J Intell Robot Syst (2007) 49:39–65
DOI 10.1007/s10846-007-9138-9

Foraging Theory for Autonomous Vehicle Decision-making System Design

Burton W. Andrews · Kevin M. Passino · Thomas A. Waite

Received: 2 October 2006 / Accepted: 29 December 2006 /
Published online: 6 March 2007

Abstract Foraging theory is typically used to model animal decision making. We describe an agent such as an autonomous vehicle or software module as a forager searching for tasks. The prey model is used to predict which types of tasks an agent should choose to maximize its rate of reward, and the patch model is used to predict when an agent should leave a patch of tasks and how to choose within-patch search patterns. We expand and apply these concepts to fit an autonomous vehicle control problem and to provide insight into how to make high-level control decisions. We also discuss extensions of the basic models, showing how a risk-sensitive version can be used to alter policies when time or fuel is limited. Throughout the applications, we examine ways an agent can estimate environmental parameters when such parameters are not known.

Key words agent · autonomous vehicle · biomimicry · control · foraging

Category (5) Intelligent Systems · Intelligent Control

B. W. Andrews (✉) · K. M. Passino
Department Electrical and Computer Engineering, The Ohio State University,
Columbus, OH 43210, USA
e-mail: andrews.191@osu.edu

K. M. Passino
e-mail: passino.1@osu.edu

T. A. Waite
Department Evolution, Ecology, and Organismal Biology, The Ohio State University,
Columbus, OH 43210, USA
e-mail: waite.1@osu.edu

1 Introduction

Foraging theory is used in the field of behavioral ecology to model animal decision making [1–3]. Evolutionary considerations often lead to optimization models of adaptive behavior, which is behavior shaped by natural selection. A wide range of characteristics have been considered including factors affecting animal survival, such as energetic shortfall and predation, and effects of constraints due to animal physiology and foraging environment. Some models have been tested experimentally while others have provided general insights and unifying principles. Here, we develop an analogy where we view an agent such as a vehicle, processor, or software module as an animal and its domain of operation as a foraging environment. We explain how to view tasks and objectives of agents as foraging activities and objectives. Then, using a "biomimicry" philosophy [4], we apply foraging theory to the development of decision-making strategies for agents. Our primary goal is to illustrate the potential significance of merging the fields of engineering and behavioral ecology and to introduce the use of simple, existing models from behavioral ecology for engineering design. In doing so, we hope to gain better insights into how to design decision-making strategies that are optimized for a given set of objectives and domain of operation, and yet have "robustness" properties like those often seen in nature.

1.1 Biomimicry of Foraging

From a biological perspective, the environment is comprised of a forager and prey or food items. Each forager's primary "goal" is to obtain energy, and the only way in which to do so is by attacking and consuming prey. The forager must make decisions about how to interact with its environment to maximize some correlate of Darwinian fitness. If there are different types of prey, which types should be attacked? Why not specialize on particular types to avoid wasting time on substandard prey? On the other hand, why not generalize and take advantage of all opportunities? How much time should be spent in a particular patch of prey or nutrients? If the next patch is far away, should more time be spent in the current patch despite the depletion of rewards within the patch? These optimal diet problems are studied using the so-called prey and patch models from foraging theory.

It is easy to view the foraging problem in terms of engineering applications. In robotics and manufacturing, a robot on a factory floor that must travel to various locations and perform a series of tasks can be thought of as a forager with the tasks being thought of as prey [5]. If the robot must handle buffers (queues) with tasks that arrive at various times, the buffers can be thought of as patches. The robot must decide how long to process each buffer. In military missions, autonomous vehicles can be thought of as foragers that must search for and engage targets or patches of targets [6]. Software agents that are distributed throughout the internet can be thought of as foragers searching for computers on which to perform different tasks. Another example is the controller of a distributed dynamical system [7]. The controller can be viewed as searching for regions of error, which, when engaged, exhibit characteristics of reward depletion. The theory presented in the rest of the article is kept general enough so that the relevance to applications such as these is clear; however, our application of interest is that of autonomous vehicles performing a search operation.

1.2 Overview and Relations to Literature

This paper discusses many of the key ideas related to the prey and patch models in classical foraging theory [1] and their applicability to agent design in engineering. We also illustrate the value of risk-sensitive versions of the models which incorporate environmental stochasticity [8–10]. Our applications of the theory focus on the problem of autonomous vehicle control. While the autonomous vehicle literature is vast, and many issues loosely related to our focus are considered, the work most relevant to the problems addressed here is in [11–13]. These works use a general framework of stochastic search and focus on establishing metrics for vehicle success; however, higher level control problems such as target type choice are not addressed.

We are able to make several novel contributions with respect to the problems not considered in the literature via our foraging-theoretic approach. We first present the prey model theory and then apply the theory to the autonomous vehicle problem. Our results illustrate what price to pay for recognition of targets and how to choose target types to attack. We then present the patch model theory followed by an application to autonomous vehicle control for a generic search problem of the type studied in [13]. Unlike in [13], we are able to show how a vehicle can determine the optimal time to leave a patch of objects given that the vehicle has knowledge of its environment. If environmental information is unknown, on-line estimation techniques can be used. Finally, we use risk-sensitive foraging theory to show how a vehicle should alter its decisions in the face of limited fuel, another problem not considered in [11–13]. Moreover we provide a novel approach that has not appeared in the literature to estimating gain functions in an uncertain environment so that a risk-sensitive version of the patch model may be used in time-limited situations. In all cases when we apply the theory, we highlight cases where underlying assumptions are not met and how this affects the prediction accuracy for the models and the design of decision-making strategies. We emphasize that the high-level control problems we consider have not been previously studied, and other known optimization techniques are not applicable. The only connection to the literature lies in the stochastic search framework we consider in our examples. Because of this it is difficult to compare with other approaches to the problems we address; however, we do consider several intuitive, alternative techniques when we test the theory in simulation, and comparison with these techniques provides insight into the usefulness of the theory in particular situations.

2 Prey Model: Search or Process?

The prey model describes an agent searching for tasks of different types in a particular environment. Each task holds a certain point value corresponding to the increment in the success level of an agent if it successfully processes (i.e., executes) the task. Processing is the equivalent of a biological forager handling prey. Point values quantify reward. The job of the agent is to search for tasks, recognize a task once it is encountered, and then decide whether or not to process the task. Both recognition and detection depend on sensing capabilities. For autonomous vehicles, these capabilities are usually defined by the technology of the sensing equipment, such as infrared, vision, doppler radar, or bistatic radar, and by a sensing footprint

which is the field of view of the sensing equipment. The prey model, in its original form, assumes that a task is recognized correctly and that no time is required for recognition. To summarize, the prey model holds that agents search for tasks, correctly recognize a task instantaneously upon encounter, and then decide whether to process the task or to pass over it and continue to search. The derivation of the basic prey model is given next and directly follows [1].

2.1 Prey Model Theory

Let there be n types of tasks described by: e_i, the expected time required to process a task of type i, v_i, the expected number of points obtained from processing a task of type i, λ_i, the rate of encounter with tasks of type i (assumed to occur via a Poisson process), and p_i, the probability of processing a task of type i if it is found and recognized. Probability of processing is the decision of the agent. The net rate of point gain of the agent is

$$J = \frac{\sum_{i=1}^{n} p_i \lambda_i v_i}{1 + \sum_{i=1}^{n} p_i \lambda_i e_i} - s \tag{1}$$

where s is the point cost of search per unit time. The goal is to find the p_i that maximizes J. It is easy to show that this is either $p_i = 1$ or $p_i = 0$ (the *zero-one rule*). In other words, to maximize its rate of point gain, an agent must either process a task of type i every time it encounters it or never process it at all. The question then is which tasks the agent should process and which tasks it should ignore.

The optimal policy for an agent is described by the prey model algorithm. First, define the rate of point gain that results from processing task type i (v_i/e_i) as the profitability of the task, and rank the tasks in the environment according to their profitability such that $v_1/e_1 > v_2/e_2 > \cdots > v_n/e_n$. If task type j is included in the agent's "task pool," those task types that the agent will process once encountered, then all task types with profitabilities greater than that of type j will be included in the task pool as well. Thus, the prey algorithm states that, after ranking the task types by profitability, include types in the task pool starting with the most profitable type until

$$\frac{\sum_{i=1}^{j} \lambda_i v_i}{1 + \sum_{i=1}^{j} \lambda_i e_i} > \frac{v_{j+1}}{e_{j+1}}.$$

The highest j that satisfies this equation is the least profitable task type in the task pool. In other words, if task types in the environment are ranked according to profitability with $i = 1$ being the most profitable, and if type $j+1$ is the most profitable type such that the agent will benefit more from searching for and processing types with profitability higher than that of $j+1$, then tasks of types 1 through j should be processed when encountered and all other tasks should not. If the equation does not hold for any j, then all task types should be processed when encountered.

The exclusion of type $j+1$ does not depend on the rate of encounter with type $j+1$. This exclusion implies that if the expected *missed opportunity* gains exceed the immediate gains of processing a particular type, then it does not benefit the agent to process the type, no matter how often the agent comes across it. Equivalently, if a type's rate of encounter exceeds a critical threshold, then less profitable types should be ignored regardless of how common they are.

2.2 Assumptions

Here, we comment on the practical use of the model and its assumptions. The model assumes that the agent has *complete information*, that is, it knows all of the parameters above. The agent is not omniscient, knowing the outcome of every situation, but it does know the parameters and rules of the model. Addressing the reasonableness of this assumption, both the expected processing time e_i and the expected points obtained v_i are parameters that are known or can be estimated by an agent in many engineering applications. On the other hand, knowledge of the rate of encounter with tasks λ_i often may not be a realistic assumption. In many situations, λ_i can be estimated or calculated. For example, a vehicle with sensor width W searching systematically for tasks can calculate the average rate of encounter with tasks as $\lambda = N_t W u/A$, where N_t is the number of tasks in a domain of area A and u is the vehicle's velocity. While this information may not hold for all situations or environments, the classical prey model makes the assumption that λ_i is known or can be calculated.

An additional assumption of the prey model is that the agent has infinite life, for example, infinite fuel or ammunition for an autonomous vehicle. Also, since the rate of encounter is constant, an infinite number of tasks are assumed to exist. This idea might imply the ability of tasks to arrive within the environment, an infinite number of spatial task arrangements, or an infinite number of ways that the agent can move through the environment.

2.3 Changed Sensing Constraints

The detection and recognition of tasks is entirely dependent upon the sensing capabilities of the agent, and the prey model assumes that tasks are immediately recognized with perfect accuracy. Extensions of the prey model may be used to address situations where this assumption does not hold. For instance, alterations of the model that incorporate nonzero recognition time, imperfect resemblance, and the value of recognition are discussed in [1].

We present an example of these extensions here by examining the value that an agent might place on the ability to recognize a task immediately upon encounter. For instance, suppose the $n = 2$ case is altered so that the agent cannot tell the types apart unless it pays a recognition cost of r_v points and r_t seconds. Assume that if $r_v = r_t = 0$, the agent would specialize on type 1 tasks. It is known [1] that the agent is willing to pay for recognition for the set of (r_v, r_t) pairs that satisfies

$$k > \alpha r_v + \beta r_t, \tag{2}$$

where

$$k = \frac{\lambda_2}{\lambda_1 + \lambda_2}[\lambda_1 v_1 e_2 - v_2(1 + \lambda_1 e_1)],$$
$$\alpha = 1 + \lambda_1 e_1 + \lambda_2 e_2, \text{ and}$$
$$\beta = \lambda_1 v_1 + \lambda_2 v_2.$$

The constant k sets an upper limit on the price the agent is willing to pay for recognition. Examining k, the agent is not willing to pay as much as v_2/e_2 approaches $\lambda_1 v_1/(1 + \lambda_1 e_1)$, the expected rate from specialization on type 1s. This result makes sense because as v_2/e_2 approaches this rate, the immediate gains resulting from processing a task of type 2 do not differ as much from the expected missed opportunity gains. This effect causes the agent to be indifferent, in which case recognition is not useful.

3 Prey Model Application to Autonomous Vehicles

The prey algorithm could be applied to an autonomous mobile robot that moves about a manufacturing facility and processes spatially distributed tasks. In that application, a task could be to assemble a part or take some other action on a product (e.g. drilling, polishing, painting). Here, we consider an autonomous vehicle in a military application, but with slight modifications it can be thought of as an underwater or space exploration application. In particular, we envision a scenario of search similar to one considered in [11, 12] (these works are similar only in the stochastic nature of search – they do not consider the task-processing decisions we do here), and we implement the scenario in simulation. A single autonomous vehicle searches for objects in a domain of operation that is 10 by 10 km. Objects may be, for example, targets, threats, decoys, etc. The vehicle must perform one or more of three activities on an encountered object: classify an object once found, attack the object, and then verify that the attack was successful and the object was destroyed. A task then is considered to be a specific activity or activities performed on a given object. In the application that follows, we define the processing of a single task as the combined activities of classification, attack, and verification of an object. Since the processing of a task is with respect to a particular object, we describe the vehicle as searching for objects and deciding whether to engage the object. In a manufacturing or exploration problem the control difference is how tasks and their processing are defined.

The objects are considered to be individual items, that is, they have no relation to each other and the vehicle can only engage one object at a time. The vehicle is a flying autonomous vehicle with the ability to move on top of objects in the domain. Searching is implemented in a fixed pattern, where the vehicle begins near the origin of a two-dimensional coordinate plane, proceeds in the vertical x_2 direction, reaches the edge of the domain ($x_2 = 10$ km), turns around and searches in the $-x_2$ direction but shifted by 500 m in the x_1 direction. We refer to this search path as a "lawnmower pattern". Sensing capabilities are defined by a sensing footprint that is a 500 by 500 m square 800 m in front of the vehicle, within which perfect sensing is assumed. The vehicle travels at a constant velocity of 120 m/s and has a minimum turning radius of 800 m, a reasonable value for autonomous air vehicles. The controller of the vehicle

determines the shortest path that completes the required activities once an object is found.

To illustrate the theoretical prey model concepts, consider the $n = 2$ case with an average of 50 type 1 objects and 80 type 2 objects. This information allows us to calculate the average rates of encounter as $\lambda_1 = (50Wu)/|A| = 0.030$ objects/s for type 1 and $\lambda_2 = (80Wu)/|A| = 0.048$ objects/s for type 2, where $|A| = 10000^2$, $W = 500$ m, and $u = 120$ m/s. Object values are chosen as $v_1 = 40$ and $v_2 = 10$, and engagement times are $e_1 = e_2 = 180$ s (180 s is the average time required to perform the three activities mentioned above and is the same for each object type). These parameter values cause the vehicle to exclude type 2 objects from its task pool as shown below.

The vehicle is assumed to have knowledge of λ_i, e_i, and v_i a priori. Ranking the objects according to profitability results in $v_1/e_1 = 40/180 > v_2/e_2 = 10/180$. According to the algorithm, the decision of whether to engage type 2 objects is based on the expected rate of point gain that results from alternatively choosing to search for and engage type 1 objects only. This expected rate is $(\lambda_1 v_1)/(1 + \lambda_1 e_1) = 0.188$ points/s. Therefore, because $\frac{\lambda_1 v_1}{1+\lambda_1 e_1} > \frac{v_2}{e_2}$, the vehicle would benefit more from continuing on and searching for type 1 objects when a type 2 object is encountered than it would from engaging the type 2 object.

3.1 Heuristic Implementation of the Prey Model

Now that we have illustrated the potential for implementing the prey model algorithm, our main interest is in studying where the algorithm may fail in real applications. First, consider the case where five type 1 and 150 type 2 objects exist in the domain and v_1 and v_2 are set to 40 and 10, respectively. The prey model algorithm predicts that type 2 objects should not be engaged when encountered. (This prediction may initially be counterintuitive due to the large number of type 2 objects; however, the choice of engaging type 1 objects is independent of the rate of encounter with type 2 objects.)

Simulating the application described above results in an average rate of point gain of 0.0326 points/s (calculated as the number of points obtained divided by the time taken to gain them) when using the algorithm's specialization prediction and a higher rate of 0.0521 points/s when engaging all types. The prey model algorithm fails to yield the optimal rate, and this is due to an inaccuracy in the vehicle model. When the vehicle searches according to the lawnmower pattern, the minimum turn radius constraint requires the vehicle to swing outside of the domain at the end of each "search strip" to begin search of the next strip. This extra movement increases the search time and decreases the overall rate of point gain. In the above case, the extra search time is enough to degrade the performance of specialization from the theoretical expected rate of point gain of 0.0779 to 0.0326 points/s. Generalization, however, only decreases from the theoretical rate of 0.0575 to 0.0521 points/s. The purpose of this example is to highlight the effect of physical constraints in real applications that potentially affect the assumptions and predictions of the model. Of course, in this case, if the extra search time is known a priori, it may be included in the model either by adjusting the rates of encounter or by including a search cost.

Another problem that arises in real applications is due to the fact that the prey model theory deals solely with averages. It uses average rates of encounter to predict

the strategy that maximizes the average rate of point gain. In real applications, deviation of an encounter rate from the average may result in suboptimal predictions of the prey model for individual missions. Such a deviation may result from the finite domain the vehicle is operating in. The model assumes infinite time, implying an infinite domain of operation which is clearly not realizable. For a given average encounter rate with a task, a finite domain cuts short the opportunity to achieve the average. In the cases we consider above, though, we assume an average number of objects in a finite domain and calculate the average rate of encounter using these values. Thus, if the number of objects encountered is the actual average, the actual rate of encounter with objects will be the average rate despite the finite domain. However, if the realization of the random number of objects is not the average, the rate of encounter will diverge from the average, potentially affecting the model predictions.

We have discussed several problems with the application of the prey model to real situations: lack of knowledge of encounter rates, physical constraints not taken into account in the model, and potential for poor performance during deviations from the average. One way to address these concerns is to use heuristic methods that exploit vehicle memory. For example, consider a vehicle that is able to keep track of the number of encounters it has had with a particular object type i, and hence has the ability to update its estimate $\hat{\lambda}_i$ of the rate of encounter for that type. The prey model algorithm can then be implemented at each time instant using the current estimates of the rates of encounters to decide whether to engage a particular object type upon an encounter [1]. Consider, for example, a situation with many low-valued type 2 objects and very few high-valued type 1 objects. In this case, the vehicle initially will not be aware of the presence of type 1 or type 2 objects and so will engage any type it encounters. However, upon encountering a type 1 object, the vehicle has the potential to decide to stop engaging type 2 objects and search only for type 1 objects. This decision is made by the prey model algorithm, which depends on v_i, e_i, and $\hat{\lambda}_1$. If the vehicle decides to set $p_2 = 0$ and then does not encounter another type 1 object for a period of time, $\hat{\lambda}_1$ will eventually decrease to the point that the vehicle decides to give up waiting for another type 1 object and resume engaging type 2 objects. If v_1 is increased, the vehicle will take longer to switch p_2 from 0 back to 1 after encountering the type 1 object because the now-more-valuable type 1 objects are worth waiting for a little longer. This change is expected because the expected rate of point gain from type 1 objects increases with increasing v_1; therefore, it will take a smaller $\hat{\lambda}_1$ to drop this expected rate of gain below the profitability of a type 2 object.

After simulating the heuristic rate-estimation approach for the original example at the beginning of this section, we find that $p_2 = 1$ for the duration of the simulation, and the vehicle obtains an average rate of point gain of 0.0519 points/s. Thus, in this case, the heuristic approach results in the rate maximizing strategy of generalization. However, after increasing the value of type 1 objects to $v_1 = 80$, the vehicle ignores type 2 objects for a period of time (approximately 800 s) after each of the first two encounters with type 1 objects, and obtains a rate of point gain of 0.0591 points/s.

3.2 The Value of Recognition for Autonomous Vehicles

Suppose the single autonomous vehicle case is modified so that the vehicle can no longer immediately recognize the type of object it has encountered. For example,

suppose there is one type of target and a decoy for it. To distinguish between the two types, the vehicle must pay a recognition cost of r_v points and r_t s. Equation 2 can then be used to gain insight into the price the vehicle is willing to pay for this recognition ability. For example, if the vehicle has the ability to continuously update a rate-of-encounter estimation $\hat{\lambda}_i$ of objects as assumed in the heuristic implementation discussed above, then $\hat{\lambda}_i$ can be used in the prey model algorithm to determine whether to engage a given object type. In addition, Eq. 2 can be used to determine whether to pay for the cost of recognition.

An autonomous vehicle following this procedure was simulated using the same parameter values as in the previous simulations. Here type 2 objects can be thought of as decoys for the higher-value type 1 objects. With this in mind, let us arbitrarily set the object type values as $v_1 = 60$ and $v_2 = 10$. We assume the vehicle knows an object's type after engagement. This allows for calculation of $\hat{\lambda}_i$ regardless of the recognition decision. The activity of classification is part of the engagement process and is not related to type recognition. Figure 1 shows p_2 (top panel) and the decision of whether to pay for recognition (bottom panel) as a function of time for $r_v = 1.0$ with $r_t = 0$. This value of r_t simplifies the simulation and does not affect illustration of the important concepts. The vehicle initially happens to encounter only type 2 objects (the decoys) so that $\hat{\lambda}_1 = 0$, which causes p_2 to be 1 and an unwillingness to pay for recognition (there is no point in paying to recognize a nonexistant object type). Once the single type 1 object is encountered, the updated $\hat{\lambda}_1$ causes the vehicle

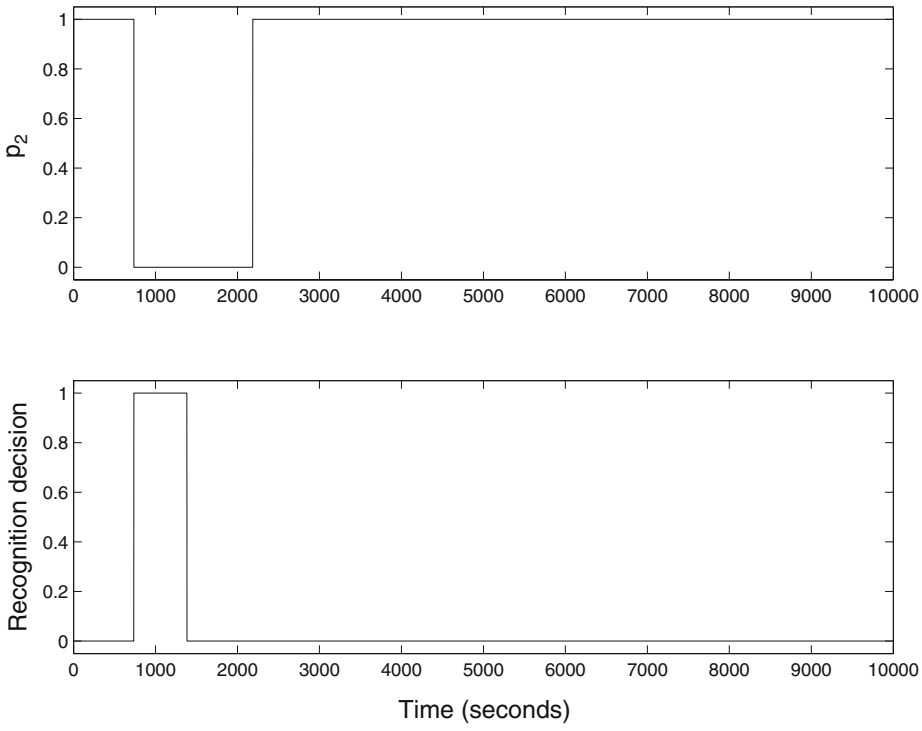

Fig. 1 Autonomous vehicle probability of engagement and recognition decision with $r_v = 1.0$. Recognition value of 1 is a willingness to pay the recognition cost

to begin paying r_v for the ability to recognize the high-valued type 1 objects. At some point, however, after not encountering any more type 1's, $\hat{\lambda}_1$ drops to the point that type 1 objects are too scarce to pay the cost of recognizing them. Although p_2 is constantly being updated based on $\hat{\lambda}_1$, it only influences the engagement decisions of the vehicle when recognition is enabled. If recognition is not enabled, the vehicle engages all objects regardless of the value of p_2. Therefore, in Fig. 1, even though the prey model algorithm tells the vehicle to ignore decoys until well after the decision to quit paying for recognition, decoys are still engaged because the cost of recognition outweighs the missed opportunity cost that accompanies engagement of the common decoys.

As the cost of recognition decreases, the amount of time the vehicle is willing to pay for recognition increases. Once r_v reaches zero, the amount of time that recognition is enabled equals the amount of time that $p_2 = 0$. This result makes sense because when $r_v = r_t = 0$, the problem becomes the original prey model problem. In addition, the derivation of Eq. 2 is based on the assumption that if $r_v = r_t = 0$, the agent will ignore type 2 objects; therefore, if this condition is met, p_2 and the recognition decision correspond exactly with one another.

4 Patch Model: How Long to Process?

The patch model describes an agent travelling between sources of point gain in the form of patches. We assume each patch is comprised of a group of tasks. The decision the agent must make is how long to remain in a patch once it is entered. The currency to be maximized is rate of point gain. The patch model we present next directly follows [1].

4.1 Patch Model and the Marginal Value Theorem

There are n types of patches which are described with three variables: λ_i is the rate at which the agent encounters patches of type i while searching (encounters are assumed to be sequential and follow a Poisson process), t_i is the time spent within patches of type i, and $v_i(t_i)$ is the gain function for patches of type i. The gain function is the expected net number of points obtained as a result of spending t_i time within patch type i. Patch residence time t_i includes search and task processing within the patch and is the decision variable for the agent.

The gain function $v_i(t_i)$ describes the net number of points that an agent expects to gain for spending a certain amount of time within a patch of type i. We assume the gain is zero when the time spent within the patch is zero ($v_i(0) = 0$) and that the gain function is negatively accelerated due to task depletion. Many types of gain functions exist, two of which are shown in Fig. 2. Figure 2a represents no within-patch search cost (unless tasks can regenerate) while b shows a case where within-patch search cost eventually causes point gain to decrease.

The average rate of point gain for an agent is

$$J = \frac{\sum_{i=1}^{n} \lambda_i v_i(t_i) - s}{1 + \sum_{i=1}^{n} \lambda_i t_i} \tag{3}$$

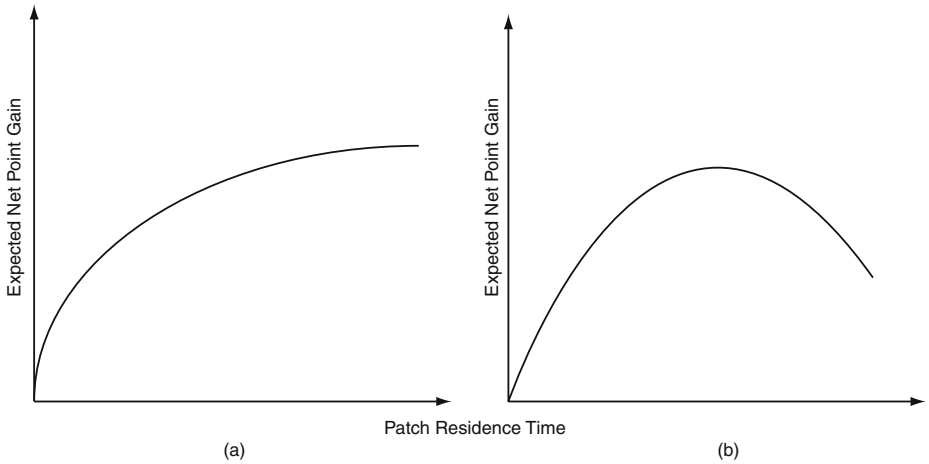

Fig. 2 Possible gain functions. As an agent spends time within a patch, its instantaneous rate of point gain decreases as a result of patch depletion

where s is the point cost of search per unit time. The patch residence time for patch type i that maximizes J is the one satisfying

$$\dot{v}_i(t_i) = J. \tag{4}$$

Thus, the optimal patch residence time $\hat{t}_i$ is the one that causes the instantaneous (marginal) rate of point gain in patches of type i to be equal to the long-term average rate of point gain. Therefore, an agent should remain in a particular patch until the marginal rate of point gain when leaving the patch is equal to the long-term average rate of point gain in the domain. This idea is known as the marginal value theorem.

4.2 Graphical Solution

The primary problem of the patch model is finding each $\hat{t}_i$ for all $i \in 1, 2, \ldots, n$ that satisfies Eq. 4. Depending on the gain function $v_i(t_i)$, this problem may or may not be a feasible task. To give some insight into the problem, we look at the simplest case: a single patch type with no search costs ($s = 0$). In this situation, Eq. 4 becomes

$$\dot{v}(\hat{t}) = \frac{\lambda v(\hat{t})}{1 + \lambda \hat{t}}, \tag{5}$$

where $\hat{t}$ is the optimal patch residence time. This equation can be solved graphically for $\hat{t}$ as shown in Fig. 3. Point gain is plotted on the vertical axis, and the horizontal axis is divided, with the travel time between patches $1/\lambda$ on the left side and the patch residence time on the right side. With this representation, the total time spent searching and processing tasks is the distance from a point $1/\lambda$ on the travel time axis to a point t on the patch residence time axis. The overall rate of point gain for the agent is $\frac{v(t)}{1/\lambda + t}$, which is the slope of the line from $(1/\lambda, 0)$ to $(t, v(t))$. The solution to Eq. 5 is then the $\hat{t}$ yielding $\dot{v}(\hat{t}) = \frac{v(\hat{t}}{1/\lambda + \hat{t}}$. This scenario is shown in Fig. 3 for two

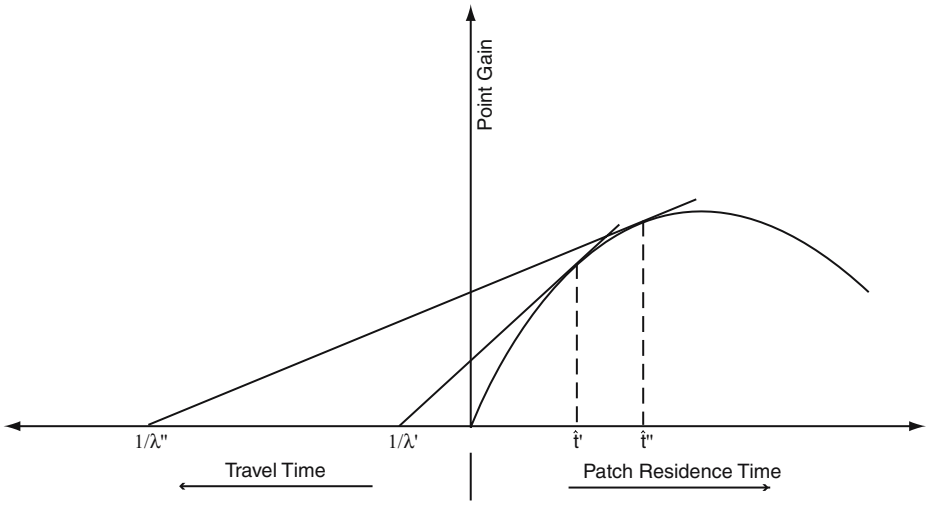

Fig. 3 Illustration of marginal-value theorem for one patch type and $s = 0$. The optimal patch residence time can be found graphically by constructing a line tangent to the gain function that passes through the between-patch travel time point on the left horizontal axis

cases. These two cases illustrate the important fact that as between-patch travel time increases (from $1/\lambda'$ to $1/\lambda''$), the agent should spend more time within the patch (compare $\hat{t}'$ to $\hat{t}''$).

For practical applications of the patch model to the control of agents, the graphical solution to the marginal value theorem is not generally feasible. However, the optimal patch residence time can be found analytically or numerically for certain gain functions. One example is an exponential function,

$$v(t) = \beta(1 - e^{-\alpha t}). \tag{6}$$

This function illustrates the case in Fig. 2a with no search cost. Plugging Eq. 6 into Eq. 5 and rearranging results in

$$\lambda e^{\alpha \hat{t}} = \alpha + \alpha\lambda\hat{t} + \lambda. \tag{7}$$

While Eq. 7 cannot be solved analytically, computational tools such as Matlab can solve specific realizations of it. It can be shown graphically that the solution $\hat{t}$ exists and is unique.

4.3 Discrete Marginal Value Theorem

A discrete version of the marginal value theorem is used to address simultaneous encounters [1]. Consider an agent that enters a patch where it simultaneously encounters tasks of two types. Suppose only one of the tasks can be processed. An example is in an autonomous vehicle application where the vehicle's sensor footprint detects more than one target at a time. If the targets have the ability to sense that they have been detected, only one of them can be engaged because the targets the vehicle decides not to engage will escape during the engagement of the chosen target. We

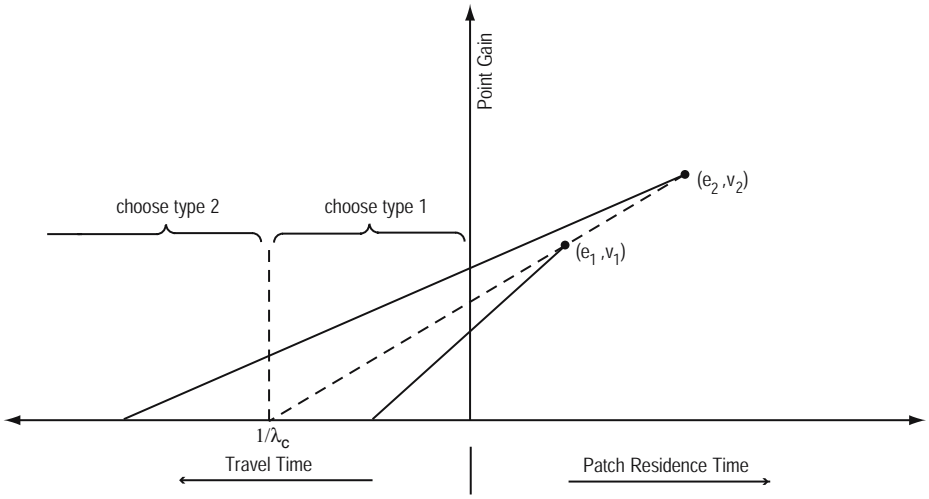

Fig. 4 Discrete marginal value theorem for simultaneous encounters. If travel time to the patch is shorter than the critical travel time $1/\lambda_c$, then the type 1 task should be processed. On the other hand, if travel time is longer than $1/\lambda_c$, then the type 2 task should be processed

can view the problem as a discrete version of the patch model; the patch residence time is the processing time of the task that is chosen.

Let v_1 and v_2 be the gains received for processing task types 1 and 2, respectively, and let e_1 and e_2 be the corresponding processing times. Suppose type 1 is smaller yet more profitable than type 2, so that $v_2 > v_1$ but $v_1/e_1 > v_2/e_2$. Under these conditions, Fig. 4 shows the discrete marginal value theorem and its prediction for the rate maximizing choice of which task to process. The axes are defined as in Fig. 3. The constant $1/\lambda_c$ is a critical travel time between patches that distinguishes between when the agent should choose task type 1 and when it should choose task type 2. This critical travel time is

$$\frac{1}{\lambda_c} = \frac{v_1 e_2 - v_2 e_1}{v_2 - v_1}.$$

An agent should choose the smaller more profitable task if $\lambda > \lambda_c$, and choose the larger less profitable task if $\lambda < \lambda_c$. If $\lambda = \lambda_c$, the agent is indifferent.

4.4 Combining the Prey and Patch Models

Both the prey and patch models assume that the decision intrinsic to the other model is already made. For example, the patch model assumes that the decision of whether to enter a particular patch is given; all that needs to be determined is when to leave the patch. Removing these assumptions results in the combination of the two models [1]. The question becomes which patches to enter and for how long. Application of the combined model to the control of agents is a direct extension of the concepts discussed here and in Section 2.

4.5 Assumptions

An agent is assumed to have knowledge of all of the model parameters a priori. This knowledge includes the rate of encounter with patches λ_i, which follows a Poisson process. The rate of encounter with tasks within the patch, however, is independent of t_i, and is not subject to any constraints except that the gain function $v_i(t)$ must hold. This issue raises the question of which gain function applies and when the parameters of that gain function are known. While a priori knowledge of $v(t)$ may not be feasible in many situations, it may be possible to estimate it. For example, Stone [13] derives a version of the exponential gain function for a vehicle searching for a single target in a domain. Under the assumptions of a uniform target distribution, a random uniformly distributed vehicle search path, and with knowledge of the vehicle sensor's sweep width and the area of the domain being searched, a detection function is derived that gives the probability of detecting the target for a given length of search path. This function is given by

$$b(z) = 1 - e^{-zW/A}, \tag{8}$$

where z is the length of the vehicle's search path, W is the sweep width of the vehicle's sensor, and A is the area of the domain being searched. Although our concern is the expected amount of points gained for a given amount of time spent within a patch, this case provides an example of how a gain function appropriate for a given environment might be derived.

5 Patch Model Application to Autonomous Vehicle Control for Search

Here we demonstrate the utility of the patch model for the control of an agent performing a search operation such as exploration of an unknown environment. We address a specific aspect of the autonomous vehicle control problem, namely how a vehicle searching for objects in a "patchy" environment should make decisions. We first describe the search problem and then discuss the applicability of the patch model.

5.1 Patch Model Application when Assumptions are Met

Suppose a single autonomous vehicle searches for objects in a domain that is 20 by 20 km. These objects are grouped in patches distributed according to a Poisson process. Within each patch, objects have a uniform random distribution. Imagine the domain as a two-dimensional plane with horizontal (x_1) and vertical (x_2) axes. The vehicle searches for patches in a "lawnmower pattern." Once a patch is encountered, the vehicle enters the patch and searches for objects. It is assumed that the vehicle knows the boundaries of a patch, there is only one patch type, and the vehicle enters all patches upon encounter. When the vehicle's footprint moves over an object, it is considered found and the appropriate points are awarded. Note that this differs from the prey model application in that no activities need to be performed on a found object. Here, the processing of a task is simply the discovery of an object, so we are studying a search problem as Stone does in [13]. The decision the vehicle must make is how long to search for objects in a particular patch.

Following [13], suppose the vehicle's search path within patches is random but uniform, so the gain function is given by Eq. 8, but modified so it is a function of time and not length of search path. In addition, it is scaled by the total number of points available in the patch V_p to provide an expected point gain for a given patch residence time. These modifications give

$$v(t) = V_p(1 - e^{-tuW/A}), \tag{9}$$

where u is the velocity of the vehicle. Note that V_p is equivalent to β in Eq. 6 and does not affect the optimal patch residence time since it does not appear in Eq. 7. Assume the width of the vehicle's sensor footprint $W = 500$ m, the area of the domain being searched (the patch area) $A = 4,000^2$ m^2, and the velocity $u = 120$ m/s. All patches contain 75 objects each with a value of 1 point, thus $V_p = 75$. Substituting these values into Eq. 9 results in the gain function given by the dashed curve in Fig. 5. The solid curve is the average realized gain function of the simulated vehicle, showing that Eq. 9 is a good approximation. One cause for the difference between the theoretical and simulated gain functions is the fact that the vehicle's footprint swings out of the patch at times. For example, even though the vehicle is still in the patch, a turn

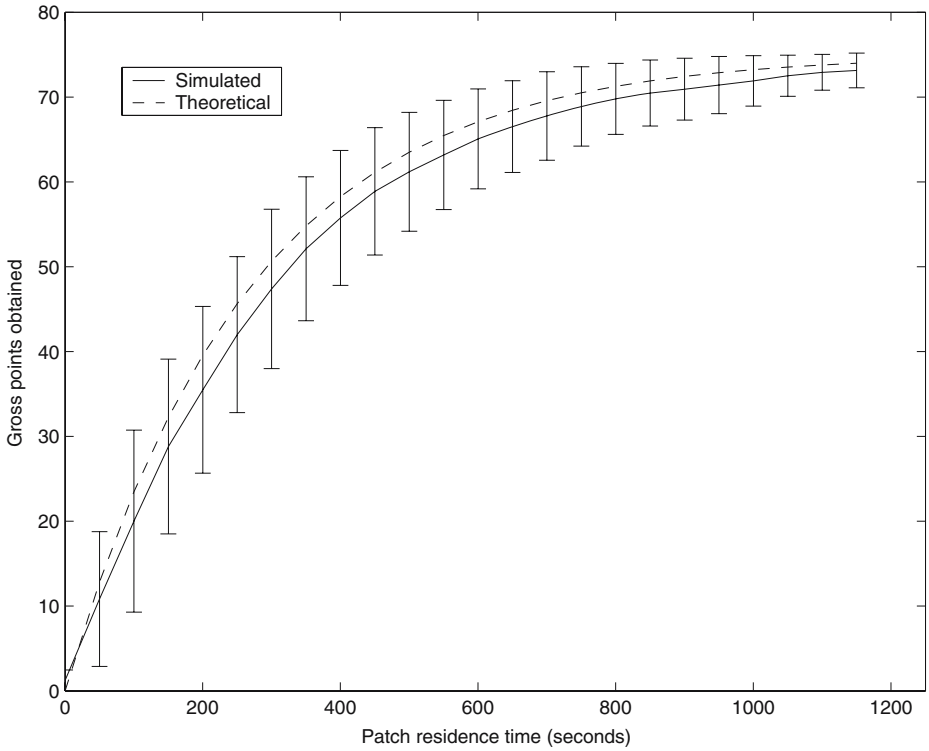

Fig. 5 Random search gain function. The *dashed curve* is the theoretical gain function for random search within patches, and the *solid curve* is the average gain function of a simulated autonomous vehicle. *Error bars* show the value of the gain function one standard deviation above and below the mean of the 150 runs

might bring its footprint out of the patch momentarily where no objects exist. This characteristic is not accounted for in Eq. 9.

If the vehicle has knowledge of the rate of encounter with patches in addition to the parameters in Eq. 9, the optimal patch residence time can be calculated by solving Eq. 7. The rate of encounter with patches is $(N_p W u)/A_d$, where N_p is the expected number of patches. Thus, if an average of 40 patches exists in the domain, $\lambda = \frac{40(500)(120)}{20{,}000^2} = 0.006$ s^{-1} (166.7 s travel time between patches) and the optimal patch residence time $\hat{t}$ from Eq. 7 is 251.5 s. However, the vehicle may not have this information. The remainder of this section discusses the situation where the vehicle has no a priori knowledge of the number of patches and thus the patch encounter rate.

5.2 Heuristic Implementation of the Patch Model

Consider a vehicle performing the same search operation described above, but with the exception that the vehicle has no information about the number of patches. Therefore, the optimal patch residence time cannot be calculated due to lack of information regarding patch encounter rate. A heuristic approach is to assume the vehicle has the ability to estimate its instantaneous rate of point gain within a particular patch as well as its overall rate of point gain. Under these assumptions, the vehicle can continuously monitor its rate of gain within a patch and decide to leave the patch when the instantaneous rate drops to the overall rate within the entire domain. This method is discussed as being the marginal-value *rule* by Stephens and Krebs in Chapter 4 of [1] and is a simple procedure that eliminates the need for a priori knowledge of gain functions for different patch types.

However, one issue that arises is how to calculate the instantaneous rate of point gain within a patch. Due to the discrete nature of the object distribution, smooth gain function curves are not characteristic of the patch environment discussed here. Derivative calculations are thus impossible. A sensible alternative is to calculate the slope between neighboring changes of the gain function. However, this slope estimation presents a problem if the vehicle finds all objects within a patch before it decides to leave. One solution is to assume the vehicle has memory not only of the previous gain function change, but the last two gain function changes. The instantaneous rate estimation can then be continuously calculated using the slope between the present gain function value and the value associated with the gain function change two changes prior to the current point in time. If all objects within a patch are found before leaving and this rate estimation approach is used, the current patch residence time will eventually become large enough to cause the instantaneous rate to drop to the point where the vehicle decides to leave the patch. The slope method given here is one simple way to approach the problem at hand. Many other methods for on-line estimation of a function exist in areas such as estimation theory and neural networks but are outside the scope of this article.

Suppose the heuristic approach described above is used as part of the vehicle's controller in the search operation. As before, the search path for patches is a lawnmower pattern, and search within patches is random but uniform. Assume an average of 30 patches exist in the domain, 20 objects exist within each patch, and patches are $2,000^2$ m^2. A Monte Carlo simulation is performed with each of 500 runs consisting of the vehicle searching the domain for 15,000 m in the horizontal direction

(30 lawnmower "strips"). At the beginning of each run, patches are distributed according to a Poisson process and objects within each patch are randomly but uniformly distributed. The average overall rate of point gain for each run is calculated as the total number of points obtained during the run divided by the total elapsed time for that run. The average overall rate of point gain of the 500 runs is 0.0146 points/s and its standard deviation is 0.0030 points/s, which suggests a sufficient number of runs.

As a means of comparison, the vehicle's patch residence time is changed so that it decides to leave the patch when its instantaneous rate of within-patch point gain is approximately zero. In other words, it leaves the patch when all objects within the patch are assumed to have been found. The Monte Carlo results for this control approach provide an average overall rate of point gain for 500 runs of 0.0094 points/s with a standard deviation of 0.0011 points/s. These results demonstrate that the marginal value theorem is superior to this alternative approach.

The vehicle's goal is to find objects within patches and in the examples above the search path used within patches is random but uniform. Another way to search within patches might be to use a lawnmower pattern, which is the same pattern used to search for the patches themselves. The average simulated gain function for this search method is shown in Fig. 6. As expected, due to the systematic nature of search, $v(t)$ is approximately a straight line, which levels off after the patch has been

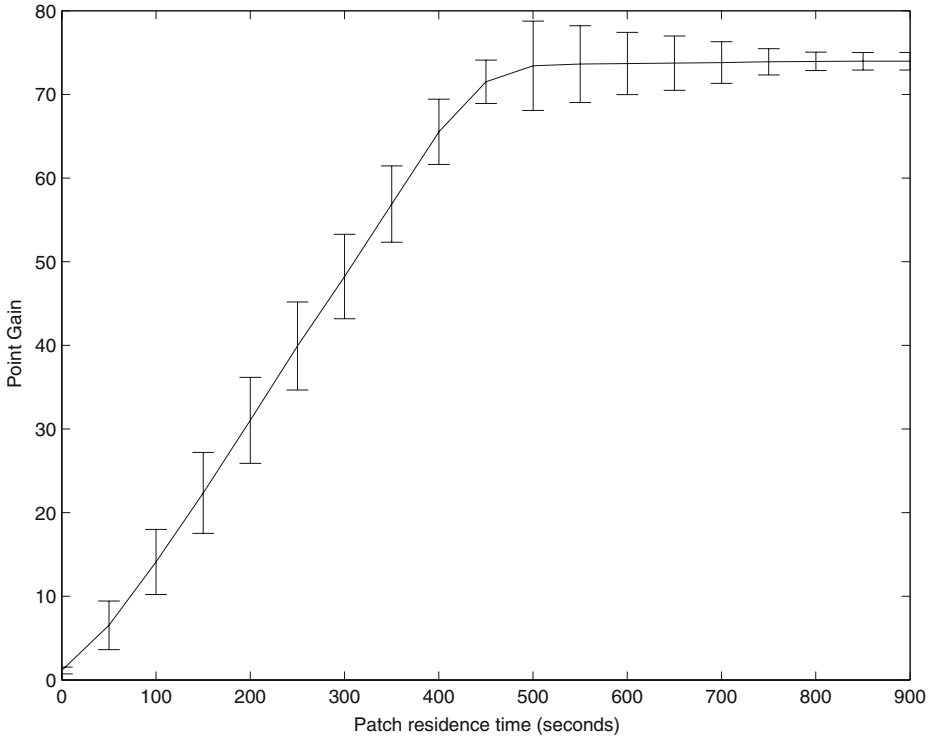

Fig. 6 Lawnmower search gain function. *Error bars* correspond to one standard deviation above and below the mean

searched extensively. Because the classical patch model's assumption of a negatively accelerated gain function does not hold in this case, there is no predicted effect of travel time on patch residence time. Instead, it can easily be seen in Fig. 7 that the optimal policy is simply to remain in the patch until all of the objects have been depleted.

Monte Carlo simulation, where the vehicle searches systematically and leaves after fully searching the patch, results in an average overall rate of point gain for 500 runs of 0.0188 points/s with a standard deviation of 0.0044 points/s. Note that this rate exceeds the rate obtained by using a random search path within patches. However, this increase in performance may come at the expense of increased complexity of the search pattern. Implementation of the lawnmower pattern might require a more accurate and complex vehicle controller. In particular, the vehicle must keep a straight path for a fixed amount of time and then repeat it with an opposite heading precisely one sensor width away from the previous path.

Whether a within-patch, lawnmower search pattern achieves a higher overall rate of point gain (the slope of the dashed lines in Fig. 7) than a random search depends on the rate of encounter with objects within the patch as well as the number of objects in the patch. Hence, this approach provides a method that can be used to determine when an agent searching within patches should do so extensively using a lawnmower pattern or should do so for a certain patch residence time using a random search. Keep in mind, though, that while the heuristic approach to the marginal value theorem can be used in environments with different patch types, graphical analyses such as those in Figs. 3 and 7 assume a single patch type. Therefore, the comparison of search methods mentioned above holds only for environments with a single patch type. A bar graph summary of the simulation results of this section is shown in Fig. 8.

These analyses provide insights regarding effects of search costs and information. The cost of searching within a patch is assumed to be embedded in the gain function

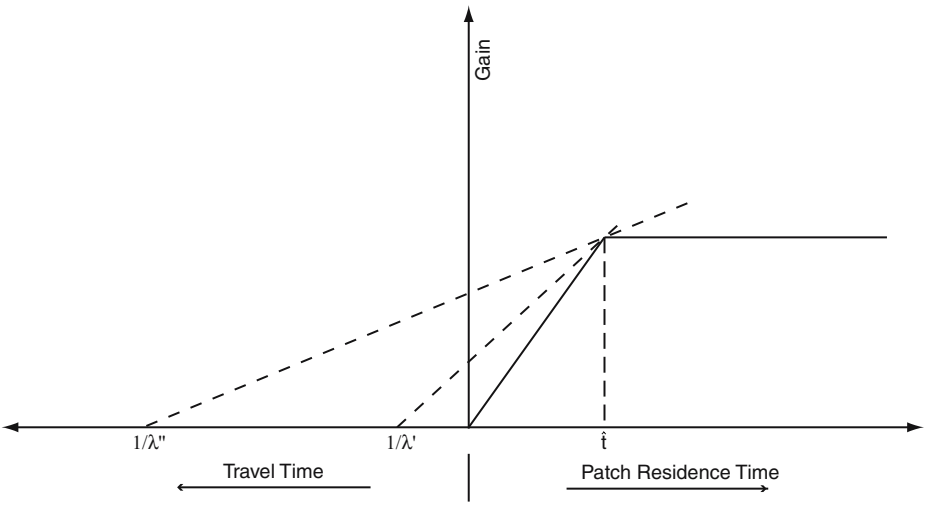

Fig. 7 Optimal patch residence time for a within-patch, lawnmower search pattern. Regardless of the encounter rate with patches, the optimal patch residence time is precisely the time at which the vehicle has exhausted the patch

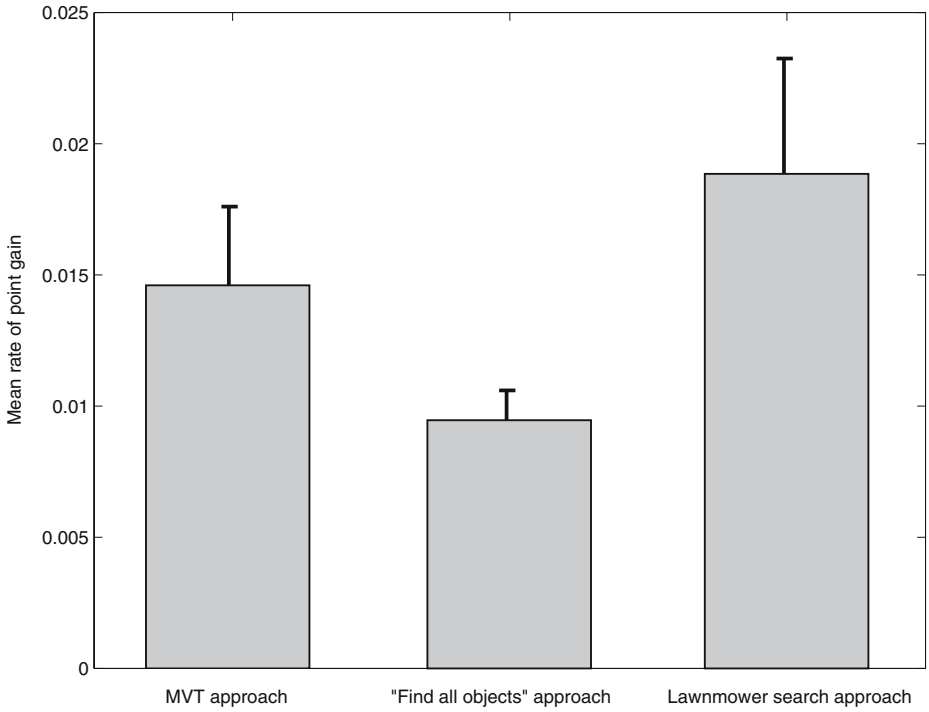

Fig. 8 Simulation summary. Monte carlo simulation results for 500 runs of an autonomous vehicle performing a search operation in a patchy environment. The within-patch search techniques considered are: random search and leaving according to the marginal value theorem (MVT), random search and leaving when all objects have been found, and lawnmower search and leaving when the patch has been searched exhaustively. *Error bars* show one standard deviation

since it is a description of net point gain. The agent only knows the relationship between time spent within a patch and its net point gain. The cost of search between patches, on the other hand, is explicitly represented by s in Eq. 3. The graphical, and in some cases analytic, solutions to the marginal value theorem discussed above are based on Eq. 5, which assumes $s = 0$. Optimal solutions may differ when $s \neq 0$. However, the heuristic approach to the marginal value theorem (the marginal value rule) discussed above is not based on Eq. 5 but rather the marginal value theorem itself. Therefore, in addition to being applicable to multiple patch-type environments, this approach is valid for nonzero search costs as long as the agent incorporates this cost into its rate estimation.

6 Time or Fuel Limitations and Risk-sensitivity

A limitation in many applications is a time constraint within which the agent must acquire a required number of points. This constraint may be due to a number of factors such as limited fuel capacity for autonomous vehicles. The question that arises is whether an agent should alter its task pool or patch residence-time decision depending on how much time or fuel remains. One approach to solving this

problem is to use risk-sensitive foraging theory [9]. The key idea is that animals may be sensitive to the variance in reward that accompanies foraging options. A dependence on variance arises when the success of a forager is a nonlinear function of the forager's state, usually energy reserves.

A simple example of risk-sensitivity is a time- or fuel-limited agent that must obtain a required number of points V_r by a critical time T_c. The currency, $J(V_f)$ where V_f is the number of points acquired during T_c, is a step function with the step occurring at $V_f = V_r$. An agent is either successful or unsuccessful depending on the number of points it has obtained by T_c. Suppose an agent has two options with the same mean, yet Option 1 is risky with some variance while Option 2 has zero variance. Assume the time remaining to T_c is small enough that the agent has just one decision remaining. Intuitively, if the agent has not yet met the point requirement and if the mean of the two options is such that obtaining it will boost the agent's points over V_r, then the agent should be risk averse because it needs only choose Option 2 to meet its requirement with certainty. However, if the mean point gain does not provide sufficient points to meet V_r, then the agent is sure to fail by choosing Option 2. Therefore, the agent should be risk prone and choose Option 1. The fitness function here is a discontinuous step function; however, a mathematical justification for our intuitive conclusion in the case of continuous convex or concave functions is described by means of Jensen's inequality [2].

Here, we use risk-sensitive foraging theory to discuss changes in the decisions predicted by both the prey and patch models when a requirement must be met within a time limit. In each case, we first provide the theory and then apply the theory to the autonomous vehicle control problem.

6.1 Risk-sensitive Prey Model

To address the fuel/time constraint problem with respect to the question of specialization, we need to evaluate the risks associated with choices predicted by the prey model. For simplicity, we initially assume just two task types. The mean net rates of point gain under the basic prey model for Option 1 (specializing on type 1 tasks) and Option 2 (processing all tasks encountered) are given in Eq. 1 as

$$\mu_1 = \frac{\lambda_1 v_1}{1 + \lambda_1 e_1} - s$$

and

$$\mu_2 = \frac{\lambda_1 v_1 + \lambda_2 v_2}{1 + \lambda_1 e_1 + \lambda_2 e_2} - s,$$

respectively. The variances in point gain per unit time associated with these decisions are

$$\sigma_1^2 = \frac{\lambda_1 v_1^2}{(1 + \lambda_1 e_1)^3}$$

and

$$\sigma_2^2 = \frac{\beta^2}{\alpha}\left[\lambda_1\left[\frac{v_1}{\beta} - \frac{e_1}{\alpha}\right]^2 + \lambda_2\left[\frac{v_2}{\beta} - \frac{e_2}{\alpha}\right]^2\right],$$

where $\alpha = 1 + \lambda_1 e_1 + \lambda_2 e_2$ and $\beta = \lambda_1 v_1 + \lambda_2 v_2$ [8]. Unlike the example given above, the decisions here do not have the same mean results. The agent thus has the choice at any point in time to choose between two mean-variance combinations so that it maximizes its probability of acquiring V_r by T_c. This problem can be modeled as a diffusion process [14] with the optimal solution for an agent with x points at time t given in [9] as choosing Option 1 if and only if

$$x < V_r + k(t - T_c), \tag{10}$$

where

$$k = \frac{\mu_2\sigma_1 - \mu_1\sigma_2}{\sigma_1 - \sigma_2}. \tag{11}$$

Equation 10 is a switching line that depends on the agent's current point gain and the time remaining. Above the line, the agent should process both types of tasks encountered and below the line the agent should process only type 1. The optimal decision depends on the number of points the agent has acquired thus far. If the agent has enough points with respect to the time remaining (above the line), it should play it safe and choose the less risky option. On the other hand, if the agent does not have enough points and time is running out, it should be risk prone.

An example for an autonomous vehicle searching for objects with a fuel-life of 2.5 h is shown in Fig. 9 with $\lambda_1 = 0.0003\ \mathrm{s}^{-1}$, $\lambda_2 = 0.0150\ \mathrm{s}^{-1}$, $v_1 = 40$, $v_2 = 1.5$, $s = 0$,

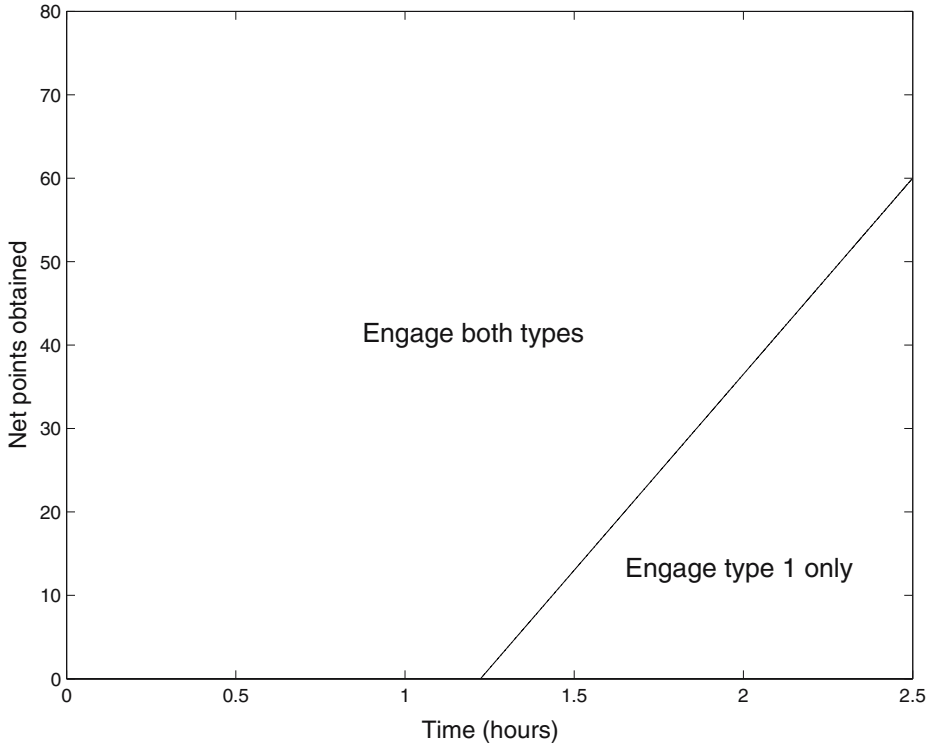

Fig. 9 Optimal object type choice for a risk-sensitive autonomous vehicle

and $V_r = 60$. If the vehicle had no time restrictions, it would be optimal to engage all objects encountered. However, as the vehicle begins to run out of fuel, it may, depending on the points it has obtained, begin skipping type 2 objects to wait for the riskier type 1 objects. As t approaches T_c, the switching line approaches V_r since an agent with just one decision remaining should be risk averse if its point total exceeds the requirement and risk prone if its point total is below the requirement. Examining Eq. 11, it can be seen that increasing $\mu_1 - \mu_2$ (increasing the mean rate of reward of Option 1 with respect to Option 2) decreases the slope of the line while the point (T_c, V_r) remains constant. Eventually, the slope becomes negative and the vehicle should engage only type 1 objects. On the other hand, as $\mu_2 - \mu_1$ increases, it eventually becomes optimal to engage both types of objects.

Monte Carlo simulations were run for a risk-sensitive and a risk-insensitive autonomous vehicle under the conditions listed for Fig. 9 but for a range T_c. Figure 10 shows the probability of a vehicle's succeeding, p_s, as a function of T_c. The value of p_s for a particular T_c is calculated by using the mean and standard deviation of the point gain of 500 simulation runs (a random normal variable due to the central limit theorem) to "standardize" the point gain to zero mean and unit variance. The probability of succeeding is then found by means of the cumulative distribution function of a standard normal random variable. At very small and very large critical

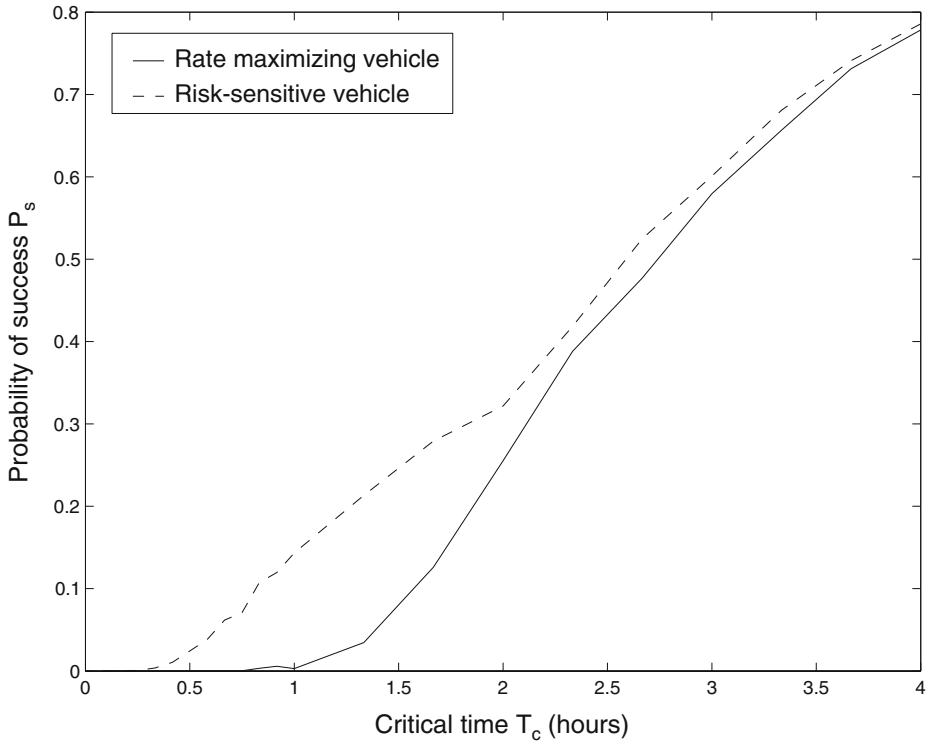

Fig. 10 Probability of success for a rate maximizing and a risk-sensitive autonomous vehicle as a function T_c

times, a risk-sensitive agent has no advantage over a rate-maximizing agent since both are either sure to fail or sure to succeed depending on the situation. However, at intermediate values of T_c, especially around 0.5–2.0 h in the autonomous vehicle example, a risk-sensitive vehicle has a higher probability of succeeding due to its exploitation of the variance in the environment.

6.2 Risk-sensitive Patch Model

Stephens and Charnov address the time constraint problem in [10] by formulating the patch model in terms of risk sensitivity. The mean and variance of the number of points acquired by an agent within a critical time limit T_c as a function of the patch residence time t are given in [10] as

$$\mu(t) = \frac{v(t)T_c}{(1/\lambda) + t} \tag{12}$$

and

$$\sigma^2(t) = \frac{T_c(1/\lambda^2)[v^2(t)]}{[(1/\lambda) + t]^3}, \tag{13}$$

respectively. Here we assume the existence of only one patch type and no between-patch search costs. Since the point gain within a patch is assumed to be deterministic by means of the gain function, the randomness arises from the stochastic encounter with patches.

The change in mean and standard deviation as an autonomous vehicle spends time searching randomly within a patch is shown in Fig. 11. This plot is obtained by evaluating Eqs. 12 and 13 for increasing values of t and plotting these values on a mean versus standard deviation graph. Suppose the critical time T_c is 10,800 s or 3 h. Suppose V_r is the number of points the autonomous vehicle must obtain by T_c. The optimal patch residence time is the one with the smallest z-score, $z = (V_r - \mu)/\sigma$, as explained in [10]. This time can be found graphically by constructing the line that originates at the point requirement V_r plotted on the mean axis and is tangent to the σ-μ curve. This line maximizes the slope and hence minimizes z. The corresponding patch residence time is the optimal choice. This z-score analysis is valid due to the fact that the sum of a large number of independent patch experiences has an approximately normal distribution (the central limit theorem).

The patch residence time predicted by the marginal value theorem, which strictly maximizes the mean, is denoted t^* in Fig. 11. It is then easy to see that if the expected point gain resulting from mean maximization, μ_{max}, is greater than V_r, the agent should be risk averse and remain in the patch longer than t^*. On the other hand, if $\mu_{max} < V_r$, the agent should be risk prone and leave sooner than t^*. The reason that leaving before t^* is risky is because a shorter patch residence time results in a lower overall average point gain yet a larger variance. Examples of the two cases discussed here are shown in Fig. 11 with patch residence times t' (risk prone) and t'' (risk averse) for requirements V_r' and V_r'', respectively. Note that if the requirement is infinite, the agent should be as risky as possible, choosing the patch residence time

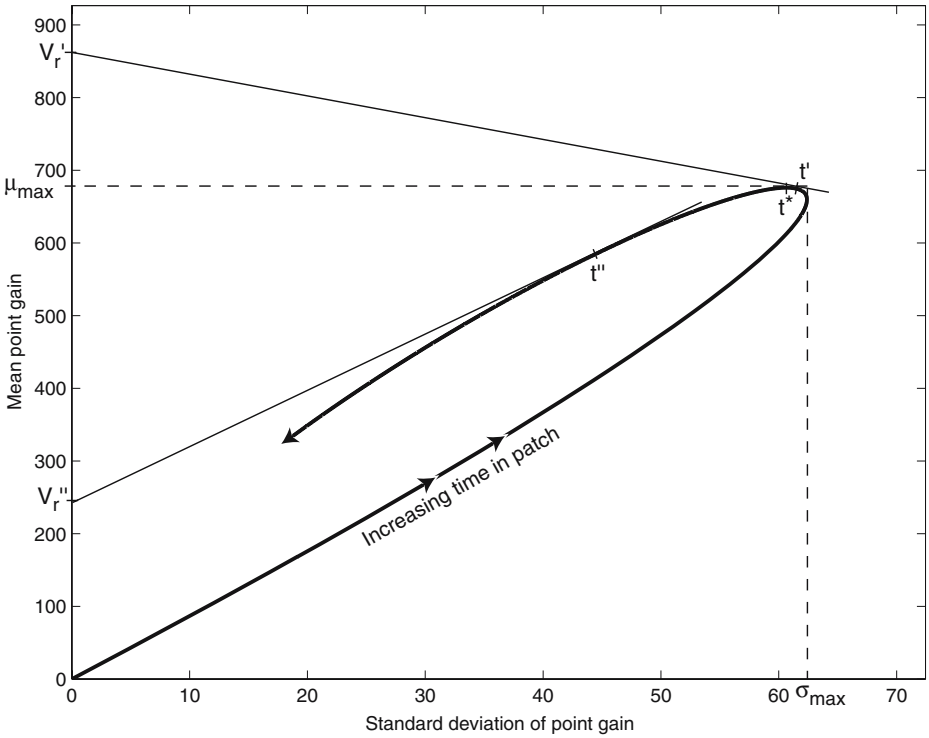

Fig. 11 Mean and standard deviation of point gain as a function of patch residence time for an autonomous vehicle. The optimal patch residence time for a risk-sensitive agent is found by constructing the line of maximal slope that originates at the requirement V_r and (*tangentially*) touches the *curve*

resulting in the maximum standard deviation. This choice is because the agent has no chance except to gamble.

If the gain function is known a priori, the optimal patch residence time for a risk-sensitive autonomous vehicle can be found by substituting Eqs. 12 and 13 into the z-score equation and minimizing the resulting function of patch residence time. While this may be difficult to do analytically, it can be achieved using various computational tools such as Matlab. On the other hand, if an agent does not have knowledge of the gain function, one potential solution is to perform on-line estimation. For example, consider an agent with memory that can be used to calculate an average gain function for the environment. The gain function estimate at a particular patch residence time is found by averaging all point gains at that patch residence time previously obtained. As time approaches infinity, the estimated gain function approaches the actual gain function. The agent can then use this gain function along with Eqs. 12 and 13 to calculate an estimated z-score for its current patch residence time. At the next time instant, a new z-score can be calculated. Once the z-score begins to increase, the agent should leave the patch and search for a new one. One potential problem arises since random perturbations may cause premature patch departure times. There are two possible approaches to solve this problem. One approach is to allow the agent to have a moving "window" of a certain

time length within which the z-score must strictly increase for the agent to leave the patch. Another approach is to force the agent to remain in the first few patches for a specified length of time so as to "learn" the environment.

A risk-sensitive autonomous vehicle performing the same search operation as in the previous applications is simulated using the z-score technique with various critical time limits to demonstrate the benefit of risk sensitivity. An average of 45 patches exist in the 20,000 by 20,000 m domain, each with 20 objects valued at 1 point. The vehicle has a point requirement $V_r = 150$ points, a z-score computation window of 25 s is used, and the vehicle must remain in the first three patches for 450 s. The results, compared with the heuristic rate-maximizing approach (the marginal value rule) are shown in Fig. 12, which plots T_c versus probability of success. Each point is obtained by using the average and standard deviation of the vehicle's point gain after 500 simulation runs to standardize the point gain, a normal random variable. The cumulative distribution function of a standard normal random variable is then used to calculate p_s. The results reveal that while an agent is sure to fail for small critical times and succeed for large critical times regardless of strategy, a risk-sensitive agent has a higher probability of succeeding than a rate-maximizing agent for intermediate values of T_c. However, a small region exists around 0.25 h where a rate maximizer has a higher p_s. This anomaly is a result of the fact that the risk-sensitive agent is

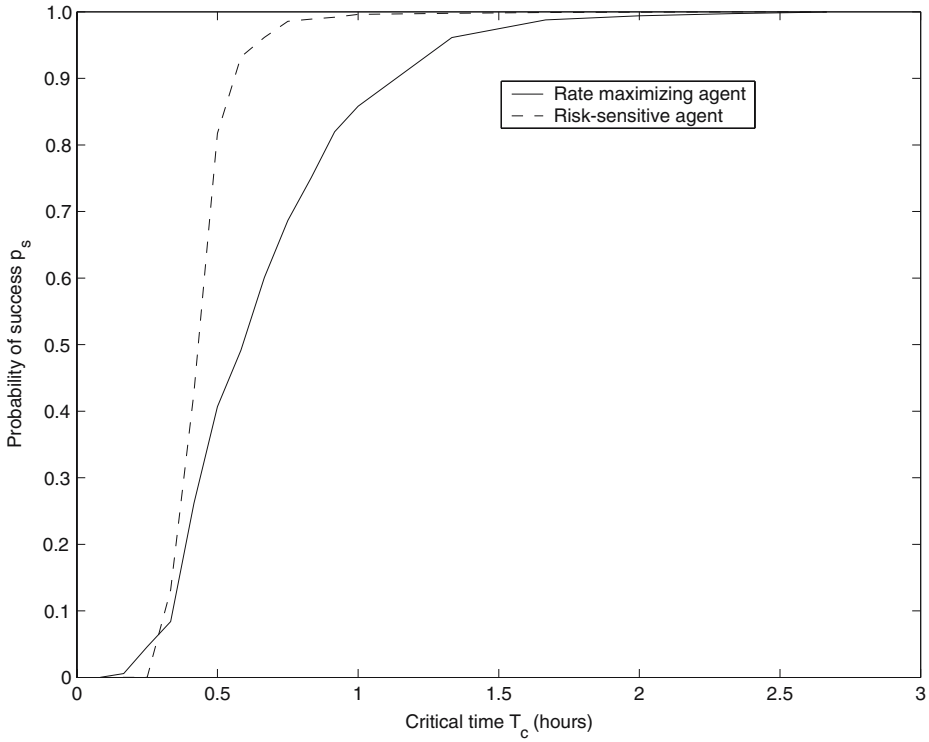

Fig. 12 Monte carlo simulation results for a rate-maximizing and a risk-sensitive autonomous vehicle as a function of T_c

forced to remain in the first three patches for a long time, which may be costly at low critical times. This issue suggests the need to select techniques in which a vehicle attempts to learn its environment. Clearly, the optimal z-score window size and initial patch departure times depend on environmental conditions and requirements. For instance, if T_c is very small or if there are very few patches, remaining in initial patches for an extended period is not beneficial.

One final point should be made regarding Fig. 12. The point requirement for the vehicle is relatively low, and the larger p_s for the risk-sensitive agent is a result of playing it safe for long critical times. Figure 11 shows that for $V_r > \mu_{max}$, the risk-sensitive and rate-maximizing patch residence times do not differ much due to the nature of the vehicle's σ-μ curve. Thus, when $V_r > \mu_{max}$, a risk-sensitive approach may not prove very productive for this particular case.

7 Conclusions

By treating decision-making agents as analogous to biological foragers, we have used foraging theory to develop high-level control strategies for an agent given certain environmental information. The problems considered here have not been previously studied. We demonstrated application of the theory in simulation for an autonomous vehicle. We have shown how the classical prey model can be used to determine which types of targets the vehicle should engage. Correspondingly, we used the patch model to predict patch residence time and make decisions about how to search within patches. If environmental information is incomplete, estimation techniques can be implemented. In addition, extensions of the classical models were used to explore, for example, recognition costs and decision adjustments under time or fuel limitations. We hope this initial attempt to apply optimality models from behavioral ecology to engineering problems will inspire much future work. An important future direction is the physical implementation of the theory in real applications. Such work will reveal issues such as sensing constraints that are either not addressed in the theory or are difficult to quantify in real systems. For example, we use the prey and patch models to control a temperature system where the controller is thought of as a forager searching for regions of error on a temperature grid in [15]. While this is not a vehicular application, it reveals issues associated with real, physical systems as well as the ability to apply the theory to a broad range of systems.

It is impossible to exploit the full range of theoretical results from behavioral ecology for engineering applications in a single article. A number of opportunities besides the ones presented here exist [16], some of which may result in cross-fertilization back to behavioral ecology. For example, Mangel and Clark in [3] present examples of dynamic, state-variable models in which dynamic programming can be used to derive the optimal policy for an animal. Also, Dukas provides an alteration of the classical prey model in [17] that accounts for prey conspicuousness and animal attention to different prey types. Finally, in [18], we show how to utilize social foraging theory [19] for multiagent engineering design problems.

Acknowledgements B.W. Andrews gratefully acknowledges the support of an Ohio Space Grant Consortium Fellowship. The work of K.M. Passino was supported in part by the AFRL/VA and AFOSR Collaborative Center of Control Science (Grant F33615-01-2-3154). We would also like to thank Ted Pavlic for his inputs.

References

1. Stephens, D.W., Krebs, J.R.: Foraging Theory. Princeton University Press, Princeton, NJ (1986)
2. Houston, A.I., McNamara, J.M.: Models of Adaptive Behaviour. Cambridge University Press, Cambridge, UK (1999)
3. Clark, C.W., Mangel, M.: Dynamic State Variable Models in Ecology. Oxford University Press, New York (2000)
4. Passino, K.M.: Biomimicry for Optimization, Control, and Automation. Springer, London (2005)
5. Passino, K.M.: Biomimicry of bacterial foraging for distributed optimization and control. IEEE Control Syst. Mag. **22**(3), 52–67 (2002)
6. Chandler, P.R., Pachter, M.: Research issues in autonomous control of tactical UAVs. In: Proceedings of the American Control Conference, Philadelphia, Pennsylvania, pp. 394–398, 24–26 June 1998
7. Quijano, N., Gil, A.E., Passino, K.M.: Experiments for dynamic resource allocation, scheduling, and control. IEEE Control Syst. Mag. **25**(1), 63–79 (2005)
8. McNamara, J.M., Houston, A.I.: Risk-sensitive foraging: a review of the theory. B. of Math. Biol. **54**(2/3), 355–378 (1992)
9. Houston, A.I., McNamara, J.M.: The choice of two prey types that minimises the probability of starvation. Behav. Ecol. Sociobiol. **17**, 135–141 (1985)
10. Stephens, D.W., Charnov, E.L.: Optimal foraging: some simple stochastic models. Behav. Ecol. Sociobiol. **10**, 251–263 (1982)
11. Jacques, D.R., Leblanc, R.: Effectiveness analysis for wide area search munitions. In: Proceedings of the AIAA Missile Sciences Conference, Monterey, CA, 17–19 Nov 1998
12. Jacques, D.R., Gillen, D.P.: Cooperation behavior schemes for improving the effectiveness of autonomous wide area search munitions. In: Murphey, R., Pardalos, P.M. (eds.) Cooperative Control and Optimization, vol. 66, pp. 95–120. Kluwer, Boston, MA (2002)
13. Stone, L.D.: Theory of Optimal Search. Academic, New York (1975)
14. McNamara, J.M.: Control of a diffusion by switching between two drift-diffusion coefficient pairs. SIAM J. Control Optim. **22**(1), 87–94 (1984)
15. Quijano, N., Andrews, B.W., Passino, K.M.: Foraging theory for multizone temperature control. IEEE Comput. Intell. Mag. **1**(4), 18–27 (2006)
16. McNamara, J.M., Houston, A.I., Collins, E.J.: Optimality models in behavioral biology. SIAM Rev. **43**(3), 413–466 (2001)
17. Dukas, R.: Constraints on information and their effects on behavior. In: Dukas, R. (ed.) Cognitive Ecology: The Evolutionary Ecology of Information Processing and Decision Making, pp. 89–127. University of Chicago Press, Chicago, IL (1998)
18. Andrews, B.W., Passino, K.M., Waite, T.A.: Social foraging theory for robust multiagent system design. IEEE Trans. Autom. Sci. Eng. **4**(1), 79–86 (2007)
19. Giraldeau, L., Caraco, T.: Social Foraging Theory. Princeton University Press, Princeton, NJ (2000)