Engineering Applications of Artificial Intelligence 22 (2009) 482–489

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Foraging theory for autonomous vehicle speed choice

Theodore P. Pavlic *, Kevin M. Passino [1]

Department of Electrical and Computer Engineering, 205 Dreese Labs, 2015 Neil Avenue, Columbus, OH 43210, USA

ARTICLE INFO

Article history:
Received 24 May 2006
Received in revised form
3 March 2008
Accepted 31 October 2008
Available online 9 January 2009

Keywords:
Intelligent control
Optimal control
Task-type choice
Speed–accuracy trade-off
Speed–cost trade-off
Decision-making algorithms

ABSTRACT

We consider the optimal control design of an abstract autonomous vehicle (AAV). The AAV searches an area for tasks that are detected with a probability that depends on vehicle speed, and each detected task can be processed or ignored. Both searching and processing are costly, but processing also returns rewards that quantify designer preferences. We generalize results from the analysis of animal foraging behavior to model the AAV. Then, using a performance metric common in behavioral ecology, we explicitly find the optimal speed and task processing choice policy for a version of the AAV problem. Finally, in simulation, we show how parameter estimation can be used to determine the optimal controller online when density of task types is unknown.

1. Introduction

Consider a vehicle that searches through a territory for tasks, and let probability of task detection depend on vehicle speed (e.g., sensor bandwidth or hysteresis requirements may prevent detections at high search speeds). On detecting a task, the vehicle may choose to process it. Both searching and processing are costly (e.g., monetary cost of depleted vehicle fuel), and search costs depend on search speed. However, the designer rewards the vehicle for processing. Hence, good vehicle control policies will set search speed and determine tasks to process in a way that balances rewards and costs.

Our vehicle description fits a variety of robotics, queueing, and other engineering applications well (e.g., Quijano et al., 2006; Passino, 2002, 2005), and so we refer to it as an *abstract autonomous vehicle* (*AAV*). The AAV is a generalization of the solitary forager from optimal foraging theory (Stephens and Krebs, 1986). Optimal foraging theorists assume that natural selection results in behaviors that optimize Darwinian fitness, and so they use analysis of proximate fitness functions to explain observed behaviors in nature. Lately, methods from optimal foraging theory have been used to analyze human behavior in nature (e.g., Rowcliffe et al., 2004) and on the Internet (Pirolli and Card, 1999; Pirolli, 2005, 2007). Following the example of Andrews et al. (2004, 2007), we apply similar methods to AAV policy design.

Early foraging theory by Schoener (1971) postulates that optimal behaviors are found on a continuum with foraging time minimization and net energetic intake maximization at opposite ends. These strategies allow time for other activities (e.g., predator avoidance and reproduction) while preventing starvation. Later, Pyke et al. (1977) argue that lifetime energetic *rate* maximization is consistent with this idea, and Charnov and Orians (1973) develop several rate maximization models that continue to be used today (Charnov, 1976a, b; Stephens and Krebs, 1986). As discussed by Houston and McNamara (1999), rate maximization is equivalent to minimizing the *opportunity cost* of each foraging decision. In our context, foraging is equivalent to task processing, and so net energetic intake is equivalent to net reward gained. Hence, we develop AAV policies that maximize long-term net *rate* of reward.

The present work combines and extends work by Andrews et al. (2004), which extends foraging theory to engineering, and Gendron and Staddon (1983), which modifies standard foraging models to include speed effects. Our AAV model subsumes those models while also including more general search and processing costs. Under mild assumptions on detection probabilities, we give an analytical solution for the optimal AAV behavior in an environment with an arbitrary number of task types. Additionally, we use computer simulation to show how online parameter estimation leads to convergence to the optimal AAV policy. While this more general AAV model can be useful in ecological analysis, we focus on its application to engineering design.

* Corresponding author. Tel.: +1 614 292 2572; fax: +1 614 292 7596.
E-mail addresses: pavlic.3@osu.edu (T.P. Pavlic), passino@ece.osu.edu (K.M. Passino).

[1] Supported in part by the AFRL/AFOSR Collaborative Center of Control Science (Grant F33615-01-2-3154).

0952-1976/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.engappai.2008.10.017

The remainder of this paper is organized as follows. In Sections 2 and 3, the foraging-based AAV model is presented. Analytical optimization results are given in Section 4, and these results are validated using results from a fixed-wing airborne vehicle simulation in Section 5. Finally, in Section 6, we make some concluding remarks and suggestions for future work.

2. Simple AAV model

First, we present an AAV model that does not explicitly depend on search speed. This model is based upon the *prey model* described by Stephens and Krebs (1986). The solitary AAV moves through an environment searching for tasks drawn from a known set of task types. Tasks from each type are found according to a Poisson process (i.e., the AAV faces a merged Poisson process). When a task is encountered, the AAV decides whether to process or ignore it. Processing each task takes time but rewards the AAV on completion. Search costs, processing costs, and processing rewards are all given in a common currency that we call *points*. Task types are characterized by average task processing gain and cost. The parameters we use to describe the task types are denoted by the following:

- $n \in \mathbb{N}$: number of task types,
- $i \in \{1,2,\ldots,n\}$: index used for referring to a specific task type,
- $g_i^p \in \mathbb{R}$: average point gain from processing a task of type i,
- $c_i^p \in \mathbb{R}$: average cost rate of processing a task of type i given in points per unit of processing time,
- $t_i^p \in \{x \in \mathbb{R} : x > 0\}$: average time required to process a task of type i, and
- $\lambda_i \in \{x \in \mathbb{R} : x > 0\}$: Poisson encounter rate of task type i given in encounters per unit time.

Likewise, AAV parameters are denoted by:

- $c^s \in \mathbb{R}$: cost rate of searching given in points per unit time,
- $q_i \in [0,1]$: probability that AAV will process an encountered task of type i, and
- $\vec{q} \in [0,1]^n$: the vector $[q_1, q_2, \ldots, q_n]^{\mathrm{T}}$.

The elements of $\vec{q}$ are set by the AAV control policy.

2.1. Average net rate of point gain

Using methods like those used by Charnov and Orians (1973) and Stephens and Krebs (1986), the long-term average rate of point gain for the AAV using task-type preference policy $\vec{q}$ can be shown to be $J : [0,1]^n \mapsto \mathbb{R}$ where

$$J(\vec{q}) \triangleq \frac{\sum_{i=1}^{n} \lambda_i q_i (g_i^p - c_i^p t_i^p) - c^s}{1 + \sum_{i=1}^{n} \lambda_i q_i t_i^p}, \tag{1}$$

which is in points per unit time.

2.2. Zero–one rule

As shown by Stephens and Krebs (1986), without loss of generality, optimization of $J(\vec{q})$ may be considered a combinatorial problem where $\vec{q} \in \{0,1\}^n$. That is, an optimal solution with extreme preferences (i.e., $q_i = 0$ or 1 for all $i \in \{1,2,\ldots,n\}$) always exists. We assume this so-called "zero–one rule," which implies that there is a set of task types such that any task encounter from one of these types should always be processed, and any task encounter outside of this set should always be ignored. We call this special set of task types a task pool.

2.3. Task-type ordering

Without loss of generality, task types are ordered (i.e., indexed) by their profitability, which is defined as the ratio of the net point gain from processing a task of that type to the processing time of that task. In other words, task types are ordered so that

$$\frac{g_1^p - c_1^p t_1^p}{t_1^p} \geqslant \frac{g_2^p - c_2^p t_2^p}{t_2^p} \geqslant \cdots \geqslant \frac{g_{n-1}^p - c_{n-1}^p t_{n-1}^p}{t_{n-1}^p} \geqslant \frac{g_n^p - c_n^p t_n^p}{t_n^p}. \tag{2}$$

The general combinatorial optimization problem involves searching through the space of 2^n task pools. However, Stephens and Krebs (1986) show that if tasks are ordered in this way, the only valid task pool candidates are the empty set and the n task pool candidates of the form $\{1,\ldots,k\}$ where $k \in \{1,\ldots,n\}$. So, this ordering reduces this combinatorial optimization problem to a search over $n+1$ possible candidates.

3. Speed-dependent AAV model

We assume that the AAV speed is described with the following parameters:

- $u_{\min} \in \{x \in \mathbb{R} : x \geqslant 0\}$: the minimum possible AAV speed given in length units per time units,
- $u_{\max} \in \{x \in \mathbb{R} : x \geqslant u_{\min}\}$: the maximum possible AAV speed given in length units per time units, and
- $u \in [u_{\min}, u_{\min}]$: constant speed of the AAV given in length units per time units.

In the following, we enhance the simple AAV model by integrating the speed u into existing AAV parameters.

3.1. Search and processing cost assumptions

For simplicity, we assume that the AAV has the same processing cost c^p for each task type (i.e., $c_i^p \equiv c^p$ for all $i \in \{1,\ldots,n\}$). So, the profitability ordering from Eq. (2) is equivalent to the ordering

$$\frac{g_1^p}{t_1^p} \geqslant \frac{g_2^p}{t_2^p} \geqslant \cdots \geqslant \frac{g_{n-1}^p}{t_{n-1}^p} \geqslant \frac{g_n^p}{t_n^p}, \tag{3}$$

which has no dependence on speed u. Note that c^p may depend on speed u; however, because every task type has the same c^p, the ordering does not.

We assume that search cost $c^s : [u_{\min}, u_{\max}] \mapsto \mathbb{R}$ is defined by

$$c^s(u) \triangleq c_\ell^s u + c_a^s \tag{4}$$

with the definitions:

- $c_\ell^s \in \mathbb{R}$: linear portion of the cost rate given in points per unit time per unit speed and
- $c_a^s \in \mathbb{R}$: constant portion of the cost rate given in points per unit time.

Our choice of $c^s(u)$ ensures analytical tractability; however, it is a reasonable model of speed-dependent fuel cost.

Similarly, the processing cost $c^p : [u_{\min}, u_{\max}] \mapsto \mathbb{R}$ is defined by

$$c^p(u) \triangleq c_\ell^p u + c_a^p \tag{5}$$

with the definitions:

- $c_\ell^p \in \mathbb{R}$: linear portion of the cost rate given in points per unit time per unit speed and

- $c_a^p \in \mathbb{R}$: constant portion of the cost rate given in points per unit time.

3.2. Encounter rate and probability of detection

We assume that encounter rate λ_i is such that

$$\lambda_i \triangleq uD_ip_i \tag{6}$$

for all $i \in \{1, 2 \ldots, n\}$ with the parameter definitions:

- $D_i \in \{x \in \mathbb{R} : x \geqslant 0\}$: linear density of tasks of type i along AAV search path given in number of tasks per unit length and
- $p_i \in [0, 1]$: probability of detecting a task of type i.

The probability of detection p_i also depends on the search speed u. Gendron and Staddon (1983) use a sophisticated relationship modeling that is valid for some specific biological contexts (Gendron, 1982). There is little reason to believe their choice applies well to all AAV problems, and so because it presents barriers to analysis for $n > 1$, we choose an affine relationship that connects minimum speed $u_{\min}$ and maximum speed $u_{\max}$ by a straight line. That is, for all $i \in \{1, \ldots, n\}$, we define:

- $p_i^{\text{slow}} \in [0, 1]$: probability of detecting task of type i at minimum speed $u_{\min}$ and
- $p_i^{\text{fast}} \in [0, 1]$: probability of detecting task of type i at maximum speed $u_{\max}$

and use linear interpolation to find the detection probability for $u \in (u_{\min}, u_{\max})$.

For example, the probability of detection function $p_i(u)$ shown in Fig. 1 is

$$p_i(u) \triangleq p_i^{\ell} u + p_i^a, \tag{7}$$

where

$$p_i^{\ell} \triangleq \frac{p_i^{\text{fast}} - p_i^{\text{slow}}}{u_{\max} - u_{\min}}, \tag{8}$$

$$p_i^a \triangleq p_i^{\text{slow}} - p_i^{\ell} u_{\min}. \tag{9}$$

We do not assume that $p_i^{\ell} \leqslant 0$. That is, the slope of each detection function may be negative, flat, or positive. However, results are most interesting when slopes are nonpositive. Otherwise, optimal behaviors are strongly biased toward operation at maximum speed.

In summary, encounter rate for task type i is $\lambda_i : [u_{\min}, u_{\max}] \mapsto \mathbb{R}$ defined by

$$\lambda_i(u) \triangleq uD_ip_i(u) = uD_i(p_i^{\ell} u + p_i^a) = D_ip_i^{\ell} u^2 + D_ip_i^a u. \tag{10}$$

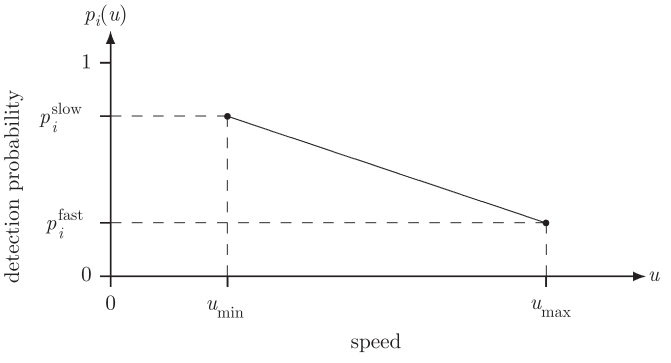

Fig. 1. Probability of detection function $p_i(u)$ for task type i.

3.3. Average net rate of point gain with speed

The average net rate of point gain used in this paper is obtained by applying Eqs. (4), (5), and (10) to Eq. (1), which gives function $J : \{0, 1\}^n \times [u_{\min}, u_{\max}] \mapsto \mathbb{R}$ defined by

$$\begin{aligned} J(\vec{q}, u) &\triangleq \frac{\sum_{i=1}^n \lambda_i(u) q_i (g_i^p - c^p(u) t_i^p) - c^s(u)}{1 + \sum_{i=1}^n \lambda_i(u) q_i t_i^p} \\ &= \frac{\sum_{i=1}^n uD_i(p_i^{\ell} u + p_i^a) q_i (g_i^p - (c_{\ell}^p u + c_a^p) t_i^p) - c_{\ell}^s u - c_a^s}{1 + \sum_{i=1}^n uD_i(p_i^{\ell} u + p_i^a) q_i t_i^p} \\ &= \frac{\left[\begin{array}{l} \sum_{i=1}^n (g_i^p D_i p_i^{\ell} q_i - t_i^p D_i c_{\ell}^p p_i^{\ell} q_i u - t_i^p D_i c_a^p p_i^{\ell} q_i) u^2 \\ \quad + \sum_{i=1}^n (g_i^p - c_{\ell}^p t_i^p u - c_a^p t_i^p) D_i p_i^a q_i u - c_{\ell}^s u - c_a^s \end{array}\right]}{1 + \sum_{i=1}^n q_i t_i^p D_i p_i^{\ell} u^2 + \sum_{i=1}^n q_i t_i^p D_i p_i^a u} \\ &= \frac{G_3(\vec{q}) u^3 + G_2(\vec{q}) u^2 + G_1(\vec{q}) u + G_0(\vec{q})}{T_2(\vec{q}) u^2 + T_1(\vec{q}) u + 1}, \end{aligned} \tag{11}$$

where

$$G_3(\vec{q}) \triangleq -\sum_{i=1}^n t_i^p c_{\ell}^p D_i p_i^{\ell} q_i, \tag{12}$$

$$G_2(\vec{q}) \triangleq \sum_{i=1}^n D_i q_i (g_i^p p_i^{\ell} - c_{\ell}^p t_i^p p_i^a - c_a^p t_i^p p_i^{\ell}), \tag{13}$$

$$G_1(\vec{q}) \triangleq \sum_{i=1}^n D_i p_i^a q_i (g_i^p - c_a^p t_i^p) - c_{\ell}^s, \tag{14}$$

$$G_0(\vec{q}) \triangleq -c_a^s, \tag{15}$$

$$T_2(\vec{q}) \triangleq \sum_{i=1}^n q_i t_i^p D_i p_i^{\ell} \tag{16}$$

and

$$T_1(\vec{q}) \triangleq \sum_{i=1}^n q_i t_i^p D_i p_i^a. \tag{17}$$

Because the functions in Eqs. (12)–(17) are independent of speed u, Eq. (11) is a simple ratio of two polynomials in u where the numerator and denominator polynomials both have a degree that is independent of n.

4. Optimization of net rate over speed and task choice

Stephens and Krebs (1986) show that the optimal task pool is determined by the encounter rates of different task types in that pool. Because encounter rate varies with speed, the optimal task pool will vary with speed. However, because Eqs. (12)–(17) are functions of the task pool, the optimal AAV speed varies with task pool. So, solving for optimal speed and optimal task pool cannot be done separately; these two problems must be solved simultaneously.

4.1. Algorithm outline

The task-type ordering in Eq. (3) does not depend on speed, and so the set of the $n + 1$ task pool candidates does not vary with speed. The set will always be the empty set and the n task pools of the form $\{1, 2, \ldots, k\}$ for all $k \in \{1, \ldots, n\}$. So, we develop an optimization algorithm that finds the optimal speed for a given task pool. By iterating this algorithm over all $n + 1$ pools, the optimal speed and task pool combination can be found.

4.2. Solving for the optimal speed using Lagrange multiplier methods

Let $\vec{q}$ represent any given task choice (i.e., a given task pool). Because $\vec{q}$ is fixed, then $J(\vec{q},u)$ in Eq. (11) is a $[u_{\min},u_{\max}]\mapsto\mathbb{R}$ function of u which is continuous and twice differentiable. For simplicity, we abbreviate $J(\vec{q},u)$ by $J(u)$ or simply J. Similarly, the $\vec{q}$ argument will be dropped from the six parameters in Eqs. (12)–(17). Here, we find the optimal speed $u=u^*$ for the function J.

The equivalent constrained optimization problem

$$\begin{aligned}&\text{minimize} \quad -J(u)\\ &\text{subject to} \quad u-u_{\max}\leqslant 0, \quad u_{\min}-u\leqslant 0\end{aligned} \tag{18}$$

can be solved using Lagrange multiplier methods (Bertsekas, 1995).

Because at most one of the two inequality constraints may be active at a time, all points $u\in\mathbb{R}$ are regular (i.e., the active constraint gradients will always be linearly independent). So, the Kuhn–Tucker necessary conditions for a local minimum may be applied in all cases. Because $-J$ is a continuous function defined over a compact set, the Lagrange multiplier method is guaranteed to find every local minimum of Eq. (18), and so all points satisfying these conditions should be considered optimal speed candidates for the given task pool.

Let $\mu_1,\mu_2\in\{x\in\mathbb{R}:x\geqslant 0\}$ be the Lagrange multipliers corresponding to the two inequality constraints in Eq. (18). The resulting Lagrangian function for this problem, its gradient, and its Hessian are

$$\begin{aligned}L(u)&=-J(u)+\mu_1(u-u_{\max})+\mu_2(u_{\min}-u)\\&=\frac{-(G_3u^3+G_2u^2+G_1u+G_0)}{T_2u^2+T_1u+1}+\mu_1(u-u_{\max})+\mu_2(u_{\min}-u),\end{aligned} \tag{19}$$

$$\begin{aligned}\nabla_u L(u)&=-\nabla_u J(u)+\mu_1-\mu_2\\&=\frac{-\begin{pmatrix}G_3T_2u^4+2G_3T_1u^3+(G_2T_1+3G_3-G_1T_2)u^2\\+2(G_2-G_0T_2)u+G_1-G_0T_1\end{pmatrix}}{(T_2u^2+T_1u+1)^2}+\mu_1-\mu_2\end{aligned} \tag{20}$$

and

$$\begin{aligned}\nabla^2_{uu}L(u)&=-\nabla^2_{uu}J(u)\\&=\frac{\left\{\begin{array}{l}(\cdot)2(T_2u^2+T_1u+1)(2T_2u+T_1)\\-(T_2u^2+T_1u+1)^2\left[\begin{array}{l}4G_3T_2u^3+6G_3T_1u^2\\+2(G_2T_1+3G_3-G_1T_2)u\\+2(G_2-G_0T_2)\end{array}\right]\end{array}\right\}}{(T_2u^2+T_1u+1)^4},\end{aligned} \tag{21}$$

respectively. The "$(\cdot)$" in Eq. (21) is the bracketed numerator in Eq. (20). Only the sign of the complicated Eq. (21) is significant. Additionally, because it will only be used in the unconstrained case at points where Eq. (20) is zero, many of its terms will disappear. In fact, only the sign of the square-bracketed expression in the numerator has any impact on the analysis.

The goal is to find a set of speed candidates that includes the optimal speed u^* for the given task pool. There are three cases to consider.

4.2.1. Unconstrained case

In this case,

$$\mu_1=\mu_2=0 \quad \text{and} \quad u_{\min}<u<u_{\max}. \tag{22}$$

Any optimal point must make the bracketed expression in the numerator of the fraction in Eq. (20) equal to zero. So, any real roots of this fourth-order polynomial that satisfy the conditions in Eq. (22) are candidates.[2]

Because all variations are feasible in the unconstrained case, second-order conditions can be used to reduce the number of candidates in this set. That is, a point in this set cannot be considered to be a candidate unless the square-bracketed expression in the numerator of the fraction in Eq. (21) is negative or zero at that point.

In summary, all real roots of the bracketed expression in Eq. (20) that satisfy Eq. (22) and additionally make the square-bracketed expression in Eq. (21) nonpositive should be included in the set of u^* candidates.

4.2.2. Lower-constrained case

In this case,

$$\mu_1=0 \quad \text{and} \quad u=u_{\min}. \tag{23}$$

Because the set of feasible variations at this constraint boundary is simply $\{0\}$, the second-order necessary conditions will always hold.

Because $\mu_2\geqslant 0$, if the bracketed expression in Eq. (20) is negative or zero at $u=u_{\min}$, then $u_{\min}$ is a u^* candidate.

4.2.3. Upper-constrained case

In this case,

$$\mu_2=0 \quad \text{and} \quad u=u_{\max}. \tag{24}$$

Because the set of feasible variations at this constraint boundary is simply $\{0\}$, the second-order necessary conditions will always hold.

Because $\mu_1\geqslant 0$, if the bracketed expression in Eq. (20) is positive or zero at $u=u_{\max}$, then $u_{\max}$ is a u^* candidate.

This method finds a small set of u^* candidates for a given task pool. The element of this set that maximizes $J(u)$ over the set will be $u=u^*$ for this $\vec{q}$. Finding the u^* for each of the $n+1$ possible optimal task pools and then maximizing $J(\vec{q},u)$ over those pool–speed combinations gives the optimal task-type choice and corresponding optimal speed.

5. Simulation

In a real scenario, an AAV may have limited information about its environment. Above, it is assumed that the task-type densities (i.e., the encounter rates) are unknown a priori. Here, the AAV estimates these densities online. Based on the estimates, the explicit solution for the optimal speed and task pool from Section 4 is used by the AAV. We show that this approach quickly converges upon the optimal task-type and speed choice for the environment, which validates the theory and demonstrates its utility in a realistic application.

5.1. Simulation setup

The algorithm presented here can be applied to an autonomous air vehicle (AAV) that searches through an environment for spatially distributed tasks. The search area is represented as a two-dimensional coordinate plane with orthogonal dimensions x_1 and x_2. The rectangular search area of interest and the AAV model

[2] Simple analytic methods for finding roots of polynomials of the fourth order are well known (e.g., Tignol, 2001) and are usually already available in mathematical computer packages.

Table 1
Search area and vehicle parameters.

Parameter	Notation	Value
Search area parameters		
Search area width (along x_1 direction)	W	2000 m
Search area depth (along x_2 direction)	D	2000 m
Vehicle parameters		
Sensor width	w	100 m
Sensor lead distance	d	122 m
Altitude	a	122 m
Maximum angular speed	$\omega_{\max}$	$\frac{1}{6}$ radians/s
Minimum speed	$u_{\min}$	11 m/s
Maximum speed	$u_{\max}$	23 m/s
Linear cost rate coefficient	c_ℓ	$\frac{11}{920}$ points/s per m/s
Constant cost rate coefficient	c_a	$\frac{1}{40}$ points/s
Smoothing filter time constant	τ	5 s

are specified by parameters in Table 1. The vehicle travels at a constant altitude a with a sensor that is triggered when an object crosses a line w wide that is distance d ahead of the vehicle. The vehicle initially starts heading in the increasing x_2 direction with the center of its sensor line positioned at $(x_1, x_2) = (w/2, 0)$. Its search pattern moves it along the x_2 direction for a distance D. When the sensor reaches $x_2 = D$, the vehicle stops searching and turns itself around so that it heads in the decreasing x_2 direction with the center of its sensor a distance w along the x_1 direction from where it stopped searching. It continually repeats the process of traveling straight for a distance D and turning around until its sensor has searched over the entire area. We call this trajectory a lawnmower pattern. It is assumed that the fuel cost for the vehicle is the same when searching as it is processing, and so $c^p = c^s = c$, which is specified by linear and constant coefficients c_ℓ and c_a, respectively. The vehicle is continuously penalized with this fuel cost. The fuel cost coefficients picked here are chosen so that during each straight pass of the search pattern the vehicle suffers a penalty approximately equal to the value of processing a small number of tasks.

The vehicle has a maximum speed of $u_{\max}$ and a minimum speed of $u_{\min}$. The vehicle's speed, orientation angle, and angular speed are denoted by u, θ, and ω, respectively. The maximum angular speed $\omega_{\max}$ effectively gives the vehicle a limited turning radius that varies with speed u. The vehicle's kinematics are given by

$$\begin{cases} \dot{x}_1 = u\cos\theta \\ \dot{x}_2 = u\sin\theta \\ \dot{\theta} = \omega \end{cases} \quad \text{with } |\omega| \leqslant \omega_{\max}. \tag{25}$$

This model was introduced by Dubins (1957) and is often called a Dubin's vehicle. The original model uses a constant velocity u and considers ω to be a single input to the system. Under these assumptions, Dubins (1957) provides a method for generating an ω control to drive the vehicle along a shortest-path trajectory from any one point to any other point. That method states that for any time the desired angular speed $\omega \in \{-\omega_{\max}, 0, \omega_{\max}\}$. That is, when the vehicle is moving along a shortest-path trajectory, it will only move in straight lines and hard turns.

Because some parameters must be estimated online, the vehicle used here must have a time-varying speed u. It is assumed that the vehicle's speed can track such a reference u as long as it is continuous; however, as parameters are estimated, the history of the speeds predicted by the algorithm may not be continuous. Thus, the algorithm's speed history is denoted by u_d which is filtered by a continuous-time first-order linear time-invariant filter with time constant τ given by the kinematics

$$\dot{u} = \tau^{-1}(u_d - u). \tag{26}$$

Hence, u will be a continuous approximation of u_d, and so the vehicle will have the ability to track this sufficiently smooth reference.

In order to simplify the generation of shortest-path trajectories, it is also assumed that the speed u is momentarily held constant whenever the vehicle is not traveling on its straight search path within the search area. So, the vehicle can use the shortest-path methods described by Dubins (1957) to pick its trajectory when moving from the end of one straight pass to the start of the next straight pass. The speed is held constant by setting $u_d = u$ at the end of a straight search pass and then resetting u_d back to the speed choice picked by the algorithm once the vehicle is at the start of the next straight search pass.

There will be four task types in this environment (i.e., $n = 4$). Whenever a task is detected, the vehicle will choose to either process or ignore the task. Processing of a task involves completing a circle at its current location in order to pass its sensor over the task an additional time (e.g., for task classification). The vehicle's speed is held constant during this processing loop and the maximum angular speed $\omega_{\max}$ is used. Processing is complete when the vehicle returns to the position it was in when it detected the task. At that point, the vehicle continues on its original search path. Notation for the important parameters for these types is given in Table 2 and the actual parameter values for each type are given in Table 3. If the vehicle has just processed a task of type i, it is rewarded with g_i^p. Because the vehicle's maximum angular speed $\omega_{\max}$ is fixed, it completes a processing circle in the same amount of time regardless of its speed u. Thus, each task type has the same processing time (i.e., $t_i^p = 2\pi/\omega_{\max}$ for all $i \in \{1, \ldots, n\}$). Under this condition, ordering by profitability is equivalent to ordering by processing gain g_i^p.

For each task type i, the actual number of tasks generated for that type is drawn according to a Poisson random variable with rate parameter N_i. Those tasks are then distributed spatially throughout the environment according to uniform distributions for both the x_1 and the x_2 directions. Thus, over multiple simulation runs, the vehicle will on average find N_i objects in

Table 2
Task-type parameter notation.

Parameter	Notation
Task-type index	i
Average number of tasks of type i in search area	N_i
Average linear density of tasks of type i in search area	D_i
Processing gain of tasks of type i	g_i^p
Processing time of tasks of type i	t_i^p
Probability of detection at $u_{\min}$ for task type i	p_i^{slow}
Probability of detection at $u_{\max}$ for task type i	p_i^{fast}

Table 3
Task-type parameter values.

Task type (i)	N_i	D_i	g_i^p	t_i^p	p_i^{slow}	p_i^{fast}
1	150 tasks	$N_1 w/(WD)$	20 points	$2\pi/\omega_{\max}$	0.8	0.5
2	45 tasks	$N_2 w/(WD)$	110 points	$2\pi/\omega_{\max}$	1.0	0.0
3	200 tasks	$N_3 w/(WD)$	100 points	$2\pi/\omega_{\max}$	0.9	0.2
4	95 tasks	$N_4 w/(WD)$	105 points	$2\pi/\omega_{\max}$	0.82	0.82

the search area for each task type $i \in \{1, \ldots, n\}$ and task encounters will follow a Poisson process.

5.2. Heuristic implementation

It is assumed that the vehicle has knowledge of every parameter except for the linear density D_i of each task type i in the search area. The vehicle initially assumes that $D_i = 0$ for all $i \in \{1, \ldots, n\}$. However, as the vehicle travels it continually updates this estimate by dividing the number of each task type it has encountered up to that time by the linear distance traveled within the search area on straight passes. While estimation could be done continually; we only ensure that an update occurs at every task detection and no more than τ time units since the previous update. At each update, a new task-type choice $\vec{q}$ and desired speed u_d is chosen according to the approach in Section 4. If the update occurs due to a task encounter, then the task-type choice $\vec{q}$ is updated before the choice to process the detected task is made.

5.3. Simulation results

5.3.1. Theoretical predictions

When the algorithm is used with these parameters, it predicts an optimal speed of $u = 12.6239\,\mathrm{m/s}$ with a task pool consisting of the three highest-profitability task types (i.e., $\vec{q} = [0, 1, 1, 1]^{\mathrm{T}}$). So, the average net rate of point gain is 1.9177 points/s.

5.3.2. Model verification

Fig. 2 shows the result of averaging 50 runs of the simulation at each of 35 speeds uniformly distributed between $u_{\min}$ and $u_{\max}$. The number of simulation runs used here and again later was chosen to ensure convergence of the first- and second-order statistics. The vehicle used in these simulations always processes tasks of the three task types of highest profitability (i.e., types 2–4) and ignores tasks of the lowest profitability task type (i.e., type 1). Each circular marker shows the net rate of point gain averaged over the 50 runs at each of the 35 speeds. Each of the square markers show the average net rate of point gain after fuel cost and additional time due to moving from the end of one search pass to the start of the next search pass are removed. That is, the curve made up of square markers is an idealized version of the curve made of circular markers. Both of these average curves are shown with standard error bars. The final curve made up of triangular markers shows the expected curve from the theory. The dashed vertical line labeled with a "u^*" shows the optimal speed predicted from the theory.

Clearly the theoretical average net rate of point gain curve matches the simulated average net rate of point gain curve provided that time and fuel losses from between-pass travel are removed. While the value of the actual average net rate of point gain is different from the idealized version, the shapes are very similar and thus the optimal speed for this particular task pool (i.e., $\vec{q}$) will still be a good choice. These observations hold in this simulation for every task pool.

In the case when the difference between the idealized and actual average curves is large, the wrong task pool may be chosen. This case will only occur when the point gains resulting from the optimal speed choice for each task pool are very close together with respect to the difference between the idealized and actual average curves. Thus, in cases when the difference between the curves is small, picking the wrong task pool will not have much of a detrimental effect. To ensure that this difference is small, the amount of extra time spent between search passes should be made very small with respect to the total time spent searching and processing. In this particular simulation, it is sufficient to increase the depth D of the search area. To compensate for the extra area the vehicle will need to cover before completing its mission, the width W of the search area can be decreased.

5.3.3. Simulation of limited-information case

Figs. 3 and 4 show the aggregate result of 50 simulation runs using the heuristic algorithm that estimates densities online and makes speed and task pool choices based on those estimates. Fig. 3 is shown with standard error bars. The stem plots in Fig. 4 represent the proportion of the total simulation runs where the vehicle chose to include each task type at any given time. Both of these have been decimated and truncated for clarity. Again, 50 simulation runs were used to ensure convergence of the first- and second-order statistics.

The statistics on the speed picked by the algorithm quickly converge to 12.6630 m/s with a standard deviation of 0.2666 m/s, which is very close to the theoretical optimal speed choice. Similarly, the inclusion proportion trajectories show that after a short time every simulation picks the expected optimal task pool

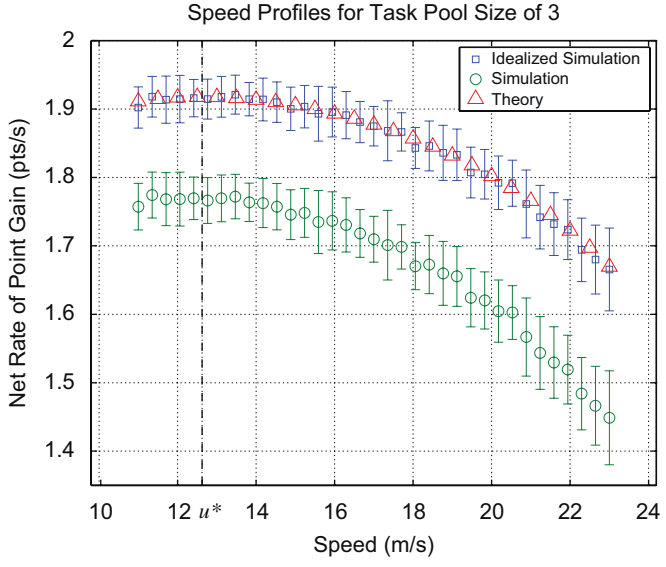

Fig. 2. Average net rate of point gain over speed.

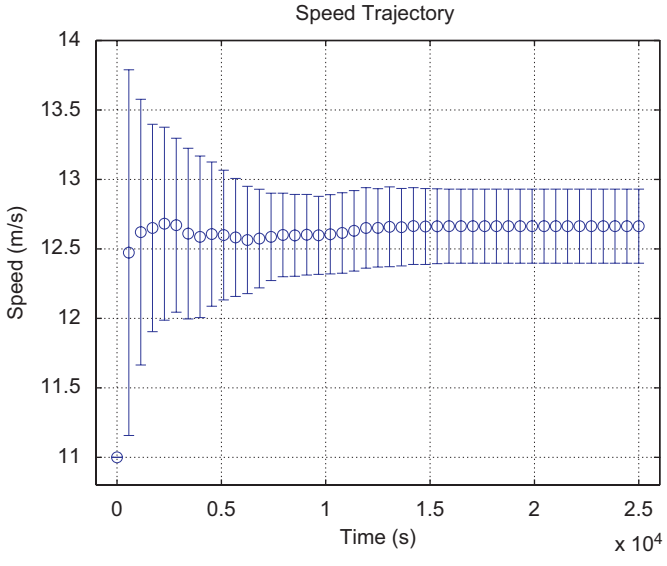

Fig. 3. Average speed trajectory.

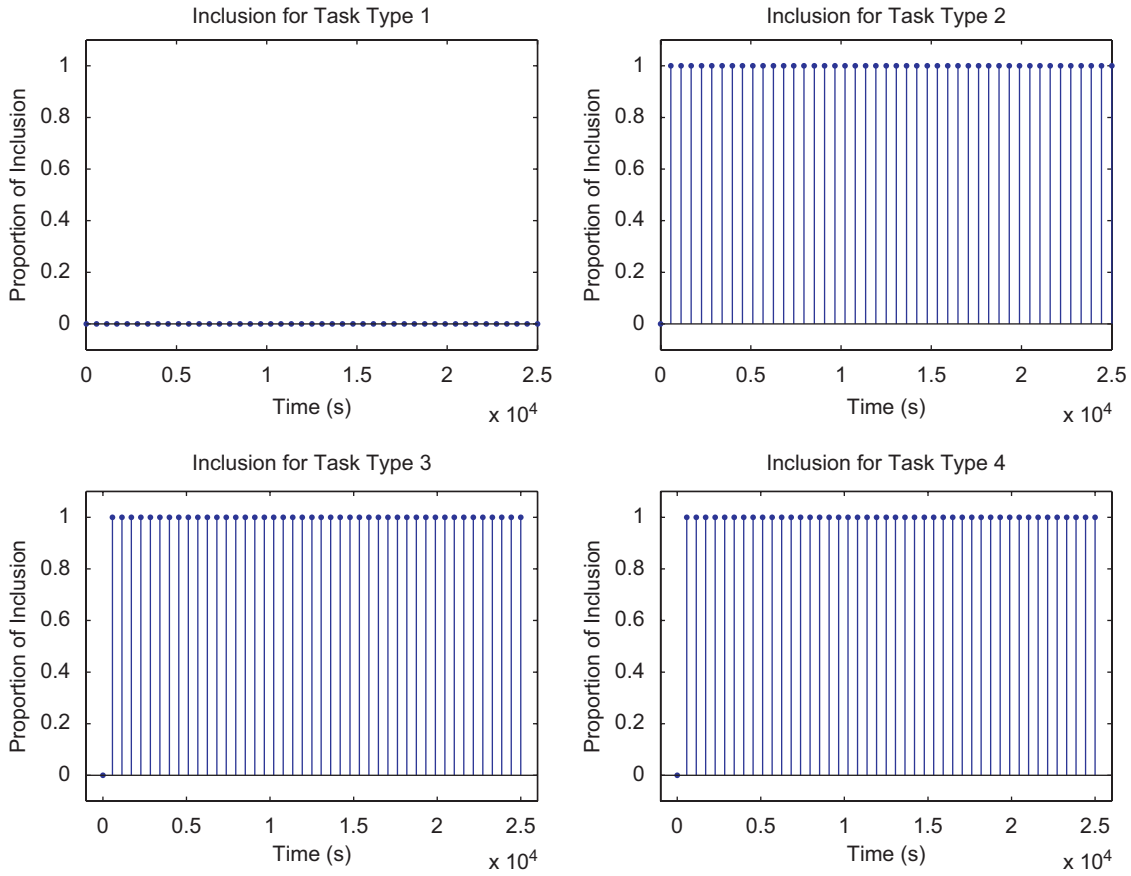

Fig. 4. Inclusion proportion trajectory.

for inclusion. The average net rate of point gain over these 50 simulation runs is 1.7677 points/s with a standard deviation of 0.0274 points/s; however, when adjusting for the extra time and fuel spent between search passes, the resulting average is 1.9157 points/s with a standard deviation of 0.0240 points/s. This result is also close to the value expected by the theory. In other words, the simulations converge so quickly that the heuristic implementation performs almost as well as an omniscient case.

6. Conclusions

Methods inspired by foraging theory can be used in the design of decision-making strategies for autonomous vehicles. We have shown that the classic prey model can be enhanced for engineering applications to include the impact of speed-dependent sensor limitations as well as speed-dependent fuel cost. The enriched prey model can be used to predict the optimal task-type choice policy and speed. When there is limited information (e.g., when the density of tasks is unknown) the information can be estimated online to perform online optimization of task-type choice and speed. We showed that this heuristic method can be applied directly to an autonomous air vehicle responsible for surveillance of a given area.

There are a number of promising directions for additional work. This work uses simple affine models of the speed dependence in detection probability and fuel cost. There is significant opportunity for future work to use more general models. Not only could more complex models be used, but models that change over time (e.g., as the AAV adjusts how it processes information from its sensors) could also be used. Additionally, the speed–accuracy and speed–cost trade-offs could also be examined in the classical patch model in foraging theory, which predicts when a forager should leave a patch of diminishing returns to search for a new patch. Of course, other numeric algorithms (e.g., genetic algorithms) can already be used to find the optimal AAV policy for these more complicated cases. However, simple algorithms informed by an analytical approach should be more efficient especially when being implemented online. Additionally, analytical solutions give insight into how the optimal solution will change as environmental parameters (e.g., task-type densities) change.

The impact of speed dependence on other foraging models (e.g., risk-sensitivity, recognition cost) could also be investigated in a similar fashion. Finally, to verify the validity of biologically inspired design methods (e.g., rate as opposed to efficiency maximization), there is a need for more examples of ideas from foraging theory being applied to actual engineering problems. In particular, the implementation of these algorithms in a mobile robot testbed would be useful to further evaluate the practical value of the ideas presented here (e.g., as Quijano et al., 2006 showed the practical value of the standard prey model for an important engineering application).

Acknowledgments

We thank Professor Thomas A. Waite of the OSU Department of Evolution, Ecology, and Organismal Biology, for teaching us about behavioral ecology and for stimulating discussions on the use of that theory in engineering applications. We also appreciate the helpful comments of an anonymous reviewer.

References

Andrews, B.W., Passino, K.M., Waite, T.A., 2004. Foraging theory for decision-making system design: task-type choice. In: Proceedings of the 43rd IEEE Conference on Decision and Control, vol. 5, pp. 4740–4745.

Andrews, B.W., Passino, K.M., Waite, T.A., 2007. Social foraging theory for robust multiagent system design. IEEE Transactions on Automation Science and Engineering 4 (1), 79–86.

Bertsekas, D.P., 1995. Nonlinear Programming. Athena Scientific, Belmont, MA.

Charnov, E., 1976a. Optimal foraging: attack strategy of a mantid. American Naturalist 110 (971), 141–151.

Charnov, E., 1976b. Optimal foraging: the marginal value theorem. Theoretical Population Biology 9, 129–136.

Charnov, E., Orians, G.H., 1973. Optimal foraging: some theoretical explorations. Ph.D. Thesis, University of Washington.

Dubins, L.E., 1957. On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal positions and tangents. American Journal of Mathematics 79, 497–516.

Gendron, R.P., 1982. The foraging behavior of bobwhite quail searching for cryptic prey. Ph.D. Thesis, Duke University.

Gendron, R.P., Staddon, J., 1983. Searching for cryptic prey: the effect of search rate. American Naturalist 121 (2), 172–186.

Houston, A., McNamara, J., 1999. Models of Adaptive Behavior. Cambridge University Press, Cambridge.

Passino, K., 2002. Biomimicry of bacterial foraging for distributed optimization and control. IEEE Control Systems Magazine 22 (3), 52–67.

Passino, K., 2005. Biomimicry for Optimization, Control, and Automation. Springer, London.

Pirolli, P., 2005. Rational analyses of information foraging on the web. Cognitive Science 29 (3), 343–373.

Pirolli, P., 2007. Information Foraging Theory: Adaptive Interaction with Information. Oxford University Press, New York.

Pirolli, P., Card, S., 1999. Information foraging. Psychological Review 106 (4), 643–675.

Pyke, G., Pulliam, H., Charnov, E., 1977. Optimal foraging: a selective review of theory and tests. Quarterly Review of Biology 52 (2), 137–154.

Quijano, N., Andrews, B.W., Passino, K.M., 2006. Foraging theory for multizone temperature control. IEEE Computational Intelligence Magazine 1 (4), 18–27.

Rowcliffe, J.M., de Merode, E., Cowlishaw, G., 2004. Do wildlife laws work? Species protection and the application of a prey choice model to poaching decisions. Proceedings of the Royal Society of London. Series B, Biological Sciences 271 (1557), 2631–2636.

Schoener, T., 1971. Theory of feeding strategies. Annual Review of Ecology and Systematics 2, 369–404.

Stephens, D., Krebs, J., 1986. Foraging Theory. Princeton University Press, Princeton, NJ.

Tignol, J.-P., 2001. Galois' Theory of Algebraic Equations. World Scientific, Singapore.