MARKET-BASED CO-EVOLUTION PLANNING FOR MULTIPLE AUTONOMOUS VEHICLES

Anawat Pongpunwattana*, Rolf Rysdyk†, Juris Vagners‡,
University of Washington, Seattle, WA 98195

David Rathbun§
The Insitu Group, Bingen, WA 98605

ABSTRACT

In a highly dynamic environment, an adaptive real-time mission planner is essential for controlling a team of autonomous vehicles to execute a set of assigned tasks. The optimal plan computed prior to the start of the operation might be no longer optimal when the vehicles execute the plan. This paper proposes a framework and algorithms for solving real-time task and path planning problems by combining Evolutionary Computation (EC) based techniques with a Market-based planning architecture. The planning system takes advantage of the flexibility of EC-based techniques and the distributed structure of Market-based algorithms. This property allows the vehicles to evolve their task plans and routes in response to the changing environment in real time.

1. INTRODUCTION

Recently many researchers have successfully applied Evolutionary Computation (EC) based techniques to many problems that include task and path planning. EC-based techniques have been used for both single-vehicle planning [1–3] and multi-vehicle planning problems .[1,4,5]

There are many advantages in using EC-based techniques for task and path planning problems. First, they are open architecture techniques; it is easy to solve optimization problems with linear or nonlinear performance functions and constraints. Second, EC-based planning approaches are on-line adaptation techniques; they can be formulated to run continuously as the vehicles execute their plans, and handle changes in the operating environment and vehicle capabilities. Third, EC-based techniques respond to the changing environment quickly because they do not have to recompute the entire plan; the current plan is adapted in response to the changes.

Although previous research has shown good results using EC-based techniques in task and path planning for multiple autonomous vehicles,[4,5] most of these approaches are based on a centralized implementation, i.e. all of the planning computation takes place in one location. The computed paths for the team are subsequently transmitted to the team members. One primary advantage of this approach is that it can produce globally optimal solutions. Nonetheless, the centralized planning approach has some disadvantages. First, it has difficulties in finding optimal plans for large scale complex problem. Second, the centralized planning approach assumes that all local information can be transmitted to a central location to update the plans corresponding to changes in the environment and vehicles capabilities. Lastly, the vehicles respond slowly to changes in the environment if there is any delay in communication. Although some of centralized approaches are implemented using parallel algorithms running on multiple processors, they assume perfect communication and require a scheme for synchronizing the processes. Therefore, the parallel algorithms do not have the robustness of distributed algorithms.

Using distributed-architecture planning can improve the drawbacks of the centralized-architecture EC-based techniques mentioned above. A promising approach is integration of EC-based techniques with a Market-based planning architecture. One of the original formulations of a Market-based approach is Smith's Contract Net Protocol (CNP).[6] Several researchers have extended the concept of the CNP algorithm[7,8] for clustering tasks and task decomposition. Market-based algorithms for task planning have been proposed by Wellman and Wurman[9] and Dias and Stentz.[10]

There are several advantages of the Market-based approach. One of its greatest advantages is the ability to deal successfully with an uncertain environment. Because of its distributed architecture, the real-time planner of each vehicle can quickly access information about the operating field and its states and capabilities detected by the on-board sensors. Therefore, the planner responds more quickly and efficiently to changes than with a cen-

*Research Assistant, Department of Mechanical Engineering, email: anawatp@u.washington.edu
†Assistant Professor, Department of Aeronautics and Astronautics, email: rysdyk@aa.washington.edu
‡Professor, Department of Aeronautics and Astronautics, email: vagners@aa.washington.edu
§Senior Scientist, email: rathbun@veribox.net

tralized planning approach. Another advantage is the scalability of the planning system since the computation is distributed among the vehicles. The system can handle large scale complex problems. The Market-based approach is also robust to single points of failure in the system. Its planning structure does not require a designated vehicle in the team to plan paths for all other vehicles. Any tasks that are assigned to disabled members are reallocated to other vehicles through a bidding process, and the entire mission can still be accomplished.

This paper proposes a framework and algorithms for solving real-time task and path planning problems for a team of autonomous vehicles by combining EC-based techniques with a Market-based planning architecture. An overview of the problem formulation is shown in section 2. Section 3 describes the architecture of the planning system. In section 4, we explain details of the evolution-based planning algorithms. Section 5 reports the simulation results of the planning algorithms. Section 6 provides concluding remarks.

2. PROBLEM FORMULATION

The main objective of the research presented here is to develop algorithms and techniques to dynamically allocate tasks and to find a set of paths for a team of heterogeneous autonomous vehicles. The generated set of paths will support the desired system states corresponding to the assigned mission objectives. It is also required that the path and tasks assigned to each vehicle are within its capabilities. In addition, the planning algorithm must provide a set of paths that minimize the probability that vehicles are destroyed due to collision with obstacles or damage from threats in the field of operation.

There are two main concepts presented in this paper. One is *real-time planning* and the other is *integrated task and path planning*. In real-time planning, the trade-off between time-available-to-plan and adaptability becomes important. One would want the non-adapting sections of the generated paths to be short in a highly dynamic environment, however that shortens the time-available-to-plan. The shorter time-available-to-plan requires faster planning algorithms. Hence, real-time planning algorithms must be fast enough to compute new plans within a specified time limit. Figure 1 illustrates this concept. In the figure, we define a *spawn point* located on the trajectory at time $t_{s_p} > t_k$. There are also other spawn points along the trajectory at time $t_{s_{p+1}}, t_{s_{p+2}}, \ldots, t_N$ where t_N is the time at the end of the planning horizon. The time difference between two adjacent spawn points ($\Delta T_s = (t_{s_{p+1}} - t_{s_p})$) specifies the maximum time-available-to-plan for the planner to update the adapting section. In this manner, the planner can account for new information about the environment that becomes available during execution. The concept of this real-time planning is very similar to that of *Model-based Predictive Control*.[11]

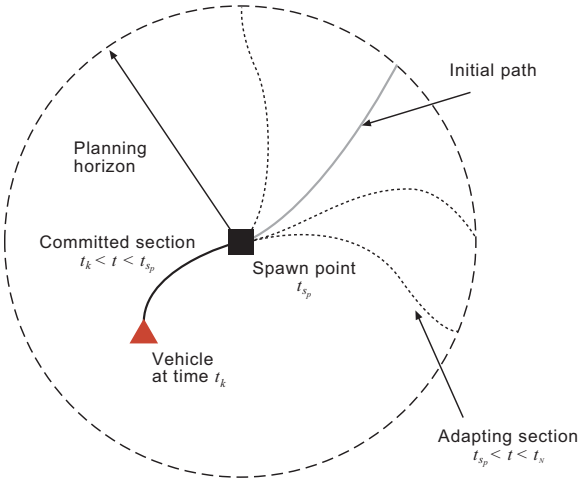

Fig. 1 Real-time planning concept. A vehicle (at time t_k) is moving along a previously computed trajectory, shown as a gray line. The spawn point at time t_{s_p}, shown as a black square, is the closest spawn point to the current position of the vehicle. This spawn point divides the committed section and the adapting section of the planned path. The committed section, which will not be altered, is a portion of the path for $t_k < t < t_{s_p}$. The committed trajectory is then sent to the vehicle's controller for execution. The adapting section of the path is free to be further refined by the planner.

The other main part of our research is solving the integrated task and path planning problem. This problem is composed of two subproblems. The first is task planning, which includes task allocation and scheduling. The second is path planning. These two subproblems are highly coupled. In order to solve for an optimal task plan, the task planner requires the cost of each optimal path to execute a particular set of tasks. Conversely, the path planner needs the allocated set of tasks before it can find an optimal trajectory for each vehicle. The coupling between these two subproblems makes this problem very complex. The block diagram shown in figure 2 illustrates the combined concepts of real-time planning and integrated task and path planning.

The objective of both the task planner and the path planner is to maximize the predicted score obtained in accomplishing each task while minimizing the predicted operation cost given the information about the world at time t_k. The objective function at time t_k can be written as

$$J(k) = \sum_{n=1}^{N-s_p} \sum_{i=1}^{N_T} \hat{S}_i(s_p + n \mid k) - \sum_{m=0}^{M-1} \sum_{v=1}^{N_V} C(P_v(s_{p+m} \mid k)) \quad (1)$$

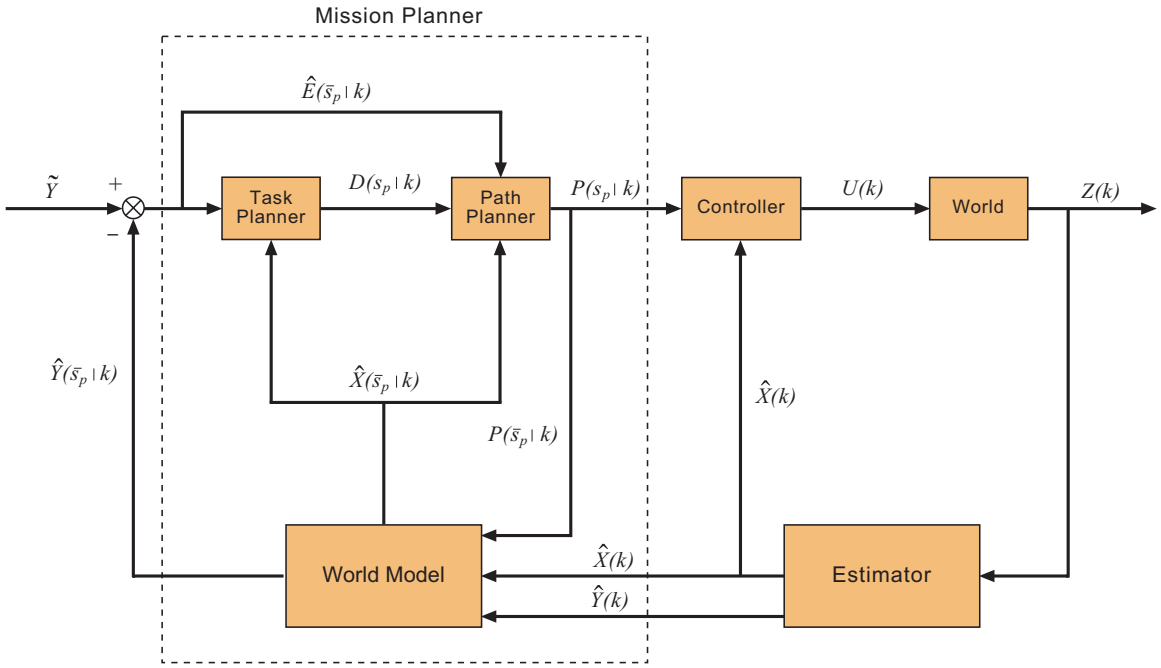

Fig. 2 Block diagram of a real-time mission control system. To simplify our notation, a sampled signal $y(t_k)$ where k can take on any integer value will be simply written as $y(k)$.

The variables represent:

$U(k)$ = controller commands at time t_k
$\hat{X}(k)$ = estimated internal states (e.g. position, velocity) of the world at time t_k
$\hat{Y}(k)$ = estimated states of the tasks indicating the percent of accomplishment of the tasks at time t_k
$\tilde{Y}$ = input commands of the planners
$Z(k)$ = observed information about the world at time t_k
$\hat{X}(\overline{s_p} \mid k)$ = predicted internal states of the world at time $t_{s_p}, t_{s_p+1}, \ldots, t_N$ given information about the world up to time t_k
$\hat{Y}(\overline{s_p} \mid k)$ = predicted states of the tasks at time $t_{s_p}, t_{s_p+1}, \ldots, t_N$ given information about the world up to time t_k
$\hat{E}(\overline{s_p} \mid k)$ = predicted errors in the states of the tasks at time $t_{s_p}, t_{s_p+1}, \ldots, t_N$ given information about the world up to time t_k
$P(s_p \mid k)$ = planned trajectory segments for time $t_{s_p} < t < t_{s_{p+1}}$ given information about the world up to time t_k
$P(\overline{s_p} \mid k)$ = planned trajectory segments for time $t_{s_p} < t < t_N$ given information about the world up to time t_k
$D(s_p \mid k)$ = task-level planning decisions at time t_{s_p} given information about the world up to time t_k.

where $\hat{S}_i(s_p + n \mid k)$ is the estimated score of task i at time t_{s_p+n} given information about the world up to time t_k, $C(P_v(s_{p+m} \mid k))$ is the cost associated with a portion of the planned path P of vehicle v for time $t_{s_{p+m}} < t < t_{s_{p+m+1}}$, N_V is the number of vehicles in the team, N_T is the number of all tasks, N is the index of sampled signals at the planing horizon t_N, and $M = (N - s_p)/(s_{p+1} - s_p)$. The estimated score of task i at time t_q given information about the world at time t_k is given by

$$\hat{S}_i(q \mid k) = -(\hat{E}_i(q \mid k) - \hat{E}_i(q-1 \mid k)) \cdot \alpha_i(q) \quad (2)$$

where $\alpha_i(q)$ is the score weighting factor of task i at time t_q. Given information about the world at time t_k, the segment of planned path P of vehicle v for time $t_{s_{p+m}} < t < t_{s_{p+m+1}}$ is a function (Γ) of the task-level planning decisions D_v at time t_{s_p}, and the predicted errors in the states of the tasks $\hat{E}$ and the predicted states of the world at time $t_{s_p}, t_{s_p+1}, \ldots, t_N$. It can be written as

$$P_v(s_{p+m} \mid k) = \Gamma(D_v(s_p \mid k), \hat{E}(\overline{s_p} \mid k), \hat{X}(\overline{s_p} \mid k)) \; ; \forall m \in \{0, 1, \ldots, M-1\} \quad (3)$$

Given the objective function in equation 1, the path planning problem can be written as

$$\max_{P_v(s_{p+m}|k)} J(k) \quad (4)$$

and the task planning problem is given by

$$\max_{D_v(s_p|k)} J(k) \quad (5)$$

Equations 1, 3, 4, and 5 clearly show the coupling between the task and path planning problems.

The task-level decision variable D is an array of decision matrices $\{D_1, D_2, \ldots, D_{N_v}\}$, where D_v is the

decision matrix for vehicle v with dimension $N_T \times N_T$. The element of D_v at row i column j is denoted by d_{ij}^v. $d_{ij}^v = 1$ if vehicle v plans to execute task i at sequence number j, otherwise $d_{ij}^v = 0$. For example,

$$D_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

means that the task plan for vehicle 1 is to execute task 1 and task 3 in sequence.

The problem considered in this paper is not finding the true optimal paths and task plans with respect to an objective function, but rather finding a near-optimal solution within the planning time available. The benefit of this approach is that the planning system can provide trajectories and task plans to the vehicles at any given time. This property allows on-line and dynamic replanning while the vehicles execute the current plans. If the conditions of the problem remain constant, and a sufficiently long planning time window is available, the solution will approach the optimum.

In this problem, we assume that each vehicle contains a guidance law to guide the vehicle along the path generated by the planner. Full communication connectivity is assumed among all the vehicles and the command and control center of the team. However, those communication channels can be limited.

3. PLANNING ARCHITECTURE

The overall distributed structure of the planning system is illustrated in figure 3. In this system, a market-based scheme is used to optimize the team's objective function. Individual vehicles are autonomous. Under the market protocol, they can choose their own tasks and plan their routes that will in turn benefit the team. Each vehicle generates its own path that allows it to accomplish the tasks in its current plan. Accomplishing tasks will bring reward to the team. However, there is cost associated with executing the task. The function of the task planning is to schedule and allocate tasks to those vehicles that can achieve them optimally.

This planning architecture considers the vehicles as if they are in a market where the trading items are tasks. However, it is not a free market that allows anyone to trade directly to others. In this system, vehicles buy or sell tasks through coordinator agents. The presence of coordinator agents may weaken the notion of distributed systems. However, this planning system largely retains the same properties as those of a completely distributed one. Coordinator agents enable organized and efficient trading in the market, as well as facilitate human operator interaction. A coordinator agent is implemented on-board each vehicle, but only one is active at any give

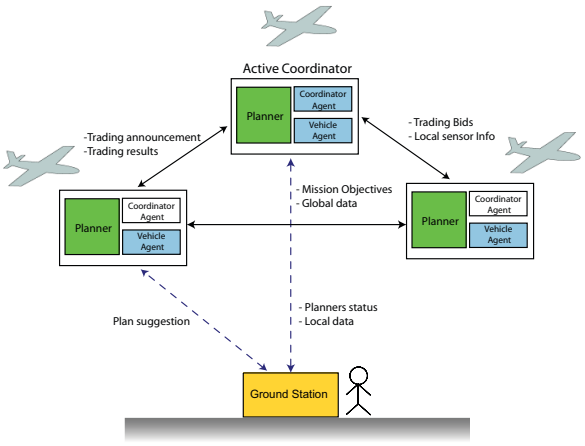

Fig. 3 Overall distributed structure of the planning system.

time. Another advantage of having coordinator agents is the monitoring of the team and the current task allocation. Therefore, if the coordinator agent finds that a vehicle is destroyed, it will put its tasks up for auction. Conversely, if the coordinator agent is damaged, a simple mechanism can be executed to elect a new coordinator agent.

The trading among the vehicles is governed by the *Market Protocol*. The listed items below describe the protocol used in the system.

1. There is only one active coordinator agent in a team.
2. Vehicle agents can only sell and buy tasks through the coordinator agent by submitting sell bids or buy bids.
3. Each vehicle agent is allowed to submit only one sell bid and one buy bid per trading cycle.
4. Each vehicle agent must submit a sell bid every trading cycle.
5. The tasks in a buy bid must be only those currently put up for sale in the market.
6. Vehicle agents must commit to every submitted sell bid.
7. A vehicle agent can remit its buy bid by sending a reject message to the coordinator when it is notified that it won the bidding. Otherwise, it must send a message to confirm the buy bid.
8. The coordinator agent will accept a sell bid only if there is at least one vehicle agent bid on the selling item with a higher price than the sell price.
9. The coordinator agent will accept a buy bid only if its price is the highest one of all the submitted bids for the bidding item.

10. The final price of a selling item for the winning bidder is equal to the second highest bid price.

The block diagram showing the architecture of each vehicle's planning system is illustrated in figure 4.

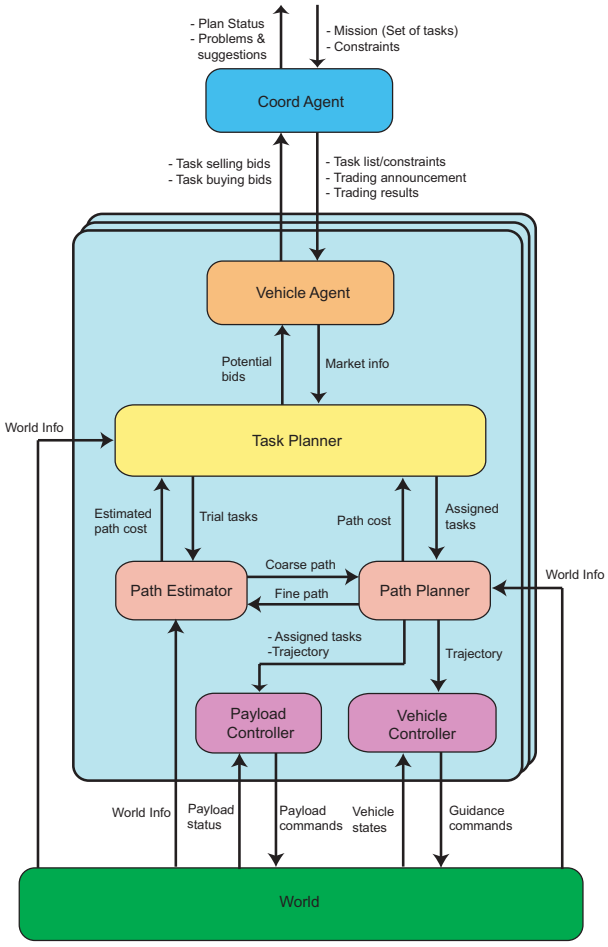

Fig. 4 Block diagram of each vehicle's planning system. Every vehicle contains a vehicle agent that sells and buys tasks for the vehicle. A task will be sold if its price is higher than the benefit for the vehicle upon executing the task. A task is bought if the benefit of its execution is more than its price. Each vehicle agent interacts with a coordinator agent and its local task planner.

In each vehicle, the task planner is the brain of the vehicle agent. It uses market price in addition to predicted environment states to make trading decisions. The task planner also schedules tasks that it currently possesses. An EC-based technique is used as the optimization engine of the task planner. One major beneficial property of EC-based techniques for coordinated systems is the availability of an intermediate solution at any time during the optimization process. A viable solution is available and continues to be improved until its output is required. Therefore, the task planner can provide a trading decision at any time to meet a deadline for submitting sell or buy bids.

To achieve the optimal solution for a task planning problem, the task planner needs to know the cost of every path for a vehicle executing each possible sequence of tasks. However, finding the optimal paths for all permutations is subject to the 'curse of dimensionality'. For example, the number of permutations of ten tasks is more than ten million. One way to approach this problem is to simultaneously optimize the path and task plan using multi-level optimization.[12] However, this approach is also computationally intensive and may be impossible to run on-line in real time. To simplify this problem, in our approach, the task planner selects a 'best' plan using estimated cost. This cost is based on the deviation of the estimated path from the straight line connecting the target locations. The data used in this estimation is stored in a table containing every cost of the best path from one target location to another. Once a set of tasks is allocated, the path planner searches for the optimal path for the assigned tasks and updates the estimates in the cost table. This technique improves the speed of the response of the planning system, and can provide viable solutions within short planning time.

The task planner of each vehicle also interacts with its path planner. It provides the task sequence as an input to the path planning module. The path planner computes the best path to execute the assigned tasks based on its knowledge of the operating field. It then sends this path to the vehicle guidance controller and feeds back the cost of the path to the task planner. The path planner can also interact with the path estimator by using the path provided by the path estimator as the initial guess in the optimization process, and it can then give the path estimator a detailed path for the current set of tasks and its cost. The task planner and path planner run simultaneously and continuously, although they may be updated at different rates. They can be used to generate off-line plans and also can run on-line while the vehicles execute the plans. Both of them use EC-based optimization algorithms which will be explained in the next section.

4. PLANNING ALGORITHMS

The general concept of the EC-based planning algorithm[4] used in this paper is illustrated in figure 5. The algorithm runs in a loop which has three phases. It starts by randomly generating a population of encoded plans. Then it evaluates the fitness value of each plan. The next step is to select the best plan to be the candidate solution for this generation and also select a portion of the plans in the population to be the parents of the next generation based on their fitness values. In our current implementation, a tournament selection scheme is used in the selection process. The last step is to produce off-

springs from the parents selected in the previous phase. An offspring is generated by cloning a parent and applying a mutation mechanism to it or by crossover of two parents. This loop is run continuously to update the plan as the optimization process proceeds.

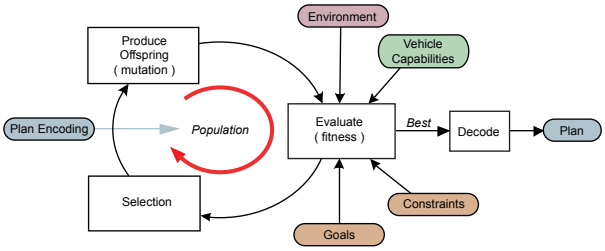

Fig. 5 Overview of Evolutionary Algorithm.

4.1. Path Planning

The path planning algorithm for a single vehicle we use in this paper is described in Rathbun and Capozzi.[3] The algorithm was further modified for dynamic replanning and presented in Rathbun et al.[13] The planning algorithm takes into account the uncertainties in the environment. The planner has knowledge of approximate locations of *sites* (targets and obstacles). Those locations are modelled as probability ellipsoids described by a probability density function. The probability that a vehicle's path will intersect with a site is approximated by digitally integrating the probability density function of the site along the vehicle path. The cost function $C(P_v)$ associated with a path P_v is based on the probability that vehicle v will hit obstacles in the field and the amount of fuel required to travel along the path.

A path is encoded as a sequence of simple *segments* chained end-to-end, shown in figure 6. The segment parameters are limited to keep motion within the vehicle capabilities. Continuity is enforced between the joining end of a segment and the starting point, heading, and speed of another joining segment. The four mutation mechanisms used in the mutation process are illustrated in figure 7.

4.2. Task Planning

In our approach, there is no centralized planner assigning tasks for each of the vehicles in the team. In contrast, the task allocation is a result of the task trading among the vehicles in the team. The trading is restricted by the Market Protocol described in section 3. Figure 8 shows the task trading sequence in a trading cycle.

The algorithm used in the task planner of each vehicle is similar to the path planning algorithm described above. A task plan for a vehicle is encoded as a sequence of tasks indicated by their task numbers. There are two parts in a task planner. One is used to schedule the current set of tasks, the other is used to determine trading bids. When the vehicle is not making a trading decision, the task planner runs the EC-based planning algorithm using the scheduling mutation illustrated in figure 9. To determine a buy bid, the scheduling process stops and the task planner generates a new population from a set of current best task plans. Each plan in the initial population is created by cloning one of the current best task plans, and then inserting a randomly selected selling task in the market at a random position. Then the task planner evolves the initial population by running the EC-based algorithm using the trading mutation mechanism shown in figure 9. The number of evolving steps depends on the time limit set by the coordinator agent. When a vehicle is asked for a sell bid, the task planner responds with a

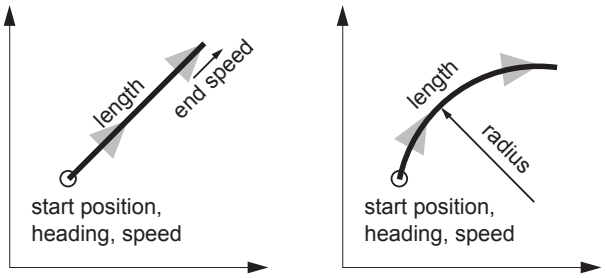

Fig. 6 Elemental path segment types. In the approach considered here, there are two elemental types of segments, straight lines and constant radius curves.

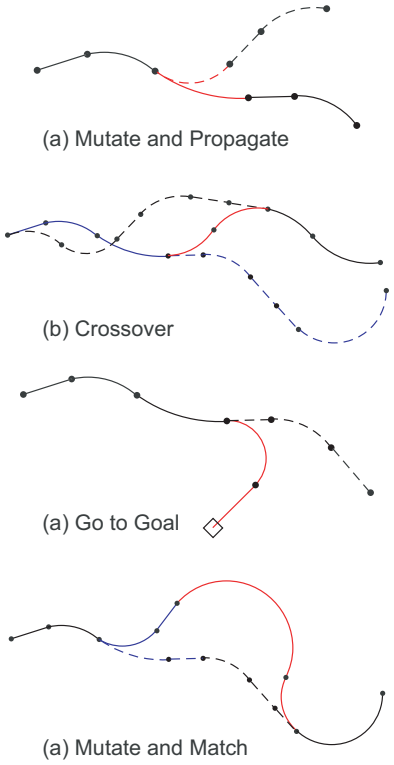

Fig. 7 Path mutation mechanisms

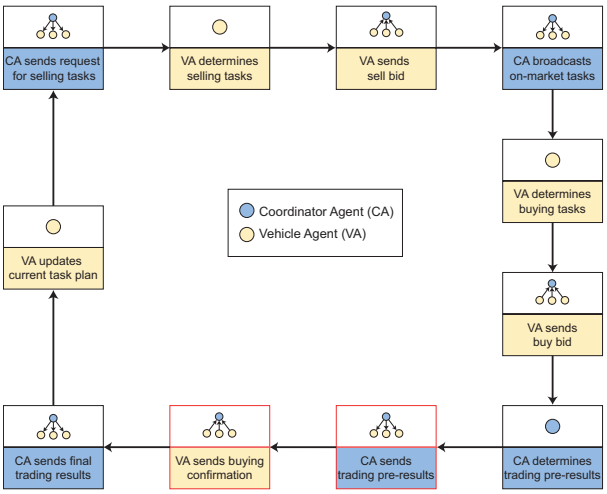

Fig. 8 Task trading sequence in one trading cycle. This process can be initialized by either randomly assigning tasks to each of the vehicles or putting all the tasks up for bidding with no reserved price until there are no more tasks left. The trading and planning processes run simultaneously.

randomly chosen current task.

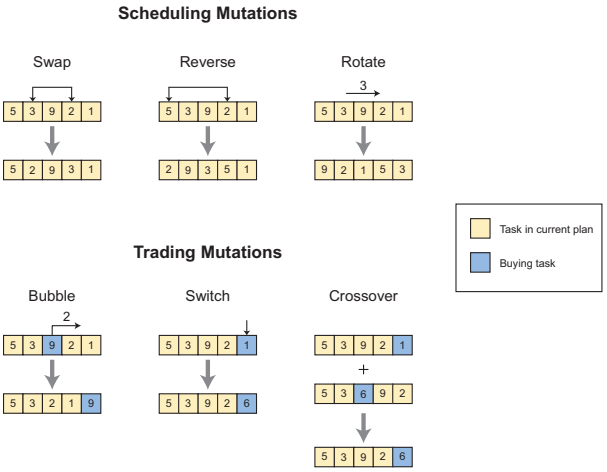

Fig. 9 Mutation mechanisms for task planning.

The fitness function used in both scheduling a task plan and determining a buy bid is the estimated team objective function described in equation 1. Using the team objective function, J, as the fitness function instead of each vehicle's individual objective function allows more cooperative behavior to emerge. Each vehicle estimates the team objective function based on the prices of the selling tasks and information acquired by communicating with the other vehicles. The more information sharing exists among the vehicles, the more cooperative behavior results in the task allocation plan. In the current implementation, the planning algorithm only uses prices to estimate the team objective function.

In the selling process, each planner sets a price for the randomly selected task using the same pricing strategy which is:

$$\text{sell price} = -\Delta\hat{J}_s$$

where $\Delta\hat{J}_s$ is an estimate of the difference in the team objective function value if the vehicle does not execute the task. The buying process is more difficult because each vehicle sells and buys in each trading cycle. The outcome of the buying decision depends on the unsettled sell bid. Therefore, the planner must make an assumption about the sell bid when submitting the buy bid. Based on that assumption, the price of a buy bid is set by

$$\text{buy price} = \Delta\hat{J}_b$$

where $\Delta\hat{J}_b$ is the estimate of the difference in the value of the objective function if the buying vehicle executes the buying task and no other vehicles do it.

Assuming the sell bid will be accepted is defined as *Type 1 assumption*. A *Type 2 assumption* assumes the sell bid will be rejected. Making a wrong assumption about the sell bid may result in an unfavorable trading outcome at the end of each trading cycle. Table 1 shows all possible outcomes after a vehicle participates in a trading cycle using type 1 assumption when there is no buying confirmation steps which are the red outlined boxes in the trading sequence shown in Figure 8. Tables 2 and 3 show the decision tables based on type 1 and type 2 assumption respectively with addition of the buying confirmation steps. These tables show that there are fewer chances for a wrong buying decision.

In fact, we can guaranteed the result of each trading cycle to be beneficial. This can be accomplished by letting each planner turns away every pre-accepted buy bid if there is a chance that the buying based on a wrong assumption results in lower objective function value. However, the search process could be struck in local minima[14] because swapping tasks between two vehicles would not occur in this situation. To avoid that problem, the vehicle using type 1 assumption has to take a risk to confirm a buy bid every time when it knows that its sell bid was pre-accepted. Therefore, the outcome in the dark shaded area in the decision table I will not happen. The outcome in the dark shaded area in the decision table II will not happen either because the planner will accept every buy bid if the initial sell bid result is according to its assumption.

After some investigation, we found that buying decisions based on the type 1 assumption are more suitable in some situations and those based on the type 2 assumption in others. Therefore, both of them must be used to achieve optimal solutions. In our implementation, the planner algorithm randomly selects an assumption about the sell bid and uses it in the buying decision process.

Buy bid / Sell bid	Accept	Reject
Accept	+	+
Reject	⊕	○

Symbols:
? Depend on decision
\+ Benefit increases
⊕ Benefit not decreases
○ Benefit unchanges
⊕ Benefit may decreases

Table 1 Agent decision table based on type 1 assumption when there is no buying confirmation steps in the trading sequence. This table shows all the possible outcomes, one of which results in reducing the value of the objective function.

Buying confirmation

Buy bid / Sell bid	Pre-Accept	Pre-Reject	Post-Accept	Post-Reject
Pre-Accept	?	⊕	⊕	⊕
Pre-Reject	?	○	⊕	○
Post-Accept	+	+	+	+
Post-Reject	⊕	○	⊕	○

Table 2 Agent decision table I based on type 1 assumption with addition of the buying confirmation steps. The dark and the light shaded areas show all the possible outcomes of a trading cycle.

Buying confirmation

Buy bid / Sell bid	Pre-Accept	Pre-Reject	Post-Accept	Post-Reject
Pre-Accept	?	⊕	⊕	⊕
Pre-Reject	⊕	○	+	○
Post-Accept	⊕	+	⊕	+
Post-Reject	⊕	○	+	○

Table 3 Agent decision table II based on type 2 assumption with addition of the buying confirmation steps. The dark and the light shaded areas show all the possible outcomes of a trading cycle.

To test our task planning algorithm, we applied it to a static multiple travelling salesman problem. In this problem there are five vehicles visiting thirteen depots. All the vehicles have the same goal location at (10,10). The score for each task is 200. The operating cost of each vehicle is the distance of the straight line connecting its starting position, allocated depots' locations and the goal multiplied by a scale factor. The results after 40 trading cycles are shown in figure 10 and 11. From the figures, we can see that the algorithms in both cases provide the optimal solution, but the algorithm in the latter case converges faster and its objective function value increase monotonically.

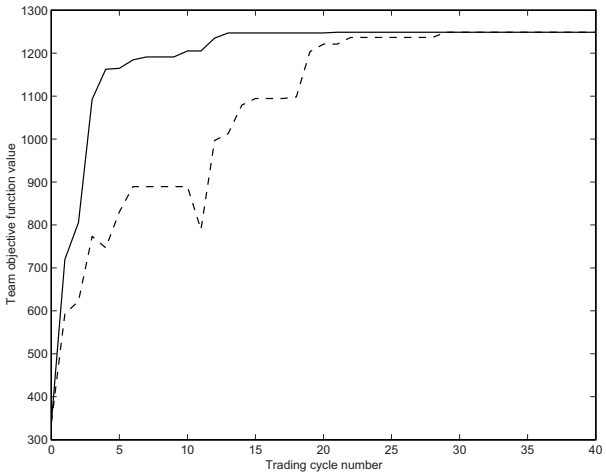

Fig. 10 Results of static multiple travelling salesman problem. The dashed line represents the value of the team objective function at each trading cycle without buying confirmation steps, and the solid line presents the value of the team objective function after addition of the buying confirmation steps.

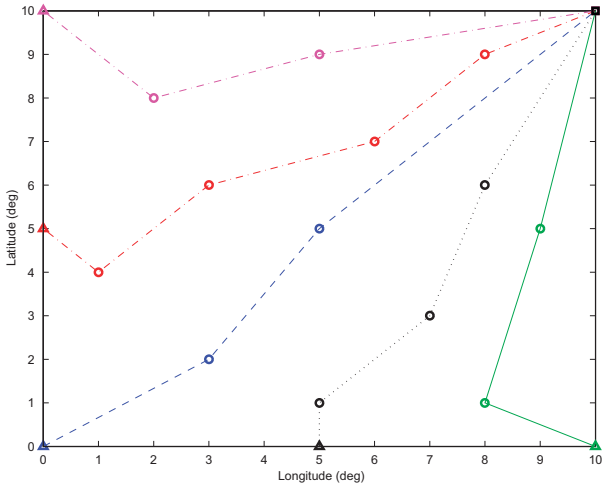

Fig. 11 Final task plan after 40 trading cycles, corresponding to figure 10. The triangles represent initial locations of the vehicles and the circles represent locations of the assigned depots. The goal location is at (10,10). The color of each depot is the same as the vehicle to which it is assigned

5. SIMULATION RESULTS

In this section, we present some of the results of simulations which demonstrate the performance of the planning algorithms described in section 4. In the scenario used for our simulations, there are three vehicles in the team. They are assigned to observe five different targets. The targets are distributed in various locations in the operating field. Each vehicle has a camera which has a limited zoom range for observing targets. We assume that the camera can rotate 360 degrees. Each vehicle also has a

sensor for detecting targets. The sensors have a specified range and a limited angle of detection. There are some obstacles in the field which the vehicles need to avoid. The vehicles are required to reach a goal location after executing their tasks.

The planning computation was initialized by randomly generating paths and assigning targets to the vehicles. Figure 12 shows the result of the off-line task and path planning before the vehicles start executing the plans. The vehicles efficiently allocate the tasks and successfully observe all the targets while avoiding the obstacles. The computation time for the off-line planning was less than one minute. In this simulation, the algorithms were implemented in C++ language and all the computation was processed on one computer with a Pentium 4 2.0 GHz processor. However, in a real system, the planner for each vehicle runs on a dedicated on-board processor.

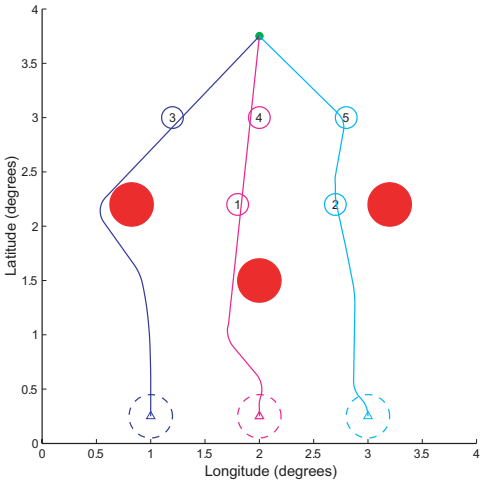

Fig. 12 Simulation result of off-line planning at time T_0. The vehicles are illustrated as small triangles with different colors. Each vehicle has a dashed circle around it to show the camera range of view. Targets are drawn as circles which have the same color as the vehicle to which they are allocated. The red circles in the figure are obstacles. The goal location is the small green circle.

Figure 13 illustrates the simulation result of the on-line planning at time T_6 after the vehicles have moved along the planned paths and just detected changes in the target locations. Figure 14 shows the result after the vehicles reallocated the tasks and replanned their paths at time T_7. Since the task and path planners of each vehicle run continuously, they adapt the task plan and path in response to the changes while the vehicle is executing the plan. The results show the vehicles successfully reallocate the tasks and replan their routes. The cycle time of the planner was about one second.

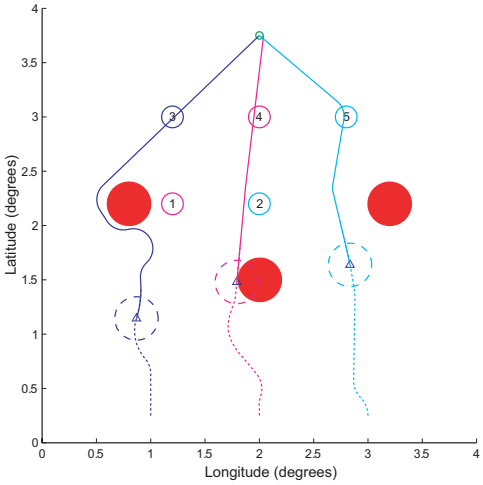

Fig. 13 Simulation result of on-line planning at time T_6. Since T_0 the vehicles have moved along the planned paths and just detected changes in the target locations. The dotted lines in the figure show the past trajectories of the vehicles.

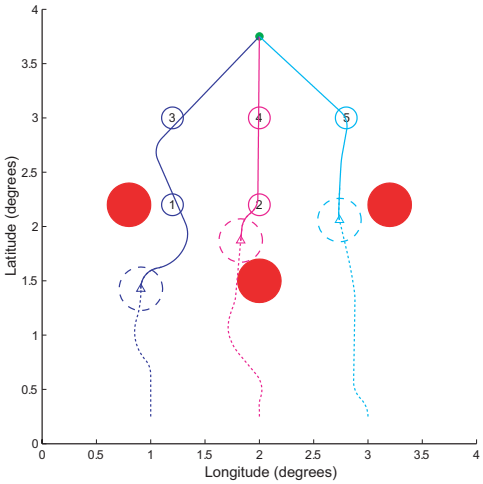

Fig. 14 Simulation result of replanning after detecting changes in target locations at time T_7 after the vehicles reallocated the tasks and replanned their paths.

6. CONCLUSION

In this paper, we have presented a framework and algorithms for combined task and path planning for multiple autonomous vehicles. The planning system takes advantage of the flexibility of EC-based techniques and the distributed structure of Market-based algorithms. This planning system can be used in both off-line planning and on-line replanning for vehicles in a dynamic environment. The results show that the algorithms are computationally feasible and capable of providing effective solutions. The research presented in this paper is an ongoing project. Future improvement of the algorithms will focus on path estimation and its interaction with path

planning, Market Protocol, the communication structure, and information flow among the vehicles and the command and control center.

7. ACKNOWLEDGMENTS

The research presented in this paper is funded by the DARPA Mixed Initiative Control of Automa-Teams (MICA) program. The authors would like to thank DARPA and the contracting agency, SPAWAR Systems Center, San Diego, CA, for the support that made this work possible (contract N66001-01-C-8074).

REFERENCES

[1]Capozzi, B. J., *Evolution-Based Path Planning and Management for Autonomous Vehicles*, Ph.D. thesis, University of Washington, 2001.

[2]Hocaoglu, C. and Sanderson, A. C., "Plannning Multiple Paths with Evolutionary Speciation," *IEEE Transctions on Evolutionary Computation*, Vol. 5, No. 3, June 2001, pp. 169–191.

[3]Rathbun, D. and Capozzi, B. J., "Evolutionary approaches to path planning through uncertain environments," *AIAA 1st Unmanned Aerospace Vehicles, Systems, Technologies, and Operations Conference and Workshop*, Portsmouth, VA, May 2002.

[4]Fogel, D. and Fogel, L., "Optimal Routing of Multiple Autonomous Underwater Vehicles through Evolutionary Programming," *Proceedings of the 1990 Symposium on Autonomous Underwater Vehicle Technology*, Washington, DC, 1990, pp. 44–47.

[5]Xiao, J. et al., "Adaptive evolutionary planner/navigator for mobile robots," *IEEE Transactions on Evolutionary Computation*, Vol. 1, April 1997, pp. 18–28.

[6]Smith, R. G., "The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver," *IEEE Transactions on Computers*, 1980, pp. 1104–1113.

[7]Fischer, K. et al., "A Model for Cooperative Transportation Scheduling," *Proceedings of the 1st International Conference on Multiagent Systems (ICMAS'95)*, San Francisco, CA, June 1995.

[8]Sandholm, T. W., "An Implementation of the Contract Net Protocol Based on Marginal Cost Calculations," *Proceedings of the 12th International Workshop on Distributed Artificial Intelligence*, Hidden Valley, Pennsylvania, 1993, pp. 295–308.

[9]Wellman, M. P. and Wurman, P. R., "Market-aware agents for a multiagent world," *Robotics and Autonomous Systems*, Vol. 24, September 1998, pp. 115–125.

[10]Dias, M. B. and Stentz, A., "A Free Market Architecture for Distributed Control of a Multirobot System," *Proceedings of the 6th International conference on intelligent autonomous systems*, July 2000.

[11]Camacho, E. F. and Bordons, C., *Model Predictive Control*, Springer, London, UK, 1999.

[12]Bialas, W. F. and Karwan, M. H., "Multilevel Linear Programming," Tech. Rep. 78-1, Department Of Industrial Engineering, State University Of New York At Buffalo, May 1978.

[13]Rathbun, D., Kragelund, S., Pongpunwattana, A., and Capozzi, B. J., "An Evolution Based Path Planning Algorithm for autonomous motion of a UAV through uncertain environments," *AIAA 21st Digital Avionics Systems Conference*, Irvine, CA, October 2002.

[14]Golfarelli, M., Maio, D., and Rizzi, S., "Multi-Agent Path Planning Based on Task-Swap Negotiation," *Proceedings 16th UK Planning and Scheduling SIG Workshop*, Durham, England, 1997, pp. 69–82.

[15]Dias, M. B. and Stentz, A., "A Market Approach to Multirobot Coordination," Technical Note CMU-RI-TR-01-26, The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, August 2001.

[16]Capozzi, B. J. and Vagners, J., "Evolving (semi)-autonomous vehicles," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Montreal, Canada, August 2001.

[17]Matos, N. and Sierra, C., "Evolutionary computing and negotiating agents," *Agent Mediated Electronic Commerce, First International Workshop on Agent Mediated Electronic Trading*, Vol. 1571 of *Lecture Notes in Computer Science*, 1999.

[18]Moree, B., Bos, A., and Tonino, H., "Cooperation by iterated plan revision," *Proceedings Fourth International Conference on MultiAgent Systems, IEEE Comput. Soc.*, 2000.

[19]Brumitt, B. and Stentz, A., "Dynamic Mission Planning for Multiple Mobile Robots," *Proceedings of the IEEE International Conference on Robotics and Automation*, April 1996.

[20]Ryan, J. et al., "Reactive Tabu Search in Unmanned Aerial Reconnaissance Simulations," *Proceedings of the 1998 Winter Simulation Conference*, 1998.

[21]Polycarpou, M. M., Yang, Y., and Passino, K. M., "A cooperative search framework for distributed agents," *Proceedings of the 2001 IEEE International Symposium on Intelligent Control*, Mexico City, Mexico, September 2001, pp. 1–6.

[22]Chien, S. et al., "Three Coordinated Planning Methods for Cooperating Rovers," *Proceedings of the World Automation Congress*, Wailea, HI, 2000.