To appear in *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*
St. Louis, Mo, USA, Oct. 11-15, 2009

Robot Task Switching under Diminishing Returns

Jens Wawerla† and Richard T. Vaughan‡

***Abstract*— We investigate the problem of a robot maximizing its long-term average rate of return on work. We present a means to obtain an estimate of the instantaneous rate of return when work is rewarded in discrete atoms, and a method that uses this to recursively maximize the long-term average return when work is available in localized patches, each with locally diminishing returns. We examine a puck-foraging scenario, and test our method in simulation under a variety of conditions. However, the analysis and approach applies to the general case.**

I. INTRODUCTION

The purpose of a robot is to perform work [8] and thus earn some reward that justifies its human owner's investment. Usually more work gives more reward, but the instantaneous rate of reward may either be independent of the amount of work done previously, e.g. I sell apples at $1 each, no matter how long my shop has been open; or it may be dependent, e.g. picking the last few apples from a tree is more time-consuming than the first few apples. This latter 'diminishing returns' is characteristic of many robot foraging tasks, such as mining, de-mining, and collecting fungus or trash.

In a previous paper [19] we investigated optimal task switching between heterogeneous tasks with constant reward gain rate. In this paper we analyze robot behaviour for homogeneous tasks that exhibit diminishing returns. We use foraging as an example task, but our analysis and approach applies to any tasks subject to local diminishing returns.

Foraging is well-studied in behavioural ecology [13], [12] and robotics. Liu [7] and Ulam [15] examine dynamic allocation of the optimal number of workers to a given foraging task. Østergaard [10], Shell [11] and Lein [6] explore methods to reduce interference between foraging robots by separating them in space in order to improve the system's performance.

Almost all the previous work examines role allocation and interference reduction in collaborative multi-robot systems. It examines *central place foraging* [12], where foraged items are delivered to a single privileged location, e.g. a nest. Very little work has been done on high-performance solitary foraging in robots, though this is well-studied in animals.

An exception, and the most similar previous work, is by Andrews et al. [1] which explores the use the Marginal-Value Theorem (see Section I-A) for task switching. The experiments in this paper are grounded in a complete low-level robot controller in a sensorimotor simulation rather than the mathematical models of [1]. Another important technical difference is mentioned below.

Simon Fraser University, School of Computing Science
Burnaby, BC, V5K 1S6, Canada
†jwawerla@sfu.ca
‡vaughan@sfu.ca
This work was supported by NSERC and DRDC in Canada.

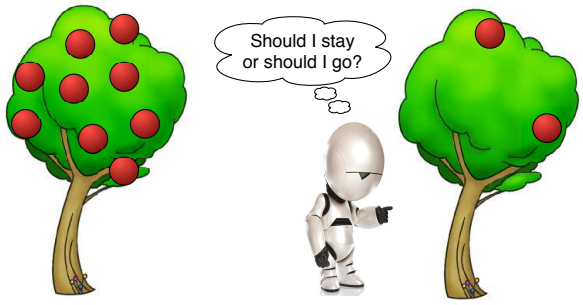


Fig. 1. The robot must maximize global collection rate in an environment of multiple work sites, each with locally diminishing returns.

In this paper we consider a single robot foraging for atomic units of resources that are consumed (and reward obtained) as they are encountered, instead of delivered centrally. This models self-feeding, for example. Crucially, units of resource are not distributed uniformly, but exist in regions of locally high density known in the biology literature as *patches*. Danchin [4] defines a patch as "an homogeneous resource containing area (or part of habitat) separated from others by areas containing little or no resources". The advantage of patches is two-fold: (1) patches give the forager *a priori* information about the likelihood of finding resources. For example we expect to find valuable apples only on apple trees, and not on the ground; (2) patches reduce the complexity of the decision process: instead of making decisions about each apple, it is sufficient to only make decisions about each patch. As the number of patches is usually significantly smaller than the number of resource units, we can expect reduced complexity of decision-making.

However, considering patches instead of atomic units of resource introduces the issue of the reward rate per patch, usually called *patch quality*. This may not be known *a priori*, and may require the forager to forage in the patch for some time to determine a good estimate of the patch quality. This sampling cost inevitably harms overall performance.

The task illustrated in Fig. 1 has the properties of interest. A robot collects apples in an orchard, and is rewarded as it collects each apple. As a tree (patch) is depleted of apples, the marginal cost of picking the next apple increases. How does the robot decide to abandon the current tree and move to the next, such that the overall rate of apple collection is maximized?

A. Marginal-Value Theorem

Charnov [2], [3] proposed the Marginal-Value Theorem (MVT) to model the decision faced by an animal foraging

in a patchy environment. The MVT says a rate-maximizing forager should leave the current patch once the marginal gain rate equals the long-term average rate of the habitat. Charnov's mathematical derivation gives two results: (1) the marginal rate of leaving must be the same for all patches in a habitat; and (2) as the cost to switch patches increases the forager should exploit each patch longer.

The simplicity of the rule makes it very appealing, but the theorem and its validity has also been widely and controversially discussed, for example by [5], [9], [13]. It has been argued that the rule is circular in that the long-term average needs to be known in order to select the best leaving threshold. But the choice of the leaving threshold influences the long-term average. We think the rule is clearly circular, in fact we will exploit this circular dependency between leaving threshold and long-term average gain rate in our robot controller. Another important issue often pointed out is that the MVT uses the instantaneous gain rate. But how does the forager measure the instantaneous rate when resources are found in discrete lumps? We will show one possible solution below.

The MVT is not the only patch-leaving rule proposed by behavioural ecologists. A number of similar rules have been discussed by [18], differing in the parameter used to make the patch-leaving decision. Candidates are (1) the total patch residence time; (2) the time since the last item was encountered; and (3) the number of collected items. We favour the instantaneous gain rate because it has some important practical advantages over the other proposed parameters. For example, if the forager encounters an exceptionally low quality (e.g. empty) patch, a patch-leaving policy based on instantaneous gain rate would leave the patch quickly (the desired behaviour), where a constant time policy would stay for the whole length of the residence interval and a policy based on the number of collected items would cause the forager to never leave the patch.

Below we present the robot control policy and demonstrate its effectiveness in series of simulation experiments.

II. ROBOT CONTROLLER

We use a generic mobile robot model in the well known simulator Stage[16]. It is equipped with a short-range colour blob tracker to sense pucks, our unit of resource. The robot knows (or equivalently can detect) the boundaries of puck patches. Patches are 620 times the size of the robot, and contain 10, 30 or 50 uniform randomly placed pucks. A minimum distance between pucks is enforced to avoid overlap. To exploit a patch, the foraging robot can use one of two foraging policies:

1) Under the **random foraging policy** the robot drives straight until it comes to the patch boundary, where it chooses a new heading that brings it back into the patch, at random. When pucks are detected, the robot servos towards the closest puck and collects it.
2) Under the **systematic foraging policy** the robot employs a regular square-wave-like search pattern from one side of the patch to the other side. The distance between legs in the pattern is small enough to guarantee that the whole patch is searched.

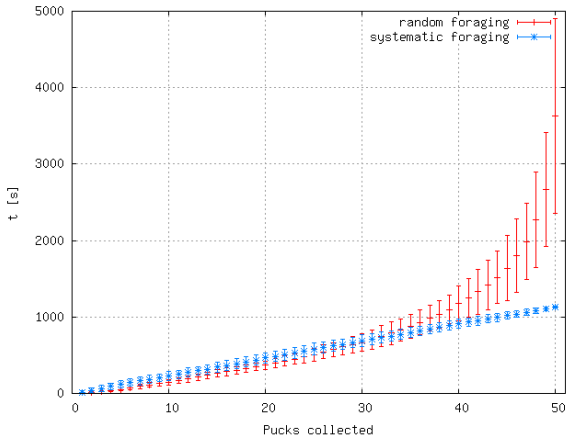


Fig. 2. Time required to collect all 50 pucks in a patch using random search and systematic search.

These policies exhibit different reward dynamics. To illustrate this we tested each policy on 100 patches of 50 uniform random placed pucks each. Fig. 2 shows the average time required to collect a certain number of pucks under the two policies. The random foraging policy suffers from diminishing returns, since over time it is increasingly more likely to re-visit previously cleared, now unproductive, parts of the patch. The systematic forager has an approximately constant collection rate. While we might prefer the systematic forager because of this property, it may be much more costly to implement than the random forager. Real-world low-cost robots like the iRobot Roomba use a randomized method that suffers from diminishing returns.

The point here is not to choose the best policy, but to illustrate that low-level robot control policy can fundamentally influence the nature of the overall optimization task, and to show that single-robot random foraging is subject to diminishing returns and thus provides a suitable model for all tasks of this domain.

Given this randomized foraging policy, the robot needs to determine when to abandon the current patch and move on. As suggested by Charnov's MVT [2], our robot abandons a patch once the instantaneous gain rate drops below a threshold. To do this we must (1) determine the instantaneous gain rate of atomic items (pucks) encountered at every time-step and (2) select a leaving threshold that maximizes the long-term average gain rate.

A. Instantaneous Gain Rate

Measuring the instantaneous rate of atomic events is impossible. [1] suggest calculating the slope using the current and the previous two gain function values. This approach may work for continuous gain function, but fails in situations where the gain function is non-continuous and stochastic such as in unit-based foraging in uniform random patches. So we resort to an alternative representation. We use the expected value of a beta distribution over time-steps in which the robot found a puck and those in which it did not.

Algorithm: forage(θ)
init ϕ_p, ϕ_q
$t = 0$
$p(0) = \phi_p$
$q(0) = \phi_q$
$\hat{\lambda}_{filt}(0) = \frac{p(0)}{p(0)+q(0)}$
repeat
  $t = t + 1$
  randomly forage for one time-step
  if *puck found* **then**
    $p(t) = p(t-1) + 1$
  else
    $q(t) = q(t-1) + 1$
  end
  $\hat{\lambda}(t) = \frac{p(t)}{p(t)+q(t)}$
  $\hat{\lambda}_{filt}(t) = \hat{\lambda}_{filt}(t-1) + k_1(\hat{\lambda}(t) - \hat{\lambda}_{filt}(t-1))$
until $\hat{\lambda}_{filt}(t) < \theta$
return (t, p)

Algorithm 1: Forage in the current patch until a proxy for the instantaneous rate $\lambda(t)$ drops below the threshold θ

Equation 1 gives this expected value of the beta distribution which we use as a proxy for the instantaneous rate $\hat{\lambda}_i(t)$.

$$\hat{\lambda}_i(t) = \frac{p_i(t)}{p_i(t) + q_i(t)} \tag{1}$$

Where i refers to the i-th patch, t is the amount of time spent in a patch, $p_i(t)$ is the number of time-steps in which the robot collected a puck and $q_i(t)$ is the number of time-steps during which the robot did not collect any pucks. This method has an interesting problem: until the first puck is found, the number of time-steps in which the robot collected pucks is zero ($p_i = 0$), and thus our estimate of the rate is zero ($\lambda_i = 0$). Hence the robot will immediately leave the patch. We avoid this by initializing $p_i(0)$ and $q_i(0)$ with a prior ϕ_p and ϕ_q respectively. These priors represent the robot's expectation of the initial gain rate of a patch. They can either be set to an environmental constant (if known) or they can be determined at run time by experience from the previous patch. The ratio of the priors represents the expected initial gain rate, the magnitude of each is an indication of the forager's confidence in these priors. The higher the magnitude the more experience is required to modify λ, i.e. change the robot's belief about the prevailing rate.

Because puck encounter is a random and infrequent process, the value of $\hat{\lambda}_i(t)$ is very variable in practise, particularly for small values of $p_i(t)$ and $q_i(t)$. We apply a low-pass filter to smooth this slightly. These steps are combined into Algorithm 1.

B. Patch-Leaving Threshold

Recall that the MVT predicts that a forager should leave a patch once λ drops to the environment's global average. In the general case it is impossible for the robot to know this true global average rate. All the robot can measure directly is the long-term average gain rate it experiences: a value which depends on the robot's past behaviour. The experienced average gain rate $\mu(\theta)$ for foraging in n patches is given by

$$\mu(\theta) = \frac{\sum_{i=1}^{n} g_i(\theta)}{\sum_{i=1}^{n} (t_i + \tau_i)} \tag{2}$$

where $g_i(\theta)$ is the gain function that gives the total number of pucks collected in patch i when the patch is abandoned according to Algorithm 1, t_i is the time spent in the i-th patch, and τ_i is the patch switching duration. The objective of the forager is to maximize μ by selecting θ.

$$\theta^* = \text{argmax}(\mu(\theta)) \tag{3}$$

Unfortunately $g_i(\theta)$ is unknown and difficult to obtain. The function depends precisely on the robot's sensorimotor interaction with the environment, the patch quality and the distribution of pucks in the patch. Hence a closed–form solution of eq. 3 is not obtainable. Fig. 3 shows the long-term average gain rates over a range of values of θ for different environmental conditions. From these graphs we can see that the observed average gain rate varies widely under different circumstances. The variance in average gain rate observed is high, but error bars are omitted for clarity.

Since we only have an approximation of the instantaneous gain rate, we cannot, as the MVT requires, leave a patch once this rate drops to the long-term average. Instead we resort to maximizing the long-term gain rate by recursively improving the leave threshold θ based on previous experience.

In practise we found that the observations have such high variance (evident in the errorbars of Fig. 2), that local gradients are not very useful and thus simple gradient descent methods resulted in poor performance. A more sophisticated heuristic approach is required in our scenario.

Action-Value methods have been shown to be effective for n-armed bandit problems; maximizing the return of a discrete set of unknown process. Sutton and Barto [14] give a overview of various action-value methods and Vermorel et al. [17] give an empirical comparison. We can adapt this framework to our continuous action-space by finding a discrete set of values of θ that approximate the input-output mapping of the true gain function. We continuously refine this set by exploring θ space, while frequently using the current best estimate of θ^* to obtain good performance as we go.

Since no generally optimal method is known, we use here a combination of *softmax* and *ε-first* adapted for continuous action-spaces. A comparative analysis of the various possible algorithms is beyond the scope of this paper. The complete method is shown as Algorithm box 2. The parameters θ_{min} and θ_{max} are the minimum and maximum patch leaving thresholds. These are determined by the environment and the robot specification, e.g. the maximum travel speed. N denotes the number of bins into which the action-space is discretized. More bins potentially allow a better approximation, at the expense of longer start-up time. Constant k_1 is used to smooth the weight update, k_2 is the *temperature* for *softmax*

1 **Algorithm:**adaptive()
2 **init** θ_{min},θ_{max}, N, k_1, k_2, k_3
3 $\delta = \frac{\theta_{max}-\theta_{min}}{N}$
4 **for** i=1 *to* ∞ **do**
5 **if** $i \leq N$ **then**
6 $a = i$
7 $\theta = i \cdot \delta$
8 **else**
9 draw a with probability $\propto \frac{e^{w(a)/k_2}}{\sum_{j=1}^{N} e^{w(j)/k_2}}$
10 draw θ uniform randomly from $[\Theta(a) - k_3\delta, \Theta(a) + k_3\delta]$
11 **end**
12 (t, p) = forage(θ)
13 $\mu = \frac{p}{t+\tau}$
14 **if** $i \leq N$ **then**
15 $\Theta(a) = \theta$
16 $w(a) = \mu$
17 **else**
18 $\Theta(a) = \frac{w(a)\cdot\Theta(a)+\mu\cdot\theta}{w(a)+\mu(a)}$
19 $w(a) = w(a) + k_1(\mu - w(a))$
20 **end**
21 **end**

Algorithm 2: Adaptive leave rate selection

action selection and k_3 determines how much we are willing to modify the leave rate threshold between adjacent patches.

To initialize our discrete approximation of the gain function we must fill the N bins. They can be initialized with prior expected values if available, but here we obtain them empirically with a start-up phase. For the first N patches visited we use the discretized leave rate threshold (lines 5-7 in ε-*first*). Here $1 \leq a \leq N$ denotes the bin number. $\Theta(a)$ describes the centre of bin a and $w(a)$ its weight. For the remaining patches we select a bin using *softmax* (line 9). The leave rate threshold θ is uniform randomly chosen from an interval centred around the centre of the bin (line 10).

The robot now forages with the new leave rate threshold θ using Algorithm 1. This results in some patch residence time t and in p pucks collected. Therefore we can calculate the average gain rate for the patch μ as the number of pucks collected divided by the sum of patch residence time and patch switching time τ (line 13). Next we update the weights and the centre of the bin. For the first N patches the weight is simply the average gain rate μ. The centre of the bin is the leave rate threshold used (lines 15-16). For the remaining patches the centre of the weight is the moving average of the average gain rate and the centre of the bin is the weighted sum of the previous centre and the just explored leave rate threshold (lines 18-19), where the previous centre is weighted by the weight of the bin and the latest leave rate threshold is weighted with the average gain rate that resulted from the use of this threshold.

Summarizing the action of Algorithm 2: *Softmax* selects the best bin in a greedy fashion while it keeps exploring the other bins with probability proportional to the weight of each bin. Since the weight is an estimate of the expected yield rate, it explores higher paying areas of the action-space more frequently then lower paying areas. The discretization step might choose bin centres that are suboptimal; to over come this problem we perform a local random walk of the bin centres. Over time this will move the bin centres towards better values.

III. EXPERIMENTS

To investigate the effectiveness of our approach, we conducted a series of simulation experiments consisting of two phases (1) generate foraging data and (2) test our adaptive task (patch) switching policy on the generated data.

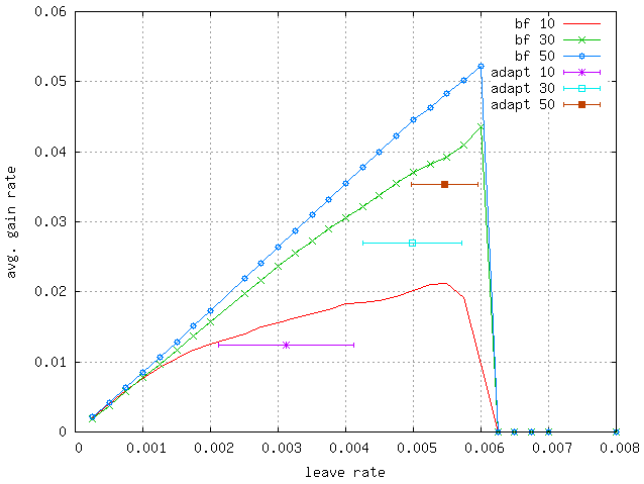
To generate the foraging data we used the method described in Section II. For each of the three puck qualities (10, 30, 50 pucks per patch) we generated 100 samples and recorded for each time-step whether the robot collected a puck or not. Note that in this phase no patch-leaving decisions are made, we just recorded data while the robot simply collected pucks until each patch was exhausted.

In the second phase we ran our adaptive patch switching policy on the recorded data and compared it against the best solution obtained by exhaustive search over the same data set. Decoupling the data generation from the analysis of the control policy allowed us to cancel any noise in the performance analysis caused by the random generation of the patches. It also made brute-force search for approximately optimal solutions feasible.

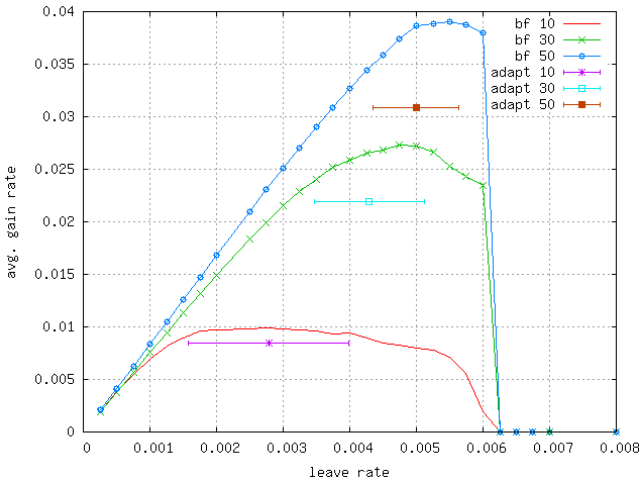
To obtain a benchmark with which to compare our online adaptive method, we choose a finite discrete set of rate thresholds over a range of θ between our preset maximum and minimum. For each threshold we had the robot forage in each patch until the instantaneous rate dropped to that selected leave rate threshold (Alg. 1). On leaving the patch, the robot takes some time in which no pucks are collected before arriving at the next patch: the patch–switching cost. This way we found the leave rate that maximizes the long-term average gain rate for the recorded data. Fig. 3 shows the long-term average gain rate over the leave rate threshold for patches of different quality and different switching costs. The prediction of the MVT that the patch residence time should increase with increasing switching cost is clearly visible. The graphs also give an idea of the search space for the adaptive algorithm. Especially low switching costs (fig. 3(a)) exhibit a sharp performance peak which is difficult to adapt to, in this situation it is desirable to stay on the side with the smaller slope. Patch quality is one factor that determines the environmental gain rate. The other is the average switching time between patches. To investigate the influence of the switching time τ we analyzed the system with 4 different switching times, $\tau = \{10, 100, 500, 1000\}$ seconds.[1].

The performance of the adaptive method is presented as mean and standard deviation of observed gain of 20 trials of 100 patches each, compared to the estimated optimum

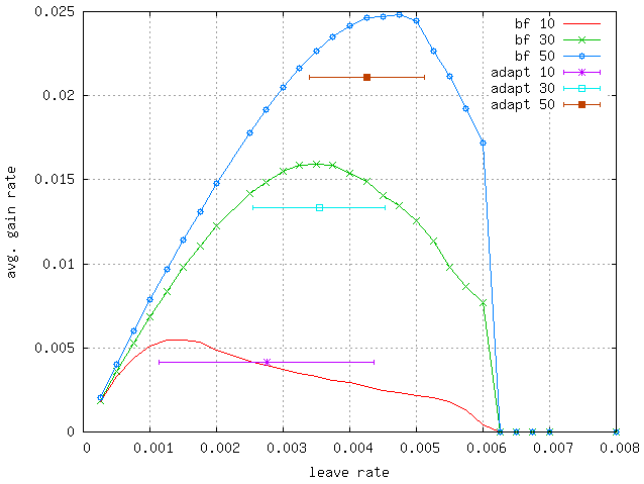
[1]The graph for switching cost 1000 sec. in Fig. 3 was omitted due to space constraints, but it is qualitatively similar to that with cost 500



(a) Average gain rate with patch switching time 10 sec.



(b) Average gain rate with patch switching time 100 sec.



(c) Average gain rate with patch switching time 500 sec.

Fig. 3. Long-term average gain rate observed versus leave rate threshold for different patch switching cost and patch qualities (errorbars in y-axis omitted to improve readability). The sharp drop-off at $\tau = 0.006$ occurs because the leave threshold is too high for the robot-environment interaction, the robot leaves a patch before it encounters the first puck

TABLE I

ADAPTIVE PATCH SWITCHING AND RANDOM FORAGING

	τ [s]							
	10		100		500		1000	
pucks	μ[%]	σ	μ[%]	σ	μ[%]	σ	μ[%]	σ
10	60.2	3.72	84.6	2.68	76.7	3.40	66.6	3.89
30	62.0	0.76	79.9	2.20	82.8	2.75	81.4	2.03
50	68.3	0.64	79.3	1.11	85.4	1.60	86.3	1.38

obtained by exhaustive search. All algorithm parameters required were set manually and kept constant without attempting to optimize them, except in one case. The confidence of the priors for the instantaneous gain rate were lowered for patches with 10 initial pucks, so that the estimate of the gain rate would change faster. The original values gave poor performance in this challenging scenario.

Table I shows the results of the first experiment in which we analyzed the adaptive patch switching policy for 3 different patch qualities and 4 different switching costs. The results are given in percentage of the bruteforce long-term gain rate. The small standard deviation of about 2.5% indicates that the system performs fairly consistently despite the high noise levels in the data. Compared to the brute-force method we expect a lower performance since the system, like any ε-greedy method, has to trade off between exploration and exploitation. Our method performs a fixed number ($N = 10$) of initial exploration trials and subsequently performs continuous exploration with a small probability. Especially during the initial exploration we expect some trials with very poor performance because the whole action-space is sampled. This does harm the overall performance.

The actual performance appears quite good: between 75 and 85% of the brute-force benchmark solution as long as the patch switching cost is not too small. The performance drops with the very small patch switching cost of 10 seconds. As Fig. 3(a) indicates, this is a challenging situation since the gain rate function has a very sharp peak. Another reason why this is a difficult situation is that the small switching cost requires the forager to leave the patch very early. For example the best threshold found by brute-force causes the robot to stay for only an average of 35 seconds in the 10 puck patch. Any error the adaptive method makes is relatively large compared to the small switching cost.

Nevertheless as Fig. 3 shows, the mean of the leaving threshold selected by the adaptive system is usually near the optimal threshold and always on the side of caution by being on the side of the smaller slope.

In a second experiment we investigated the performance of the system when the patch switching time varies from one patch to the next. For this experiment we drew patch switching times from a normal distribution with a mean of 100 seconds and variance of 30, 50 and 80 seconds respectively. As in the previous experiment we obtain a near-optimal threshold by brute-force search as a benchmark and compare it with the average of 20 instances of the adaptive system. Table II shows the results. The performance is around 80% for all situations with low deviation, indicating that the

TABLE II

ADAPTIVE PATCH SWITCHING AND RANDOM FORAGING UNDER VARYING PATCH SWITCHING TIMES WITH MEAN 100 SEC

	switching cost variance							
	0		30		50		80	
pucks	μ[%]	σ	μ[%]	σ	μ[%]	σ	μ[%]	σ
10	84.6	2.68	82.6	3.30	79.5	4.79	80.0	2.97
30	79.9	2.20	80.0	1.48	79.5	2.55	80.8	3.01
50	79.3	1.11	79.5	1.16	79.9	0.90	80.9	1.18

TABLE III

ADAPTIVE PATCH SWITCHING FOR RANDOM FORAGING UNDER CHANGING PATCH QUALITY AND SYSTEMATIC FORAGING

	τ [s]							
	10		100		500		1000	
pucks	μ[%]	σ	μ[%]	σ	μ[%]	σ	μ[%]	σ
30→50	82.0	2.87	92.8	3.02	90.9	2.92	90.8	1.67
50→30	80.0	2.44	90.7	2.44	91.3	1.50	89.3	1.50
10	55.2	4.54	81.2	6.32	84.4	6.50	83.4	7.03
30	80.7	3.46	80.5	3.76	79.1	3.89	79.2	2.05
50	76.5	1.80	80.1	1.49	84.9	2.64	83.7	2.79

system is not sensitive to variance in the switching cost.

To analyze the system under changing patch quality conditions we set up two simulations in which patch quality changes from 30 pucks per patch for the first 100 patches, to 50 pucks per patch for another 100 patches and vice versa. We ran these simulations with a patch switching cost of 10, 100, 500 and 1000 seconds respectively. We compare the performance of our method with that observed when switching between the best fixed thresholds for 30 and 50 pucks obtained by brute force search. The results are shown in the top two rows of table III. The data suggest that the system handles the patch quality switching well. The performance seems slightly better than in the stationary situation in table I. This is probably due to a relatively longer exploitation duration: the cost of the initial 10 patch exploration is amortized over 190 patches rather than 90. Despite that fact that the bruteforce method is given the advantage of actually knowing when the patch quality switch occurs, something our method has to detect and adapt to, the system performs very well.

In a last experiment we investigated the adaptive method's performance in situation in which the forager searches each patch systematically. Recall that under this foraging policy the instantaneous gain rate is constant, that is until the patch is empty and the gain rate drops to zero. In this situation the robot has not to decide which leave rate maximizes the overall reward but only to determine when the patch is empty. To see how our system copes with this situation we compared the brute-force and the adaptive method when robots forage patches using the systematic foraging policy. As the last three rows of table III shows the system can also handle situations in with the instantaneous gain rate is given by a step-function.

IV. CONCLUSION

In this paper we investigated the problem of a robot maximizing its long-term average rate of return on work. We presented a means to obtain an estimate of the instantaneous rate of return when work is rewarded in discrete atoms, and suggested one way to use this to recursively maximize the long-term average return when work is available in localized patches, each with locally diminishing returns.

We examined a puck-foraging scenario, and tested our method in simulation under a variety of conditions. However, the analysis and approach applies to the general case.

The validity and applicability of the Marginal-Value Theorem to animal behaviour is widely and controversially discussed in the behavioural ecology literature. Here we have provided evidence that the underlying idea of making task switching decisions based on thresholding the instantaneous gain rate is a valid approach, at least for artificial systems. Whether biological systems make decisions on this principle is an open question.

REFERENCES

[1] B. W. Andrews, K. M. Passino, and T. A. Waite. Foraging theory for autonomous vehicle decision-making system design. *J. of Intelligent and Robotic Systems*, 49:39–65, 2007.

[2] E. L. Charnov. Optimal foraging: Attack strategy of a mantid. *The American Naturalist*, 110:141–151, 1976.

[3] E. L. Charnov. Optimal foraging, the marginal value theorem. *J. of Theo. Biology*, 9(2):129–135, 1976.

[4] É. Danchin, L.-A. Giraldeau, and F. Cézilly, editors. *Behavioural Ecology*. Oxford University Press, 2008.

[5] R. F. Green. Stopping rules for optimal foragers. *The American Naturalist*, 123(1):30–43, January 1984.

[6] A. Lein and R. T. Vaughan. Adaptive multi-robot bucket brigade foraging. In *Proc.of the 11th Int. Conf. on the Simulation and Synthesis of Living Systems*, 2008.

[7] W. Liu, A. F. T. Winfield, J. Sa, J. Chen, and L. Dou. Towards energy optimization: Emergent task allocation in a swarm of foraging robots. *Adaptive Behavior*, 15(3):289–305, 2007.

[8] D. J. McFarland and E. Spier. Basic cycles, utility and opportunism in self-sufficient robots. *Robotics and Autonomous Systems*, 20:179–190, 1997.

[9] J. M. McNamara. Optimal patch use in a stochastic environment. *Theo. Population Biology*, 21(2):269–288, 1982.

[10] E. Østergaard, G. S. Sukhatme, and M. J. Matarić. Emergent bucket brigading - a simple mechanism for improving performance in multi-robot constrained-space foraging tasks. In *Proc. of the Int. Conf. on Autonomous Agents*, pages 29–30, 2001.

[11] D. A. Shell and M. J. Matarić. On foraging strategies for large-scale multi-robot systems. In *Proc. of the Int. Conf. on Intelligent Robots and Systems*, pages 2717–2723, 2006.

[12] D. W. Stephens, J. S. Brown, and R. C. Ydenberg, editors. *Foraging - Behavior and Ecology*. University of Chicago Press, 2007.

[13] D. W. Stephens and J. R. Krebs. *Foraging Theory*. Princeton University Press, 1986.

[14] R. S. Sutton and A. G. Barto. *Reinforcement learning: an introduction*. MIT Press, 1998.

[15] P. Ulam and T. Balch. Using optimal foraging models to evaluate learned robotic foraging behavior. *Adaptive Behavior*, 12(3-4):213–222, 2004.

[16] R. T. Vaughan. Massively multi-robot simulations in Stage. *Swarm Intelligence*, 2(2-4):189–208, 2008.

[17] J. Vermorel and M. Mohri. Multi-armed bandit algorithms and empirical evaluation. In *Proc. of the 16th European Conf. on Machine Learning (ECML)*, pages 437–448, 2005.

[18] J. K. Waage. Foraging for patchily-distributed hosts by the parasitoid, nemeritis canescens. *Animal Ecology*, 48(2):353–371, 1979.

[19] J. Wawerla and R. T. Vaughan. Optimal robot recharging strategies for time discounted labour. In *Proc. of the 11th Int. Conf. on the Simulation and Synthesis of Living Systems*, 2008.