A Linear Programming Relaxation and a Heuristic for the Restless Bandits Problem with General Switching Costs

Jerome Le Ny, Munther Dahleh and Eric Feron *

May 11, 2008

Abstract

We extend a relaxation technique due to Bertsimas and Niño-Mora for the restless bandit problem to the case where arbitrary costs penalize switching between the bandits. We also construct a one-step lookahead policy using the solution of the relaxation. Computational experiments and a bound for approximate dynamic programming provide some empirical support for the heuristic.

1 Introduction

We study the restless bandit problem (RBP) with general switching costs between the bandits, which could represent travel distances for example. This problem is an intractable extension of the multi-armed bandit problem (MABP), which can be described as follows. There are \( N \) projects, of which only one can be worked on at any time period. Project \( i \) is characterized at (discrete) time \( t \) by its state \( x_i(t) \), which belongs to a finite state space.

*This work was supported by Air Force - DARPA - MURI award 009628-001-03-132 and Navy ONR award N00014-03-1-0171. J. Le Ny and M. Dahleh are with the Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA 02139-4307, USA. E. Feron is with the School of Aerospace Engineering, Georgia Tech, Atlanta, GA 30332, USA. jleny@mit.edu, eric.feron@aerospace.gatech.edu
If project \( i \) is worked on at time \( t \), one receives a reward \( \alpha^t r(x_i(t)) \), where \( \alpha \in (0, 1) \) is a discount factor. The state \( x_i(t) \) then evolves to a new state according to given transition probabilities. The states of all idle projects are unaffected. The goal is to find a policy which decides at each time period which project to work on in order to maximize the expected sum of the discounted rewards over an infinite horizon. The MABP problem was first solved by Gittins [5]. He showed that it is possible to define separately for each project an index which is a function of the project state only, and that the optimal policy operates at each period the project with the greatest current index. Moreover, these indices can be calculated efficiently, as shown for example in [19].

Whittle [20] proposed an interesting modification of the model, called the restless bandit problem (RBP), which extends significantly the range of applications. In the RBP, one can activate several projects at each time period, and the projects that are not activated continue to evolve, possibly using different transition probabilities. Finding an optimal policy efficiently for the RBP is unlikely to be possible however, since the problem is PSPACE-hard [14], even in restricted cases. Nevertheless, Whittle proposed an index policy for the RBP which performs well in practice.

Another extension of the MABP concerns the addition of costs for changing the currently active project. This problem, which we call the multi-armed bandit problem with switching costs (MABPSC), is of great interest to various applications, as discussed by [8], [9], [18], [10], in order to model for example set-up and tear-down costs in queuing networks, transition costs in a job search problem or transaction fees in a portfolio optimization problem. It is easy to see that the MABPSC is NP-hard, since the HAMILTON CYCLE problem is a special case of it [12]. The MABPSC has been studied in particular by Asawa and Teneketzis [1], and very recently by Glazebrook et al. [6] and Niño-Mora [13]. These authors are concerned with the case where the switching costs have a separable form \( c_{ij} = c_i + c_j \), preserving the separable structure from the MABP, and design approximate index policies.

Our work was motivated by an optimal aerial surveillance problem, where switching costs correspond to travel distances between inspection sites. Hence, the assumption on the separable form of the switching costs does not hold. This introduces additional coupling between the projects, and it is not clear then how to design index policies. Moreover, the sites continue to evolve while not visited, and thus we are led to consider the restless bandit problem with switching costs (RBPSC).
We adopt a computational approach to the RBPSC. We impose no restriction on the switching costs, not even the triangle inequality. In Section 2, we formulate the problem as a Markov decision process (MDP), using the state-action frequency approach [4]. This yields a linear program, which we relax in section 3 by following an idea that Bertsimas and Niño-Mora developed for the RBP [3], optimizing over a restricted set of marginals of the occupation measure. The coupling introduced by the switching costs makes this relaxation significantly more challenging to develop than in the classical case, and the first contribution of the paper is to present valid constraints on the marginals improving the quality of the relaxation. This relaxation provides an efficiently computable bound on the achievable performance. Section 4 describes how the relaxation can also be used to motivate a heuristic policy. This heuristic is based on approximate dynamic programming (ADP) techniques, but we also show how to recover it from the linear programming theory used by Bertsimas and Niño-Mora to design their primal-dual heuristic for the RBP. Section 5 presents numerical experiments comparing the heuristic to the performance bound.

The advantage of using the approximate dynamic programming point of view is that a recently developed performance bound provides additional support for our heuristic. However, we do not consider in this paper the development of policies with an priori performance bound. Few results exist in the literature concerning such bounds. As remarked by Guha et al. [7], even the standard RBP is PSPACE-Hard to approximate to any non-trivial factor, unless some assumptions are made on the reward functions.

## 2 Exact Formulation of the RBSC Problem

We formulate the RBPSC using the linear programming approach to Markov decision processes [4], [16]. \( N \) projects are distributed in space at \( N \) sites, and \( M \leq N \) servers can be allocated to \( M \) different projects at each time period \( t = 1, 2, \ldots \). In the following, we use the terms project and site interchangeably; likewise, agent and server have the same meaning. At each time period, each server must occupy one site, and different servers must occupy distinct sites. We say that a site is active at time \( t \) if it is visited by a server, and is passive otherwise. If a server travels from site \( k \) to site \( l \), we incur a cost \( c_{kl} \). Each site can be in one of a finite number of states \( x_n \in S_n \), for \( n = 1, \ldots, N \), and we denote the Cartesian product of the individual state spaces by \( S = S_1 \times \ldots \times S_N \). If site \( n \) in state \( x_n \) is visited, a reward \( r_n^1(x_n) \) is earned, and its state changes to \( y_n \) according to the transition
probabilities \( p_{x_ny_n} \). If the site is not visited, then a reward (potentially negative) \( r^0_{x_n}(x_n) \) is earned for that site and its state changes according to the transition probabilities \( p_{x_ny_n} \). \textit{We assume that all sites change their states independently of each other.}

Let us denote the set \( \{1, \ldots, N\} \) by \([N]\). We consider that when no agent is present at a given site, there is a fictitious agent called passive agent at that site. We also call the real agents active agents, since they collect active rewards. The transition of a passive agent between sites does not involve any switching cost, and when a passive agent is present at a site, the passive reward is earned. Therefore, we have a total of \( N \) agents including both the real and passive agents, and we can describe the positions of all agents by a vector \( s = (s_1, \ldots, s_N) \), which corresponds to a \textit{permutation} of \([N]\). We denote the set of these permutation vectors by \( \Pi_{[N]} \), and we let the \( M \) first components correspond to the active agents. For example, with \( M = 2 \) and \( N = 4 \), the vector \((s_1 = 2, s_2 = 3, s_3 = 1, s_4 = 4) \in \Pi_{[4]} \) means that agent 1 is in site 2, agent 2 in site 3 and sites 1 and 4 are passive.

For an agent \( i \in [N] \), we refer to the set of the other agents by \(-i\). If we fix \( s_i \in [N] \) for some \( 1 \leq i \leq N \), then we write \( s_{-i} \) to denote the vector \((s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_N) \), and \( \Pi_{[N]-s_i} \) to denote the permutations of the set \([N] - \{s_i\} \). In particular, we write \( \sum_{s_{-i} \in \Pi_{[N]-s_i}} \) to denote the sum over all permutations of the positions of the agents \(-i\), over the set of sites not occupied by agent \( i \). We also write \( S_{-i} \) to denote the cartesian product \( S_1 \times \ldots S_{i-1} \times S_{i+1} \times \ldots \times S_N \).

The state of the system at time \( t \) can be described by the state of each site and the positions \( s \in \Pi_{[N]} \) of the servers, including the passive ones. With this state description, we are able to handle any number \( M \leq N \) of agents as a parameter within the same framework. We denote the complete state by \((x_1, \ldots, x_N; s_1, \ldots, s_N) := (x; s) \). We can choose which sites are to be visited next, i.e., an action \( a \) belongs to the set \( \Pi_{[N]} \) and corresponds to the assignment of the agents, including the passive ones, to the sites for the next time period. Once the sites to be visited are chosen, there are costs \( c_{s_i a_i} \) for moving the active agent \( i \) from site \( s_i \) to site \( a_i \), including possibly a nonzero cost for staying at the same site. The immediate reward earned is

\[
R((x; s), a) := \sum_{i=1}^{M} (r^1_{a_i}(x_{a_i}) - c_{s_i a_i}) + \sum_{i=M+1}^{N} r^0_{a_i}(x_{a_i}).
\]

We are given a distribution \( \nu \) on the initial state of the system, and \textit{we will assume a product
form
\[ \nu(x; s) = \prod_{i=1}^{N} \nu_i(x_i) \delta_{d_i}(s_i), \]  
(1)
i.e., the initial states of the sites are independent random variables and server \( i \) leaves initially from site \( d_i \), with \( d \in \Pi_{[N]} \).

The transition probability matrix has a particular structure, since the sites evolve independently and the transitions of the agents are deterministic. Let us write its elements
\[ P_{(x'; s')a(x; s)} = P_{x' a x'} \prod_{i=1}^{N} \delta_{s_i}(a_i), \]  
where
\[ P_{x' a x'} := \prod_{i=1}^{M} p_{x_i, x'_{a_i}} \prod_{i=M+1}^{N} p_{x_i, x'_{a_i}}. \]

The optimal infinite horizon discounted reward, multiplied by \((1 - \alpha)\), is the optimal value of the following linear program (LP) [4]
\[
\begin{align*}
\text{maximize} & \quad \sum_{s \in \Pi_{[N]} \atop a \in \Pi_{[N]} \atop x \in \mathcal{S}} R((x; s), a) \rho_{(x; s), a} \\
\text{subject to} & \quad \sum_{s' \atop a \in \Pi_{[N]} \atop x' \in \mathcal{S}} \rho_{(x'; s')a} [\delta_{(x; s)}(x'; s') - \alpha P_{(x'; s')a(x; s)}] \\
& \quad = (1 - \alpha)\nu(x; s), \quad \forall (x, s) \in \mathcal{S} \times \Pi_{[N]} \\
& \quad \rho_{(x; s), a} \geq 0, \quad \forall ((x; s), a) \in \mathcal{S} \times \Pi_{[N]}^2.
\end{align*}
\]
(2)

The variables \( \{\rho_{(x; s), a}\} \) of the LP, called state action frequencies or occupation measure, form a probability measure on the space of state-action pairs and an optimal policy can be recovered from an optimal solution for the LP. The formulation above is of little computational interest however since the number of variables and constraints is of the order of \(|\mathcal{S}| \times (N!)^2\), that is, exponential in the size of the input.

We can obtain the linear program dual to (2) by constructing it directly, or starting from Bellman’s equation and using standard dynamic programming arguments [2, vol. 2, p. 53].
The decision variables $\{\lambda_{x,s}\}_{x,s}$ of the dual correspond to the reward-to-go vector. We get

$$\minimize (1 - \alpha) \sum_{x,s} \lambda_{x,s} \nu(x; s)$$

subject to

$$\lambda_{x,s} - \alpha \sum_{\tilde{x} \in S} \mathcal{P}_{x,a} \tilde{x} \lambda_{\tilde{x},a} \geq R((x; s), a), \forall x \in S, \forall (s, a) \in \Pi_{[N]}^2.$$

3 LP Relaxation of the RBPSC

We compute a bound on the performance achievable by any assignment policy by relaxing the LP formulation. We start by rewriting the objective function (2) and we identify the relevant marginals of the occupation measure:

$$\sum_{s=1}^{N} \sum_{a=1}^{N} \sum_{x_a \in S_a} \left[ (r_1^{i}(x_a) - c_{sa}) \left( \sum_{i=1}^{M} \rho_{x_{a;sa}}^i \right) + r_0^{i}(x_a) \left( \sum_{i=M+1}^{N} \rho_{x_{a;sa}}^i \right) \right]$$

where the marginals appearing above are obtained as follows:

$$\rho_{i;sa}^i = \sum_{x_{-a} \in S_{-a}} \sum_{s_{-i} \in \Pi_{[N]} - s} \sum_{a_{-i} \in \Pi_{[N]} - a} \rho(x; s, a),$$

$$\forall (i, s, a) \in [N]^3, x_a \in S_a,$$

and the superscripts refer to the agents.

Now to express the constraints, we will also need the following variables:

$$\rho_{i;sa}^i = \sum_{x_{-a} \in S_{-a}} \sum_{s_{-i} \in \Pi_{[N]} - s} \sum_{a_{-i} \in \Pi_{[N]} - a} \rho(x; s, a),$$

$$\forall (i, s, a) \in [N]^3, x_{s} \in S_{s}.$$

The variables in (5) (respectively (6)) can be interpreted as the frequency with which agent $i$ switches from site $s$ to site $a$ and the destination (resp. origin) site is in state $x_a$ (resp. $x_s$). Note that this notation is somewhat redundant, since we can write the variables $\rho_{i;sj}^j$ as in (5) or (6).

It is straightforward to see that the constraints (3) imply

$$\sum_{a=1}^{N} \rho_{i;sa}^i - \alpha \sum_{\tilde{x}_a \in S_{a}} \sum_{s'=1}^{N} \rho_{i;sa}^i \tilde{x}_{a;sa} \mathcal{P}_{x_{a;sa}}^1 \tilde{x}_{a;sa}$$

$$= (1 - \alpha) \nu_s(x_s) \delta_{d_i}(s), \forall (i, s) \in [N]^2, \forall x_s \in S_s,$$
on the marginals, where $1\{\cdot\}$ is the indicator function. However, there are additional relations that must exist because the marginals are obtained from the same original occupation measure. These relations must be found to insure a sufficiently strong relaxation. Another intuitive way to think about this type of constraints is that we enforce sample path constraints only in average [20]. First, from the definitions we have immediately:

$$
\sum_{x_s \in S} \rho_{(x_s,s),a}^i = \sum_{x_a \in S} \rho_{(x_a,s),a}^i, \quad \forall (i, s, a) \in [N]^3.
$$

(8)

Now for the RBPSC, exactly one agent (active or passive) must be at each site at each period. The frequency with which the agents leave site $j$ in state $x_j$ should be equal to the frequency with which the agents move to site $j$ in state $x_j$. So we expect that the following constraints should hold:

$$
N \sum_{s} = 1 N \sum_{a} = 1 \rho_{\tilde{x}_j,j,a}^i = \sum_{x_i \in S} \sum_{s \in \Pi_{[N]-j}} \rho_{(x_i,s),\tilde{x}_j};a \quad \forall \tilde{x}_j \in [N], \tilde{a} \in S_j.
$$

(9)

We now show that (9) are indeed valid constraints. We use the notation $(x_{-j}, \tilde{x}_j)$ to mean that the $j^{th}$ component of the vector is $\tilde{x}_j$, and similarly for $(s_{-i}, j)$. We have, starting from the definition (6)

$$
\sum_{i=1}^{N} \sum_{a=1}^{N} \rho_{\tilde{x}_j,j,a}^i = \sum_{x_{-j} \in S_{-j}} \sum_{a \in \Pi_{[N]}} \sum_{s \in \Pi_{[N]-j}} \rho_{(x_{-j},\tilde{x}_j);(s_{-i},j),a}^i
\sum_{k \in [N]-i} \rho_{(x_{-j},\tilde{x}_j);s,a}^k
$$

(10)

The first equality comes from the fact that we count all the permutation vectors $a$ by varying first the $i^{th}$ component $a_i$ from 1 to $N$. The second equality comes from the fact that we count all the permutations $s$ by varying the position $i$ where the component $s_i$ is equal to $j$ (exactly one of the components of a permutation vector of $\Pi_{[N]}$ has to be $j$). The proof that the right hand side of (9) is also equal to the quantity in (10) is identical.

Here are two additional sets of valid constraints:

$$
\sum_{s=1}^{N} \sum_{x_s \in S} \sum_{a \in [N]-\tilde{a}} \rho_{x_s,s,a}^i = \sum_{k \in [N]-i} \sum_{s=1}^{N} \sum_{x_s \in S} \rho_{x_s,s,\tilde{a}}^k, \quad \forall (i, \tilde{a}) \in [N]^2,
$$

(11)

$$
\sum_{a=1}^{N} \sum_{x_a \in S} \sum_{s \in [N]-\tilde{s}} \rho_{x_a,a,s}^i = \sum_{k \in [N]-i} \sum_{a=1}^{N} \sum_{x_a \in S} \rho_{x_a,a,\tilde{s}}^k, \quad \forall (i, \tilde{s}) \in [N]^2.
$$

(12)
Intuitively, on the left hand side we have the probability that agent \( i \) does not go to site \( \tilde{a} \) (respectively does not leave from site \( \tilde{s} \)), which must equal the probability that some other agent \( k \) (passive or not) goes to site \( \tilde{a} \) (respectively leaves from site \( \tilde{s} \)). Again, these relations can be verified by inspection of (6). Indeed, in (11) for \((i, \tilde{a})\) fixed, similarly to (9), we have two equivalent ways of summing the occupation measure over all indices \((x, s, a)\) such that none of the permutation vectors \( a \) with \( \tilde{a} \) in position \( i \) appears. On the left hand side of (11), we vary the coefficient \( a_i \) in the set \( \{1, \ldots, N\} \setminus \{\tilde{a}\} \), whereas on the right hand side, we obtain the same result by forcing the element \( \tilde{a} \) to be in a position different from position \( i \). Similarly, in (12), we have two ways of summing over all indices such that none of the permutation vectors \( s \) with \( \tilde{s} \) in position \( i \) appears.

Finally, we have obtained a relaxation for the RBPSC:

**Theorem 3.1.** We can obtain an upper bound on the optimal reward achievable in the RBPSC by solving the following linear program:

\[
\begin{align*}
\text{maximize} & \quad \sum_{s=1}^{N} \sum_{a=1}^{N} \sum_{x_a \in S_a} \left[ (r^1_a(x_a) - c_{sa}) \left( \sum_{i=1}^{M} \rho^i_{x_a,s,a} \right) + r^0_a(x_a) \left( \sum_{i=M+1}^{N} \rho^i_{x_a,s,a} \right) \right] \\
\text{subject to} & \quad (7), (8), (9), (11), (12), \\
& \quad \rho^i_{(x,s),a} \geq 0, \quad \forall (i, s, a) \in [N]^3, x_s \in S_s \\
& \quad \rho^i_{(x,a),s} \geq 0, \quad \forall (i, s, a) \in [N]^3, x_a \in S_a.
\end{align*}
\]

There are now \( O(N^3 \times \max_i |S_i|) \) variables \( \rho^i_{(x,s),a}, \rho^i_{(x,a),s} \), and constraints in the relaxed linear program, which is polynomial in the size of the input. From the remarks about the complexity of the problem, it is unlikely that a polynomial number of variables will suffice to formulate the RBPSC exactly. However, the addition of the constraints tying the marginals together helps reduce the size of the feasible region spanned by the decision vectors and improve the quality of the relaxation. Computing the optimal value of this linear program can be done in polynomial time, and provides an upper bound on the performance achievable by any policy for the original problem.
3.1 Dual of the Relaxation

It will be useful to consider the dual of the LP relaxation obtained in the previous paragraph, which we derive directly from (13). This dual program could be obtained by dynamic programming arguments, in the spirit of the original work of Whittle, incorporating the constraints (8), (9), (11), and (12) using Lagrange multipliers. We obtain:

\[
\begin{align*}
\text{minimize} & \quad (1 - \alpha) \sum_{i=1}^{N} \sum_{s=1}^{S} \nu_s(x_s) \delta_{d_i}(s) \lambda^i_{s,x_s} \\
\text{subject to} & \quad \lambda^i_{s,x_s} + \mu^i_{s,a} + \kappa^i_{s,x_s} - \sum_{i' \neq i} \zeta^i_{s,a'} + \sum_{s' \neq s} \xi^i_{s,s'} \geq 0, \quad \forall (i, s, a) \in [N] \times [N], s \neq a, \\
& \quad -\alpha \sum_{\tilde{x}_a} p^1_{x_a \tilde{x}_a} \lambda^i_{a,\tilde{x}_a} - \mu^i_{s,a} - \kappa^i_{a,x_a} - \sum_{i' \neq i} \zeta^i_{s,a'} + \sum_{s' \neq s} \xi^i_{s,s'} \geq r^1_{a\{i \leq M\}}(x_a) - c_{sa}1\{i \leq M\}, \\
& \quad \forall (i, s, a) \in [N]^3, s \neq a, \\
& \quad \lambda^i_{s,x_s} - \alpha \sum_{\tilde{x}_a} p^1_{x_a \tilde{x}_a} \lambda^i_{a,\tilde{x}_a} - \sum_{i' \neq i} (\zeta^i_{s,a'} + \xi^i_{s,s'}) + \sum_{s' \neq s} (\zeta^i_{s,a'} + \xi^i_{s,s'}) \geq r^1_{a\{i \leq M\}}(x_s) - c_{ss}1\{i \leq M\}, \\
& \quad \forall i \in [N].
\end{align*}
\]

The optimal dual variables \(\bar{\lambda}^i_{s,x_s}\) are Lagrange multipliers corresponding to the constraints (7) and have a natural interpretation in terms of reward-to-go if site \(s\) is in state \(x_s\) and visited by agent \(i\). The optimal dual variables \(\bar{\mu}^i_{s,a}, \bar{\kappa}^i_{a,x_a}, \bar{\zeta}^i_{a,s}, \bar{\xi}^i_{s,s}\) correspond to the additional constraints (8), (9), (11), and (12) respectively. We can obtain the optimal primal and dual variables simultaneously when solving the relaxation. For \(j = s\) or \(a\), we also obtain the optimal reduced costs \(\bar{\gamma}^i_{x_j, s, a}\): \(\bar{\gamma}^i_{x_s, s, a}\) are equal to the left hand side of the constraints (15), whereas \(\bar{\gamma}^i_{x_s, s, a}\) are equal to the difference between the left hand side and the right hand side of (16), or (17) if \(s = a\). There is one such reduced cost for each variable \(\rho^i_{x_j, s, a}\) of the primal, and by complementary slackness, \(\bar{\rho}^i_{x_j, s, a}, \bar{\gamma}^i_{x_j, s, a} = 0\), where \(\{\bar{\rho}^i_{x_j, s, a}\}\) is the optimal solution of the primal.

4 A Heuristic for the RBPSC

The relaxation is also useful to actually design assignment policies for the agents. We present here a one-step lookahead policy and its relationship with the primal-dual heuristic.
of Bertsimas and Ninó-Mora, developed for the RBP.

4.1 One-Step Lookahead Policy

Consider the multi-agent system in state \((x; s)\), with \(s\) a permutation of \([N]\). Given the interpretation of the dual variables \(\lambda^i_{s_i, x_{s_i}}\) in terms of reward-to-go mentioned in section 3.1, it is natural to try to form an approximation \(\tilde{J}(x; s)\) of the global reward-to-go in state \((x; s)\) as

\[
\tilde{J}(x_1, \ldots, x_N; s_1, \ldots, s_N) = \sum_{i=1}^{N} \lambda^i_{s_i, x_{s_i}},
\]

(18)

where \(\lambda^i_{s_i, x_{s_i}}\) are the optimal values of the dual variables obtained when solving the LP relaxation. The separable form of this approximate cost function is useful to design an easily computable one-step lookahead policy [2], as follows. In state \((x; s)\), we obtain the assignment \(\tilde{u}(x; s)\) of the agents as

\[
\tilde{u}(x; s) \in \arg \max_{a \in \Pi_{[N]}} \left\{ R((x; s), a) + \alpha \sum_{x' \in S} P_{x' \to x} \bar{J}(x'; a) \right\}.
\]

(19)

In this computation, we replaced the true optimal cost function, which would provide an optimal policy, by the approximation \(\tilde{J}\). Using (18), we can rewrite the maximization above as

\[
\max_{a \in \Pi_{[N]}} \sum_{i=1}^{N} m_{i,a_i},
\]

(20)

with

\[
m_{i,a_i} = x^1_{a_i}(x_{a_i}) - c_{s_i a_i} 1\{i \leq M\} + \alpha \sum_{x'_{a_i} \in S_{a_i}} \bar{J}^i_{a_i, x'_{a_i}} p_{x_{a_i} x'_{a_i}} 1\{i \leq M\}.
\]

Assuming that the optimal dual variables have been stored in memory, the evaluation of the terms \(m_{i,a_i}\), for all \((i, a_i)\), takes a time \(O(N^2 \max_i |S_i|)\). The maximization (20) is then a linear assignment problem, which can be solved by linear programming or in time \(O(N^3)\) by the Hungarian method [17]. Thus, the assignment can be computed at each time step in time \(O(N^2 \max_i |S_i| + N^3)\) by a centralized controller.
4.2 Equivalence with the Primal-Dual Heuristic

Recall from paragraph 3.1 that, when solving the linear programming relaxation, we can obtain the optimal primal variables \( \{ \rho_{x_i, s, a}^j \} \), the dual variables \( \{ \lambda_{s_i, x_i}^i, \bar{\mu}_{s, a}^i, \bar{\kappa}_{a, x_i}^i, \bar{\zeta}_{a, s}^i, \bar{\xi}^i \} \), and the reduced costs \( \{ \bar{\gamma}_{x_i, s, a}^i \} \). These reduced costs are nonnegative. Bertsimas and Ninomia [3] motivated their primal-dual heuristic for the RBP using the following well-known interpretation of the reduced costs: starting from an optimal solution, \( \bar{\gamma}_{(x_j; s, a)}^i \) is the rate of decrease in the objective value of the primal linear program (13) per unit increase in the value of the variable \( \rho_{(x_j; s, a)}^i \).

We use this interpretation and the following intuitive idea: when agent \( i \) is in state \( x_s \) and we decide to send it to site \( a \) in state \( x_a \), in some sense we are increasing the values of \( \rho_{(x_i; s, a)}^i \) and \( \rho_{(x_a; s, a)}^i \), which are the long-term probabilities of such transitions. In particular, we would like to keep the quantities \( \rho_{(x_j; s, a)}^i \) found to be 0 in the relaxation as close to 0 as possible in the final solution. By complementary slackness it is only for these variables that we might have \( \bar{\gamma}_{(x_j; s, a)}^i > 0 \). Hence, when the system is in state \((x; s)\), we associate to each action \( a \) an index of undesirability

\[
I((x; s), a) = \sum_{\{i \in [N]: s_i \neq a_i\}} (\bar{\gamma}_{(x_i; s_i), a_i}^i + \bar{\gamma}_{(x_a; s_i), a_i}^i) + \sum_{\{i \in [N]: s_i = a_i\}} \bar{\gamma}_{(x_i; s_i), s_i}^i,
\]

that is, we sum the reduced costs for the \( N \) different projects. Then we select an action \( a_{pd} \in \Pi_{[N]} \) that minimizes these indices:

\[
a_{pd}(x; s) = \arg\min_{a} \{ I((x; s), a) \}.
\]

We now show that this policy is in fact equivalent to the one-step lookahead policy described earlier. Using the expression for the reduced costs from paragraph 3.1, we can rewrite the indices in (21) more explicitly. The term \( \bar{\gamma}_{(x_i; s_i), a_i}^i + \bar{\gamma}_{(x_a; s_i), a_i}^i \) of the sum (21) is equal to

\[
\bar{\lambda}_{s_i, x_i}^i + \bar{\kappa}_{x_i, s_i}^i - \sum_{s' = 1}^{N} \bar{\epsilon}_{s_i}^i + \sum_{s' = 1}^{N} \bar{\epsilon}_{s_i}^i + \sum_{a' = 1}^{N} \bar{\zeta}_{a_i}^i - \bar{\kappa}_{a, x_i}^i - \alpha \sum_{x_{a_i}} p_{x_{a_i}, x_{a_i}}^i \bar{\lambda}_{a_i, x_i}^i - r_{a_i}^1 \{ x_{a_i} \} + c_{a, a_i}^i \{ i \leq M \},
\]

after cancellation of \( \bar{\mu}_{s, a}^i \) and adding and subtracting \( \bar{\zeta}_{a_i}^i \) and \( \bar{\xi}_{s_i}^i \). This expression is valid for the terms \( \gamma_{(x_i; s_i), s_i}^i \) as well. Now after summation over \( i \in [N] \), the first two lines in
expression (23) do not play any role in the minimization (22). This is obvious for the terms that do not depend on \( a \). For the terms involving the \( \zeta i^a_j \), we can write

\[
\sum_{i=1}^{N} \sum_{i'=1}^{N} \bar{\zeta}_{ai}^i = \sum_{i'=1}^{N} \sum_{i=1}^{N} \bar{\zeta}_{a'i}^i = \sum_{i'=1}^{N} \sum_{a=1}^{N} \bar{\zeta}_{a'i}^a,
\]

the last equality being true since \( a \) is just a permutation of \( \{1, \ldots, N\} \). Hence the sums involving the \( \bar{\zeta}_{a'i}^i \) cancel (in fact we even see that each individual sum is independent of the choice of \( a \)). As for the term \( \sum_{i=1}^{N} \bar{\kappa}_{ai}x_{ai} \), it is equal to \( \sum_{j=1}^{N} \bar{\kappa}_{j}x_{j} \), and so it is independent of the choice of \( a \in \Pi[N] \). We are left with the following optimization problem:

\[
\mathbf{a}_{pd}(x; s) \in \arg\min_a \left\{ -\sum_{i=1}^{N} (r_{ai}^{1[i \leq M]}) (x_{ai}) - c_{s,ai} 1\{i \leq M\} + \alpha \sum_{x_{ai}} p_{x_{ai}, \bar{x}_{ai}}^{1[i \leq M]} \bar{\lambda}_{ai} \bar{x}_{ai} \right\},
\]

which after a sign change is seen to be exactly (20). We have shown the following

**Theorem 4.1.** The primal-dual heuristic (22), based on the interpretation of the reduced costs of the LP relaxation, is equivalent to the one-step lookahead policy (20) assuming the separable approximation (18) for the reward-to-go.

In view of this result, we obtain an alternative way to compute the one-step lookahead policy. The minimization (22) is again a linear assignment problem. If we can store the \( \mathcal{O}(N^3 \max_i(|S_i|)) \) optimal reduced costs instead of the \( \mathcal{O}(N^2 \max_i(|S_i|)) \) optimal dual variables, there is just a linear cost involved in computing the indices \( I((x; s), a) \) of the problem at each period resulting in an overall \( \mathcal{O}(N^3) \) computational cost for the on-line maximization at each period.

## 5 Numerical Experiments

Table 1 presents numerical experiments on problems whose characteristics differently affect the performance of the heuristic described in section 4. Linear programs are implemented in AMPL and solved using CPLEX. Due to the size of the state space, the expected discounted reward of the heuristics is computed using Monte-Carlo simulations. The computation of each trajectory is terminated after a sufficiently large, but finite horizon: in our case, when \( \alpha^t \) times the maximal absolute value of any immediate reward becomes less than \( 10^{-6} \). To
reduce the amount of computation in the evaluation of the policies, we assume that the
distribution of the initial states of the sites is deterministic.

In a given problem, the number $|S_i|$ of states is chosen to be the same for all projects. $c/r$ is
the ratio of the average switching cost divided by the average active reward. This is intended
to give an idea of the importance of the switching costs in the particular experiment. The
switching costs are always taken to be nonnegative. $Z^*$ is the optimal value of the problem,
computed using (2), when possible. $Z_r$ is the optimal value of the relaxation and so provides
an upper bound on the achievable performance. $Z_{osl}$ is the estimated expected value of the
one-step lookahead policy. $Z_g$ is the estimated expected value of the greedy policy which is
obtained by fixing the value of the $\lambda_{a_i,x_{a_i}}$ in (20) to zero, i.e., approximating the reward-to-
go by zero. This greedy policy is actually optimal for the MABP with deteriorating active
rewards, i.e., such that projects become less profitable as they are worked on [2, vol. 2, p.69].
Problem 2 is of this type and shows that the one-step lookahead policy does not perform
optimally in general.

Problem 1 is a MABP. The heuristic is not optimal, so we see that we do not recover
Gittins’ policy. Hence the heuristic is also different from Whittle’s in general, which reduces
to Gittins’ in the MAB case. In problem 3, we add transition costs to problem 2. The greedy
policy is not optimal any more, and the one-step lookahead policy performs better in this
case. Problem 4 is designed to make the greedy policy underperform: two remote sites have
slightly larger initial rewards (taking into account the cost for reaching them), but the active
rewards at these sites are rapidly decreasing and the agents are overall better off avoiding
these sites. The greedy policy does not take into account the future transition costs incurred
when leaving these sites. In this case, it turns out that the one-step lookahead is quasi-
optimal. Problem 7 and 8 are larger scale problems, with up to 30 sites. The relaxation is
computed in about 20 minutes on a standard desktop, showing the feasibility of the approach
for this range of parameters.

6 A “Performance” Bound

In this section, we present a result that offers some insight into why we could expect the
one-step lookahead policy to perform well if the linear programming relaxation of the original
problem is sufficiently tight. We begin with the following
| Problem | Problem | Problem | Problem | Problem | Problem | Problem | Problem | Problem | Problem |
|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|
|         |         |         |         |         |         |         |         |         |         |
| (N, M, | (4,1,3, | (4,1,3, | (4,1,3, | (4,2,5, | (6,2,4, | (6,2,4, | (6,2,4, | (20,15, | (30,15, |
|         | 0.5     | 0.9     | 0.9     | 0.9     | 0.5     | 0.9     | 0.9     | 0.9     | 0.9     |
| α       | Z*      | Z_r     | Z_g     | Z_r     | Z*      | Z_r     | Z_g     | Z*      | Z*      |
|         | Zosl    |         |         |         | Zosl    |         |         | Zosl    |         |
| 0.5     | 84.69   | 85.21   | 84.5    | 85.1    | 84.1    | 84.1    | 84.1    | 84.1    | 84.1    |
| 0.9     | 299.6   | 301.4   | 276     | 231.0   | 245.1   | 231     | 228     | 231     | 228     |
| 0.99    | 2614.2  | 2614    | 2324    | 2614    | 2614    | 2614    | 2614    | 2614    | 2614    |
| 0.5     | 57.54   | 59.32   | 56.0    | 57.5    | 56.0    | 57.3    | 56.0    | 57.3    | 57.3    |
| 0.9     | 184.5   | 185.0   | 177     | 184.5   | 185.0   | 177     | 183     | 183     | 183     |
| 0.99    | 1279    | 1280    | 1273    | 1279    | 1280    | 1273    | 1277    | 1277    | 1277    |
| 0.5     | 165.7   | 115     | 165     | 165.7   | 115     | 165     | 165     | 165     | 165     |
| 0.9     | 767.2   | 661     | 767     | 767.2   | 661     | 767     | 767     | 767     | 767     |
| 0.95    | 1518    | 1403    | 1518    | 1518    | 1403    | 1518    | 1518    | 1518    | 1518    |
| 0.5     | 39.25   | 38.5    | 36.5    | 39.25   | 38.5    | 36.5    | 36.5    | 36.5    | 36.5    |
| 0.9     | 214.0   | 205     | 198     | 214.0   | 205     | 198     | 198     | 198     | 198     |
| 0.95    | 431.6   | 414     | 396     | 431.6   | 414     | 396     | 396     | 396     | 396     |
| 0.5     | 9.727   | 6.93    | 8.24    | 9.727   | 6.93    | 8.24    | 8.24    | 8.24    | 8.24    |
| 0.9     | 62.80   | 38.0    | 47.0    | 62.80   | 38.0    | 47.0    | 47.0    | 47.0    | 47.0    |
| 0.95    | 128.7   | 78.0    | 99.0    | 128.7   | 78.0    | 99.0    | 99.0    | 99.0    | 99.0    |
| 0.5     | 196.5   | 189     | 194     | 196.5   | 189     | 194     | 194     | 194     | 194     |
| 0.9     | 952.7   | 877     | 900     | 952.7   | 877     | 900     | 900     | 900     | 900     |
| 0.95    | 1899    | 1747    | 1776    | 1899    | 1747    | 1776    | 1776    | 1776    | 1776    |
| 0.5     | 589.4   | 566     | 564     | 589.4   | 566     | 564     | 564     | 564     | 564     |
| 0.9     | 2833    | 2640    | 2641    | 2833    | 2640    | 2641    | 2641    | 2641    | 2641    |
| 0.95    | 5642    | 5218    | 5246    | 5642    | 5218    | 5246    | 5246    | 5246    | 5246    |

Lemma 6.1. The approximate reward-to-go (18) is a feasible solution for the original dual linear program (4).

Proof. Consider one constraint in the original dual LP (4), for a fixed state-action tuple
(x, s, a). We consider a situation where $s_i \neq a_i$ for all $i \in \{1, \ldots, N\}$. Summing the constraints (15) over $i$ for the given values of $x_{s_i}, s_i, a_i$, we get

$$\sum_{i=1}^{N} \lambda_{s_i,x_{s_i}}^i + \sum_{i=1}^{N} \mu_{s_i,a_i}^i + \sum_{i=1}^{N} \kappa_{s_i,x_{s_i}}^i - \sum_{i=1}^{N} \sum_{j'=1}^{N} \zeta_{a_i}^j + \sum_{i=1}^{N} \sum_{a'=1}^{N} \zeta_{a_i}^i$$

$$= \sum_{i=1}^{N} \lambda_{s_i,x_{s_i}}^i + \sum_{i=1}^{N} \mu_{s_i,a_i}^i + \sum_{i=1}^{N} \kappa_{s_i,x_{s_i}}^i \geq 0.$$ 

The cancellation follows from the discussion preceding theorem 4.1. Now summing the constraints (16) over $i$, we also get

$$-\alpha \sum_{i=1}^{N} \sum_{x_{a_i}} P_{1{i \leq M}} \lambda_{x_{a_i},x_{a_i}}^i - \sum_{i=1}^{N} \sum_{a_i} \mu_{s_i,a_i}^i - \sum_{i=1}^{N} \sum_{a_i} \kappa_{s_i,x_{s_i}}^i \geq \sum_{i=1}^{N} r_{a_i}^{1{i \leq M}}(x_{a_i}) - c_{s_i,a_i} 1\{i \leq M\}.$$

Finally, we add these two inequalities. We obtain

$$\sum_{i=1}^{N} \lambda_{s_i,x_{s_i}}^i - \alpha \sum_{i=1}^{N} \sum_{x_{a_i}} P_{1{i \leq M}} \lambda_{x_{a_i},x_{a_i}}^i \lambda_{a_i,x_{a_i}}^i \geq \sum_{i=1}^{N} \left(r_{a_i}^{1{i \leq M}}(x_{a_i}) - c_{s_i,a_i} 1\{i \leq M\}\right),$$

which is the inequality obtained by using the vector (18) in the constraints of (4).

The case where $s_i = a_i$ for some $i$ is almost identical, considering the constraints (17) for the corresponding indices.

In the following theorem, the occupation measure $F_\alpha(\nu, \tilde{u})$ is a vector of size $|S|$, representing the discounted infinite horizon frequencies of the states under policy $\tilde{u}$ and initial distribution $\nu$ [4]. The proof of the theorem follows from the analysis presented in [15], see [11] for more details.

**Theorem 6.2.** Let $\nu$ be an initial distribution on the states, of the product form (1). Let $J^*$ be the optimal reward function, $\tilde{J}$ be an approximation of this reward function which is feasible for the LP (4), and $\tilde{u}$ be the associated one-step lookahead policy. Let $F_\alpha(\nu, \tilde{u})$ and $J_{\tilde{u}}$ be the occupation measure vector and the expected reward associated to the policy $\tilde{u}$. Then

$$\nu^T(J^* - J_{\tilde{u}}) \leq \frac{1}{1 - \alpha} F_\alpha(\nu, \tilde{u})^T(\tilde{J} - J^*).$$

(24)

From lemma 6.1, the theorem is true in particular for $\tilde{J}$ formed according to (18). In words, it says that starting with a distribution $\nu$ over the states, the difference in expected rewards
between the optimal policy and the one-step lookahead policy is bounded by a weighted $l^1$-distance between the estimate $\tilde{J}$ used in the design of the policy and the optimal value function $J^*$. The weights are given by the occupation measure of the one-step lookahead policy. It provides some motivation to obtain a good approximation $\tilde{J}$, i.e., a tight relaxation, which was an important element of this paper.

7 Acknoledgements

The authors thank Michel Goemans, Sean Meyn and John Tsitsiklis for valuable comments. This work was supported by Air Force - DARPA - MURI award 009628-001-03-132 and Navy ONR award N00014-03-1-0171.

References

[1] M. Asawa, D. Teneketzis, Multi-armed bandits with switching penalties, IEEE transactions on automatic control 41 (3) (1996) 328–348.

[2] D. Bertsekas, Dynamic Programming and Optimal Control, vol. 1 and 2, 2nd ed., Athena Scientific, 2001.

[3] D. Bertsimas, J. Niño-Mora, Restless bandits, linear programming relaxations, and a primal-dual index heuristic, Operations Research 48 (2000) 80–90.

[4] C. Derman, Finite State Markovian Decision Processes, Academic Press, 1970.

[5] J. Gittins, D. Jones, A dynamic allocation index for the sequential design of experiments, in: J. Gani (ed.), Progress in Statistics, North-Holland, Amsterdam, 1974, pp. 241–266.

[6] K. D. Glazebrook, D. Ruiz-Hernandez, C. Kirkbride, Some indexable families of restless bandit problems, Advances in Applied Probability 38 (3) (2006) 643–672.

[7] S. Guha, K. Munagala, P. Shi, On index policies for restless bandit problems (2007). URL http://arxiv.org/abs/0711.3861

[8] T. Jun, A survey on the bandit problem with switching costs, De Economist 152 (4) (2004) 513–541.
[9] T. Jun, Costly switching and investment volatility, Economic Theory 25 (2) (2005) 317–332.

[10] G. Koole, Assigning a single server to inhomogeneous queues with switching costs, Theoretical Computer Science 182 (1997) 203–216.

[11] J. Le Ny, M. Dahleh, E. Feron, Multi-agent task assignment in the bandit framework, in: Proceedings of the 45th IEEE Conference on Decision and Control, San Diego, CA, 2006.

[12] J. Le Ny, E. Feron, Restless bandits with switching costs: linear programming relaxations, performance bounds and limited lookahead policies, in: Proceedings of the American Control Conference, Minneapolis, MN, 2006.

[13] J. Niño-Mora, A faster index algorithm and a computational study for bandits with switching costs, INFORMS Journal on Computing. Accepted.

[14] C. Papadimitriou, J. Tsitsiklis, The complexity of optimal queueing network control, Mathematics of Operations Research 24 (2) (1999) 293–305.

[15] D. Pucci de Farias, B. Van Roy, The linear programming approach to approximate dynamic programming, Operations Research 51 (6) (2003) 850–856.

[16] M. Puterman, Markov Decision Processes, Wiley, 1994.

[17] A. Schrijver, Combinatorial Optimization - Polyhedra and Efficiency, Springer, 2003.

[18] M. Van Oyen, D. Pandelis, D. Teneketzis, Optimality of index policies for stochastic scheduling with switching penalties, Journal of Applied Probability 29 (1992) 957–966.

[19] P. Varaiya, J. Walrand, C. Buyukkoc, Extensions of the multiarmed bandit problem: the discounted case, IEEE transactions on automatic control 30 (5) (1985) 426–439.

[20] P. Whittle, Restless bandits: activity allocation in a changing world, Journal of Applied Probability 25A (1988) 287–298.