COLLABORATIVE TASKING OF TIGHTLY CONSTRAINED MULTI-ROBOT MISSIONS*

Douglas C. MacKenzie
Mobile Intelligence Corporation
Livonia, Michigan, 48150–1646
doug@mobile–intelligence.com

Abstract Collaborative Tasking provides the ability to automatically task and re-task a group of heterogeneous unmanned vehicles, producing a stable, robust, tightly integrated unit while reducing manpower requirements. A variant of a market economy-based approach is used, where each vehicle computes constraint-based estimates of its cost to perform a particular task. These cost projections are sent back to the entity offering the task, which uses a cost minimization algorithm to select the appropriate vehicle(s) and constraint values in a unified manner. Each vehicle's computation of its cost to perform a task takes into account remaining consumables, required effort, its other pending tasks, and user specified preferences. Costs are reported as functions of constraints, such as location and time, to provide a unified approach to assigning time-sensitive, tightly coordinated tasks.

Keywords: multi-robot, collaborative tasking, coupled tasks

1. Introduction

A great deal of effort is required of the human commander to individually task each vehicle (and re-task when things change). Manned military units are tasked at the unit level (*e.g.*, Company Alpha bomb the bridge, Platoon Bravo scout the ridge) while unmanned units are currently tasked at the vehicle level (*e.g.,* UAV12 bomb the bridge, UGV42 scout the ridge). Our goal is to provide teams of unmanned vehicles with the same capability as manned units to receive tasks at the unit level. Automatic team-level tasking and re-

*In *Multi-Robot Systems: From Swarms to Intelligent Automata (Proceedings Second International Workshop on Multi-Robot Systems)*, Washington D.C., A.C. Schultz et al. Editors, Kluwer Academic Publishers. vol. 2, (2003): 39-50.

tasking will be used to produce a stable, robust, tightly integrated unit, thereby reducing manpower requirements.

This goal is complicated by the adversarial military environment. In this domain we assume there will be lossy communications, unexpected loss and addition of vehicles, tasks of widely varying priority, as well as retraction and modification of tasks currently underway. Multiple users must be able to quickly inject tasks at any time while minimizing their radio emissions and required attention. One example is a collection of loitering munitions which attack targets designated by soldiers on the ground that are replenished periodically with additional vehicles.

To solve these problems we look to a variant of market economy-based approaches which have proven useful in multi-robot tasks. To better fit this domain, we have developed a variant of the standard Contract Net Protocol (Smith, 1980) and its extensions (Sandholm and Lesser, 1995) that we call the Collaborative Tasking Protocol (CTP) targeted for benign agents and adversarial environments. Rather than agents striving to maximize their monetary gain as in CNP, our agents strive to complete tasks while minimizing the total cost to the robot team.

When a task is injected, each vehicle estimates its cost to perform the task taking into account remaining consumables, required effort, its other pending tasks, and user specified preferences. Costs are reported as multi-dimensional functions of constraints, such as location and time, to provide a unified approach to assigning time-sensitive, tightly coordinated tasks. These cost projections are evaluated using a cost minimization algorithm to select the appropriate vehicle(s) and constraint values in a unified manner.

Several others have also used negotiation strategies to coordinate multi-robot tasks. Stentz(Dias and Stentz, 2002; Zlot et al., 2002) has pioneered using the market-based approach to tasking robots, with a focus on optimizing exploration coverage and similar independent tasks. Coordinated mapping demonstrated by (Thayer et al., 2000) uses a CNP variant to provide unit-level tasking of a robot mapping team. The most closely related work to ours is found in (Gerkey and Matarić, 2002). Their MORDOCH system provides a unit-level tasking capability similar to our system. Their box pushing and long-term autonomy experiments demonstrate the utility of this approach for providing robustness when failure-prone vehicles must accomplish critical tasks. An important difference is our use of multi-dimensional cost functions to model the impact of task constraints. We also distribute their contract renewal process among the robots using a peer process so the user is not required to maintain connectivity with the robot team after injecting the task.

A Collaborative Tasking Module that implements the Collaborative Tasking Protocol has been developed to allow experimental evaluation. Preliminary analysis of the protocol and simulation results from the implementation

are presented. A simulated three-phased bombing mission (lead vehicle verifies target, second vehicle bombs target, third vehicle assesses bomb damage) is demonstrated within MIC's Societal Agent Robot Architecture (SARA) (MacKenzie et al., 1997). These results highlight the utility of this approach in realistic settings.

2. The Collaborative Tasking Protocol

The Collaborative Tasking Protocol (CTP) defines the message syntax and semantics required of agents participating in the team. Figure 1a shows the State Transition Diagram for the actions required of a user injecting a task: a transmission of the task followed by acknowledgment of a claim from one of the vehicles (Figure 1b) which becomes the lead for that task.

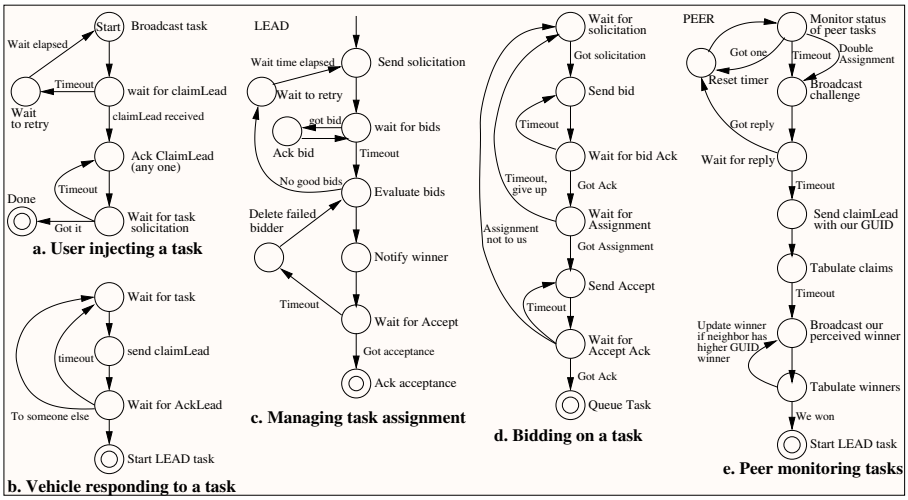


Figure 1. State transition diagrams in the Collaborative Tasking Protocol

The lead vehicle serves as an advocate for the user, offering the task to the vehicles and assigning the winner. It starts the bidding process, as shown in Figure 1c, by broadcasting a solicitation. All interested vehicles respond with bids, as shown in Figure 1d, and monitor whether they won. The vehicle able to carry out the task with the least cost is assigned as the winner by the task lead and queues the task for execution. All other vehicles start the peer monitoring process shown in Figure 1e to periodically verify that the assigned vehicle is still planning to carry out the task.

Task monitoring is accomplished by periodic broadcasts of task status reports by the winning vehicle. These reports note its successful or estimated time of completion. If these reports are missed for a period of time, the peer vehicles broadcast a challenge. If the assigned vehicle does not respond to the

challenge, the challenging vehicle will try to claim the lead for that task by broadcasting a claimLead message. If successful, it then re-offers the task for execution and the process of assigning a vehicle to carry out the task begins anew.

2.1 Protocol Analysis

A new protocol demands a discussion of its strengths and weaknesses. In this section we explore three facets of collaborative tasking: the ability to complete tasks, prevent duplicated effort, and evaluate the communications load.

Claim 1: *Task Completion*
If there is at least one operational vehicle v_0 able to exchange messages with the user at time t_0, and a chain of intermediary vehicles $v_1 \dots v_n$ able to exchange messages ($v_i \Leftrightarrow v_{i+1}$ where $A \Leftrightarrow B$ means A can communicate with B) and the termination of vehicle i, $T(v_i)$ occurs at least t_{sync} after the start of vehicle v_{i+1}, $S(v_i)$ to allow vehicle v_{i+1} to have acquired the list of pending tasks. Then, if vehicle v_n is operational and able to execute task x at time t_1, task x will be carried out.

Proof: For task x to "fall on the floor" there must be some period of time $t_j | t_0 \leq t_j \leq t_1$ where no vehicle $v_i | v_0 \leq v_i \leq v_n$ is aware of task x. There are two cases to consider.

The first is when v_{i-1} has been assigned task x. The protocol requires that a vehicle v_{i-1} which is assigned task x broadcast a status report every time t_{status}, and that the status report contains sufficient detail to allow a new team member to instantiate a peer monitoring process. Therefore, if $t_{sync} \geq t_{status}$, v_{i-1} will broadcast at least one status report related to task x and v_i will become aware of task x.

The second case is when v_{i-1} is a peer for task x and no operational vehicle is assigned task x. The protocol requires that a peer monitor for status reports and broadcast a challenge if time $t_{peer} | t_{peer} \gg t_{status}$ passes without a status report. The challenge contains sufficient detail to allow a new team member to instantiate a peer monitoring process. Therefore, if $t_{sync} \geq t_{peer}$, v_{i-1} will broadcast at least one challenge related to task x and v_i will become aware of task x.

A complicating factor is periodic communications failure. To minimize its impact we posit that the vehicles' communications cognizantly fail, to allow queuing and retransmission of the latest status report when the link is again available. The transmission time of status and challenge messages should also be randomized to preclude synchronization with periodic outages. Of course, the primary reaction to poor communications must be to expand the timeout periods.

Claim 2: *Task Duplication*
If a vehicle v is assigned task x, and v is able to get a status message to a vehicle v_i once each time t_{peer}, then only v will execute task x.

Proof: There is only one case where two vehicles, v_i and v_j, could execute task x. There must be some point in time t_p after v_i was assigned task x and before it executed it at time t_x when it became unable to communicate with v_j. With v_i unaware, task x was challenged and reassigned to v_j. Further, this partitioning of the communications network must have persisted until one of the vehicles executed the task. In this case there is no way to know that the vehicle executed the task, unless it can be discerned from the environment. To recover from temporary partitioning, if a vehicle receives status reports from two vehicles for the same task it will issue a challenge and the task will be rebid to a single vehicle.

In general, t_{peer} will vary based on the type of task and assigned vehicle. Some tasks and/or vehicles require a significant period of stealth leading up to execution, which would indicate a rather large value for t_{peer} for these tasks.

Claim 3: *Communication Load*
The communications load for this protocol during normal execution is $L = T(2V + 3) + (t_{dur} * T)/t_{status} + T$, where T is the number of tasks, V is the number of vehicles, t_{dur} is the average time a task is pending, and t_{status} is the time between status reports.

Proof: The load from bidding a new task is $L_{bidding} = 2V + 3$, since each vehicle V replies to the solicitation and is sent a bid acknowledgment, and one vehicle is sent and acknowledges the assignment. The load from monitoring a pending task is $L_{pending} = 1/t_{status}$, since a status report is broadcast during each time period t_{status}. The load from executing a task is $L_{execute} = 1$, since a completion report is broadcast just before execution.

3. The Collaborative Tasking Module

The Collaborative Tasking Protocol (CTP) has been implemented in the Collaborative Tasking Module within our multi-agent Societal Agent Robot Architecture (SARA) to support experimental evaluation. A block diagram of our Collaborative Tasking Module (CTM) is shown in Figure 2. The Comms Manager implements the communications protocol and interacts with other instances of the CTM executing on other vehicles and operator consoles. It manages the pool of Active Solicitations, posting bids and receiving assignments. The CTM Controller manages the process of generating and responding to solicitations and sending assigned tasks to the Execution Subsystem. The Cost Estimator generates cost functions based on the operational state of the vehicle and the capabilities of the Execution Subsystem. The Task Assigner computes

a minimum cost assignment of tasks and constraints based on the bids. Each vehicle and operator console has an identical instance of the CTM.

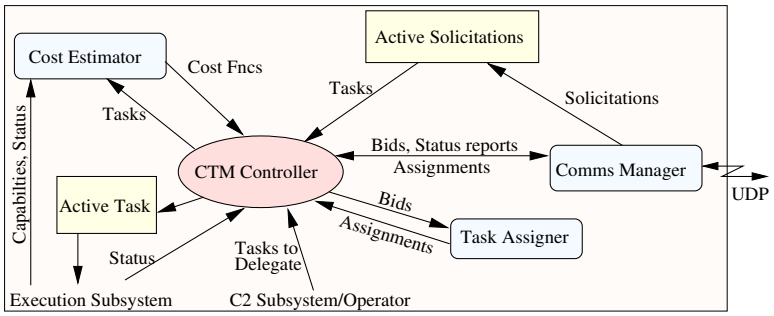


Figure 2. Block diagram of the Collaborative Tasking Module

The cost functions are multi-dimensional functions of the constraint values, where the value of the function at a particular point represents the cost to perform the task given those values of the constraints. For example, consider a cost function where the constraints are time and approach heading. These two constraints result in a 3-dimensional cost surface, where given any $\langle time, heading \rangle$ values we can extract the cost for a vehicle to approach on the specified *heading*, arriving at the target at the specified *time*.

To control communications bandwidth in our current implementation, the cost function is sampled and only the samples are transmitted. The approach heading is sampled at 45 degree increments and the performance time is sampled at 5 second increments, from 0 to 100 seconds. This allows transmitting the cost functions in only 21 (time samples) x 8 (heading samples) x 2 (bytes per sample) = 336 bytes, which fits in a broadcast UDP communications packet. Future work will improve the sampling algorithm to consider the derivative of the cost function and focus the samples in areas where the slope is changing rapidly. This will maximize information content of the cost function reconstructed from the samples.

Assignment of tasks is performed using a cost minimization algorithm. Our current implementation only examines constraint values at the sampled points of the cost functions. This precludes generating optimal solutions, but gets very close for the smooth functions we expect. Our prototype algorithm uses a brute-force approach, where all consistent pairings at these constraint sample points are evaluated and the lowest cost assignment of tasks and constraint values is selected. Future work will explore more scalable evaluation strategies, including cost trees, to improve scalability of the Task Assignment algorithm. However, the current implementation is suitable for the size of teams ($\sim$ 10 vehicles) we have targeted initially.

Task assignments could also be performed before the mission using the same protocol. This would allow, for example, deploying a squadron of UCAVs with pre-planned ingress corridors, loiter areas, and initial targets that could be executed in radio silence. When radio silence is broken, new tasks or constraints could be added to these pre-planned missions using the same protocols.

4. Example Three-Phased Bombing Mission

Consider a situation where a forward observer has reported the coordinates of an enemy tank and requests an air strike. To begin, the operator adds the new target to the workspace by clicking on its location on a map display. The operator then attaches a coordinated three-phased air strike task to the target. This task requires three tightly coordinated vehicles: the first verifies a valid target, the second bombs the target, and the third performs bomb damage assessment. The vehicles must arrive at the target only a few seconds after each other to prevent enemy reaction. This is a challenging task, since it requires accurate timing of each vehicle's arrival, as well as close coordination of their ingress and egress flight corridors to prevent collisions. Figure 7a on Page 10 shows the operator console with all symbology and the map overlay turned off to reduce visual clutter. There are three vehicles shown, *doug2* is in the lower left, *doug3* is in the upper left, and *doug4* is on the right. The bombing target has been added near the middle of the display. Once the operator has attached the task to the target the CTM begins negotiating its disposition.

First, the Collaborative Tasking Module resident on the taskLead decomposes the high-level bombing task into three executable tasks and broadcasts these tasks to the vehicles. Figure 3 shows the generated solicitation. Task0 requests bids on verifying the target located at coordinates $< 31.8, 27.9 >$, and notes that the cost functions should specify constraints on performance time and travel corridors (the "tt" flags). Task1 similarly requests bids on bombing the target and Task2 on bomb damage assessment (BDA).

task[0]	=	VerifyTarget tt 0 0 1 L $< 31.8, 27.9, 0 >$
task[1]	=	Bomb tt 0 0 1 L $< 31.8, 27.9, 0 >$
task[2]	=	BDA tt 0 0 1 L $< 31.8, 27.9, 0 >$

Figure 3. The solicitation sent to the vehicles for the three tasks

Each vehicle receiving the solicitation computes its cost function to execute each task it is capable of performing using the constraints. Each vehicle then sends its cost functions back to the taskLead. Figure 4 shows a plot of the cost function returned by *doug2* for the VerifyTarget task. The radial distance from the central tower is the performance time, the rotational angle around the center is the approach heading, and the height is the cost to perform the task. There is a minimum time required to get to the target that varies based on the approach heading. The top of the central tower shows <time,heading> values that are

not feasible, because *doug2* can't get there quickly enough. Minimum costs occur along paths that reach the target just in time. If the performance time is delayed beyond this minimum, the vehicle wastes time waiting to complete the task, so costs rise as you move farther from the center.

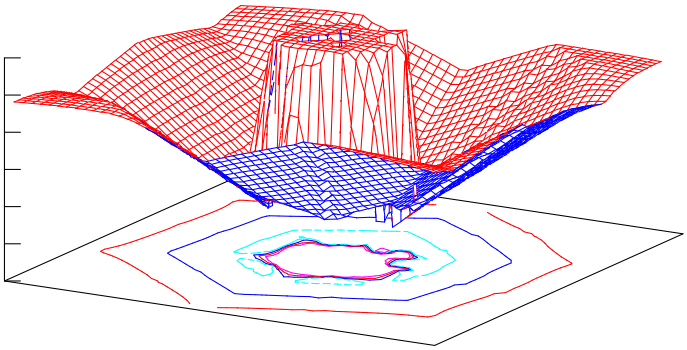


Figure 4. Graph of UCAV *doug2*'s VerifyTarget cost function (polar coordinates). Height is the cost, performance time is the radial distance from the center, angle around the center axis (the central tower) is the approach heading, and the contour lines show changes in cost of 50 units.

Figure 5 shows the graph of each vehicle's minimum cost versus its approach heading to the target. These are "aerial views" of Figure 4, showing only the minimum cost ring around the central tower. Comparing Figure 5 with Figure 7a you can see that UCAV *doug2* (in the lower left) can in fact perform with a lower cost if it approaches the target at a heading of 45 degrees (moving towards the upper right of the screen). Other headings would require *doug2* to partially circle the target before starting its approach.

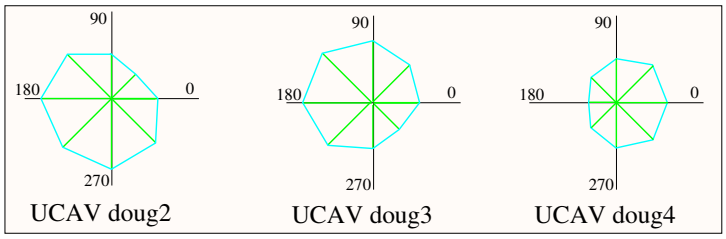


Figure 5. Slices of the generated cost functions along a particular heading

Figure 6 shows how a slice of the cost functions at a particular approach heading varies with time. For example, the left curve for UCAV *doug2* shows its costs as a function of time when approaching the target with its preferred heading of 45 degrees. Notice that it takes at least 20 seconds to reach the target; however, delays beyond the minimum time to reach the target increase the costs. Specifically, the cost for UCAV *doug2* is "40" with heading 45 degrees and performance time t=20. Its cost increases to "50" if the performance time is delayed until time t=25.

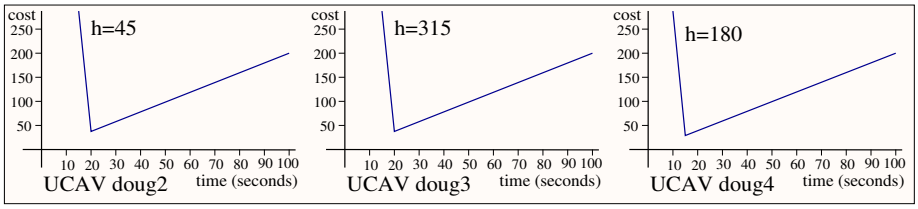


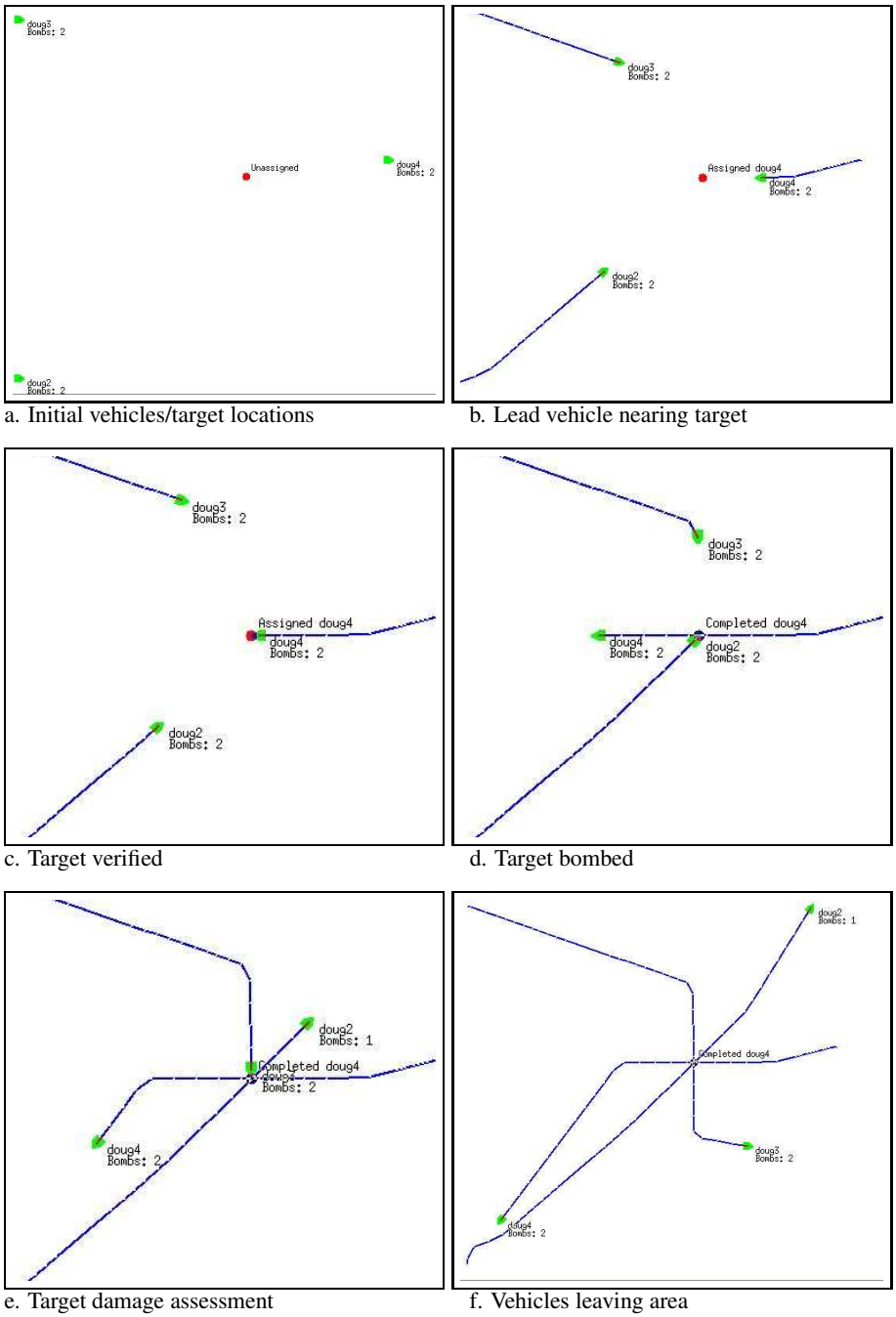
Figure 6. Slices of the generated cost functions along the minimum cost curve

Once the lead CTM has gathered the bids (cost functions), a minimization algorithm is used to select the best vehicle for each task, with mutually consistent performance times, and de-conflicted approach headings. In this example, all vehicles have similar capabilities. The minimum cost solution selected by the Collaborative Tasking Module (CTM) is for UCAV *doug4* to perform the VerifyTarget task at time t=15 seconds. It will approach with a heading of 180 degrees (moving left on the screen). The bombing task will be performed by *doug2*, approaching with a heading of 45 degrees and crossing the target at time t=20. The BDA task is assigned to *doug3*, with a heading of 270 (down the screen) and a performance time of t=25.

Once computed, the assignments are sent to the vehicles and execution of the task begins. Figure 7a shows the vehicles in their initial positions, before they receive their assignments. Figure 7b shows the mission starting to execute in the SARA simulator, with the lead vehicle, *doug4*, approaching the coordinates to verify there is a valid target at that location. Figure 7c shows that *doug4* has followed its assigned flight corridor and completed its VerifyTarget task; in this run crossing the target 1 second early. The behavior controller is time-aware and is managing both the speed and routes of the vehicles to achieve the specified time over target. (The timing system used in the behavior controller only has a one second resolution.) In Figure 7d, UCAV *doug2* has bombed the target, also 1 second early. Notice that the lead vehicle continues to fly its egress along the assigned corridor. In Figure 7e the final vehicle, *doug3*, has crossed the target to perform bomb damage assessment; in this case it was 2 seconds early. Notice that *doug4* exited its flight corridor to move on to another task. Figure 7f concludes this demonstration with all three vehicles completing their assigned tasks, leaving the controlled area, and moving on to new tasks.

5. The Societal Agent Robot Architecture

We are developing the collaborative tasking module within our Societal Agent Robot Architecture (SARA). Mobile Intelligence has recently licensed the Georgia Tech *MissionLab* system(MissionLab, 2002) to commercialize it. We are currently working to combine *MissionLab*'s extensive behavior library



a. Initial vehicles/target locations

b. Lead vehicle nearing target

c. Target verified

d. Target bombed

e. Target damage assessment

f. Vehicles leaving area

Figure 7. Example three-phased bombing mission

(Arkin, 1998) and proven user interfaces(MacKenzie and Arkin, 1998) with the robustness and scalability of Mobile Intelligence's Swarm run-time system. The resulting SARA toolset will provide a robust, deployment-grade robotics software package with state of the art capabilities and competence.

6. Future Work

Our current focus is to expand the Collaborative Tasking Module to more generically support constraint-based negotiations. We also intend to expand the user's capability to inspect and modify cost functions based on personal preferences. The most difficult technical challenge is to create cost functions that are suitably generic to work for many vehicles and tasks, while ensuring robustness and stability.

An implication of this work is that an abstract tasking language can be formulated for use by the CTM. Our current tasking language is static and doesn't easily support new kinds of tasks. We are currently exploring DAML and XML in our search for a suitable language for communicating tasking requests and constraints.

7. Summary

This effort's primary technical innovation is the use of extended dimensional cost functions to provide a unified representation of the impact of constraints on tightly coupled tasks. This allows a single process to select optimal assignments of both tasks to vehicles and values to constraints (*e.g.*, assigning tasks along with performance times and flight corridors).

The Collaborative Tasking Protocol with its focus on benign agents and adversarial environments was presented and discussed. Our implementation of CTP in the Collaborative Tasking Module is used to demonstrate a simulated three-phased bombing mission (lead vehicle verifies target, second vehicle bombs target, third vehicle assesses bomb damage). Real-world tasks such as these demonstrate the power of collaborative tasking and provide a focus for future research.

Acknowledgments

The author wishes to thank Karen Brehob MacKenzie for her help and patience with this project.

References

Arkin, R. (1998). *Behavior-Based Robotics*. MIT Press, Cambridge, Massachusetts.

Dias, M. B. and Stentz, A. T. (2002). Opportunistic optimization for market-based multirobot control. In *Proc. Intelligent Robotics and Systems (IROS 2002)*.

Gerkey, B. P. and Matarić, M. J. (2002). Sold!: Auction methods for multi-robot coordination. *IEEE Trans. on Robotics and Automation, Special Issue on Multi-robot Systems*, To Appear.

MacKenzie, D. and Arkin, R. (1998). Evaluating the usability of robot programming toolsets. *The Intl. Journal of Robotics Research*, 14(4):381–401.

MacKenzie, D., Arkin, R., and Cameron, J. (1997). Multiagent mission specification and execution. *Autonomous Robots*, 4(1):29–52. Also appears in *Robot Colonies*, ed R. Arkin and G. Bekey, Kluwer Academic Publishers, 1997.

MissionLab (2002). *Mobile Robot Laboratory*. College of Computing, Georgia Institute of Technology, Available via *http://www.cc.gatech.edu/ai/robot-lab/research/MissionLab*, Version 5.0 edition.

Sandholm, T. and Lesser, V. (1995). Issues in automated negotiation and electronic commerce: Extending the contract net protocol. In *First International Conference on Multiagent Systems (ICMAS-95)*, pages 694–701, San Fransisco. Also in *Readings in Agents*, Huhns, M.N. and Singh, M.P. Editors, Morgan Kaufmann Publishers, 1997.

Smith, R. G. (1980). The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12):1104–1113. Also in *Readings in Distributed Artificial Intelligence*, Bond, A.H. and Gasser, L. Editors, Morgan Kaufmann Publishers, 1988.

Thayer, S. et al. (2000). Distributed robotic mapping of extreme environments. In *Proc. SPIE Conf. on Mobile Robots XV*, volume 4195.

Zlot, R., Stentz, A., and Dias, M. (2002). Multi-robot exploration controlled by a market economy. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, pages 3016–3023, Washington D.C.