100
18	100	100	100	100	100
19	100	100	100	100	100
20	86.47	100	100	65.24	100

Table 2. Network coverage with CVaR constraints. The total jamming level (%) for each scenario is shown.

Node	Scenario 1	Scenario 2	Scenario 3	Scenario 4	Scenario 5
1	100	100	100	100	100
2	100	100	100	100	100
3	100	100	100	100	100
4	100	100	100	100	100
5	100	100	100	100	100
6	100	100	100	100	100
7	100	100	100	100	100
8	100	100	100	100	100
9	100	100	100	100	100
10	100	100	100	100	100
11	100	100	100	100	100
12	100	100	100	100	100
13	100	100	97.25	100	100
14	100	100	100	100	100
15	100	100	100	100	100
16	100	100	100	93.93	100
17	100	100	100	100	100
18	100	100	100	100	100
19	100	100	100	100	100
20	100	100	100	100	100

4.2 Connectivity Index Problems

Now we discuss the results of the case study for the RSCIP with VaR and CVaR constraints. In this case, both maximum connectivity indices are set to $L = 3$. Again, the VaR threshold is .90. The optimal solution for this problem (without percentile constraints) is shown in Figure 3. This solution requires 4 jamming

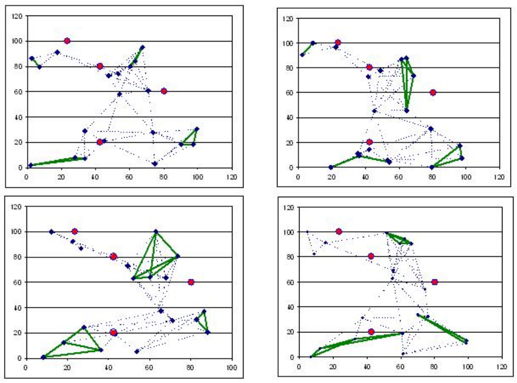


Fig. 5. The optimal solution with VaR constraints for the networks in Figure 2 is given for the case when the maximum connectivity index is 3 for all nodes

devices and was computed in 3 minutes, 58 seconds. The solution using VaR constraints is shown in Figure 5. This solution also required 4 jamming devices, but took 8 hours, 49 minutes, 43 seconds to compute. The same solution was also obtained using CVaR constraints in a time comparable to the original formulation. Even for this small example, we see that including VaR constraints in an optimization model often leads to drastic increases in computation times. This provides more evidence that using CVaR constraints instead is usually more efficient and provides appropriate solutions.

5 Conclusion

In this chapter, we develop models for jamming communication networks under uncertainty. This work extends prior work by the authors in which deterministic cases of the problems were considered [4,6]. In particular, we have developed formulations for jamming wireless networks when the exact topology of the underlying network is unknown. We have used scenario based techniques which provide robust solutions to the problems considered. Future areas of research include investigating the required number of scenarios to accurately model the statistical properties of the data. Due to the complexity of the problems considered, heuristics and advanced cutting plane techniques should also be investigated.

Acknowledgements

The authors gratefully acknowledge the Air Force Office of Scientific Research for providing funding under project FA9550-05-1-0137.

References

1. Arulselvan, A., Commander, C.W., Elefteriadou, L., Pardalos, P.M.: Detecting critical nodes in social networks. Computers and Operations Research (accepted, 2008)
2. Arulselvan, A., Commander, C.W., Pardalos, P.M., Shylo, O.: Managing network risk via critical node identification. In: Gulpinar, N., Rustem, B. (eds.) Risk Management in Telecommunication Networks. Springer, Heidelberg (submitted, 2007)
3. Boginski, V., Commander, C.W.: Detecting critical nodes in protein-protein interaction networks. In: Butenko, S.I., Chaovilitwongse, W.A., Pardalos, P.M. (eds.) Clustering Challenges in Biological Networks. World Scientific, Singapore (to appear, 2008)
4. Commander, C.W.: Optimization Problems in Telecommunications with Military Applications. Ph.D. Dissertation, University of Florida (2007)
5. Commander, C.W., Pardalos, P.M., Ryabchenko, V., Shylo, O., Uryasev, S.: Jamming communication networks under complete uncertainty. Optimization Letters 2(1), 53–70 (2008)
6. Commander, C.W., Pardalos, P.M., Ryabchenko, V., Uryasev, S.: The wireless network jamming problem. Journal of Combinatorial Optimization 14(4), 481–498 (2007)

7. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman and Company, New York (1979)
8. Holton, G.: Value-at-Risk: Theory and Practice. Academic Press, London (2003)
9. King, J.: Three problems in search of a measure. American Mathematical Monthly 101, 609–628 (1994)
10. Krokhmal, P., Palmquist, J., Uryasev, S.: Portfolio optimization with conditional value-at-risk objective and constraints. The Journal of Risk 4(2), 11–27 (2002)
11. Peake, J.B.: Beyond the Purple Heart - Continuity of care for the wounded in Iraq. New England Journal of Medicine 352(3), 219–222 (2005)
12. Resende, M.G.C., Pardalos, P.M.: Handbook of Optimization in Telecommunications. Springer, Heidelberg (2006)
13. Rockafellar, R.T., Uryasev, S.: Optimization of conditional value-at-risk. The Journal of Risk 2(3), 21–41 (2000)
14. Rockafellar, R.T., Uryasev, S.: Conditional value-at-risk for general loss distributions. Journal of Banking and Finance 26, 1443–1471 (2002)
15. Uryasev, S.: Conditional value-at-risk: Optimization algorithms and applications. Financial Engineering News 14, 1–5 (2000)

A Token-Based Approach to Sharing Beliefs in a Large Multiagent Team[*]

Prasanna Velagapudi[1], Oleg Prokopyev[2], Paul Scerri[1], and Katia Sycara[1]

[1] The Robotics Institute, Carnegie Mellon University
Pittsburgh, PA 15213, USA
{pkv,pscerri,katia}@cs.cmu.edu

[2] Department of Industrial Engineering, University of Pittsburgh
Pittsburgh, PA 15261, USA
prokopyev@engr.pitt.edu

Abstract. The performance of a cooperative team depends on the views that individual team members build of the environment in which they are operating. Teams with many vehicles and sensors generate a large amount of information from which to create those views. However, bandwidth limitations typically prevent exhaustive sharing of this information. As team size and information diversity grows, it becomes even harder to provide agents with needed information within bandwidth constraints, and it is impractical for members to maintain any detailed information for every team mate. Building on previous token-based algorithms, this chapter presents an approach for efficiently sharing information in large teams. The key distinction from previous work is that this approach models differences in how agents in the team value knowledge and certainty about features. By allowing the tokens passed through the network to passively estimate the value of certain types of information to regions of the network, it is possible to improve token routing through the use of local decision-theoretic models. We show that intelligent routing and stopping can increase the amount of locally useful information received by team members while making more efficient use of agents' communication resources.

1 Introduction

Emerging and envisioned systems involve hundreds or thousands of autonomous platforms cooperating in an environment to achieve complex joint objectives [6,1]. These systems will generate incredible volumes of sensor data that need to be fused and disseminated to the platforms that need the produced information. In a distributed system, getting sensor data to platforms that require it in a timely manner while respecting tight communication constraints is a key problem. Algorithms to achieve this goal are critical to the success of

[*] This research has been sponsored in part by AFOSR FA9550-07-1-0039 and AFOSR FA9620-01-1-0542.

teams of autonomous systems for applications ranging from tracking [6] to force protection [1] to searching for lost hikers [2].

The problem of distributed data fusion is a broad one, where many successful solutions have been developed for specific problem instances. The specific problem addressed in this chapter is characterized by three properties: (1) large numbers of agents generate a large volume of data about many features, (2) each agent gains different value from information about different features, e.g., an agent will be more interested in features in its spatial vicinity, and (3) the value an agent gains from information about a particular feature is not precisely known to the rest of the team in advance. While previous solutions have dealt with problems having one or two of these properties, using techniques ranging from particle filters [7] to Bayesian filters [3] to decision theory [8], there are no previous solutions that can handle distributed fusion problems with all three characteristics.

In many situations, especially in heterogeneous teams, agents will require information at varying levels of precision. In many approaches, this distinction is lost altogether, and information is distributed equally among teammates in this situation. However, this is highly inefficient when most of the agents do not need high precision about most of the features. For instance, a ground robot may require very extensive information about the position of nearby ground-level obstacles, while an aerial robot might need only rough estimates of large ground obstacles over a wide area. Notice that the agents that need particular information are not always the same as those that sense it. In this case, the aerial robot can use its perspective to get very detailed ground obstacle data over a wide area. Often, very few agents will attribute high value to precise knowledge of a particular feature, while equally few will be able to sense relevant information about that feature. For example, in a large team in a large outdoor environment, a few robots might be descending into a canyon. Information about the canyon, which might only be visible to a few robots exploring the cliffs above, is extremely valuable to this small subset. In situations like these, information must be shared *asymmetrically*, i.e., not all agents will receive the same information or even the same amount of information.

In this work, individual sensor readings are encapsulated into *tokens* and "pushed" from agent to agent. Each agent decides whether to continue to forward the token based on how useful to the team it believes the reading is. By encapsulating sensor readings and utilizing unique identifiers, we can altogether avoid the "double-counting" problem faced by other methods that condense or splice belief representations. In addition, sensor readings are usually compact, can be exchanged between agents with different filter algorithms, and are atomic and unordered.

In previous work [9], if the agent chose to forward the token, it did so randomly. In an asymmetric information environment this strategy is inefficient since, on average, all agents will receive the same information regardless of their need for that information. In this chapter, this inefficiency is addressed by building on the following two key ideas.

First, instead of considering the problem of *whether* to forward a token independent of *where* to forward it, we can combine both into a decision problem. Given a known cost of communication, we can estimate the value to the team of either forwarding the token to some teammate or terminating it, and use decision-theoretic methods to determine the optimal action. Previously received tokens encapsulate observations that influence the probabilities for the decision problem, e.g., receiving an observation indicating that a token was not useful to the previous agent along with information about a particular feature may indicate that it would not be useful to send another reading of the same feature to that agent in the near future.

However, if an agent only considers the impact of its routing on neighbors, its routing decisions can significantly impair communications throughout the network by redirecting tokens to greedy, suboptimal paths. The second key idea is to avoid making these myopic decisions about if and where to route the token by requiring the agent that has the token to estimate the token's impact on agents in other parts of the network. Our key insight here is that the token itself is an ideal carrier and accumulator for this information. If each token captures information about the agents it visits, it might be possible for recipient agents to make better estimates of the value of that information to the team. This, in turn, has the potential to improve the routes that agents use for subsequent tokens. However, in a large team, it is impractical for tokens to carry individual information about each visited agent. Instead, we have the token itself contain the aggregated estimate of the value to the team along a given path, calculated and updated by each agent it visits. This way, the decisions made by an agent can be based on summarized network statistics, while the estimate itself involves only constant time computations at each agent. We demonstrate that these simple estimates can improve the routing of tokens through a small worlds network (in which each node has relatively few neighbors, but any pair of nodes can be connected by a short path [10]) while maintaining or reducing the number of communications required.

2 Problem Statement

This section formally describes the problem addressed by this chapter. Agents $A = \{a_1, \ldots, a_m\}$ are a team with a joint objective in a partially observable domain. Decisions about actions by the agents are based on state variables $X(t) = \{x_1(t), \ldots, x_n(t)\}$ that describe their environment. These state variables can have any mathematical type (e.g., discrete, continuous, boolean) as long as the following functions are defined appropriately.

Agents take readings with their sensors. These sensors are imperfect, thus agents must use a filter to estimate a distribution over each of the state variables. Agent a has a probability distribution over X at time t of $P_a(X(t), t)$. While agents need not be homogeneous, it is assumed that each agent's filter can handle sensor readings from any other agent's sensor. Communication between agents is assumed to be fixed cost and point-to-point, with agents able to communicate directly with a known, static subset of teammates at any time, which are referred to as *neighbors*. No assumptions are made about the spatial or informational association between agents, but the communications network as a whole is assumed to have a small world property [10]. Denote by N_a the set of neighbors with which agent a can communicate. Let κ be the cost of a single communication, and denote by M_t the cumulative number of communications made before time t.

The performance of the team will be adversely affected whenever its members' estimates of the state of environment differ from the actual state of the environment. The information difference of a single agent is $\varDelta^a(X, P_a(X(t), t))$ (e.g., Kullback-Leibler divergence, or a similar measure). The bigger $\varDelta^a(\bullet)$, the higher the value of the divergence. However, depending on their current activities, individual agents' performance will not be equally effected by divergence. In general, they will only need to know precisely some values, while others can be coarsely understood or not known at all. Specifically, the cost of $\varDelta^a(\bullet)$ divergence to an agent a at a particular time is: $c(a, \varDelta^a(\bullet)) \rightarrow \mathcal{R}$. For example, if a ground robot were traveling quickly across rugged terrain, the cost of divergence for a state variable representing the position of a nearby obstacle might be quite high, since it could potentially endanger the robot. The same variable might have a low cost of divergence for an aerial robot, as it would pose no threat and would not affect the decisions that the robot needed to make.

As agents receive sensor readings, they are integrated into $P_a(X(t), t)$ via some filter ϕ, $P'_a(X(t), t) = \phi(P_a(X(t), t), s)$. The only assumption made about the filter is that it is always better to have more sensor readings. Using the cost of information divergence and filter equations, the value of that sensor reading to a is

$$v(s, a) = c(a, \varDelta^a(X, P'_a(X(t), t))) - c(a, \varDelta^a(X, P_a(X(t), t))) \ ,$$

i.e., the change in cost. We assume $v(s, a) \geq 0$. In a situation where a team of robots were estimating the location of a landmark, this value would be high for a high precision readings that could significantly improve robots' estimates. In contrast, if the robots already knew the position of the landmark to a high degree of accuracy, a noisy sensor reading would change very little, and so it would map to a small value. The value of s to the whole team is

$$V(s) = \sum_{a \in A} v(s, a) \ .$$

The objective is to pass sensor readings around the team such that the total cost to the team is minimized after communication costs are considered:

$$\min \sum_{a \in A} c(a, \varDelta^a(X, P_a(X(t)))) + \kappa \cdot M_t \ . \tag{1}$$

Although omitted in the above expression for clarity, this minimization covers the space of potential paths through the network of every sensor reading generated over the time interval. Note that agents cannot directly measure $\Delta^a(X, P_a(X(t), t))$. Instead, agents estimate $v(s, a)$ directly, using their filters and domain-specific knowledge of x. For example, aerial robots receiving ground obstacle information would realistically be able to determine that such information was of little value to them, regardless of the accuracy of their filter. Denote this estimation function as $\hat{v}(s, a)$.

3 Algorithm

An agent cannot directly observe the value gained by its neighbors for a given token. However, it can get observations of the value gained by previous tokens of the same type that previously visited its neighbors. There also exists some transition function that determines how the value at the neighbors decreases as they receive tokens from various sources. An agent a is then faced with choosing an action from the set of possible actions Ψ, defined as

$$\Psi = \left\{ \bigcup_{b \in N} \mathcal{A}_b \right\} \bigcup \mathcal{S}$$

where $\mathcal{A}_b$ is the action of forwarding the token to neighbor b, and $\mathcal{S}$ is the action of stopping the token.

This token-based algorithm will be effective if tokens are delivered to team members who gain information from the sensor reading on the token. Thus, a policy for propagating a token around the team has two components: (1) determining whether to further propagate the token and, if so, (2) to whom to send the token.

The problem of estimating future value from only passively observed token traffic is hard. For divergence-based value functions, the state transition function upon receiving a token is often non-linear. In addition, there is the problem of repeating visits. If an agent a receives a token from an agent b that has a history of high-value, it is likely that sending a token of the same type to b will also result in high value. However, a token that has already been sent to b will not gain much additional value, as it will likely be revisiting nodes. Thus, we also need to estimate the likelihood that agents in b's neighborhood have already received a token, and either converge to a tree-like structure or rapidly update value estimates to compensate.

Given these issues, it is impractical to try and create an exact value or transition model based on the actual movement of tokens. We instead create a local heuristic framework for solving this decision problem based on the following insights: (1) the average value that an agent gains from tokens of a particular type is distributed in a roughly Gaussian way, and the mean of this value can be reasonably approximated, (2) we can represent a near-optimal solution to this problem as a probability distribution over the discrete action space, (3) many tokens move through the network, so at least a few will travel over each link, and (4) while agent-token value will change over time, it will change slowly with respect to the token movement through the network.

We have tokens of various types $\Gamma = \{T_1, T_2, \ldots, T_{|\Gamma|}\}$. Each type of token contains data related to the corresponding state variable, i.e. T_1 contains data about x_1, T_2 about x_2, and so forth. A token τ of type $T \in \Gamma$ is defined as a tuple containing three elements: $\tau =< s, b, E >$, where s is the sensor reading about x_i, b is the previous agent visited, and $E \triangleq E^T_{ab}$ is the expected value of sending a token of type T from agent a to its neighbor b. This is computed every time a token is to be forwarded from an agent b to an agent a. We describe the details of this process below.

Each agent a maintains a decision matrix D_a, of dimension $|\Gamma| \times (|N_a| + 1)$ from which it samples its routing actions. Each column of the matrix represents the possible actions that agent a has for a token. Columns 1 to $|N_a|$ are the actions of routing to the neighbors of a, while column $|N_a| + 1$ is the action of stopping the token. Each row in the matrix represents a type of token from set Γ. Element $D_a[T, \psi]$ is the probability of executing an action $\psi \in \Psi$ when given a token of type $T \in \Gamma$. Thus, each row of D_a is a well-formed probability distribution, satisfying

$$\begin{aligned} D_a[T, \psi] \geq 0 \quad \forall \psi \quad \forall T \\ \sum_{\psi} D_a[T, \psi] = 1 \quad \forall T \quad . \end{aligned}$$

Each agent also maintains a value matrix V_a of dimension $|\Gamma| \times (|N_a| + 1)$ that contains its value estimates for each potential routing action. Once again, each column of the matrix is an action, and each row denotes a type of token. Element $V_a[T, \psi]$ is the estimated value gained by the team if action $\psi \in \Psi$ is performed on a token of type $T \in \Gamma$.

When an agent a receives a token of type $T \in \Gamma$, it samples an action from the distribution in the respective row of D_a. If that action is to send to a neighbor b, then the agent computes the expected value estimate

$$E^T_{ba} = \hat{v}(s, a) + \sum_{c \in N \ \setminus \{b\}} (D_a[T, \mathcal{A}_c] \cdot V_a[T, \mathcal{A}_c] - \kappa) \quad . \tag{2}$$

Intuitively, this is simply the expected value of sending a packet from agent b to agent a, using an expectation estimate that incorporates a split-horizon. This value is then stored in the token, and the selected action of forwarding is performed. If the stopping action is selected, the token is simply deleted. While this estimate does not include an explicit discount factor, the probabilities used in the expectation are later constrained to be strictly less than one. In practice, they effectively act as implicit discount factors.

3.1 Heuristic Updates of D_a and V_a

As tokens are received by an agent $a \in A$, D_a and V_a are updated based on the incoming value estimates. We define two heuristic functions that govern this behavior. First, a value update function λ is applied to incorporate the estimate contained in a received token τ into the value matrix V_a:

$$V_a \leftarrow \lambda(V_a, \tau.E) \ . \tag{3}$$

Then, a decision update function π uses the updated value matrix to refine the decision matrix D_a:

$$D_a \leftarrow \pi(D_a, V_a) \ . \tag{4}$$

Value estimation (λ). Each agent maintains an estimate of the value of each type of token, based on a simple adaptive learning rule. This estimate is calculated using a learning rule of the form

$$V_a[T, \mathcal{A}_b] \overset{\lambda}{\leftarrow} (1-\alpha) \cdot V_a[T, \mathcal{A}_b] + \alpha \cdot \tau.E \tag{5}$$

where α is the *value learning factor*, and $\tau.E$ is the expected value estimate in the received token τ. This will compute a weighted average of the new measurement and the current estimate. The larger the value of α, the more sensitive the value estimation to noise. If α is too small then value estimation will not respond fast enough to the system dynamics.

Routing (π). We define a heuristic for updating the decision matrix D_a that follows the intuition: *If routing action ψ is n-times more useful than action ψ', then action ψ should be n-times more likely to occur than action ψ'.* Here, "usefulness" is represented by the value estimate. This intuition is embodied in the simple update rule

$$w_\psi^T = V_a[T, \psi] + 1 - \min_{\phi \in \Psi} V_a[T, \phi] \tag{6}$$

$$D_a[T, \psi] \overset{\pi}{\leftarrow} (1-\epsilon) \cdot \frac{w_\psi^T}{\sum\limits_{\phi \in \Psi} w_\phi^T} + \frac{\epsilon}{|\Psi|} \tag{7}$$

where w_ψ^T is the weight of action ψ for a token of type T. Here, the minimum value is subtracted and an offset of one is added to project the estimated value $V_a[T, \psi]$ into the range $[1, +\infty)$. This conveniently allows the weight to be normalized to obtain probabilities in the row of D_a corresponding to T. A small constant factor ϵ is used to ensure that every route is selected with some non-zero probability, thus preventing situations where agents never receive tokens from a particular neighbor.

Combining the two update rules, we formulate the token handling procedure that each agent performs when receiving or generating a token. The resulting algorithm is summarized in Algorithm 1.

4 Experimental Results

In this section, we describe an empirical evaluation of our approach, which we refer to below as a *proportional routing* policy. A simulator was constructed to

Algorithm 1. Handle token.

```
1: $\tau \leftarrow getToken()$
2: $\psi \sim D_a[\tau.T]$
3: if $(\psi \neq \mathcal{S})$ then
4:     Calculate $E_{ba}^{\tau.T}$ using Equation 2
5: end if
6: $V_a = \lambda(V_a, \tau.E)$
7: $D_a = \pi(D_a, V_a)$
8: if $(\psi \neq \mathcal{S})$ then
9:     $\tau.E = E_{ba}^{\tau.T}$
10:    $send(\tau)$
11: else
12:    $delete(\tau)$
13: end if
```

evaluate this policy in a scenario of 500 agents performing 1-D target tracking. This simple setup was sufficiently complex to test the basic dynamics of a large team without introducing irrelevant effects that could obfuscate the performance of the policy. The agents were connected by a small worlds communication network, with an average of about 7 communications links to other agents. 1% of the agents were equipped with simulated sensors that generated Gaussian estimates of the target position every 5 time steps, emulating a discrete sampling rate. Every agent maintained a local Kalman filter using a static motion model with high process noise. As there was only one state variable, we used only one type of token, i.e. $|\Gamma| = 1$. The target was initialized at a random Gaussian position, and proceeded to follow a constant velocity trajectory for the duration of the simulation. Each trial was run for 10^4 time steps. Results were averaged over 10 trials for each condition.

Agent a's need for an accurate estimate was represented by a weight C_a. The distribution of information need over the team was bimodal, with a small proportion of the agents having high need, $\mathcal{N}(10^6, 10^5)$, and the remainder having low need, $|\mathcal{N}(0,1)|$. The objective was to minimize the weighted KL-divergence of the team at the end of the simulation, defined as the following sum:

$$WD = \sum_{a \in A} C_a \cdot \Delta^a(X, P_a(X(t), t)) \bigg|_{t=10^4}$$

Note that this is the first term in the original cost function in Equation 1, applied using KL-divergence as the divergence metric. The value approximation function used by the agents was the covariance of the agent's filter multiplied by the information need constant. By separating this from the communications cost, it was possible to analyze the tradeoffs agents made between communications and value under the test conditions.

Two policies were tested, the proportional routing policy, and a random routing policy. The random policy simply routed each token uniformly randomly for a constant number of transmissions, then terminated it.

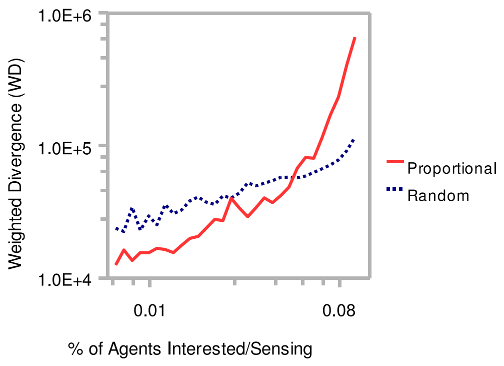


Fig. 1. A comparison of the random and proportional routing policies over changing interest levels

In the first experiment, the weighted divergence of the policies was studied as the percentage of high-C_a and sensor-equipped agents was simultaneously varied. For fair comparison, each trial of the proportional policy was matched with a trial of the random policy which was adjusted such that policies had the same average number of token communications. The resulting weighted divergence in Figure 1 shows that the proportional policy outperforms the random policy in situations where both few agents are high-C_a and few are producing the relevant sensor readings. Specifically, this was the case when the high-C_a and sensor-equipped percentages were below 5%.

As a baseline, the random policy was tested to determine the effects of average token transmissions on weighted divergence. Figure 2 shows that the random policy's weighted divergence drops rapidly as the average number of communications increases at small numbers of transmissions, but flattens asymptotically after about 10 transmissions. This suggests that there may not be much benefit to tokens having extremely long lifetimes, as they will not improve the weighted divergence any further.

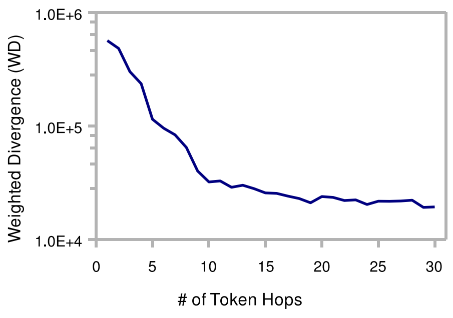
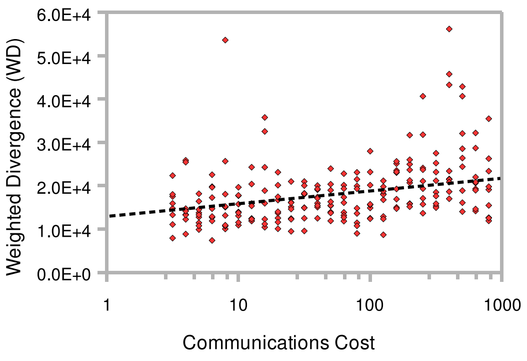
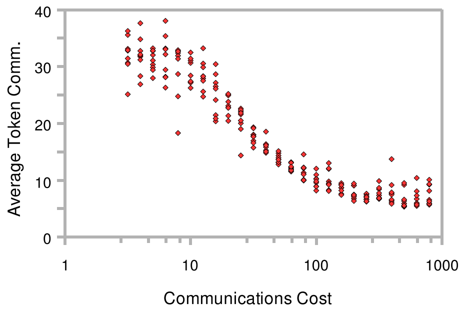


Fig. 2. The effects of increasing token lifetime while using the random policy



(a) Weighted divergence is logarithmically affected by cost. (some outliers are not visible)

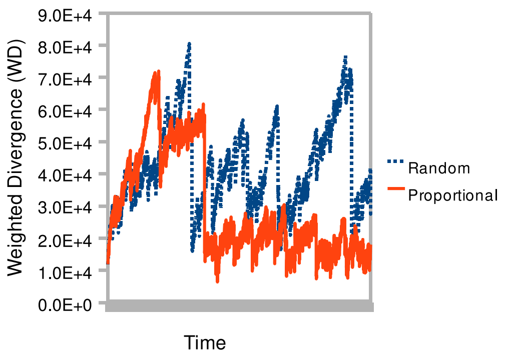


(b) Number of token transmissions decreases log-sigmoidally as cost increases.

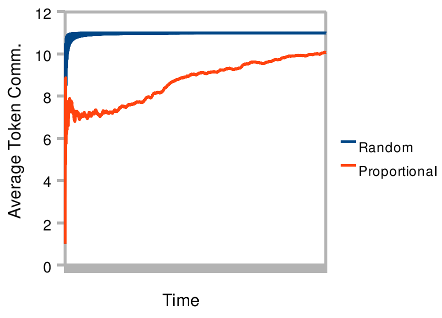
Fig. 3. Proportional routing policy behavior over changing communications cost

Next, an experiment was done to evaluate the effects of the communications cost on the proportional policy. By varying the cost, the behavior of the policy can be adjusted to prioritize decreasing communications over increasing value. In Figure 3, communications cost proportionally impacts the average number of communications, while the relationship with weighted divergence is less evident. It is possible that the reduction of transmissions more directly impacts longer, suboptimal routes before shorter, high-C_a ones, mitigating the effects of the cost increase. This is consistent with the previous observation that token lifetimes above some threshold provided diminishing returns in weighted divergence.

Closer examination of an individual trial, shown in Figure 4, provides insight into the behavior of the algorithm. Weighted divergence steadily increased during periods when high-C_a agents did not receive sensor readings, then dropped sharply when readings were delivered. The spacing and magnitude of these ramps suggest that the proportional routing policy reduces weighted divergence by being more consistent in its delivery of sensor readings to high-C_a agents.



(a) The proportional policy begins routing to high-C_a agents to reduce divergence sharply.



(b) Both policies use the same total number of token transmissions, but the proportional policy uses its transmissions more conservatively.

Fig. 4. A comparison of the random and proportional routing policies over a single trial

These experiments show that the proportional policy is relatively more effective than the random policy when low percentages of the team are producing and consuming sensor readings. This is a region where many current methods are inefficient, hence, this situation exemplifies the type of problem for which this algorithm was designed. The policy is also capable of dynamically adjusting the number of communications as cost increases, while maintaining effective routing. Finally, we see diminishing returns in value as token lifetimes are increasing, suggesting that reasonably short lifetimes are sufficient to maintain low weighted divergence over the team.

5 Related Work

Much previous work focuses on sending beliefs after filtering has occurred, which requires precautions to be taken to avoid "double-counting", in which multiple filter updates include information derived from the same sensor reading. One frequent solution to this problem is imposing an acyclic structure over the network [4] [5], which introduces scalability and dynamics issues. In contrast, this work assumes updates are generated for individual sensor readings. By treating each reading atomically, hashing mechanisms can trivially handle the problem of receiving the same information multiple times by storing a history of token hashes and ignoring revisiting tokens, while still allowing information to be forwarded anonymously, which is useful in a large team [3]. These token histories need only be temporary, as individual tokens have relatively short lifetimes. Note that individual readings could be replaced by any other atomic method of information sharing, such as exchanging particles between team members' particle filters [7].

Token-based methods have been shown to be effective in large-scale team coordination tasks, including the task of information sharing [12]. Using only information pushed forward in tokens, Xu demonstrates that adaptive probabilistic routing can improve team performance [11]. However, these previous approaches to information sharing do not explore token termination conditions, and require tokens or agents to store path histories over their lifetime. Other related methods include a dynamic optimization-based strategy for computing token lifetimes under assumptions of random routing with peer- to-peer communication [9], and an adaptive routing method that uses learning rules similar to Equation 5 to self-optimize routing over dynamic networks.

6 Conclusions

This chapter presented a novel approach to sharing information in large teams using tokens. In contrast to previous work, it was assumed that members of the team had vastly different needs for the information generated by other agents in the team. In our experiments, this need was concentrated among a small percentage of the team. Under these situations, this approach adjusts local routing and stopping probabilities to improve information sharing performance over the team. Empirical results demonstrate this efficiency in a simple simulated tracking problem despite the sparse information agents had with which to make routing decisions.

The experiments also show a number of interesting properties of this approach and the problem of information sharing in teams with asymmetric need. One surprise was the unexpectedly high performance of a random routing policy in reducing divergence, even in highly asymmetric situations. Analytically, it may be possible to bound the optimality of random routing in this problem, and use it as a baseline for comparing other techniques. In contrast, this approach seemed to lose efficiency when faced with large numbers of interested agents. However, hybrid approaches might be possible within this token framework that can use the proportional routing heuristic when routing information destined for a small subset of agents in a team, then switch to a different heuristic when routing more commonly desired information, all using the same estimation methods. If this framework can be extended to work across a wider range, it will provide a lightweight, dynamic approach to sharing information in teams with asymmetric information needs.

References

1. Uryasev, S.: Conditional value-at-risk: Optimization algorithms and applications. Financial Engineering News 14, 1–5 (2000)
2. Drury, J.L., Richer, J., Rackliffe, N., Goodrich, M.A.: Comparing situation awareness for two unmanned aerial vehicle human interface approaches. In: Proceedings of the IEEE International Workshop on Safety, Security and Rescue Robotics (2006)
3. Grocholsky, B.: Information-Theoretic Control of Multiple Sensor Platforms. PhD thesis, The University of Sydney (2002)
4. Makarenko, A., Brooks, A., Williams, S.B., Durrant-Whyte, H.F., Grocholsky, B.: An architecture for decentralized active sensor networks. In: IEEE International Conference on Robotics and Automation (ICRA 2004), New Orleans, LA, USA (2004)
5. Nettleton, E., Thrun, S., Durrant-Whyte, H.: Decentralized slam with low-bandwidth communication for teams of airborne vehicles. In: Proc. of International Conference on Field and Service Robotics (2003)
6. Ortiz, C.L., Vincent, R., Morisset, B.: Task inference and distributed task management in centibots robotic systems. In: AAMAS (2005)
7. Rosencrantz, M., Gordon, G., Thrun, S.: Decentralized sensor fusion with distributed particle filters. In: Proceedings of the Conference on Uncertainty in AI (UAI) (2003)
8. Tambe, M.: Agent architectures for flexible, practical teamwork. National Conference on AI (AAAI 1997), pp. 22–28 (1997)
9. Velagapudi, P., Prokopyev, O., Sycara, K., Scerri, P.: Maintaining shared belief in a large multiagent team. In: Proceedings of Tenth International Conference on Information Fusion (2007)
10. Watts, D., Strogatz, S.: Collective dynamics of small world networks. Nature 393, 440–442 (1998)
11. Xu, Y., Lewis, M., Sycara, K., Scerri, P.: Information sharing in very large teams. In: AAMAS 2004 Workshop on Challenges in Coordination of Large Scale Multi-Agent Systems (2004)
12. Xu, Y., Scerri, P., Yu, B., Okamoto, S., Lewis, M., Sycara, K.: An integrated token-based algorithm for scalable coordination. In: AAMAS 2005 (2005)

Development of a Mission Simulator for Design and Testing of C2 Algorithms and HMI Concepts across Real and Virtual Manned-Unmanned Fleets*

Oktay Arslan, Bahadir Armagan, and Gokhan Inalhan

Controls and Avionics Laboratory
Istanbul Technical University
Istanbul, Turkey
{oktay.arslan,bahadir.armagan,inalhan}@itu.edu.tr

Abstract. The increasing use of unmanned vehicles in civilian (metropolitan traffic monitoring, rapid assessment of disaster areas) and military (reconnaissance, target identification, tracking and engagement) domains has driven critical requirements for the interoperability of manned-unmanned systems. In this work, we provide the design and the development of a flight mission simulator structured for joint real-time simulation across manned-unmanned fleets, and the mission control center. The hardware structure within the network simulator is tailored to mimic the distributed nature of each of the vehicle's processors and communication modules. Open-source flight simulation software, FlightGear, is modified for networked operations and it is used as the 3D visualization element for the pilot and the mission controls. The UAV dynamics and low-level control algorithms are embedded within the xPC target computers. Equipped with 3D flight simulation displays and touch-screen C2 interface at the desktop pilot level, the platform also allows us to rapidly prototype and test pilot-unmanned fleet supervisory control and pursuit-evasion game designs. In addition, the unique design enables seamless integration of real unmanned air vehicles within a simulated scenario. Hardware-in-the-loop testing of network bus compatible mission computers and avionics systems provides us with validation of the C2 architectures and the hardware designs on a realistic lab-scale platform before the actual flight experiments.

1 Introduction

During the last decade, Unmanned Air Vehicles (UAVs) have enjoyed practical success at scenarios involving reconnaissance, surveillance and active tracking. Independent of the role of the unmanned system (ranging from support scenarios to scenarios that are only achievable by customized unmanned vehicles), there

* A part of this work is funded under the DPT HAGU program administered through ITU ROTAM.

are two major drivers that make such systems favorable over manned vehicles. These are the physical working conditions (such as remote locations with harsh terrain or chemical or radioactive spill) and the scenario constraints (such as uninterrupted day long service) that make the operation of manned systems or vehicles impractical, risky or just cost ineffective.

With the ever growing involvement of UAVs in complex application areas (such as dynamically changing urban rescue operations), the types and the number of tasks easily outgrow one vehicle's (or a set of UAV operators' command and control) limited capabilities and thus require a fleet of UAVs to work autonomously in a collaborative fashion to achieve desired operational goals. In addition, the increasing use of unmanned vehicles in civilian and military airspace domains has driven critical requirements for the interoperability of manned-unmanned systems – rather this be for pure collision-avoidance purposes (sense-avoid) or for achieving common mission goals that require significant coordinated actions (mark-image) [7].

In order to benefit from all of these growing capabilities of UAVs, it is essential to develop new control structures and algorithms that allow manned-unmanned system interoperability [4] as abstracted in Figure 1. In addition, the increasing complexities in missions and the safety-critical requirements on the avionics systems demand that all the control and coordination algorithms, the avionics hardware and the vehicles be tested on realistic testbeds [2,9] before using them in the real mission scenarios [6,19].

Towards this end, an experimental network mission simulator, as shown in Figure 2, is developed for joint real-time simulation across manned-unmanned fleets, and the mission control center. The in-house developed mission simulator allows rapid prototyping, software-in-the-loop (SIL) and hardware-in-the-loop (HIL) testing of various coordination algorithms among manned and unmanned fleets, and the mission control center. Open-source flight simulation software, FlightGear [16], is modified for networked operations and it is used as the 3D visualization element for the pilot and the mission controls. The manned vehicle dynamics, UAV dynamics and low-level control algorithms are embedded within the xPC computers using Matlab/Simulink rapid prototyping technique for real-time execution of the mathematical models and control algorithms. Equipped with touch-screen C2 interface at the pilot station, the platform also allows us to rapidly prototype and test pilot-unmanned fleet supervisory control designs [3].

The mission simulator is structured around two distinct bus systems represented by the visualization and the mission layer. In addition, the hardware structure within the network simulator is tailored to mimic the distributed nature of each of the vehicle's processors and communication modules. This allows rapid integration of new hardware elements to both the simulation units, and to the simulation scenario. As such, hardware-in-the-loop testing of network bus compatible in-house developed micro-avionics systems [14] is carried out at flight controls and mission planning levels. One distinct feature of the mission simulator is the ability to integrate real unmanned air or ground vehicles within the

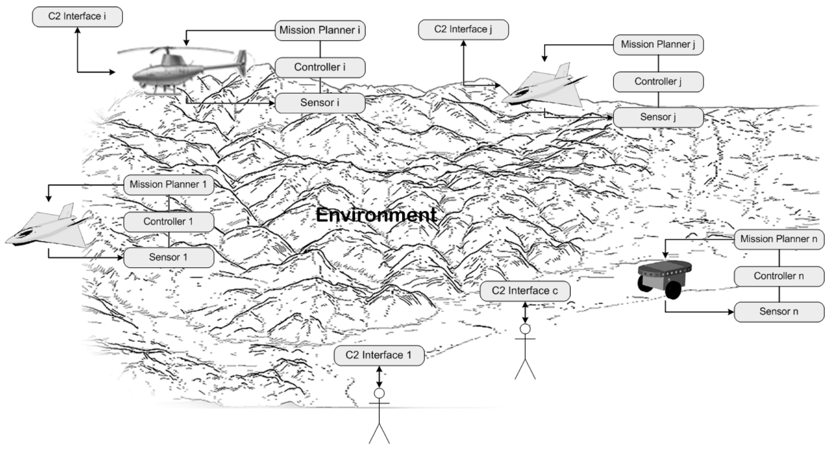


Fig. 1. An overview of the distributed command-control (C2) problem in which a mission is carried out by human operators, and unmanned-manned vehicles. All entities, both humans and vehicles, can be considered as members of a cooperative team working towards accomplishing a common goal. From a software engineering point of view, entities carry common software objects, such as human operators (who can be airborne co-pilots or commanders in the mission control center) can be represented by Command and Control Interfaces (C2I). This functional commonality is carried out to the mission simulator design requirements.

simulation scenario coupled with virtual unmanned and manned vehicles. This allows in-flight testing of vehicles in simulated scenarios.

However these capabilities require data exchange across not only similar units (UAVs), but also across dissimilar units (ground stations, manned vehicles). In addition, the communication and data distribution in the mission layer of the cooperative unmanned-manned vehicle network is complicated by mission-driven information flow requirements. While achieving fleet-level coordination, the unmanned-manned airborne fleet network must also remain in communication with other non-fleet entities such as commander units as well as ground stations, information nodes [13] which forward requests and enhance situational awareness.

Figure 3 illustrates the cooperative unmanned-manned vehicle network from a communication and information distribution perspective. Each entity within this network, such as unmanned-manned vehicle or ground asset, has several local information sources such as sensors, embedded-computers or C2 interfaces. This information may be purely payload data about the environment or could involve the states of the vehicle. In addition, this information can be propagated to the network for data fusion, or this data set may be commands or mission related requests that need to be delivered to an individual vehicle or a group of vehicles in the network in order to achieve the predefined mission collaboratively. Because of such mission-driven information flow requirements, a standalone and multi-functional module is needed for communication and information

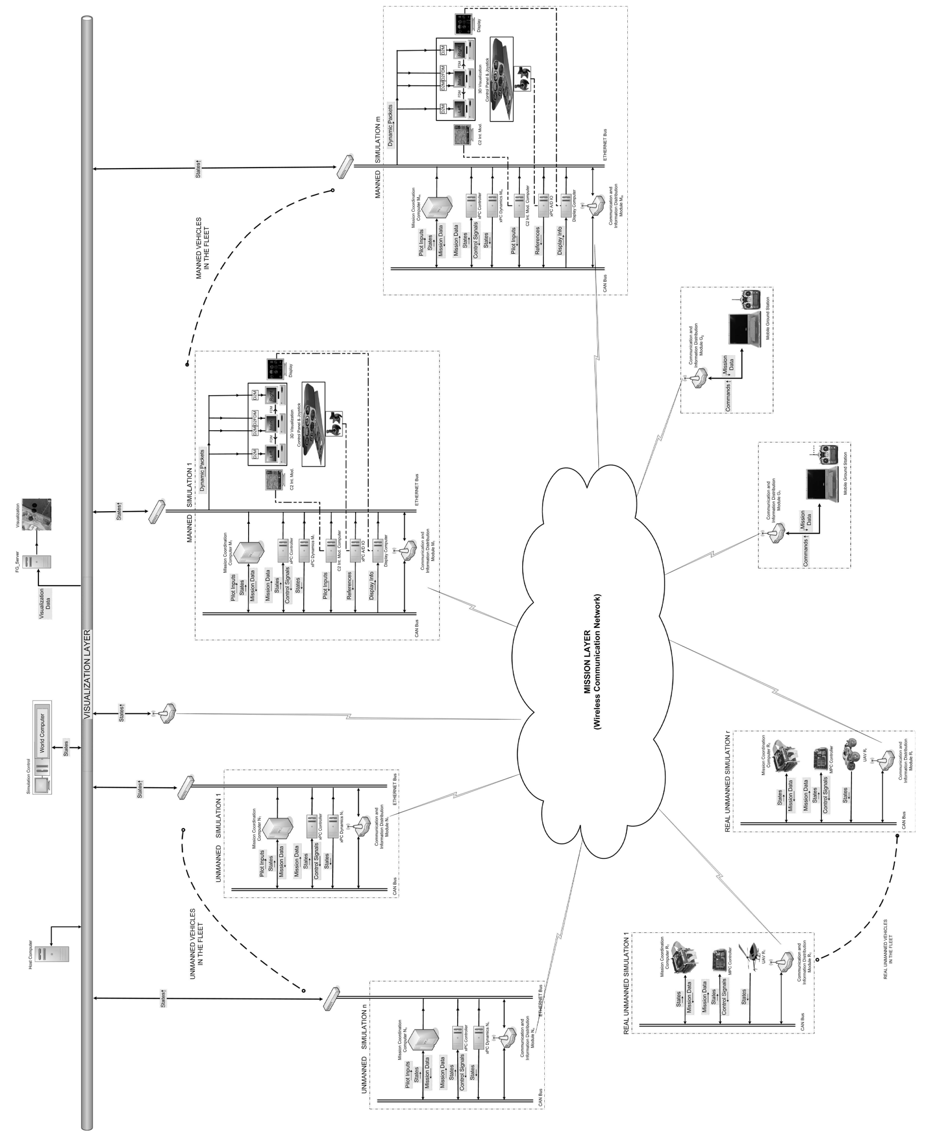


Fig. 2. General architecture of Cooperative Manned-Unmanned Network Simulator

distribution with other entities in the overall network. Driven by this need, we have developed a standardized Command and Information Distribution Module (CID) as a part of the mission simulator hardware. Figure 4 shows the working concept diagram of this module. The module has several communication interfaces including Ethernet, CAN and serial link allowing rapid integration of a

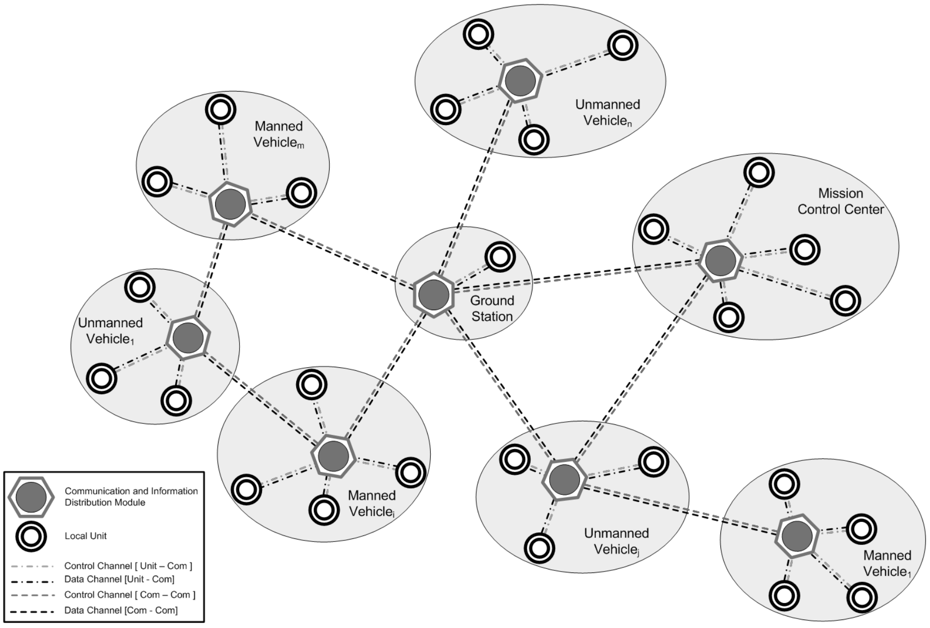


Fig. 3. A conceptual view of the Communication and Information Distribution Problem shows vehicles as nodes, embedded computers and sensors as local information sources, and communication modules

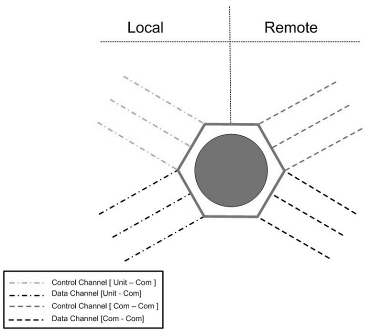

Local
Remote
Control Channel [Unit – Com]
Data Channel [Unit - Com]
Control Channel [Com – Com]
Data Channel [Com - Com]

Fig. 4. Working Concept of the Communication and Information Distribution Module

variety of additional piloted simulation vehicles, virtual or real unmanned vehicles to the existing simulation both at the visualization and the mission layer.

The organization of this work is as follows: In Section 2, we present the general design and the architecture of the mission simulator. First, the unique elements of the visualization layer: simulation control center and the networked Flight Gear implementation is introduced. Second, the CID module which plays an important role within the mission layer is described. In Section 3, the manned simulator and its software/hardware elements are introduced. In this section, as a part of C2 implementations, the prototype of a Human-Machine-Interface display system is described. These units not only serves as the primary flight display, but also serves as airborne (in manned simulation) pilot/operator supervisory control device for unmanned air vehicle fleets in coordinated missions. In Section 4, unmanned vehicle simulator components including virtual and real vehicles are illustrated. In this section, two distinct mission simulator experiments are described. The first experiment involves hardware-in-the-loop testing of a fleet level autonomous target/task assignment algorithm implemented across a UAV network [10]. The second experiment is software-in-the-loop testing of real-time probabilistic trajectory planning [11,12] and mode-based nonlinear control [18] algorithms for a UAV operating within a city-like complex and dense environment. The chapter concludes with the on-going research activities aimed at extending the capabilities of the mission simulator.

2 General Design and Architecture of the Simulator

The general design of the mission simulator is structured around two layers: the visualization and mission layer. These two layers represent two different data bus structures and data flows. As seen in Figure 2, simulation elements such as piloted vehicle simulator, unmanned vehicles, real unmanned vehicles (ground vehicles and micro-UAVs), ground stations and the simulation control computers carry distinct wired and wireless connections to these two data layers.

Visualization Layer entails the passage of the visualization and simulation related data packets (i.e. packets which result in a coherent visual picture of the whole scenario to the operators and the simulator pilot) across the wired ethernet network using UDP packets. The visualization layer uses open-source FlightGear flight simulator packet structure to allow direct integration to the flight simulator visualization elements. These visualization elements include the three panel environment display for the pilot of the manned simulator (as shown in Figure 5) and the pilot/operator panel for tactical/simulation displays. The *Mission Layer* is accomplished via wireless communications (serial and Ethernet) across each unique entity existing within the simulation using predefined data packet numbers and structures. Mission critical data such as target assignments, payload readings, commands and requests are delivered through this wireless mission layer link.

The reason for this layered approach stems from the need to differentiate the real operational wireless communication links in mission scenarios from the wired communication links to obtain visualization and simulation information for manned vehicle simulator and the operators. This break-up of layers not only represents a realistic mission implementation, but also allows HIL testing and analysis of wireless network structures (point-to-point, point-to-multi point, ad-hoc) that mimic the real operational communication links as it would be implemented and experienced in a real mission. The modular design of the mission simulator elements is tailored according to the need for joint real-time simulation

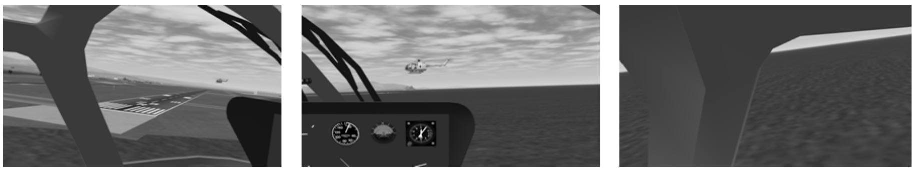


Fig. 5. Multiple monitor visualization as seen by the pilot during a joint mission with unmanned helicopters

across groups of manned and unmanned fleets, and the mission control centers. As such, the hardware structure within the network simulator is tailored to mimic the distributed nature of each of the vehicle's processors and communication modules. Note that the layered bus approach to the visualization and the mission layer allows direct integration of new manned and unmanned simulation elements within the scenario once linked to the corresponding wired and wireless buses.

In the next two subsection, we will review the structure of the simulation control center and basic building blocks of the visualization and the mission layers. These are the FlightGear software implementation used in the visualization layer, and the communication and information distribution modules used in the mission layer. In the two sections to follow the hardware and software elements for both manned and unmanned vehicle simulations, and the C2 implementations in typical mission scenarios, will be discussed.

2.1 Simulation Control Center and FlightGear: Visual Layer Implementation

Network simulator visualization requires integration of components with different specific purposes. Visualization includes cockpit simulation with multiple displays and a tactical interface for visualizing simulation objects on a 2D map. Integration of these components requires multiple node communication over the Ethernet network. Underlying network packet structure is the key point of the communication. Well defined packet structure of the network also enables the expendability of visualization with any additional visualization component such as a 3D projected visualization or another display system for piloted simulation. For this reason, an open source flight simulator with networked operation option, FlightGear and its OpenSceneGraph visualization element, is used for both simulation and visualization of the mission scenario. The packet structure within the visualization layer follows the FlightGear native data format identified by three distinct packet structures: Dynamic States Packet (D), FlightGear Multiplayer Packet (M) and Flight Dynamics Packet (FDM). D packets represent vehicle dynamic states such as position, orientation and velocity information supplied by the computers running the mathematical model of each of the manned and unmanned vehicles. M packets include information about position, orientation, velocity and the model of aerial vehicle. FDM packets include generic type and configuration information about the aerial or ground vehicles.

The simulation and the visualization is controlled by the simulation control center consisting of three elements: simulation operator computer, world computer and the FlightGear server:

Operator Panel is used for configuring the network setting of the simulator. All information about the simulation (numbers of manned-unmanned computers, IP addresses of the machines) are input from this GUI.

World Computer is used for controlling of the packet traffic of the visualization layer. Basically, an in-house developed program called State Sender runs on this computer and makes packet conversion from Dynamic (D) packet to the Multiplayer (M) or Flight Dynamic Model (FDM) packets. M and FDM packets are needed for visualization of the simulation. State Sender, being an add-on module to FlightGear, basically is a piece of C++ code that takes dynamic states information of a vehicle or any other simulation object and when it receives such information it makes the packet conversion. All the unmanned-manned computers which run the mathematical models of the vehicles (xPC computers) in the simulator send their states as Dynamic (D) packet to these programs at a specific frequency and State Sender programs convert the dynamic (D) packet to the multiplayer (P) packet and send them to the FlightGear server and the 3D Multi-monitor visualization systems of manned simulator. In addition to that, for the manned simulator, the State Sender program makes a dynamic (D) to Flight Dynamic Model (FDM) packet conversion and sends the FDM packets to the 3D Multi-monitor visualization system for cockpit visualization of the manned simulator.

FlightGear Server (fg server) is "an Internet and LAN based multiplayer server for the FlightGear Project. It provides services to link pilots together so they can see each other when flying"[1]. Fg server is designed for communicating with FlightGear only. However, if the data structures of FlightGear are implemented, other programs may send packets to, and thus talk with fg server. Since both FlightGear and fg server projects are open-source projects, group simulation of aerial and ground vehicles can be accomplished easily with a separate program that carries the main data structures as implemented in FlightGear. Fg server in mission simulator runs in a separate computer, listening for incoming connections. The protocol for exchanging packets in fg server is UDP. Through the Fg server not only we can create a complete list and picture of all the simulation elements within the mission, but also generate tactical display information to the pilot/operator and the simulation controllers by linking to the server.

2.2 Communication and Information Distribution Module (CID): Mission Layer Implementation

CID module is an in-house developed communication and information distribution module for heterogenous teams of unmanned and manned vehicles, and

[1] http://fgms.sourceforge.net/

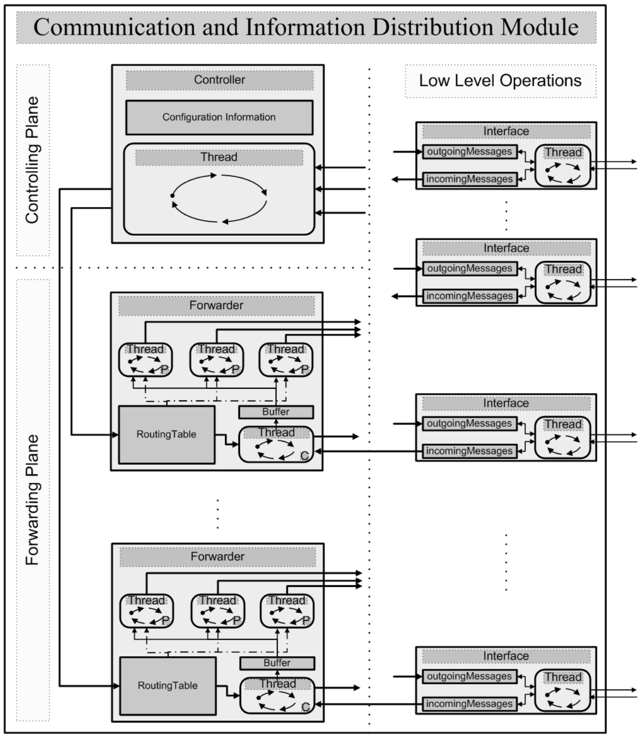


Fig. 6. Basic CID module structure: thread interactions and shared data

ground assets. The module implements a customized routing algorithm that takes the data from these different communication interfaces and routes to the destination addresses according to the real-time updated routing tables. CID module's objective is to gather control messages which are sent by other entities in the network from control channels, and serve the requests or commands according to the control message types. Figure 4 shows the working concept diagram of this module.

The module has several communication interfaces including Ethernet, CAN and serial link. Our first hardware implementation of the module is based on a PC-104 structured card stack which includes ethernet, CAN and serial links. Currently, the module code is being ported to run on Gumstix hardware modules to result in considerable weight advantage as a stand-alone unit. Each local processor unit of a vehicle and the remote CID modules of other vehicles are connected to the module with both data and control channels. Data about environment, states, or other specific information are gathered from local units and distributed to specific destinations via data channels. Also, routing of information flow is changed according to the commands, requests and special messages taken via the control channels.

The software CID module is written in C++ and boost thread library was used for thread management [8,17]. General architecture of the software was developed as modular as possible by using a highly object oriented design methodology. Figure 7 illustrates an overview of the class hierarchy used in the software. Since there are numerous threads need to be synchronized during execution of the code, classical thread synchronization problems were encountered (producers-consumers and readers-writers problems). Monitor concept with Mesa semantic is utilized in order to solve these classic problems.

Functional Semantics. The operation of the CID module can be separated into four systems: configuring, controlling, forwarding and transporting information systems. There are three main packet types as depicted in Figure 7 Use Case diagram: configuration packet, control packet, and data packet. First of all, the CID module has to be configured according to special configuration data which includes the name of the communication interfaces, parameters, address of units and the other CID modules in the networks. When the module turns on, *Configure System* handles all operations related with initialization of connection tables, creation of controller objects and all other control communication interfaces.

Basically, the CID module operates in two different planes: the control plane, and forward plane. Parsing of control messages taken via control channels and controlling of the data channels are handled in the control plane. *Control Information* use case handles operation related with parsing of control messages. One controller objects running in this plane manages the preparation of the data communication interface object and the forwarder objects according to control messages. This object can control the information flow by updating routing tables of forwarder objects. Forwarding plane is responsible for the continuous or periodic routing of the data taken via data channel. *Forwarding Information* use case has several subtasks related with updating of routing table, periodic and continuous routing operations. There may be some data routing requests from other clients, and in this case controller object can update the routing table by using routines of forwarding plane. Routing of information may be requested as continuously or periodically. These two planes heavily use low level functions which are implemented in Transport Information use case and these two planes perform the actual process of sending and receiving a packet received on a logical interface to an outbound logical interface. *Transport Information* use case includes operations about configuration of several physical communication interfaces including CAN, Ethernet and Serial. Low level send and receive operations are implemented by functions in this use case. The overall module software implementation is tested under several cases transmitting data between not only similar units, but also dissimilar units. The specific test cases used for inter-vehicle and intra-team data include ad-hoc network structures resulting from Ground Station and Mission Coordination Computer connections.

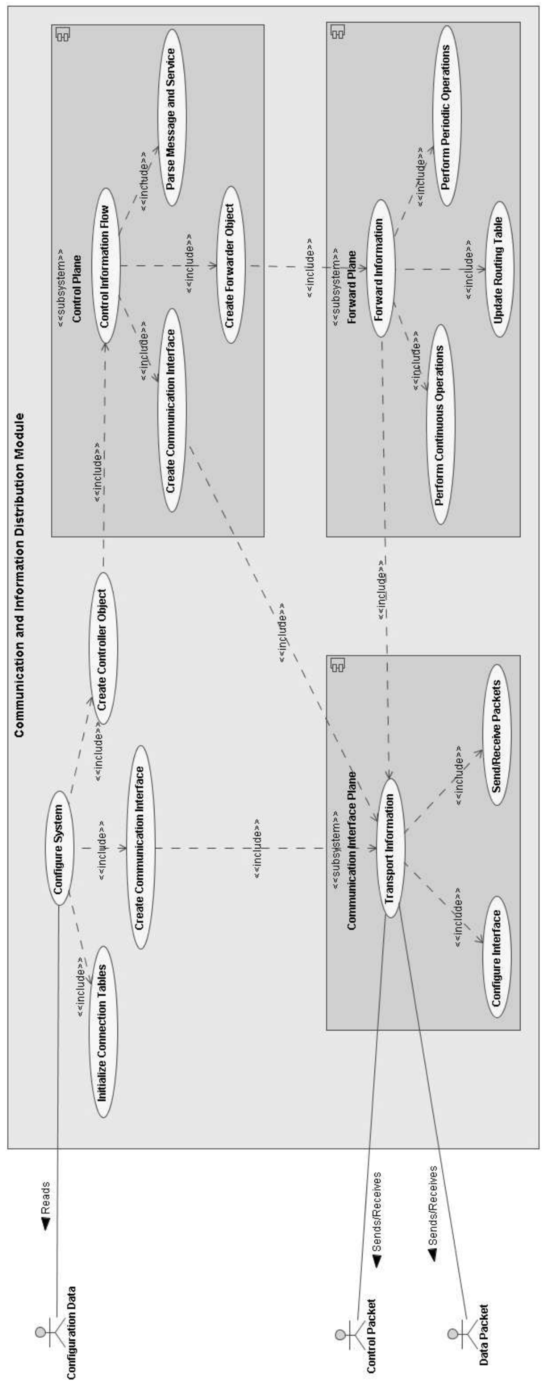


Fig. 7. Level 2 use case diagram for Communication and Information Distribution Module

3 Manned Vehicle Simulation

Manned vehicle simulation system as shown in Figure 8 consists of a generic pilot station and a dedicated computer rack. The generic pilot station is equipped with both standard helicopter (cyclic, pedal, collective, throttle) and aircraft (thrust, flap) pilot input devices. In addition, the pilot could employ a four degrees of freedom joystick as a side stick. The pilot console includes a selection of switches (such as engine, autopilot, stability augmentation system (SAS) on-off) and warning lights (such as altitude, engine failure) that can be configured for different operational needs and conditions. This is illustrated in Figure 9. The pilot station is equipped with a three-monitor visualization system that provides scenery information of the flight scenario from FlightGear. The computer rack system includes a set of dual bus (CAN and ethernet) connected computers for pilot input A/D conversion, real-time execution of the vehicle mathematical model, mission and flight controls, and C2 pilot/operator interface. Following the schematic in Figure 8, we present in detail each of the elements within the pilot station, the computer rack system, and their respective working schemes.

3D Multi-monitor Visualization. Multiple Display System enables 3D cockpit simulation in FlightGear environment. Multiple monitor configurations require three separate personal computers to be connected with each other through the Ethernet network. Multiple display synchronization property of FlightGear provides flight dynamics (rather this be native to FlightGear or external driven by a computer) exchange between two or more FlightGear programs running in

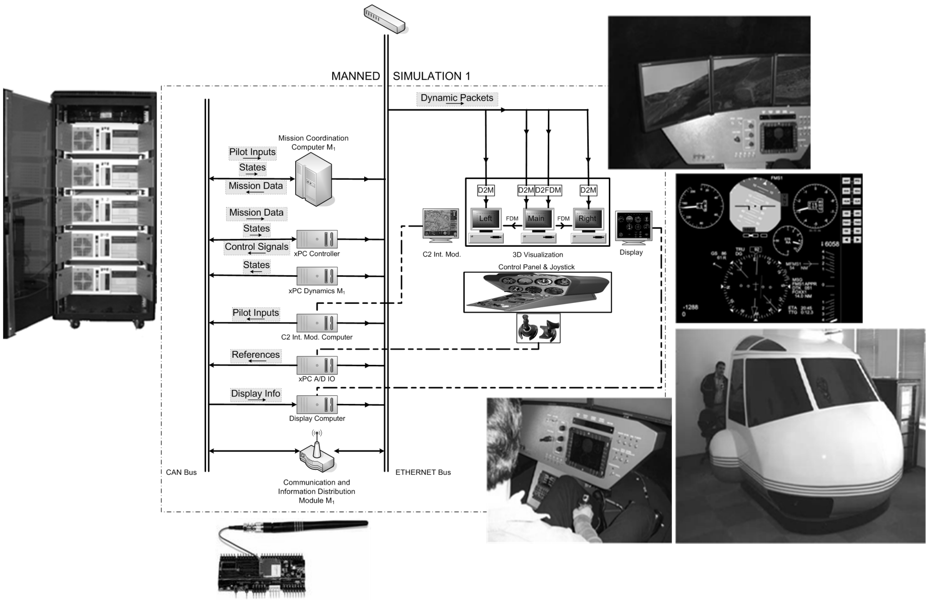


Fig. 8. Components of the manned simulation

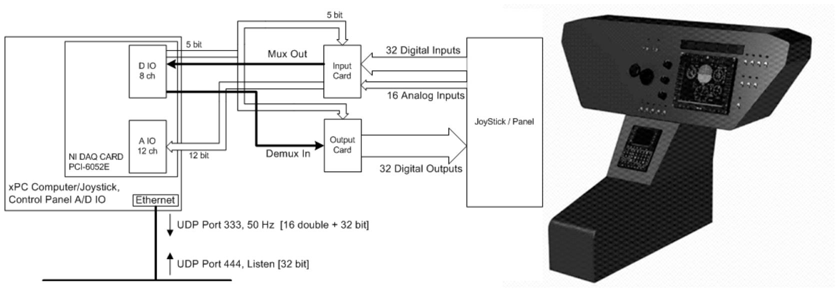


Fig. 9. The pilot panel and the connection of the pilot controls to the simulator

different computers. Computers used for multiple display system are equipped with high capacity graphics cards, and these cards are connected to wide screen LCD monitors. Each of these monitors are configured and located to certain positions for establishing realistic cockpit simulation. In cockpit simulation mode, two computers run their simulation with the FDM packets that they receive from the main computer which is accepted to be the central (or main) computer. Specifically for the piloted simulator, central computer running FlightGear (rather this be in native or external dynamics mode) captures the flight dynamics of the current aircraft, and sends them to the left and right computer over the visualization layer.

xPC Computer–Dynamics. The mission simulator has the capability of simulating a wide-range of vehicle dynamics including both FlightGear's built-in and in-house developed aircrafts' and helicopters' models. The Mathworks's xPC Target product is used for real-time simulation of different vehicle dynamics and testing of different control algorithms. Therefore, dynamic models of several types of air vehicles such as helicopters (EC-120, Vario) and fighter aircrafts (F-16, F-4) have been developed in Matlab/Simulink environment, and then these models are compiled into C code via Real-Time Workshop. Finally, the automatically generated C code is downloaded to a low-cost computer, named as xPC Computer, running the xPC Target real-time kernel via Ethernet connection.

One of the dynamic models that is frequently used in the mission simulator is a high fidelity model of an F-16 aircraft from a NASA report [15]. This model includes a six degrees of freedom full envelope flight dynamics, implements engine and atmosphere models, and has four direct input channels of an aircraft; elevator, aileron, rudder and throttle. In addition, the model is enhanced to include actuator saturations and rate limitations.

This model can also be used as a good representation of flight dynamics of an unmanned combat air vehicle. By designing reference tracking control laws, it is possible to autonomously fly agile maneuvers with the aircraft. In general it is a very challenging task to describe general motion of an unmanned air vehicle, and design a single control law that handles the complete reference tracking needs for such maneuvers. From the inspection of the well known smooth aerobatic

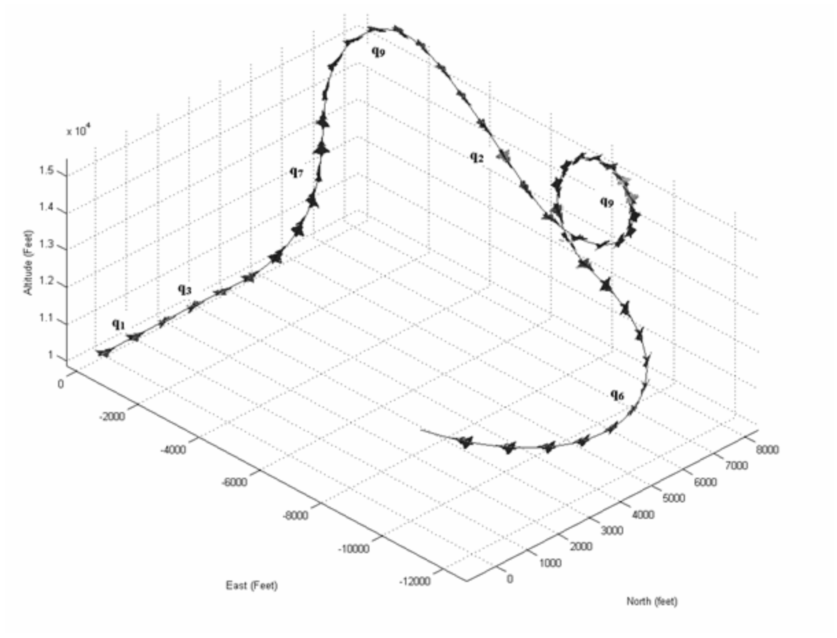


Fig. 10. Mode Based Agile Maneuvering

maneuvers and more complex combat maneuvers, we see that this task can be quantized by decomposing general maneuvers to maneuver modes. Via these modes both the system dynamics and the control task is simplified. Using these modes, we have developed a structured finite state automaton that spans the full-flight-envelope maneuvers and designed nonlinear sliding manifold control system to the maneuver sequences resulting from this automaton [18]. Figure 10 represents the agile maneuverability of this control approach, flying the F-16 autonomously through a series of complex combat maneuvers.

xPC Computer/Pilot Input, Control Panel A/D IO. This xPC computer employs a set of National Instruments A/D input-output card, CAN interface card, and in-house built multiplexers. As illustrated in Figure 9, the control inputs from the pilot including flight controls and the console come in the form of both analog and digital signals. The signals are converted to CAN messages through this unit. Later, the pilot input CAN messages are fed back to the xPC computer executing the dynamic model.

xPC Computer/Controller. This computer provides the ability to implement the controller (stability augmentation or autopilot algorithm) of the manned vehicle external to the dynamic model. Due to its fast development and rich built-in blocks, Matlab/Simulink environment is chosen for development of the controller algorithms.

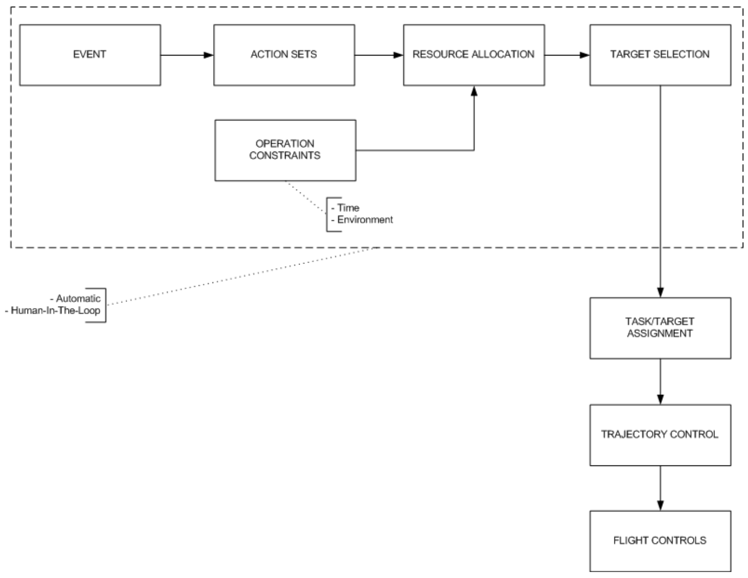


Fig. 11. Event-Driven Decision Support Architecture

Mission Coordination Computer. (MCC) is used for coordination between other manned-unmanned vehicles in the fleet when performing a mission collaboratively. Higher level planning and coordination algorithms are run on this computer. Specific usage of the MCC, which is also an integral component of the unmanned vehicles will be described in the next section. One of the unique applications associated with the mission simulator is development of new human-machine-interface designs that provide unmanned vehicle group command and control capability to the pilot/operator screens. An extension of the primary flight displays with this capability is illustrated in the next subsection.

3.1 C2 Application: Expansion of the Human-Machine-Interface (HMI) to Decision-Support System for Manned and Unmanned Fleets

Distributed command and control of vehicles in a dynamically changing environment is a challenging problem as the number of the vehicles and complexity of the mission increases. In order to solve this challenging problem, one approach that can be used is to change the role of manned vehicle from low level commanding of unmanned vehicle to supervisory commanding [3]. However, it is not always feasible to apply basic supervisory command and control for a large number of unmanned vehicles performing complex missions with strict time constraint. Therefore, a decision support or autonomous decision-making system can be designed for the commander that decrease the workload. Figure 11 illustrates the overall process of such a decision making support system.

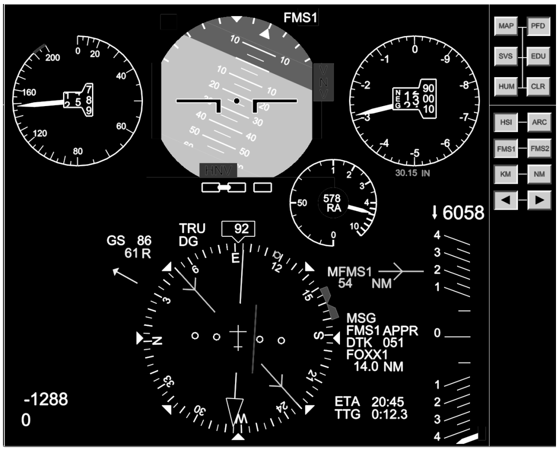


Fig. 12. Primary Flight Display of the manned simulation

A key part of such an event-driven process interaction hinges on the UAVs to autonomously coordinate the target/task assignments and distributions. The experimental illustration of this within the mission simulator will be illustrated in the next section. However, it is important to note that supervisory control and interruption is desired in all mission critical phases. Towards this goal, we have developed a Human-Machine-Interface display systems which allow this supervisory control capability over unmanned vehicle networks over the manned simulation platform. In addition to showing manned vehicle flight information data, this computer also tracks and monitors the action of unmanned vehicle fleet within the joint mission.

On the primary pilot interface flight and mission information is displayed in 5 separate screens. These screens are MAP, Primary Flight Display (PFD), Synthetic Vision System (SVS), Engine Display Unit (EDU), and Health and Usage Monitoring System (HUMS). The primary display functionality is illustrated in Figure 12. SVS [1,5] screen provides significant situational awareness with 3D view of terrain and aerial vehicles. A view of SVS system is shown in Figure 14.

In addition, a C2 pilot/operator interface module is designed as a GUI based program that interacts with the pilot. This module can be seen in Figure 13. Basically, this interface takes dynamic model packets of each vehicle and other commands from different components of network simulator as inputs, and then serves this information on a 2D map to the pilot as basic as possible. This interface mainly has two parts; a 2D map which shows other vehicles, way points, trails, and a command panel that contains several commands like assigning a vehicle for a specific task. It has capability of interactive assignment of next waypoint by simply "touching" an aerial vehicle in view of the screen.

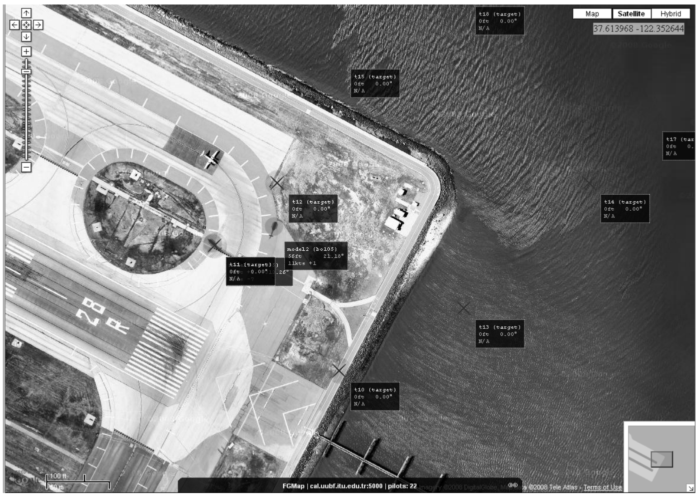


Fig. 13. Command and Control Interface GUI

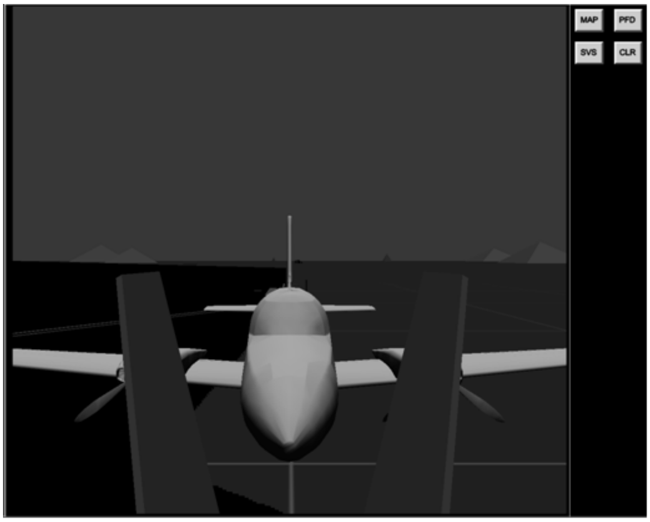


Fig. 14. SVS screen view

The display hardware consists of a PC with touchscreen LCD displays. The PC has TCP port for intercommunication between the display and the mission simulator, and USB ports are used for various data loadings such as streaming video, terrain databases, and data recordings such as flight data. The touchscreen input capability provides smooth operator interfacing and a flexible software design environment. The interface software consists of separately developed graphical objects tightly coupled with each other based on a state machine. Every object in the object oriented hierarchy is run by a state machine. The operator can navigate though screens by an adaptive menu, which changes its navigation button positions and functions.

Current work on the interfaces is aimed at a) increasing ergonomics of tactical operations and navigational experience by means of an 3D artificial tunnel-in-the-sky which has real-time reshaping capability for pursuit-evasion and navigation scenarios, and b) enabling voice control commands for the C2 interfaces across manned-unmanned vehicle joint operations.

4 Unmanned Vehicle Simulation

Unmanned vehicle simulation integration within the mission simulator can be achieved through virtual vehicles and real unmanned vehicles. The virtual unmanned simulation is run on a rack of standard PCs which has four components: mission coordination computer (MCC), xPC computer/controller, xPC computer/dynamics, and CID module. As there are two types of information flow in the network, namely mission information and visualization information, there are two different communication buses: CAN and Ethernet bus. CAN bus is used for mission related information flow, while Ethernet is used for data flow to the visualization system. The components of the virtual unmanned vehicle (VUV) simulation is shown in the Figure 15. The real unmanned vehicles connect to the mission layer through their own CID module and vehicle information data is transported to the visualization layer through the mission simulator's own dedicated CID module which collects data from micro-air or ground vehicles naturally flying or driving outside the lab environment.

In the next subsections, we review the C2 mission algorithmic implementations on both the virtual and the real unmanned vehicles.

4.1 Virtual Unmanned Air Vehicle

The simulation rack structure of the virtual unmanned air vehicle is identical to the manned system except for the pilot controls and the C2 interface computers. The system consists of four computers including the CID module. Each individual vehicle has to use CID module in order to join the mission network. This CID module implements a simple communication protocol that is used among components of the network for communication and information distribution. xPC Computer/Dynamics is used for executing the dynamics algorithm of the unmanned vehicle in real-time, while the control computer provides the closed-loop control and command functionality. Mission Coordination Computer is used for coordination between other manned-unmanned vehicles in the fleet when performing a mission collaboratively. The program structure that is implemented on MCC, has three sub modules; communication module, flight mission module, and task module. As shown in Figure 16, the Flight Mission Module

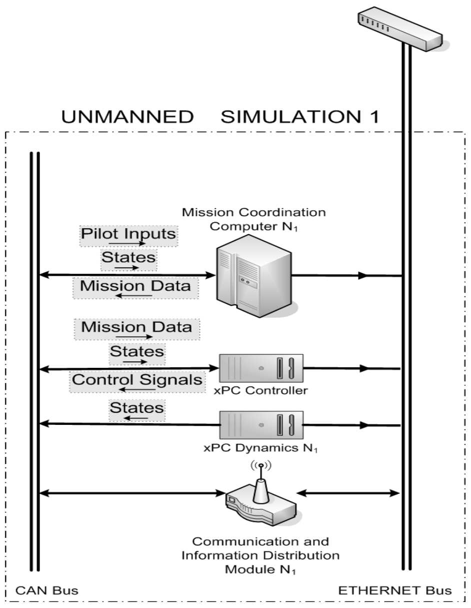


Fig. 15. Components of Virtual Unmanned Vehicle Simulation

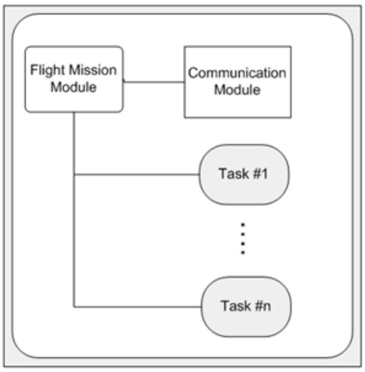


Fig. 16. Mission Coordination Computer Program Architecture

is responsible for data flow control and sequencing between tasks. CID module is used for communication with other MCCs, and the task modules may be any special coordinated planning algorithm such as task assignment or a specific pursuit and evasion implementation between two vehicles.

In the next subsection, we review a specific implementation of an autonomous large-scale target-task assignment algorithm as implemented across a UAV network within the mission simulator.

4.2 C2 Implementation: Hardware-in-the-Loop Testing of a Large-Scale Autonomous Target-Task Assignment Problem for a UAV Network

As a first step coordination algorithm implementation within the mission simulator, we considered one of the basic problems. The problem consists of n targets to be visited by m UAVs and the vehicles should autonomously find the way-point selections that results in the minimum total path traveled by the UAV fleet. In addition, all the mission coordination and the communication between the units had to be done autonomously without any human intervention. To address the first step of this problem, we have developed a large-scale distributed task/target assignment method that allows autonomous and coordinated task-assignment in real-time for a UAV fleet. By using delayed column generation approach on the most primitive non-convex supply-demand formulation, a computationally tractable distributed coordination structure (i.e. a market created by the UAV fleet for tasks/targets) is exploited. This particular structure is solved via a fleet-optimal dual simplex ascent in which each UAV updates its respective flight plan costs with a linear update of way-point task values as evaluated by the market. The complete theoretical treatment and algorithmic structure of this problem can be found in [10]. The basic implementation of this within the network simulator is given in Algorithm 1:

In this particular experiment, we use the Mission Coordination Computers (MCCs) to embed the real-time implementation of the task/target assignment algorithm and run simulations if they are in a real mission. Vehicle dynamics and the low level control algorithms for each UAV are embedded within their unique xPC target modules. These modules simulate an autonomous UAV which receives a sequence of targets from a UDP based communication channel and executes these commands to reach the targets. MCCs run a C program that is used as a bridge for the communication layer between the UAVs and the Matlab based computation layer of each of the UAVs. Each UAVs MCC send its respective targets to the UAV control systems during the execution of the target assignment algorithm. The world computer which is responsible for simulation of the mission informs the MCCs if a new target appears in the scenario. Figure 17 provides a detailed look at the computational behavior of the algorithm. The data points correspond to average values of 600 different simulation scenarios. Note that a total of 250 targets can be solved across 5 UAVs in approximately 50 seconds. This illustrates the real-time implementation capability of the algorithm in comparison to the only off-line capable standard integer programming approach [10]. The numeric solution to a scenario covering 500 waypoints with dynamic pop-ups is given in Figure 18. Figure 19 provides the extensive numerical verification of the restricted horizon polynomial-time computational growth property of this distributed formulation.

4.3 Real Unmanned Air and Ground Vehicle Simulation

One of the unique features of the mission simulator is the fact that the existing real unmanned micro-air and ground vehicles within the laboratory can connect

Algorithm 1. Distributed Synchronized Implementation of Autonomous Target-Task Selection Algorithm for i_{th} UAV

```
Input: $n$ waypoint positions and the $i_{th}$ UAVs position.
Output: The fleet shortest path optimal waypoint assignments for $i_{th}$ UAV
    PHASE I: Initialization
 1: Compute the distances between the waypoints and form the $n \times n$ matrix $V$
 2: Compute the distance to the waypoints and form the $n \times 1$ matrix $d_i$
 3: for every possible path $i_{th}$ UAV can travel do
 4:     Add the path to the matrix $D_i$
 5:     Calculate the cost of this path to vector $c_i$
 6: end for
 7: Read the $(n+m) \times (n+m)$ initial predefined $B$ route matrix from memory
 8: Communicate The costs for corresponding paths in the initial $B$ matrix to start
    the master iteration
    PHASE II: Dual Simplex Ascent
 9: repeat
10:     Solve the restricted master iteration and find the dual variables vector: $[\pi \quad \mu]' = [\bar{c}_1' \dots \bar{c}_m']' B^{-1}$
11:     Remove the ineffective flight path (if any) with maximum cost
12:     Solve the sub-problem corresponding to the $i_{th}$ UAV by updating the flight
        costs: $c^q_{i\,updated} = c^q_i - \pi d^q_1 - \mu_i$
13:     Select the minimum negative $c^q_{i\,updated}$ (sub-problem cost) and the associated
        path
14:     Exchange the selected path and its flight cost and the paths and the associated
        costs generated by other vehicles
15:     Insert the first feasible flight path with minimum cost solution and its cost to
        the restricted master problem.
16: until There are no paths with negative subproblem costs
```

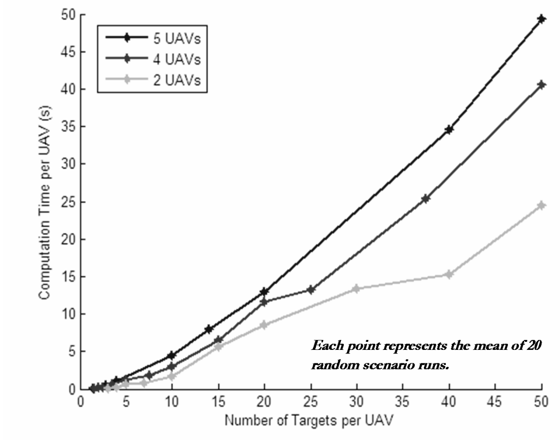


Fig. 17. The computational behavior of the algorithms shows strong correlation with the number of way-points per UAV given a particular snap-shot of scenario

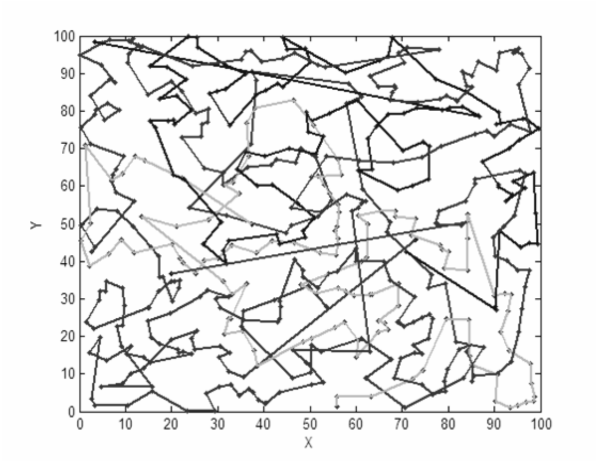


Fig. 18. The waypoint routes for a random pop-up task-target assignment for four vehicles. The algorithm is implemented further in the receeding horizon mode for five hundred waypoints.

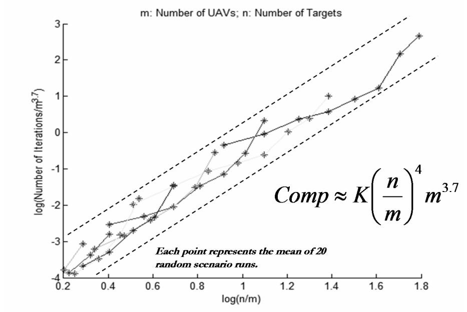


Fig. 19. The numeric computational complexity analysis suggests polynomial-time growth for restricted time horizons

to the mission simulation scenario using their customized CID modules. This feature not only provides in-flight verification capability of mission coordination algorithms, but also creates a financially less cumbersome testing environment in which some of the vehicles are simulated instead of being flown. Figure 20 shows how the laboratory unmanned air and ground vehicles are perceived within the mission network once connected to the mission simulator through their respective CID modules.

Mission Coordination Computer. In the avionics system of the unmanned vehicles, instead of a desktop PC, an embedded PC104 computer and an ARM processor is used for mission planning algorithms.

MPC555/Controller. Similarly, in the real avionics system an embedded MPC555 computer is used for running the real-time controller algorithms. Since

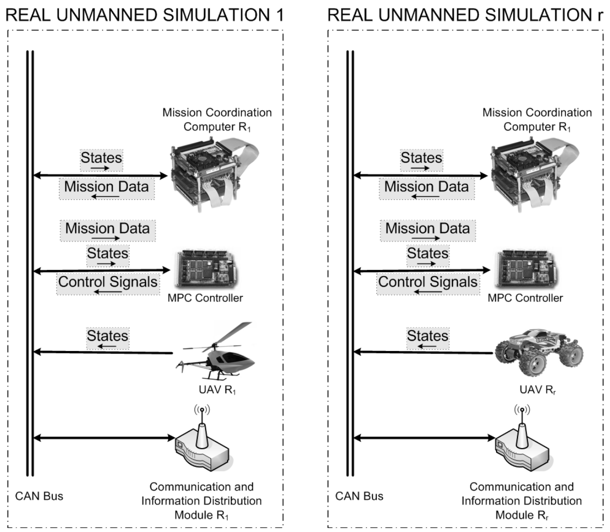

REAL UNMANNED SIMULATION 1
REAL UNMANNED SIMULATION r
Mission Coordination Computer R1
States
Mission Data
Mission Data
States
Control Signals
MPC Controller
States
UAV R1
UAV Rr
CAN Bus
Communication and Information Distribution Module R1
Communication and Information Distribution Module Rr

Fig. 20. Components of Real Unmanned Simulation

this embedded processor can also be programmed by Matlab/Simulink, there is almost no extra effort to modify the controller algorithms used in xPC computer in virtual simulations. This results in the ability of software-in-the-loop verified algorithms to be directly ported to the real hardware for hardware-in-the-loop and real flight testing.

Real UAV/UGV Avionics System and the hardware-in-the-loop integration to the mission simulator. A collection of in-house modified real UAVs/UGVs (including a Vario Acrobatic Helicopter, Trainer 60 fixed wing platform and a 1/8 scale Humvee [9]) each equipped with sensors, embedded processors can be connected to the mission simulator during simulated scenarios. The cross-platform compatible architecture of the in-house developed microavionics system [14] is illustrated in Figure 21 and it used on all the unmanned platforms with minor modifications. This bus based microavionics system is structured around a main CAN bus and all the components including flight and mission controller boards and sensors communicate with each other via this bus. Since each sensor has different type communication interface such serial or I2C links, a simple circuit named SmartCan with universal interface is designed to connect these sensors to the CAN Bus. SmartCAN basically makes packet conversion between different types of communication interfaces to the CAN interface. These vehicles can be operated autonomously or remotely by a human pilot. The operation mode selection is achieved by the means of a switch board that allows switching of the controls. In addition to the standard hardware-in-the-loop testing capability for flight controllers, the mission simulator also allows debugging of flight test problems through hook-up of the CAN Simulator to the network. The

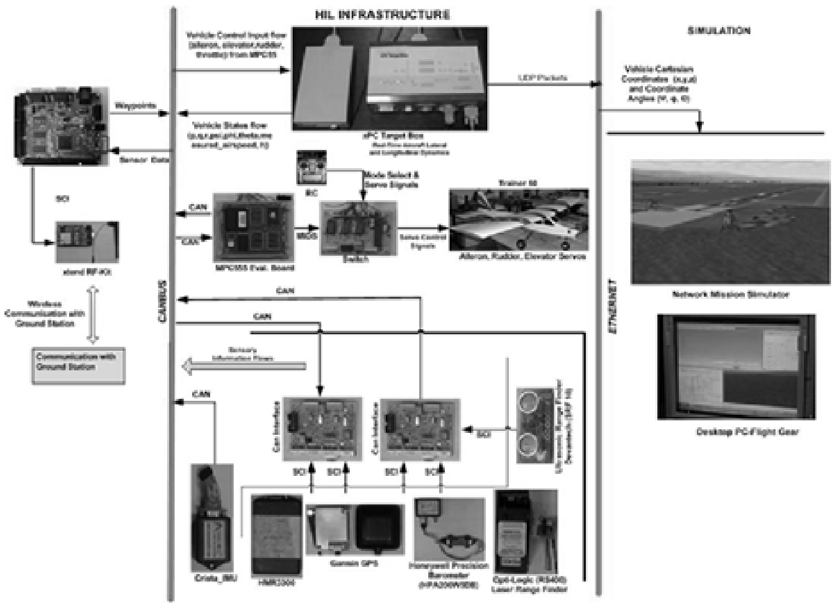

HIL INFRASTRUCTURE
SIMULATION
Vehicle Control Input flow (aileron, elevator, rudder, throttle) from MPC555
Waypoints
Vehicle States flow (u,g,r,psi,phi,theta,measured_airspeed, h)
Sensor Data
xPC Target Box
UDP Packets
Vehicle Cartesian Coordinates (x,y,z) and Coordinate Angles (ψ, φ, θ)
SCI
RC
Mode Select & Servo Signals
CAN
MIOS
MPC555 Eval. Board
Switch
Servo Control Signals
Trainer 60
Aileron, Rudder, Elevator Servos
Network Mission Simulator
Wireless Communication with Ground Station
Communication with Ground Station
CANBUS
ETHERNET
Sensors Information Flows
Can Interface
Desktop PC-Flight Gear
Crista_IMU
HMR2300
Garmin GPS
Honeywell Precision Barometer (HPA200W5DB)
Opti-Logic (RS400) Laser Range Finder

Fig. 21. Cross-platform compatible micro-avionics system and the hardware-in-the-loop integration to the mission simulator

CAN Simulator machine is capable of playing back and emulating the sensory information of flight test data collected in the flight tests.

Ground Station. Ground station computer, as seen in Figure 22, is used for tracking the vehicle and the mission data, creating new missions, and sending special commands to any vehicle while the mission is being performed. This computer is also connected to the network via CID module like other vehicles.

The complete integration of the real unmanned vehicle platform (including ground stations) also enables us to do complete mission related performance

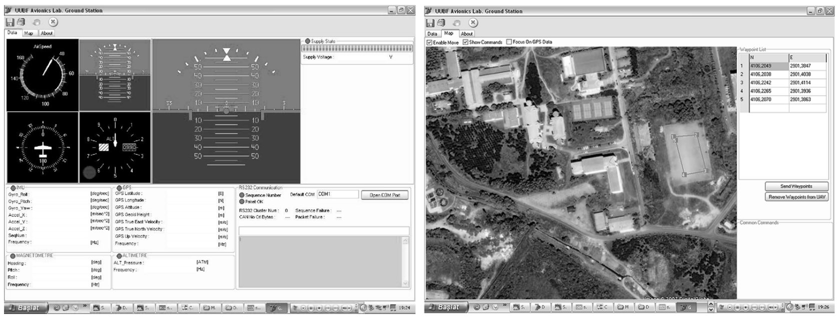


Fig. 22. Ground Station GUI

tests of the real-time control algorithms on actual hardware not only outside, but also in the laboratory. For complex and risky tests involving agile maneuvers in city-like dense environments, mission simulator is utilized for software in-the-loop testing of path planning and control algorithms. In the next subsection, we review two probabilistic flight trajectory planning and controls algorithms designed as a part of the mission simulator development.

4.4 Software-in-the-Loop Testing: Flight Trajectory Planning and Controls in Complex Environments

Motion planning and trajectory tracking control system design for autonomous execution maneuvers in complex environments in search of tasks/targets is a low level enabling technology for UAV operations driven by performance goals. Towards this goal, we have developed a probabilistic trajectory planner that uses distinct flight modes from which almost any aggressive maneuver (or a combination of) can be created. This allows significant decreases in control input space and thus search dimensions, resulting in a natural way to design controllers and implement trajectory planning using the closed-form flight modes.

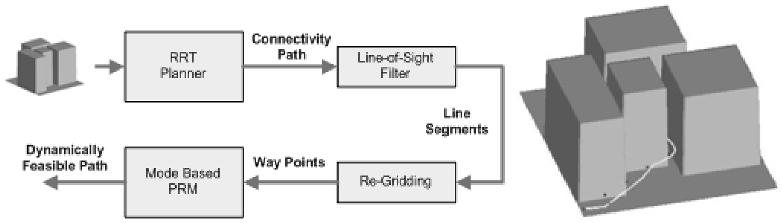


Fig. 23. Mode-Based trajectory generation strategy and complete solution of and dynamically feasible path planner

For fixed-wing UAVs a three-step probabilistic trajectory planner, as shown in Figure 23, is utilized to solve the motion planning problem. In the first step, the algorithm rapidly explores the environment through a randomized reachability tree (RRT) search using approximate line segment models. In order to decrease the computational time, we disregard the vehicle's dynamic constraints and use RRT for searching only the configuration space of vehicle. The resulting connecting path is converted into flight way points through a line-of-sight segmentation. These way points are refined with a single-query Probabilistic Road Map (PRM) planner that creates dynamically feasible flight paths by applying the feasibility constraints of the aircraft model. These feasibility constraints are characterized by a hybrid representation of aircraft dynamics. The general motion of the aircraft has been divided into a discrete set of maneuver modes which has continuous flight dynamics and with each mode there is associated feasibility constraints such as flight envelope and actuator saturation limitations. The problematic issue of narrow passages is addressed through non-uniform distributed

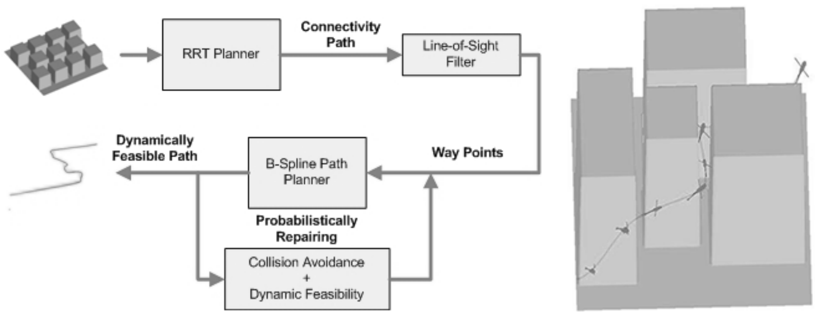

RRT Planner
Connectivity Path
Line-of-Sight Filter
Way Points
B-Spline Path Planner
Dynamically Feasible Path
Probabilistically Repairing
Collision Avoidance + Dynamic Feasibility

Fig. 24. Probabilistic B-Spline trajectory generation strategy and example solution path result of an unmanned helicopter

capture regions, which prefer state solutions that align the vehicle to enter the milestone region in line with the next milestone to come. Detailed explanations of the related algorithms for this implementation can be found in [12].

For helicopter like Vertical Take-Off and Landing (VTOL) systems, a variant two step planner as illustrated in Figure 24 is used for solving the motion planning problem. In the first step, the planner explores the environment using RRT algorithm. The resulting connecting path is converted into flight way points through a line-of-sight segmentation to filter long detours. Remaining points, named as way points, generally appear in entering and exiting regions of the narrow passages that are formed between the obstacles. We segment the hard path planning problem to a series of small path generating problems on these locations by adding way-points. In the second step, each consecutive way point is connected with third-order B-spline curves that represent constant inertial acceleration paths and these curves are repaired probabilistically until a dynamically feasible path is obtained. Since B-Spline curves have local support property, these repairing processes can be made on local interest of path. Detailed demonstrations of this algorithm can be found in [11].

With these two distinct flight planning and controls approaches, we can test the coordination algorithms while ensuring that the target/task assignments in complex and dense environments can be achieved within the existing flight envelopes of the vehicles.

5 Conclusions

In this work, an experimental mission simulator has been developed for joint simulation of manned and unmanned vehicle fleets. This platform serves not only for testing of distributed command and control algorithms and interfaces, but also provides real device hardware-in-the-loop test and real flight test vehicle integration capabilities within simulated scenarios. As such, it provides a realistic test platform to demonstrate and validate research on C2 algorithms and enabling technologies for joint manned-unmanned operations. Current research on this platform is focused on modifying hardware and algorithms with STANAG compliant devices and message formats, adding speech command capability for human operator and pilots, and providing augmented reality concept to the SVS screen by integration of real time streaming video from real unmanned vehicles flying in simulated scenarios.

Acknowledgements

The authors would like to thank ASELSAN, Nazim Kemal Ure, Emre Koyuncu, Sertac Karaman, Ahmet Cetinkaya, Mirac Aksugur, Ismail Bayezit, Serdar Ates, Taner Mutlu, and Fatih Erdem Gunduz for their valuable contributions to this work.

References

1. Behringer, R., Tam, C., McGee, J., Sundareswaran, V., Vassiliou, M.(eds.): System for synthetic vision and augmented reality in future flight decks. In: The International Society for Optical Engineering. Proceedings of SPIE, vol. 4023 (2000)
2. Christiansen, R.S.: Design of an Autopilot for Small Unmanned Aerial Vehicles. Msc. thesis, Brigham Young University (2004)
3. Cummings, M.L., Guerlain, S.: An interactive decision support tool for real-time in-flight replanning of autonomous vehicles. In: AIAA 3rd Unmanned Unlimited Technical Conference, Workshop and Exhibit, pp. 1–8 (2004)
4. Hollan, J., Hutchins, E., Kirsh, D.: Distributed cognition: Toward a new foundation for human-computer interaction research. ACM Transactions on Computer-Human Interaction 7, 174–196 (2000)
5. Horne, W.F., Tucker, R.R., Ekiert, S., Hannan, C.H., Radke, J., Zak, J.A.: Synthetic vision system technology demonstration methodology and results to date. Technical Report 1-19, SAE Technical Paper Series (1992)
6. How, J., Kuwata, Y., King, E.: Flight demonstrations of cooperative control for UAV teams. In: 3rd Conference Unmanned Unlimited Technical Conference, Chicago, Illinios (September 2004)
7. Inalhan, G., Stipanovic, D.M., Tomlin, C.: Decentralized optimization, with application to multiple aircraft coordination. In: The Proceedings of the 41st IEEE Conference on Decision and Control, Las Vegas (2002)
8. Jones, E.D., Roberts, R.S., Hsia, T.C.S.: STOMP: a software architecture for the design and simulation of UAV-based sensor networks. In: Proceedings of IEEE International Conference on Robotics and Automation, ICRA 2003, vol. 3, pp. 3321–3326 (2003)
9. Karaman, S., Aksugur, M., Baltaci, T., Bronz, M., Kurtulus, C., Inalhan, G., Altug, E., Guvenc, L.: Aricopter: Aerobotic platform for advances in flight, vision controls and distributed autonomy. In: IEEE Intelligent Vehicles Symposium, Istanbul, Turkey (June 2007)
10. Karaman, S., Inalhan, G.: Large-scale task/target assignment for UAV fleets using a distributed branch and price optimization scheme. In: Int. Federation of Automatic Control World Congress (IFAC WC 2008), Seoul, South Korea (June 2008)

11. Koyuncu, E., Inalhan, G.: A probabilistic b-spline motion planning algorithm for unmanned helicopters flying in dense 3d environments. In: International Conference Intelligent Robots and Systems, Nice (September 2008)
12. Koyuncu, E., Ure, N.K., Inalhan, G.: A probabilistic algorithm for mode based motion planning of agile air vehicles in complex environments. In: Int. Federation of Automatic Control World Congress (IFAC WC 2008), Seoul, South Korea (June 2008)
13. Makarenko, A.A.: A decentralized Architecture for Active Sensor Networks. PhD thesis, The University of Sydney (November 2004)
14. Mutlu, T., Comak, S., Bayezit, I., Inalhan, G., Guvenc, L.: Development of a cross-compatible micro-avionics system for aerorobotics. In: IEEE Intelligent Vehicles Symposium, Istanbul, Turkey (June 2007)
15. Nguyen, T.L., Ogburn, M.E., Gilbert, W.P., Kibler, K.S., Brown, P.W., Deal, P.L.: Simulator study of stall/post-stall characteristics of a fighter airplane with relaxed longitudinal static stability. Technical paper, NASA (December 1979)
16. GPL OpenSource. The flightgear flight simulator (1996)
17. GPL OpenSource. Boost c++ libraries (2001)
18. Ure, N.K., Inalhan, G.: Mode based hybrid controller design for agile maneuvering F-16 aircraft. Journal of Process Control (manuscript, 2008)
19. Valenti, M., Schouwenaarsy, T., Kuwataz, Y., Feronx, E., How, J.: Implementation of a manned vehicle - UAV mission system. In: AIAA Guidance, Navigation, and Control Conference and Exhibit (2004)

Author Index