International Journal of Artificial Intelligence, 9, 1–20 (1998)
© 1998 Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

Multiple Adaptive Agents for Tactical Driving

RAHUL SUKTHANKAR, SHUMEET BALUJA

rahuls@jprc.com, baluja@jprc.com

Justsystem Pittsburgh Research Center, 4616 Henry St., Pittsburgh PA 15213

and

The Robotics Institute, Carnegie Mellon University, Pittsburgh PA 15213-3891

JOHN HANCOCK

jhancock@ri.cmu.edu

The Robotics Institute, Carnegie Mellon University, Pittsburgh PA 15213-3891

Received ??; Revised ??

Editors: ??

Abstract. Recent research in automated highway systems has ranged from low-level vision-based controllers to high-level route-guidance software. However, there is currently no system for tactical-level reasoning. Such a system should address tasks such as passing cars, making exits on time, and merging into a traffic stream. Many previous approaches have attempted to hand construct large rule-based systems which capture the interactions between multiple input sensors, dynamic and potentially conflicting sub-goals, and changing roadway conditions. However, these systems are extremely difficult to design due to the large number of rules, the manual tuning of parameters within the rules, and the complex interactions between the rules. Our approach to this intermediate-level planning is a system which consists of a collection of autonomous agents, each of which specializes in a particular aspect of tactical driving. Each agent examines a subset of the intelligent vehicle's sensors and independently recommends driving decisions based on their local assessment of the tactical situation. This distributed framework allows different reasoning agents to be implemented using different algorithms.

When using a collection of agents to solve a single task, it is vital to carefully consider the interactions between the agents. Since each reasoning object contains several internal parameters, manually finding values for these parameters while accounting for the agents' possible interactions is a tedious and error-prone task. In our system, these parameters, and the system's overall dependence on each agent, is automatically tuned using a novel evolutionary optimization strategy, termed Population-Based Incremental Learning (PBIL).

Our system, which employs multiple automatically trained agents, can competently drive a vehicle, both in terms of the user-defined evaluation metric, and as measured by their behavior on several driving situations culled from real-life experience. In this article, we describe a method for multiple agent integration which is applied to the automated highway system domain. However, it also generalizes to many complex robotics tasks where multiple interacting modules must simultaneously be configured without individual module feedback.

Keywords: intelligent vehicles, evolutionary algorithms, simulation, distributed AI

1. Introduction

The task of driving can be characterized as consisting of three levels: strategic, tactical and operational [13]. At the highest (strategic) level, a route is planned and goals are determined; at the intermediate (tactical) level, maneuvers are selected to achieve short-term objectives — such as deciding whether to pass a blocking vehicle; and at the lowest (operational) level, these maneuvers are translated into control operations.

Mobile robot research has successfully addressed the three levels to different degrees. Strategic-level planners [18, 24] have advanced from research projects to commercial products. The operational level has been investigated for many decades, resulting in systems that range from semi-autonomous vehicle control [7, 11] to autonomous driving in a variety of situations [4, 15]. Substantial progress in autonomous navigation in simulated domains has also been reported in recent years [17, 3, 16]. However, the decisions required at the tactical level are difficult and a general solution remains elusive.

Consider the situation depicted in Figure 1. Car A, under computer control, is approaching its desired exit when it comes upon a slow moving blocker (Car B), in its lane. Car A's tactical reasoning system must determine whether to pass Car B and risk missing the exit. Obviously, the correct decision depends on a number of factors such as the distance to the exit, Car A's desired velocity, and the density and speed of surrounding traffic.

Such scenarios are of particular relevance to intelligent vehicles operating in mixed-traffic environments. In these environments, computer- and human-controlled cars share the roadway, and tactical decisions must be made without relying on communication-based protocols. This short-term planning problem is challenging because real-time decisions must be made based on incomplete, noisy information about the state of the world. Furthermore, the penalty for bad decisions is severe since errors in judgment may result in high-speed collisions.

SAPIENT, described in Section 3, is a tactical reasoning system designed to drive intelligent vehicles, such as the Carnegie Mellon Navlab [23], in mixed-traffic environments. In SAPIENT, decisions are made by a collection of independent agents, termed *reasoning agents*, each of which specializes in a particular aspect of the tactical driving task. This article focuses on how these agents automatically configure themselves to optimize a user- specified evaluation function using a novel evolutionary algorithm termed Population Based Incremental Learning (PBIL).

This article is organized as follows. Section 2 presents the simulated highway environment used to train the SAPIENT agents. Section 3 details the SAPIENT architecture, describing the reasoning agents and their voting language. Section 4 introduces PBIL, and explains the encoding scheme used to represent agent parameters. Subsequent sections present our results, both on small-scale tactical scenarios (such as the one shown in Figure 1), and on larger highway configurations. Finally, Section 8 summarizes the research and outlines areas for further research.

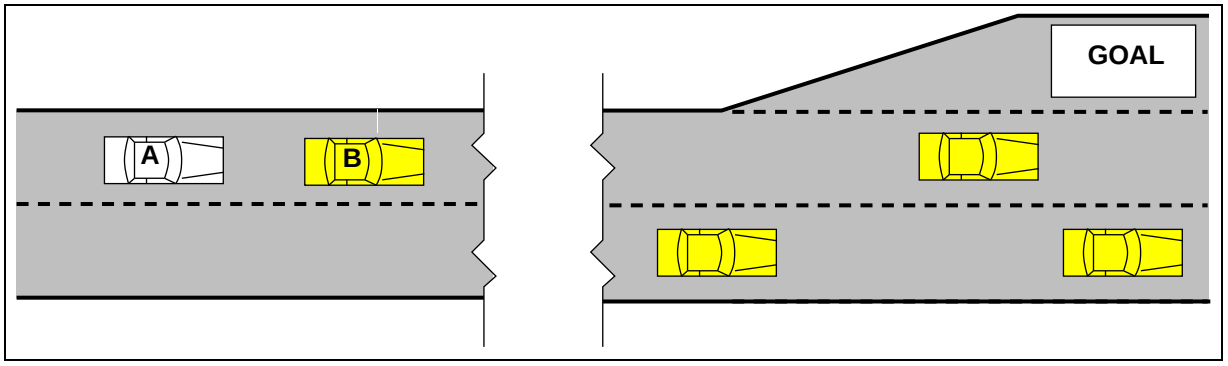


Fig. 1. An example of tactical-level reasoning. Car A is approaching its desired exit behind a slow vehicle B. Should Car A attempt to pass?

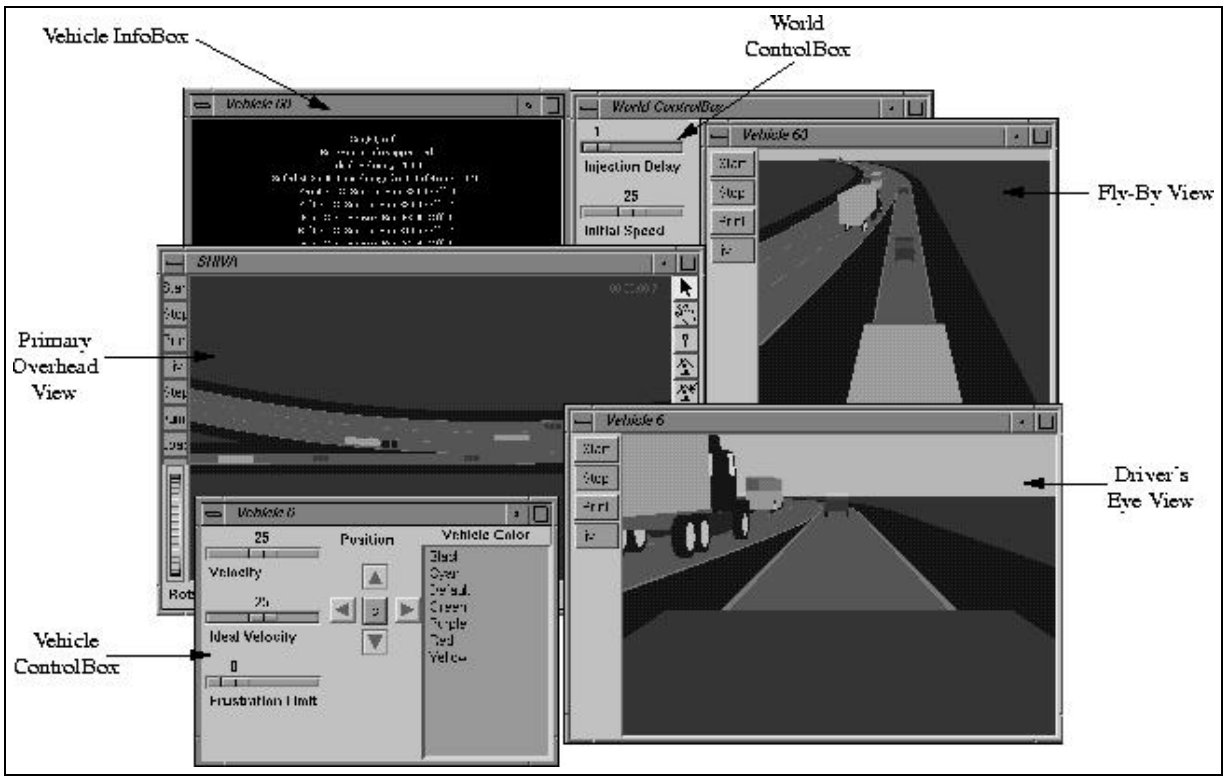


Fig. 2. SHIVA: A design and simulation tool for developing intelligent vehicle algorithms.

2. The SHIVA Simulator

Simulation is essential in developing intelligent vehicle systems because testing new algorithms in real traffic is expensive, risky and potentially disastrous. SHIVA[1] (**S**imulated **H**ighways for **I**ntelligent **V**ehicle **A**lgorithms) [22, 21] is a kinematic micro-simulation of vehicles moving and interacting on a user-defined stretch of roadway that models the elements of the tactical driving domain most useful to intelligent vehicle designers. The vehicles can be equipped with simulated human drivers as well as sensors and algorithms for automated control. These algorithms direct the vehicles' motion through simulated commands to the accelerator, brake, and steering wheel. SHIVA's user interface provides facilities for visualizing and specifying the interactions between vehicles (see Figure 2). The internal structure of the simulator is comprehensively covered in [22], and details of the design tools may be found in [21].

All simulated vehicles are composed of three subsystems: perception, cognition, and control. The perception subsystem consists of a suite of simulated functional sensors (e.g., global positioning systems, range-sensors, lane-trackers), whose outputs are similar to real perception modules implemented on the Navlab vehicles. Simulated vehicles use these sensors to obtain information about the road geometry and surrounding traffic. Vehicles may control the sensors directly, scanning and panning the sensors as needed, encouraging active perception. Some sensors also model occlusion and noise, forcing cognition routines to be realistic in their input assumptions.

While a variety of cognition modules have been developed in SHIVA, this article only discusses two types: rule-based reasoning and SAPIENT. The rule-based reasoning system, which was manually designed, is implemented as a monolithic decision tree. It consists of a collection of tactical driving rules such as:

> "Initiate a left lane change if the vehicle ahead is moving slower than $f(v)$ m/s, and is closer than $h(v)$, and if the lane to your left is marked for legal travel, and if there are

no vehicles in that lane within $g(v)$ meters, and if the desired right-exit is further than $e(x, y, v)$ meters."

where: $f(v)$ is the desired car following velocity, $h(v)$ is the desired car following distance (headway), $g(v)$ is the required gap size for entering an adjacent lane, and $e(x, y, v)$ is a distance threshold to the exit based on current lane, distance to exit and velocity. While this system performs well on many scenarios, it suffers from four disadvantages: 1) as the example above illustrates, realistic rules require the designer to account for many factors; 2) modification of the rules is difficult since a small change in desired behavior can require many non-local modifications; 3) hand-coded rules perform poorly in unanticipated situations; 4) implementing new features requires one to consider an exponential number of interactions with existing rules. Similar problems were reported by Cremer *et al.* [3] in their monolithic state-machine implementation for scenario control. The SAPIENT distributed architecture, discussed in the next section, was developed to address some of these problems.

The control subsystem is compatible with the controller available on the Carnegie Mellon Navlab II robot testbed vehicle. Commands to the controller are issued by the cognition modules at a rate of 10 Hz.

3. SAPIENT

SAPIENT (**S**ituation **A**wareness **P**lanner **I**mplementing **E**ffective **N**avigation in **T**raffic) [20] consists of a collection of independent modules (termed *reasoning agents*), each of which is an expert on a specific aspect of the tactical driving task. Each agent is assigned to monitor a relevant physical entity in the environment and is responsible for assessing the repercussions of that entity on the intelligent vehicle's upcoming actions (see Figure 3). For example, the reasoning agent associated with a vehicle ahead monitors the motion of that vehicle and determines whether to continue car following, initiate a lane change, or begin braking. Similarly, a reasoning agent associated with an upcoming exit is concerned with recommending the lane changes and speed changes needed to successfully maneuver the intelligent vehicle to the off-ramp.

3.1. System Overview

The SAPIENT architecture is shown in Figure 4. The *perception modules* (depicted as ellipses) are connected to the intelligent vehicle's sensors, and perform functions such as lane tracking or vehicle detection. Wherever possible, they correspond to existing systems available on the Carnegie Mellon Navlab (e.g., the lane tracker is based on ALVINN [15]). Each *reasoning agent* (shown as a dark rectangle) obtains information about the situation from one or two perception modules, and independently calculates the utility of various courses of action. This information is then sent to the *voting arbiter*, which integrates these recommendations and selects the appropriate response. Finally, the tactical action is translated into steering and velocity commands and executed by the operational-level controller.

As seen in Figure 4, reasoning agents can be classified into classes based on their area of specialization. SAPIENT's loosely-coupled architecture allows new classes to be developed without modifying the existing reasoning agents. Our current implementation spans the following tactical-level aspects:

- Road properties: local geometry, legal lanes, speed limits, etc.
- Nearby vehicles: sizes, positions, and velocities
- Exits: distance, exit lane, speed constraints
- Self-state: current velocity, lateral position, explicit goals

Each reasoning agent tracks the associated physical entity's attributes by monitoring the appropriate sensors. For example, a reasoning agent associated with a nearby vehicle normally tracks its longitudinal position and velocity, and its lateral position (mapped into road coordinates). The tracking has two important implications. First, it allows the reasoning agent to obtain a better estimate of the relevant attribute. Second, the reasoning agent can accumulate statistics that can help influence decisions. For instance, based on the ir-

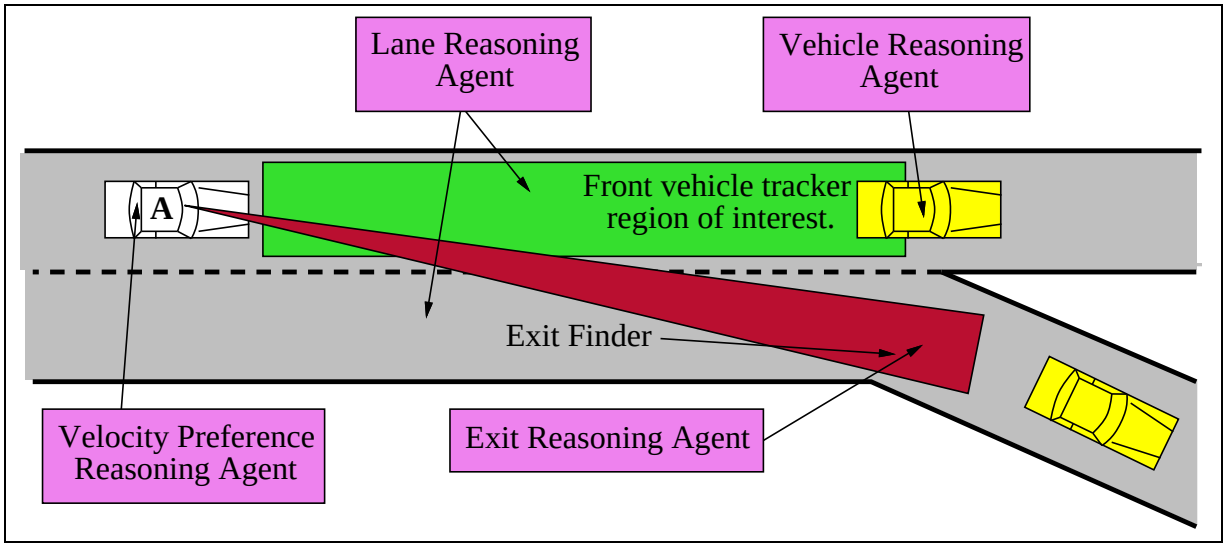


Fig. 3. SAPIENT reasoning agents are associated with relevant physical entities in the environment. In this situation, the intelligent vehicle (A) is following a car and approaching its desired exit.

regular lane-keeping performance of a nearby vehicle (an indication of an inexperienced or intoxicated driver), the reasoning agent associated with that vehicle could favor actions that maintain a greater distance from that vehicle. Thus, SAPIENT is not a purely reactive system; the local state associated with each reasoning agent allows SAPIENT to make decisions based on past history. The relevant history is maintained by each agent.

Externally, all reasoning agents share a similar structure — each agent accepts inputs from a subset of the intelligent vehicle's perception modules and sends outputs to the voting arbiter as a set of votes over the entire *action space* (See Section 3.2). Internally, however, SAPIENT's reasoning agents are heterogenous, maintaining local state and using those representations that are most applicable to the assigned subtask. For example, the reasoning agents responsible for exit management are rule-based while the reasoning agent monitoring other vehicles use generalized potential fields [9, 10]. The different reasoning agent types and their associated algorithms are detailed in [20].

SAPIENT reasoning agents are myopic in their outlook. For example, the Exit Reasoning Object's votes are not influenced by the presence of the blocking vehicle; conversely, the reasoning agent associated with the blocking vehicle is oblivious to the exit. Finally, the arbiter is completely ignorant of the driving task. Yet the combination of these local reasoning schemes leads to a distributed awareness of the tactical-level situation. Before discussing how a knowledge-free arbiter can combine these local views of the tactical driving task, a closer look at the action space is warranted.

3.2. Actions

Tactical maneuvers (such as lane changing) are composed by concatenating several basic actions. Reasoning agents indicate their preference for a basic action by assigning a vote to that action. The magnitude of the vote corresponds to the intensity of the preference and its sign indicates approval or disapproval. Each reasoning agent must assign some vote to every action in the action space. All actions have velocity (longitudinal) and lane-offset (lateral) components; for example, "brake hard while changing left" or "increase speed and maintain your current lane position".

Since different reasoning agents can return different recommendations for the next action, conflicts must be resolved. SAPIENT uses a voting arbiter to perform this integration. At each timestep, the reasoning agents synchronously submit votes or vetoes for each action in the action space (see Table 1). During arbitration, all of the votes for a given action are summed together (after

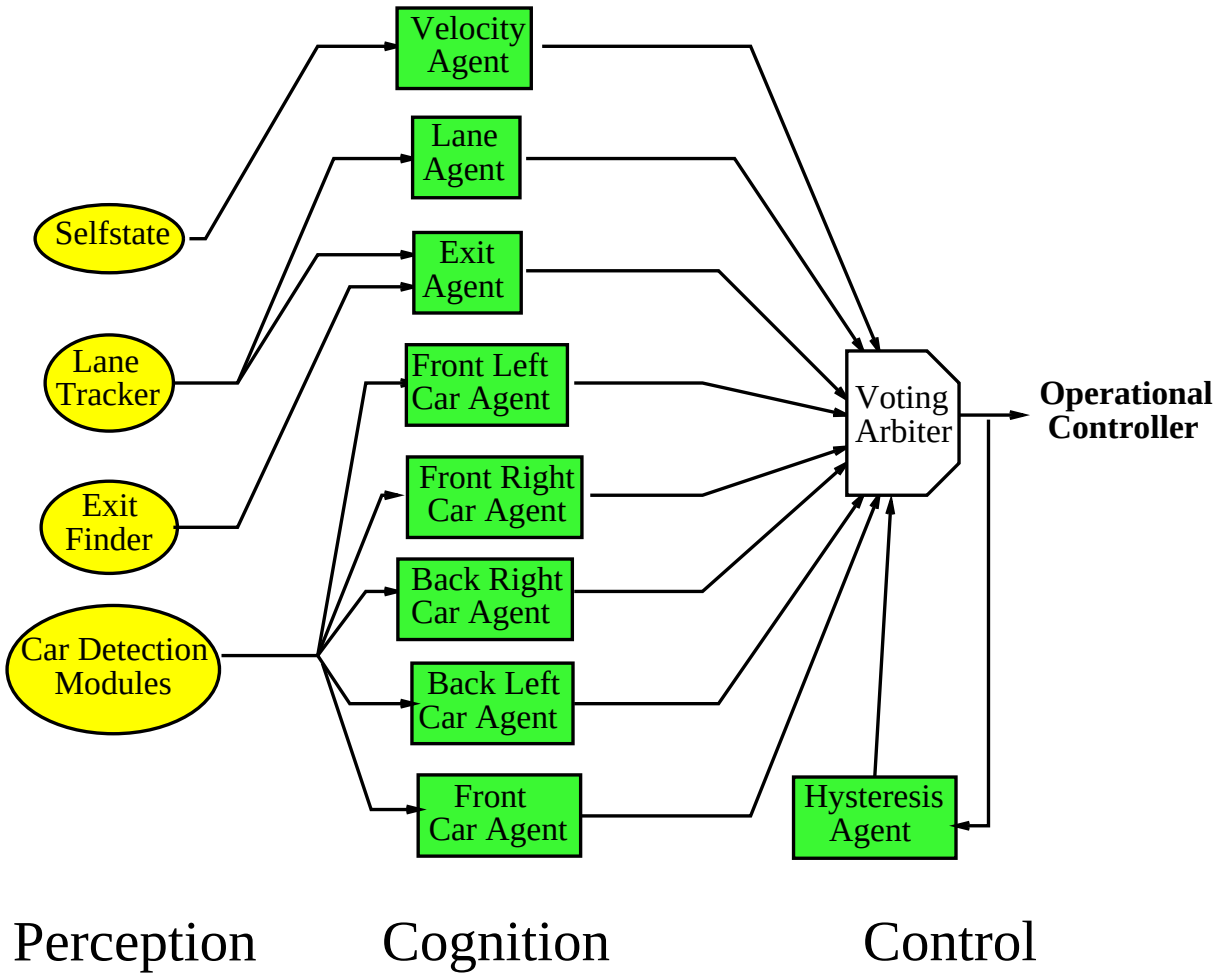


Fig. 4. SAPIENT consists of a collection of reasoning agents which recommend actions based upon local considerations. Each reasoning agent monitors a subset of the vehicle's sensors and votes upon a set of possible actions. The hysteresis reasoning agent is responsible for maintaining consistency over time (especially in cases where multiple actions are equally advantageous); this is done by voting in favor of the action selected in the previous time step. Action fusion is performed by a domain-independent, voting arbiter.

Table 1. The action space is a 3×3 discretization of the lateral/longitudinal space. The labels are translated at the operational level into specific numbers. Thus, "left" and "right" map to lateral positions (e.g., move left/right by 0.1 lane widths) while "accelerate" and "decelerate" map to changes in velocity (e.g., speed up/slow down by 0.1 m/s).

accelerate/shift-left	accelerate/straight	accelerate/shift-right
coast/shift-left	coast/straight	coast/shift-right
decelerate/shift-left	decelerate/straight	decelerate/shift-right

being scaled by the reasoning agent's *influence weight*), and the action with the most accumulated votes (which has not been vetoed by any agent) is executed. The actions used in the implementation described in this article are summarized in Table 1. Finer discretizations and alternate action spaces are discussed in [20]. Although the action space restricts reasoning objects to voting on their adjacent lanes, the reasoning agents can internally plan longer-range courses of action. For example, the *exit agent* can vote for lane changes towards the exit, even when the exit is several lanes away.

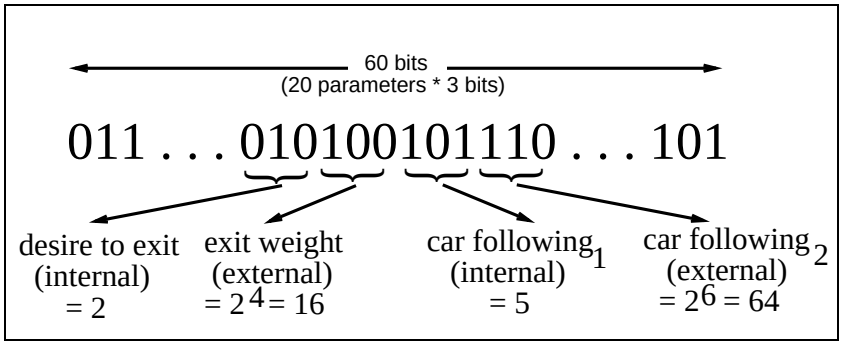


Fig. 5. The three-bit encoding scheme used to represent parameters in the search space: internal parameters are linearly scaled while external ones are exponentially scaled.

3.3. Parameters

Different reasoning agents use different internal algorithms. Each reasoning agent's output depends on a variety of *internal parameters* (e.g., thresholds, gains, etc.). Before going to the arbiter, each agent's outputs are scaled by its influence weight (*external parameters*).

When a new reasoning agent is being implemented, it is difficult to determine whether a vehicle's poor performance should be attributed to a bad choice of parameters in the new agent, a bug in the logic of the new reasoning agent or, more seriously, to a poor representation scheme (inadequate configuration of reasoning agents). To overcome this difficulty, we have implemented a method for automatically configuring the parameter space. A total of twenty parameters, both internal and external, were selected for the tests described here, and each parameter was discretized into eight values (represented as a three-bit string). For internal parameters, whose values are expected to remain within a certain small range, we selected a linear mapping (where the three bit string represented integers from 0 to 7); for the external parameters, we used an exponential representation (with the three-bit string mapping to eight values in the range 0 to 128). The latter representation increases the range of possible weights at the cost of sacrificing resolution at the higher magnitudes. A representation with more bits per parameter would allow finer tuning but increase the training time. The encoding is illustrated in Figure 5. In the next section, we describe the evolutionary algorithm used for the learning task.

4. Population-Based Incremental Learning

Population-Based Incremental Learning (PBIL) is a combination of genetic algorithms (GAs) [8] and competitive learning [1, 2]. The PBIL algorithm attempts to explicitly maintain statistics about the search space and uses them to direct its exploration. The object of the algorithm is to create a real valued probability vector which, when sampled, reveals high quality solution vectors with high probability. The full algorithm is presented in Figure 6.

Initially, each element of the PBIL probability vector is initialized to 0.5. Sampling from this vector yields random solution vectors because zeros and ones are generated with equal probability in each bit position. As training progresses, the values in the probability vector gradually shift to represent high evaluation solution vectors through the following process. A number of solution vectors are generated based upon the probabilities specified in the probability vector. The probability vector is pushed towards the generated solution vector with the highest evaluation. After the probability vector is updated, a new set of solution vectors is produced by sampling from the updated probability vector, and the cycle is continued. As the search progresses, the entries in the probability vector move away from their initial settings of 0.5 towards either 0.0 or 1.0. *The best solution ever generated in the run is returned as the final solution.* Note that because the algorithm only returns the best solution generated during the run, convergence of the probability vector is not a prerequisite for the success of the algorithm.

```
for i := 1 to LENGTH do P[i] = 0.5;

while (NOT termination condition)
      ****** Generate Samples ******
      for i := 1 to SAMPLES do
            sample_vectors[i] := generate_sample_vector_according_to_probabilities(P);
            evaluations[i] := Evaluate_Solution( sample_vectors[i]; );
      best_vector := find_vector_with_best_evaluation( sample_vectors, evaluations );

      ****** Update Probability Vector towards best solution ******
      for i := 1 to LENGTH do
            P[i] := P[i] * (1.0 − LR) + best_vector[i] * (LR);

      ****** Mutate Probability Vector ******
      for i := 1 to LENGTH do
            if (random (0,1) < MUT_PROBABILITY) then
                  if (random (0,1) > 0.5) then mutate_direction := 1;
                  else mutate_direction := 0;
                  P[i] := P[i] * (1.0 − MUT_SHIFT) + mutate_direction * (MUT_SHIFT);

return the best solution found in run;

USER DEFINED CONSTANTS (Values Used in this Study):
SAMPLES: the number of vectors generated before update of the probability vector (100)
LR: the learning rate, how fast to exploit the search performed (0.1).
LENGTH: the number of bits in a generated vector (3 * 20)
MUT_PROBABILITY: the probability of a mutation occuring in each position (0.02).
MUT_SHIFT: the amount a mutation alters the value in the bit position (0.05).
```

Fig. 6. The PBIL algorithm used to train SAPIENT reasoning agent parameters. Here, the explicit preservation of the best solution from the previous generation (elitist selection) is not shown.

However, empirically, the probability vector has converged in all of the runs conducted.

The probabilistic generation of solution vectors does not guarantee the creation of a good solution vector in every iteration. This problem is exacerbated by the small population sizes used in our experiments. Therefore, in order to avoid moving towards unproductive areas of the search space, the best vector from the previous population is included in the current population (by replacing the worst member of the current population) — in GA literature, this is termed *elitist selection* [8].

Since space limitations preclude a complete discussion about the relationship between GAs and PBIL, we can only provide a brief intuition. Diversity in the population is crucial for GAs. By maintaining a population of solutions, the GA is able — in theory at least — to maintain samples in many different regions. In genetic algorithms, crossover is used to merge these different solutions. However, when the population converges, crossover is deprived of the diversity that it needs to be an effective search operator. When this happens, crossover begins to behave like a mutation operator that is sensitive to the convergence of the value of each bit. If all individuals in the population converge at some bit position, crossover leaves those bits unaltered. At bit positions where individuals have not converged, crossover will effectively mutate values in those positions. Therefore, crossover creates new individuals that differ from the individuals it combines only at the bit positions where the mated individuals disagree. This is analogous to PBIL which creates new trial solutions that differ mainly in bit positions where prior good performers have disagreed. More details can be found in [1].

Our application challenges PBIL in a number of ways. First, since a vehicle's decisions depend on the behavior of other vehicles which are not under

its control, each simulation can produce a different evaluation for the same bit string. We evaluate each set of vehicle parameters multiple times to compensate for the stochastic nature of the environment. Second, the PBIL algorithm is never exposed to all possible traffic situations (thus making it impossible to estimate the "true" performance of a PBIL string). Third, since each evaluation takes considerable time to simulate, minimizing the total number of training evaluations is important.

5. Training Specifics

All of the tests described below were performed on the track shown in Figure 8, known as the SHIVA *cyclotron*. While this configuration does not resemble a real highway, it has several benefits as a testbed: 1) It is topologically identical to a highway with equally spaced exits; 2) Taking the nth exit is equivalent to traveling n laps of the course; 3) One can create challenging traffic interactions at the entry and exit merges with only a small number of vehicles.

During training, each scenario was initialized with one SAPIENT/PBIL vehicle, and eight rule-based cars (with hand-crafted decision trees). The SAPIENT car was directed to take the second exit (1.5 revolutions) while the other cars had goals of zero to five laps. Whenever the total number of vehicles on the track dropped below nine, a new vehicle was injected at the entry ramp to maintain the desired traffic density. Only one SAPIENT vehicle was permitted on the course at a time.

At the start of the run, the PBIL algorithm suggested a candidate bit-string which was converted into SAPIENT parameters, and instantiated as a simulated vehicle. Each evaluation of a PBIL parameter string required one run of a simulated vehicle. At the end of the vehicle's run, the score that it received was sent to PBIL as the evaluation of that candidate bit-string. It should be noted that the population size in PBIL affected the number of evaluations required in each generation of the PBIL algorithm. The population size does not correspond to the number of SAPIENT vehicles present on the track since each candidate vehicle was independently evaluated (as stated earlier, only one SAPIENT vehicle was permitted on the track at a time).

Whenever a SAPIENT vehicle left the scenario (upon taking an exit, or crashing 10 times), its evaluation was computed based on statistics collected during its run. This score was used by the PBIL algorithm to update the probability vector — thus creating better SAPIENT agents in the next generation.

While the definition of good driving is largely subjective, the following characteristics are strongly correlated with *bad* driving: 1) collisions; 2) taking the wrong exit; 3) deviating from the desired speed; 4) weaving (poor lane tracking). Many possible evaluation functions could be constructed from these characteristics. For our evaluation function, we combined them in a simple weighted sum, to be maximized:

$$\begin{aligned} \text{Eval} \;=\; & -(10000 \times \text{all-veto}) \\ & -(1000 \times \text{num-crashes}) \\ & -(500 \times \text{if-wrong-exit}) \\ & -(0.02 \times \text{speed-deviation}) \\ & -(0.02 \times \text{lane-deviation}) \\ & +(\text{dist-traveled}) \end{aligned}$$

where: `all-veto` indicates that the SAPIENT vehicle objects to all actions (with good parameters, this should never happen); `num-crashes` is the number of collisions involving the SAPIENT vehicle; `if-wrong-exit` is a flag — true if and only if the SAPIENT vehicle exited prematurely, or otherwise missed its designated exit; `speed-deviation` is the difference between desired and actual velocities, integrated over the entire run; `lane-deviation` is the deviation from the center of a lane, integrated over the entire run; `dist-traveled` is the longitudinal distance covered by the vehicle, in meters (an incremental reward for partial completion)

While the evaluation function is a reasonable measure of performance, it is important to note that there can be cases when a "good" driver becomes involved in unavoidable accidents; conversely, favorable circumstances may enable "bad" vehicles to score well on an easy scenario. To minimize the effects of such cases, we tested each candidate string in the population on a set of four scenarios. In addition to traffic, these test cases in-

cluded some pathological situations with broken-down vehicles obstructing one or more lanes.

6. Training Results

We performed a series of experiments using a variety of PBIL population sizes, evaluation functions and initial conditions. More details about individual experiments are presented in the next section. This section focuses on evaluation metrics for the training algorithm.

Figure 7 shows the results of a training run with the evaluation function described earlier, and a PBIL population size of 100. These 3-D histograms display the distribution of scores for each generation. It is clear that as the parameters evolve in successive generations, the average performance of SAPIENT vehicles increases, and the variance of evaluations within a generation decreases. In the experiments with population size 100, good performance of some vehicles in the population is achieved early (by the fifth generation) although consistently good evaluations are not observed until generation 15. The number of vehicles scoring poor evaluations drops rapidly until generation 10, after which there are only occasional low scores. The PBIL strings converge to a stable set of SAPIENT parameters, and by the last generation, the majority of the vehicles are able to take the proper exit, and avoid crashes in all scenarios. The results of experiments with different population sizes were similar.

Figure 8 shows a scenario on the cyclotron track. This scenario is pathological, in that it contains many broken-down vehicles, scattered over the roadway. The trace shows a trained SAPIENT vehicle successfully navigating the course by avoiding the obstacles.

Above, we described the overall performance of the SAPIENT vehicles in terms of a global evaluation function. Here, we examine how the individual components of the scoring metric improve over time. Three observable quantities play a significant role in the SAPIENT training evaluation function: κ, the total number of near-collisions; β, whether the vehicle made its exit; and, ζ , the distance traveled by the intelligent vehicle in the scenario. Thus, for a given population of SAPIENT vehicles, the quantities: $K = \sum_{\forall v} \kappa$, $B = \sum_{\forall v} \beta$, $Z = \sum_{\forall v} \zeta$ (over all vehicles, v, in the population), reflect the "goodness" of the population. The three graphs in Figure 9 show how K, B, and Z change over successive generations. Each PBIL population contains 40 vehicles, and each vehicle is evaluated on four different scenarios. The graphs show that:

- The number of near-collisions, K, drops steadily as PBIL tunes the SAPIENT reasoning agent parameters. In the final generation, *none* of the vehicles in the population are involved in *any* near-collisions over the entire set of four scenarios.
- The fraction of vehicles in the population which missed their exit also decreases steadily over time as the SAPIENT vehicles learn. This too is zero in the final generation.
- The third quantity, Z, reflects the incremental improvement in performance of the vehicles during training. It can be seen that the early vehicles are eliminated from the scenario (either through timeout, taking the wrong exit, or crashing) before they travel very far. Over time, the vehicles are able to travel greater distances. Note that Z has an upper bound (since ζ cannot be greater than the distance to the desired exit).

To investigate the robustness of this training method, two additional sets of experiments were performed, where the coefficients in the evaluation function were perturbed. In the first set, six experiments were conducted, and in each experiment, one coefficient was multiplied by 10. Some of these results are shown in Figures 10 and 11. Somewhat surprisingly, the SAPIENT reasoning agents generated from these perturbed evaluation functions are still successful. We hypothesize two reasons for this: 1) the tactical-level sub-tasks are closely linked: it is quite likely that a vehicle which makes the correct exit has also learned to avoid collisions — otherwise it would have been eliminated in a collision earlier on the track; 2) although PBIL is responsible for setting the internal and external parameters for each reasoning agent, the underlying algorithms are predefined; thus, a small perturbation in reasoning agent parameters does not cause catastrophic failures in the system.

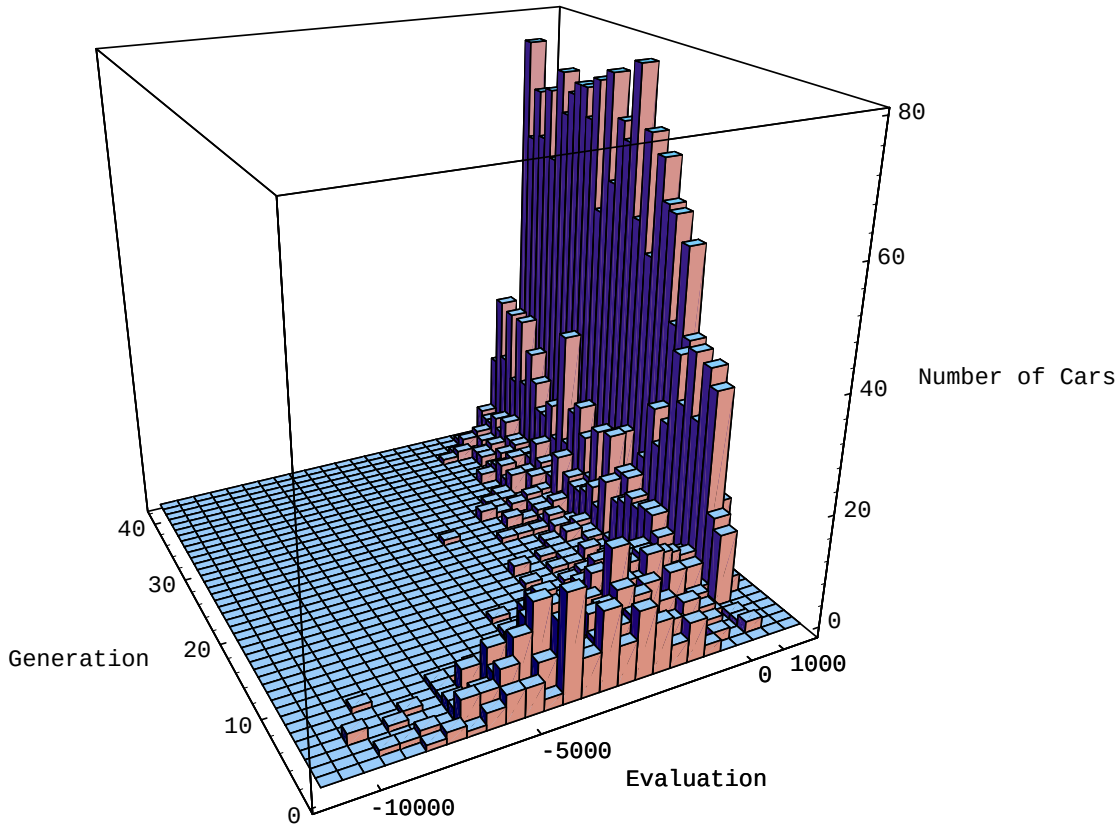


Fig. 7. 3-D Histogram showing increase of high-scoring PBIL strings over successive generations. Population size is 100 cars in each generation.

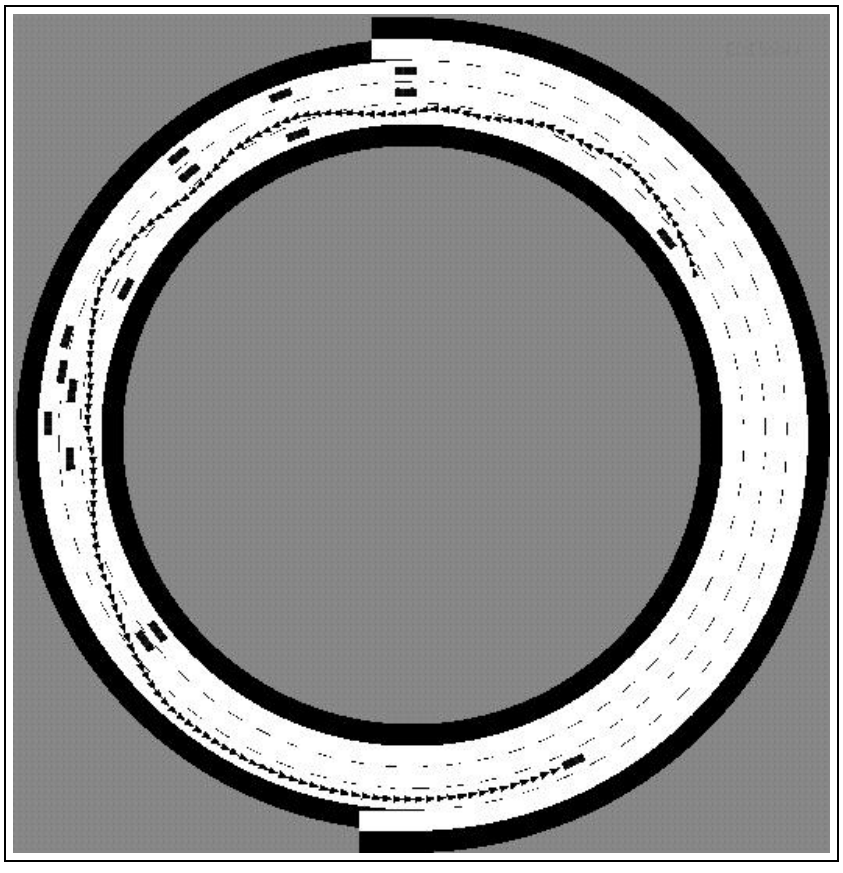


Fig. 8. A pathological scenario on the cyclotron track with 15 obstacles. The trace shows a SAPIENT vehicle successfully navigating the course by avoiding the obstacles.

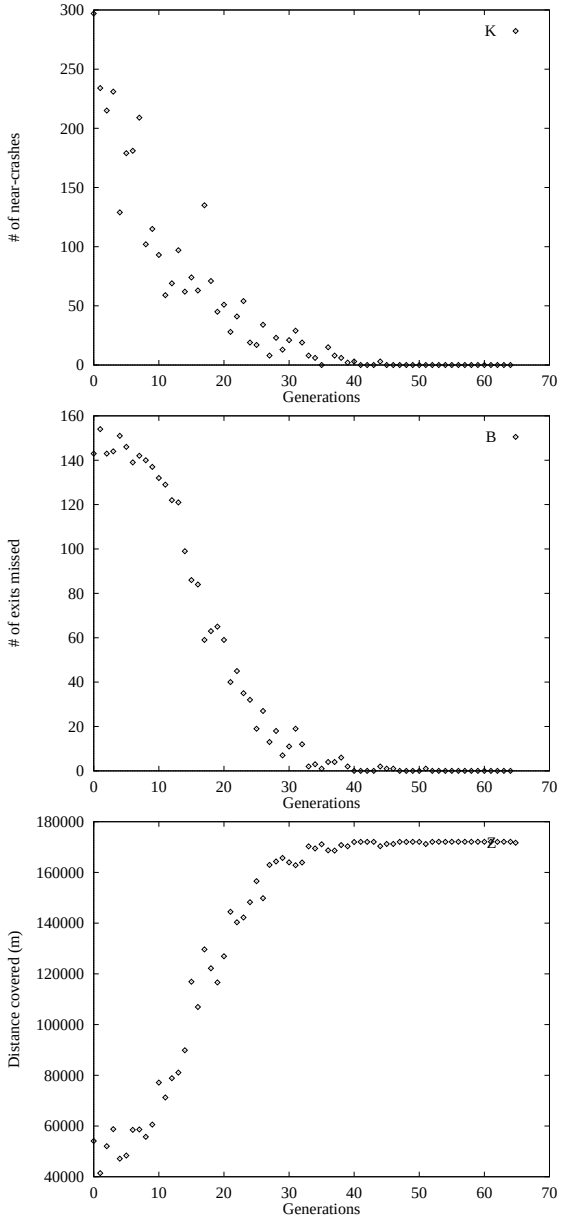
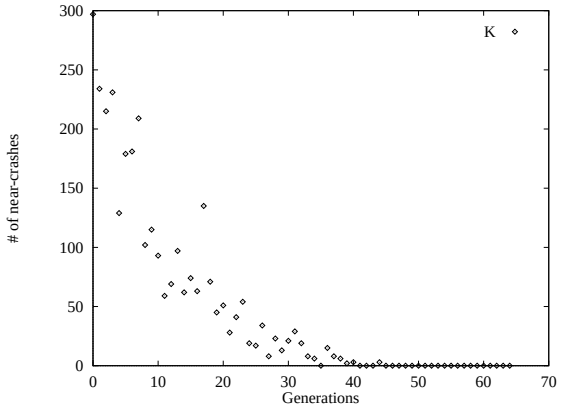
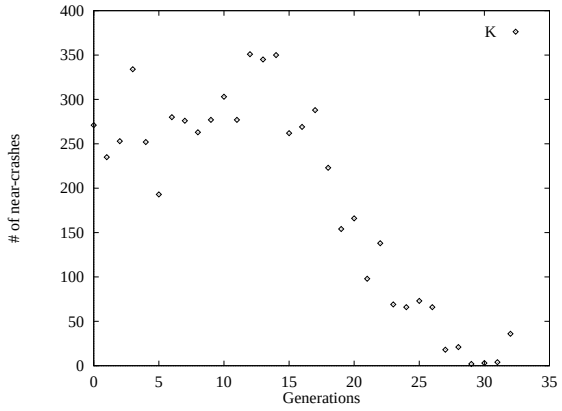


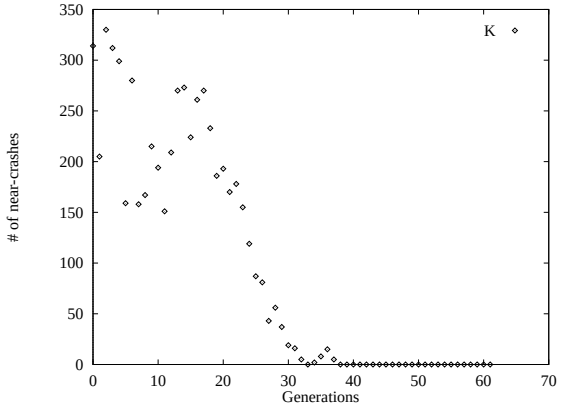
Fig. 9. This graph shows how the number of near-collisions (K), number of missed-exits (B), and distance traveled (Z) in a population of learning SAPIENT vehicles varies with successive generations. In these tests, the population size was set to 40, and each vehicle was evaluated on four scenarios. The graphs show the *accumulated* statistics for all of the vehicles in the given generation, over all four scenarios. Note that both K and Z decrease to zero, while Z, the incremental reward, rises.



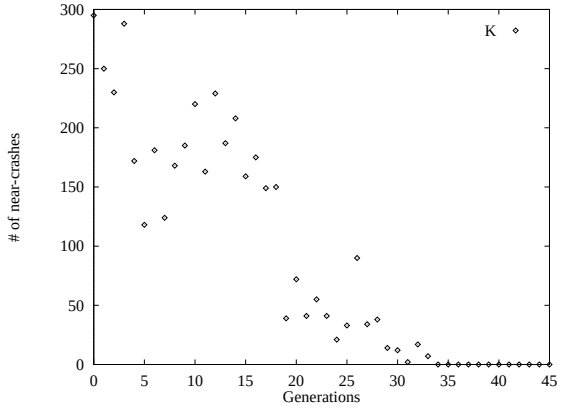
$E = \ldots\ -(10000 \times \text{num-crashes})\ldots$



$E = \ldots\ -(5000 \times \text{if-wrong-exit})\ldots$



$E = \ldots\ +(10 \times \text{dist-traveled})$



$E = \ldots\ -(0.2 \times \text{speed-deviation})\ldots$

Fig. 10. This graph shows that the SAPIENT parameters learned by PBIL converge to competent vehicles despite variations in the evaluation function used. Each of these graphs shows the total number of near-collisions (K), in a population of 40 cars, evaluated on four scenarios. In each experiment, a coefficient weighting one observable was multiplied by 10. The changes to the evaluation function are shown above.

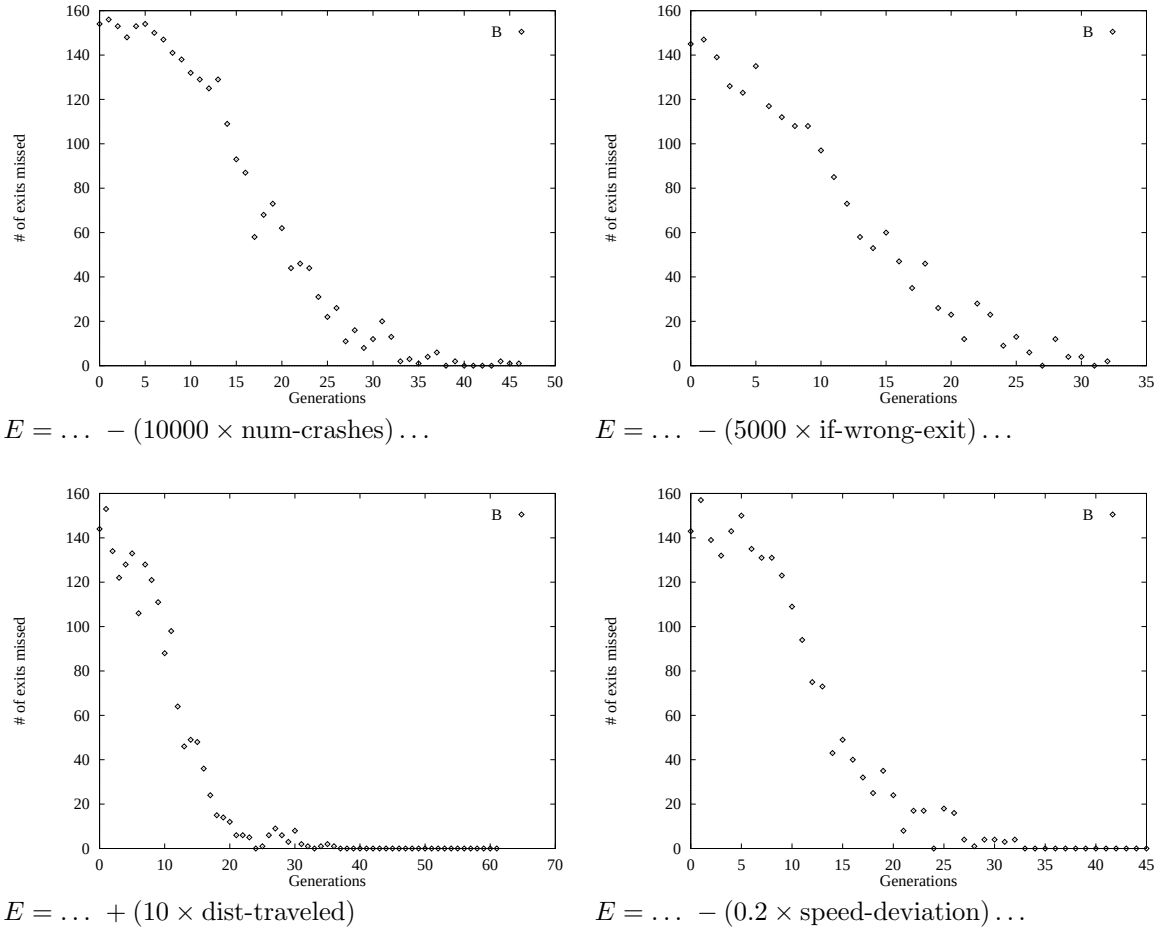


Fig. 11. This graph shows that the SAPIENT parameters learned by PBIL converge to competent vehicles despite variations in the evaluation function used. Each of these graphs shows the total number of missed exits (B), in a population of 40 cars, evaluated on four scenarios. In each experiment, a coefficient weighting one observable was multiplied by 10. The changes to the evaluation function are shown above.

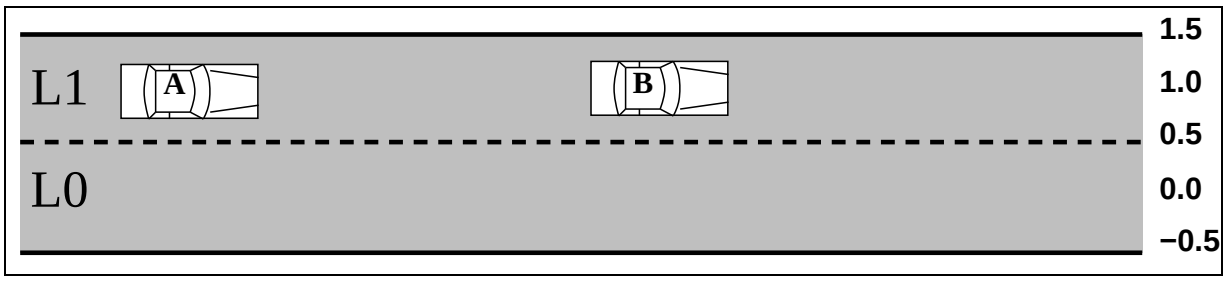


Fig. 12. This scenario tests if the tactical reasoning system can overtake a slower-moving vehicle.

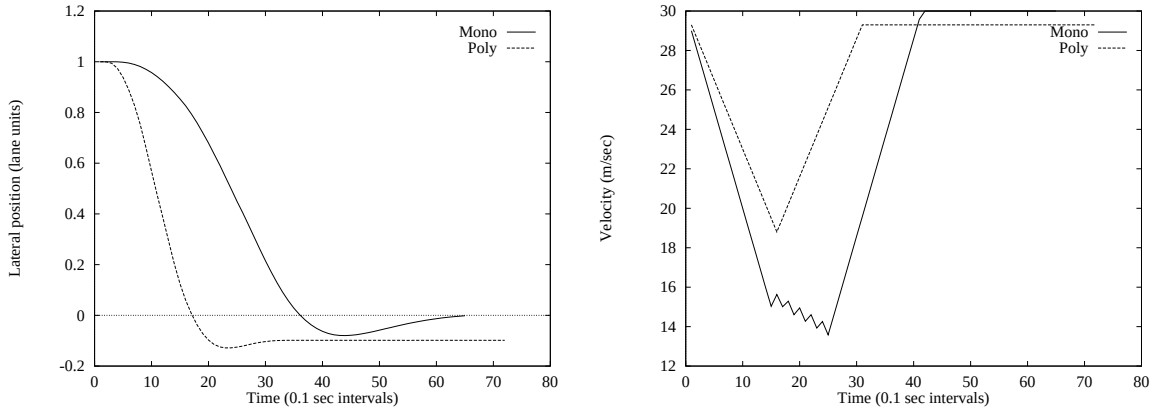


Fig. 13. Lateral displacement (left) and velocity (right) as a function of time, for rule-based (denoted as *Mono*) and SAPIENT (denoted as *Poly*) vehicles on the overtaking scenario (See Figure 12). See the text for a discussion of these graphs.

generated competent vehicles. For example, increasing the penalty of a collision from -1000 to -1000000 does not affect vehicles since they learn how to avoid all collisions. By contrast, radically increasing the penalty for speed deviations in a similar manner leads to vehicles that are willing to collide with others in a desperate effort to avoid the penalties incurred in dropping below the target velocity.

7. Scenario-Based Evaluation of Tactical Driving

Scenarios are widely used in driving research to evaluate the performance of human subjects [12, 14]. Similar techniques are also used to measure situation awareness in other domains [6, 5, 19]. Here, we use micro-scenarios to examine the performance of SAPIENT's reasoning agents in situations where tactical-level decisions are required. A more comprehensive discussion of these scenarios is available in [20].

In each of the following scenarios, we focus on the vehicle marked A in the respective diagrams. SAPIENT's performance is compared to the behavior of the default rule-based vehicle. In the accompanying graphs, the monolithic, hand-coded, rule-based vehicle is denoted as *Mono*, while the multi-agent, adaptive, SAPIENT system is marked *Poly*. It should be emphasized that the SAPIENT vehicles have not been exposed to any tactical scenarios — they were trained (using PBIL) exclusively on obstacle courses in the cyclotron environment.

The first scenario (See Figure 12) involves a simple overtaking maneuver, and is a common occurrence on the highway. Initially, both vehicles are moving at normal highway speeds, but the lead vehicle begins braking (as it approaches its exit, for example). There is no other traffic, so Car A should safely overtake. As seen in the lateral displacement and velocity profiles (See Figure 13), both types of cognition module are able to solve

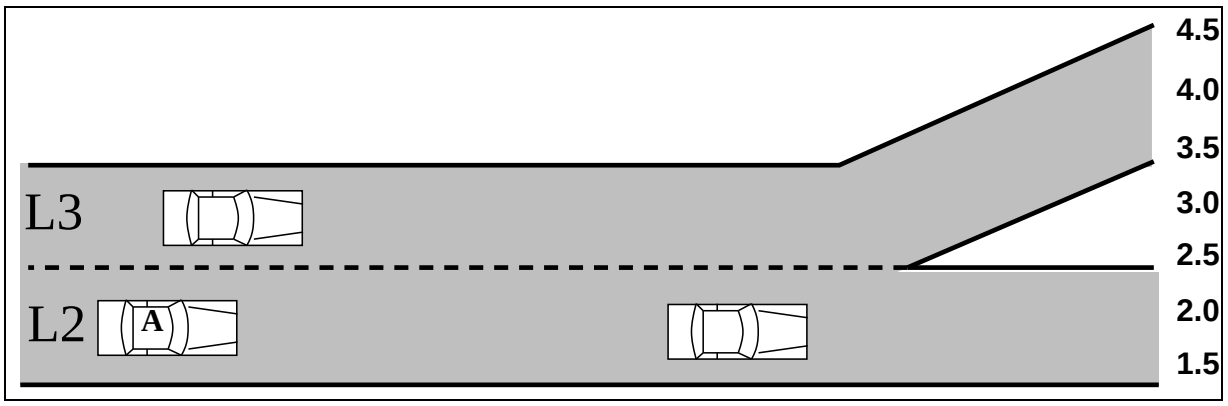


Fig. 14. Exit scenarios add complexity to the tactical driving domain because they introduce additional strategic-level goals. The conflicts between two strategic-level goals leads to interesting tactical behavior.

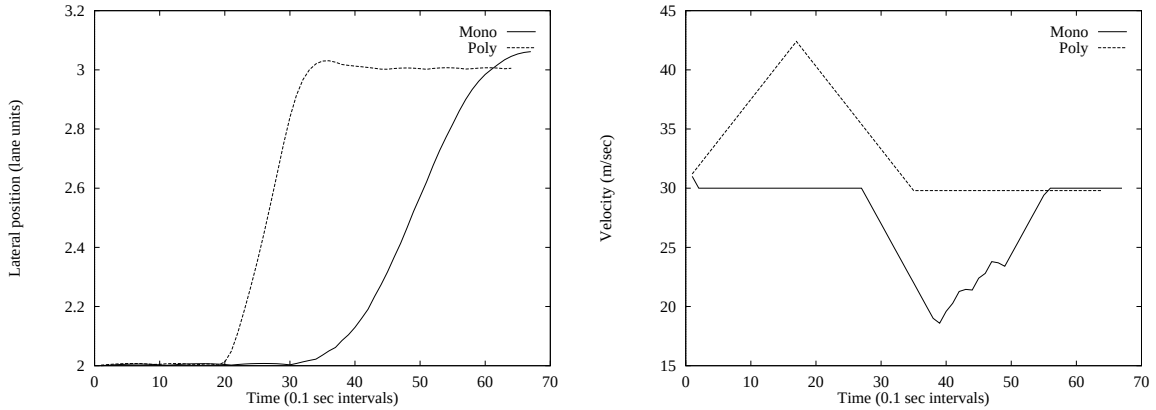


Fig. 15. Lateral displacement (left) and velocity (right) as a function of time, for rule-based and SAPIENT vehicles on the exit scenario (See Figure 14). See the text for a discussion of these graphs.

this scenario successfully. However, note that the SAPIENT vehicle is more aggressive, maintaining a smaller headway during the maneuver than the hand-tuned, rule-based vehicle. This is because the SAPIENT reasoning agent responsible for car following has tuned its generalized potential fields relative to the other vehicle based on a time-to-impact metric, as opposed to using a constant headway. The other notable feature is the oscillation in the rule-based vehicle's velocity profile. This is caused by a combination of two factors: a discrete velocity controller and brittle car-following rules. Note that the SAPIENT vehicle is not perfectly centered in the passing lane during the overtaking maneuver. This is because the potential field surrounding the obstacle votes for additional space, and since there is sufficient space in the target lane, the SAPIENT vehicle is able to drive off-center. This behavior can also be observed in the other scenarios.

The second scenario (See Figure 14) introduces a second (possibly conflicting) strategic goal: taking an exit; also, ambient traffic is introduced. Vehicle A must now change lanes to make its desired exit without colliding with other cars. Figure 15 shows an interesting difference in driving behavior. The rule-based car slows down until it can find a gap in the exit lane, and then changes lanes. In contrast, SAPIENT speeds up to overtake the vehicle in the exit lane. This maneuver allows it to maintain its desired speed while making the exit.

The final scenario, shown in Figure 16, is identical to the one discussed in the Introduction. Recall that Car A may take its desired exit by either staying behind the slow blocker, or by passing. Unlike the situation shown in Figure 14, chang-

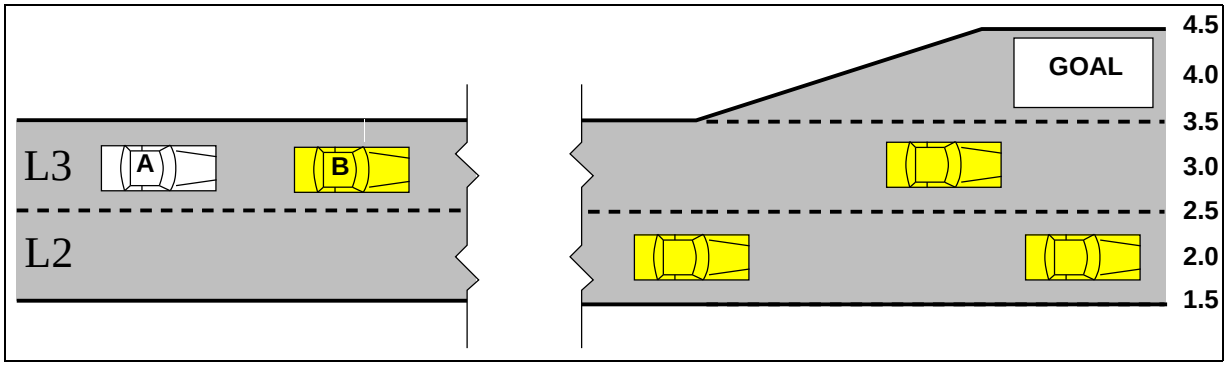


Fig. 16. This exit scenario is difficult because the lane changes are optional. To address the strategic-level goal of maintaining speed, the intelligent vehicle must decide whether or not to attempt the overtaking maneuver at the risk of missing the desired exit.

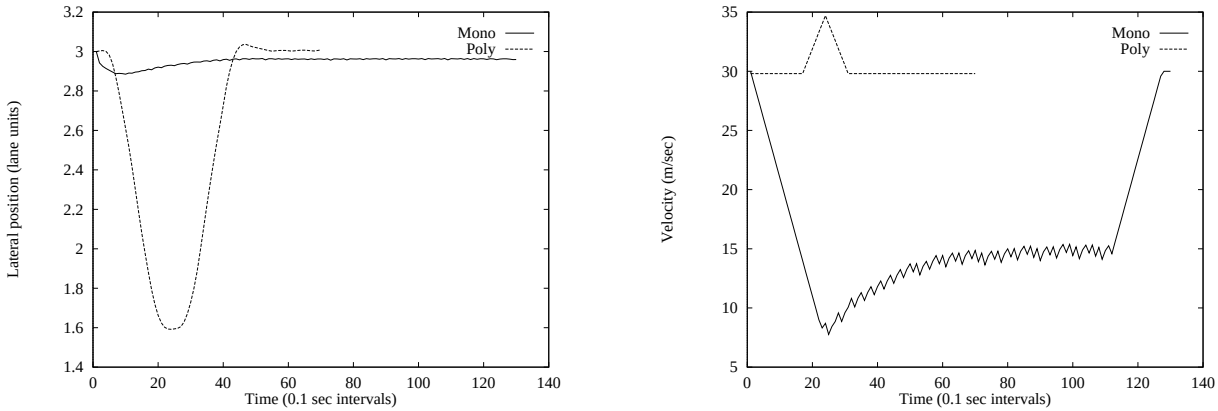


Fig. 17. Lateral displacement (left) and velocity (right) as a function of time, for rule-based and SAPIENT vehicles on the more difficult exit scenario (See Figure 16).

ing lanes is not mandatory; in fact, should Car A decide to pass, it will have to complete two lane changes before exiting. Once again, the two different vehicle types choose differently. The rule-based vehicle opts to stay in its lane, based solely on a rule which depends on the distance to the exit. On the other hand, the SAPIENT vehicle chooses to overtake the blocker.

7.1. Heterogeneous Traffic Experiments

In the final set of experiments, vehicles were injected into an initially empty cyclotron track from the on-ramp at regular intervals, τ. Each vehicle was given two strategic-level goals: 1) make exactly one circuit of the track before exiting; 2) maintain the speed at which it was injected whenever possible. The aim of the experiment was to see how the two tactical driving systems, rule-based, and SAPIENT, would behave as the roadway became more congested. Three sets of experiments with different traffic configurations were performed: all-rule-based cars, all-SAPIENT cars, and a uniform mix of rule-based and SAPIENT cars.

As expected, the number of vehicles on the roadway increased until the rate of vehicles entering the track was equal to the rate of vehicles leaving (either because the vehicles had successfully completed their circuit, or because the vehicles were unable to merge into the traffic stream). At low rates of traffic flow (e.g., $\tau > 6$ seconds), all of the three traffic configurations safely negotiated the scenario (with no missed exits). However, once the traffic flow was increased, the behavior of the three traffic configurations diverged.

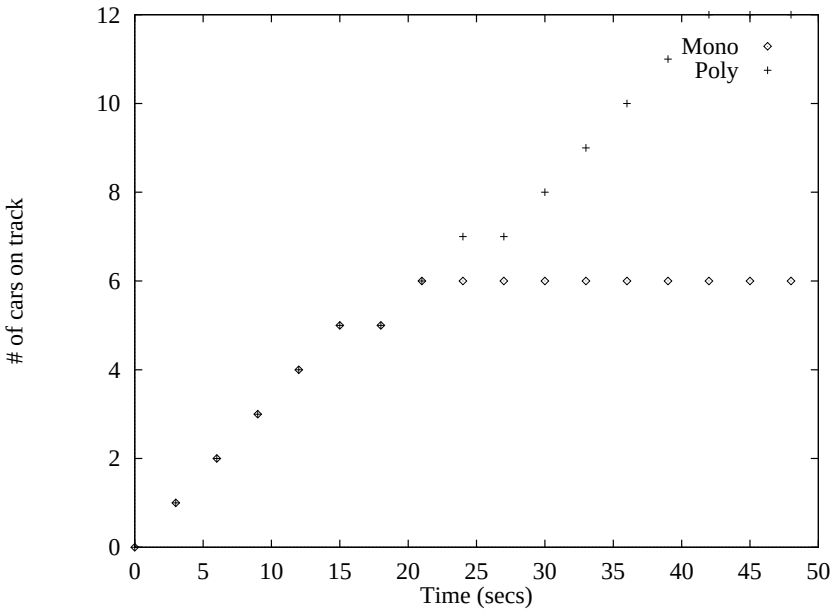


Fig. 18. This graph shows how the number of vehicles on the cyclotron varies as a function of time in heavy traffic (traffic injection rate $\tau = 3$ secs between cars). Note that only six rule-based vehicles were able to merge onto the cyclotron loop; by contrast, all of the SAPIENT vehicles were able to merge and complete the scenario.

The graph in Figure 18 shows how the number of vehicles varies as a function of time when $\tau = 3$ seconds for the all-rule-based and all-SAPIENT cases. Even in heavy traffic, neither of the pure traffic types have any collisions. Although both types of vehicles perform well initially (when the roadway is clear), once the number of vehicles on the track increases to about 6, the conservative rule-based drivers are unable to merge into the traffic stream (since they require a guaranteed headway of two seconds on both sides of the gap). Thus they are unable to change lanes, and exit the scenario prematurely. To make matters worse, the rule-based vehicles that were already on the roadway become trapped in the inner loops of the cyclotron (due to the high rate of traffic in the entry/exit lane).

The all-SAPIENT traffic, on the other hand, is able to drive successfully. This can probably be attributed to two factors: 1) the aggressive driving style, relying on time-to-impact reasoning agents, is willing to merge into smaller gaps; 2) the distributed reasoning system is better at making tradeoffs — the negative votes for merging into a potentially unsafe gap are tolerated since the alternative (missing the exit) is seen to be worse. The brittle decision tree used in the rule-based cars, on the other hand, rejects these gaps outright.

Interleaving rule-based and SAPIENT cars in the heavy traffic scenario leads to a stable heterogenous behavior with no collisions. While the more aggressive SAPIENT vehicles still miss fewer exits, even the rule-based vehicles perform better than they did in the pure-rule-based case because of the reduced congestion. This may have positive implications for the deployment of automated vehicles in mixed traffic conditions.

8. Conclusion and Future Directions

Our experiments have demonstrated: 1) The potential for intelligent behavior in the tactical driving domain using a set of distributed reasoning agents; 2) The ability of evolutionary algorithms to automatically configure a *collection* of these modules for addressing their *combined* task. While the evaluation sections compared SAPIENT's performance with a rule-based vehicle, the results should not be taken out of context: clearly it is possible to encode SAPIENT's current knowledge in the form of rules to create a more competent rule-based vehicle. The difference is that

creating a monolithic rule-based vehicle is a much more difficult task due to the interactions between large number of rules, the manual tuning of parameters within the rules, and the complex interactions between the rules.

In this study, we used a simple evaluation function. By introducing alternative objective functions, we plan to extend this study in at least two directions. First, for automated highways, we would like the cars to exhibit altruistic behavior. In a collection of PBIL vehicles, optimizing a *shared* evaluation function (such as highway throughput) may encourage cooperation. Second, we are developing reasoning agents to address additional complications which will arise when these vehicles are deployed in the real world, such as complex vehicle dynamics and noisy sensors.

Our system, which employs multiple automatically trained agents, can competently drive a vehicle, both in terms of the user-defined evaluation metric, and as measured by their behavior on several driving situations culled from real-life experience. In this article, we described a method for multiple agent integration which is applied to the automated highway system domain. However, it also generalizes to many complex robotics tasks where multiple interacting modules must simultaneously be configured without individual module feedback.

9. Acknowledgments

The authors would like to acknowledge the valuable discussions with Dean Pomerleau and Chuck Thorpe which helped to shape this work. Thanks also to Gita Sukthankar for the data processing scripts and graphs. This research was partially supported by the Automated Highway System project, under agreement DTFH61-94-X-00001, and was started while Shumeet Baluja was supported by a graduate student fellowship from NASA, administered by the Lyndon B. Johnson Space Center. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the AHS Consortium or NASA.

Notes

1. More information and an interactive demo are available at: `<http://www.cs.cmu.edu/rahuls/shiva.html>`

References

1. S. Baluja. Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Technical Report CMU-CS-94-163, Carnegie Mellon University, 1994.
2. S. Baluja and R. Caruana. Removing the genetics from the standard genetic algorithm. In *Proceedings of the International Conference on Machine Learning (ML-95)*, 1995.
3. J. Cremer, J. Kearney, Y. Papelis, and R. Romano. The software architecture for scenario control in the Iowa driving simulator. In *Proceedings of the 4th Computer Generated Forces and Behavioral Representation*, 1994.
4. E. Dickmanns and A. Zapp. A curvature-based scheme for improving road vehicle guidance by computer vision. In *Proceedings of the SPIE Conference on Mobile Robots*, 1986.
5. M. Endsley. Towards a theory of situation awareness. Technical report, Texas Technical University, Department of Industrial Engineering, 1993.
6. M. Fracker and S. Davis. Explicit, implicit, and subjective rating measures of situation awareness in a monitoring task. Technical report, Wright-Patterson Air Force Base, 1991.
7. K. Gardels. Automatic car controls for electronic highways. Technical Report GMR-276, General Motors Research Labs, June 1960.
8. D. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, Reading, MA, 1989.
9. B. Krogh. A generalized potential field approach to obstacle avoidance control. In *Proceedings of Robotics Research: The Next Five Years and Beyond*, 1984.
10. B. Krogh and C. Thorpe. Integrated path planning and dynamic steering control for autonomous vehicles. In *Proceedings of IEEE Conference on Robotics and Automation*, 1986.
11. I. Masaki, editor. *Vision-Based Vehicle Guidance.* Springer-Verlag, 1992.
12. J. McKnight and B. Adams. Driver education and task analysis volume 1: Task descriptions. Technical report, Department of Transportation, National Highway Safety Bureau, November 1970.
13. J. Michon. A critical view of driver behavior models: What do we know, what should we do? In L. Evans and R. Schwing, editors, *Human Behavior and Traffic Safety.* Plenum, 1985.
14. National Safety Council. Coaching the experienced driver II, 1995. Defensive driving course training manuals.
15. D. Pomerleau. *Neural Network Perception for Mobile Robot Guidance.* PhD thesis, Carnegie Mellon University, February 1992.

16. A. Ram, R. Arkin, G. Boone, and M. Pearce. Using genetic algorithms to learn reactive control parameters for autonomous robotic navigation. *Adaptive Behavior*, 2(3), 1994.
17. D. Reece. *Selective Perception for Robot Driving*. PhD thesis, Carnegie Mellon University, May 1992.
18. J. Rillings and R. Betsold. Advanced driver information systems. *IEEE Transactions on Vehicular Technology*, 40(1), 1991.
19. N. Sarter and D. Woods. How in the world did we ever get into that mode? mode error and awareness in supervisory control. *Human Factors*, 37(1), 1995.
20. R. Sukthankar. *Situation Awareness for Tactical Driving*. PhD thesis, Carnegie Mellon University, January 1997. Also available as CMU Tech Report CMU-RI-TR-97-08.
21. R. Sukthankar, J. Hancock, D. Pomerleau, and C. Thorpe. A simulation and design system for tactical driving algorithms. In *Proceedings of AI, Simulation and Planning in High Autonomy Systems*, 1996.
22. R. Sukthankar, D. Pomerleau, and C. Thorpe. SHIVA: Simulated highways for intelligent vehicle algorithms. In *Proceedings of IEEE Intelligent Vehicles*, 1995.
23. C. Thorpe, M. Hebert, T. Kanade, and S. Shafer. Vision and navigation for the Carnegie Mellon Navlab. *IEEE Transactions on PAMI*, 10(3), 1988.
24. R. von Tomkewitsch. Dynamic route guidance and interactive transport management with ALI-Scout. *IEEE Transactions on Vehicular Technology*, 40(1), 1991.

Rahul Sukthankar is currently a Research Scientist at Justsystem Pittsburgh Research Center, and an adjunct faculty member at The Robotics Institute, Carnegie Mellon University. He completed his Ph.D. in Robotics from Carnegie Mellon in 1996, and received a B.S.E. *summa cum laude* in Computer Science from Princeton in 1991.

His research interests include intelligent vehicles, simulation, agents, and applications of machine learning.

Shumeet Baluja was born in New Delhi, India, on February 17, 1971. He received a B.S. degree in computer science from the University of Virginia, Charlottesville, Virginia, in 1992, and completed his Ph.D. in computer science from Carnegie Mellon University, Pittsburgh, Pennsylvania, in 1996. He is currently a research scientist at Justsystem Pittsburgh Research Center, and an adjunct faculty member in the Computer Science Department and The Robotics Institute at Carnegie Mellon University.

His research interests include the integration of machine learning and computer vision, mechanisms for selective attention in vision, artificial neural networks and their applications, and high-dimensional optimization.

John Hancock is a Ph.D. student in Robotics at the Robotics Institute, Carnegie Mellon University. He earned his B.S. degree in 1993 in Electrical Engineering from Harvard University. His thesis work is on obstacle detection for automated highway applications using laser intensity and a predictive stereo system. Other work at CMU includes improvements to ELVIS, an eigenvector-based-learning road-following system, and the development of a highway simulator, SHIVA, for testing tactical driving algorithms.