Appendix A
Proofs

A.1 The Best-So-Far Procedure Is Markovian

Here, I prove Proposition 2 of Section 3.4.1. This proposition gives us a way to apply a natural patience-based termination criterion to a nonmonotonic search procedure such as WALKSAT—yet still maintain the Markov property that makes the target of STAGE's learning well-defined—by using the device of the *best-so-far abstraction* $\text{BSF}(\pi)$, as defined in Definition 6 on page 61. The statement is as follows:

Proposition (**#2**, p. 62). *If local search procedure π is Markovian over a finite state space X, and π' is the procedure that results by adding patience-based termination to π, then procedure $\text{BSF}(\pi')$ is proper, Markovian, and strictly monotonic.*

In outline, the proof follows these three steps:

1. The procedure π' is proper, but not necessarily Markovian in the state space X. However, π' is proper and Markovian in an augmented state space Z.
2. We apply a lemma that the Y*-abstraction* (defined below) of any proper and Markovian procedure is also proper and Markovian. $\text{BSF}(\pi')$ is such an abstraction, which proves that it is proper and Markovian in the augmented state space Z.
3. Finally, we show that the Markov property still holds when trajectories of $\text{BSF}(\pi')$ are projected back down to the original state space X. The property of strict monotonicity also follows trivially.

Before explaining these steps in detail, I present the definition of a Y-abstraction and the lemma that Step 2 requires.

Definition 7. Let π be a proper local search procedure defined over a state space $Z \cup \{\text{END}\}$, and let $Y \subset Z$ be an arbitrary subset of states. Then the Y*-abstraction* of procedure π is a new policy π', defined over the smaller state space $Y \cup \{\text{END}\}$, which is derived from π as follows: starting from any state $y_0 \in Y$, π' generates trajectories by following π *but filtering out all states belonging to* $Z \setminus Y$ (see Figure A.1).

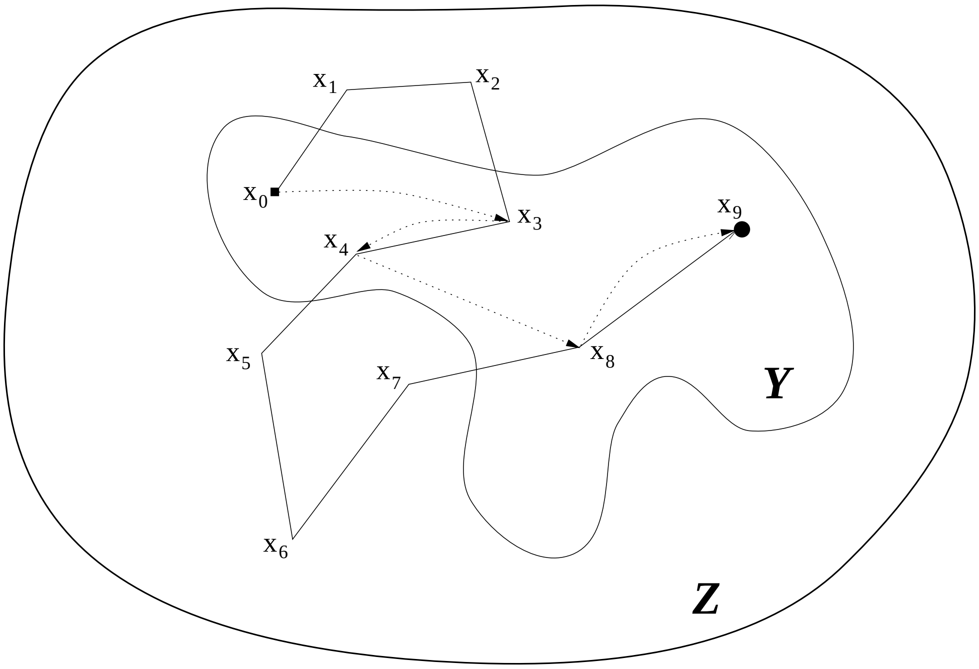

FIGURE A.1. Illustration of the Y-abstraction procedure. Given a trajectory $(x_0, \ldots, x_9)$ (straight solid lines) generated by procedure π in state space Z, the Y-abstraction produces a corresponding shorter trajectory (dotted lines) that is restricted to the subspace Y.

Lemma 1. *Suppose procedure π is proper and Markovian over state space $Z \cup \{\text{END}\}$, and that procedure π' is the Y-abstraction of Z for any given subset $Y \subset Z$. Then π' is proper and Markovian over the subspace $Y' = Y \cup \{\text{END}\}$.*

Proof of Lemma 1. First, note that if we apply π starting from any state $x_0 \in Z$, we get a trajectory τ which must eventually visit a state $y \in Y'$, since π is assumed proper and Y' includes the terminal state. Let $\text{ENTER}(\tau, Y')$ be the first state in Y' that τ visits after leaving x_0. Then, over the full distribution of trajectories that π may generate from x_0, the probability that y is the first state subsequently visited in Y' is given by $p(y|x_0)$, computed as follows:

$$\forall x_0 \in Z, y \in Y' : \quad p(y|x_0) \stackrel{\text{def}}{=} \sum_{\{\tau \in \mathcal{T} : \text{ENTER}(\tau, Y') = y\}} P(\tau | x_0^\tau = x_0) \tag{A.1}$$

Now consider a trajectory of the Y-abstraction policy π', starting at a state $y_0 \in Y$. By definition of π', this trajectory will be built by applying the original policy π and filtering out the states in $Z \backslash Y$. Thus, the first transition will be to some state $y_1 \in Y'$

with probability given precisely by $p(y_1|y_0)$ as defined in Equation A.1. Furthermore, by the Markov assumption on π, the future transitions from y_1 will be independent of the trajectory history and obey the same probabilities $p(y_{i+1}|y_i)\ \forall i \geq 0$. Thus, π' is Markovian over Y'. π' is also clearly proper, by inheritance from π. □

I now fill in the details of the proof of Proposition 2, following the three-part outline sketched above.

Proof of Proposition 2.

1. In the statement of the proposition, π' is defined to be the procedure that results when patience-based termination is imposed on a procedure π that is Markovian, but not necessarily proper. Recall that patience-based termination means, for a given patience level $\textsc{Pat} \geq 1$, that any trajectory will end deterministically as soon as $\textsc{Pat}$ consecutive steps are taken with no improvement over the best state found so far. That is, for all trajectories τ produced by π':

$$(t \geq \textsc{Pat}) \wedge (x^\tau_{t-\textsc{Pat}} = \min_{\{j:0\leq j\leq t\}} \mathrm{Obj}(x^\tau_j)) \;\Rightarrow\; x^\tau_{t+1} = \textsc{end}.$$

   Since the state space X is assumed finite, π' is certainly proper: the Obj values cannot keep decreasing forever. However, π' is not necessarily Markovian. Because of the new termination condition, the transition probability

$$\mathrm{P}(x^\tau_{i+1} = x_{i+1} \mid x^\tau_0 = x_0, x^\tau_1 = x_1, \ldots, x^\tau_i = x_i)$$

   is *not* independent of the history $x_0, \ldots, x_{i-1}$ as required by Definition 3 (page 57). In particular, the probability of termination depends not just on the current state, but also on how recently the best-so-far state was found, and what the Obj value at that state was.

   However, the procedure π' can be made Markovian by augmenting the state space with these relevant extra variables. Define the augmented state space $Z = X \times \Re \times \mathbb{N}$, where the three components of a state $z \in Z$ are given by

   $a(z) \in X$, representing a state x in the original state space
   $b(z) \in \Re$, representing the best Obj value seen on the current trajectory
   $c(z) \in \mathbb{N}$, representing the count of non-improving steps since finding $b(z)$.

   In this augmented space, a trajectory of procedure π' begins at the state $z^\tau_0 = \langle x^\tau_0, \mathrm{Obj}(x^\tau_0), 0\rangle$. All future state transitions depend only on the current state

z_i^τ. There are three kinds of possible transitions: (1) the patience counter may expire, causing deterministic termination; (2) a new best-so-far state may be discovered, in which case $b(z)$ is updated and $c(z)$ is reset to zero; or (3) an ordinary transition to a non-best-so-far state occurs, in which case $c(z)$ is simply incremented. These give rise to the following Markovian transition probabilities: $\forall z \in Z, z' \in Z \cup \{\text{END}\}$,

$$\mathrm{P}(z_{i+1}^\tau = z' \mid z_i^\tau = z) =$$
$$\begin{cases} 1 & \text{if } (c(z) = \text{PAT}) \wedge (z' = \text{END}) \\ p(a(z')|a(z)) & \text{if } (\text{Obj}(a(z')) < b(z)) \wedge (b(z') = \text{Obj}(a(z'))) \wedge (c(z') = 0) \\ p(a(z')|a(z)) & \text{if } (\text{Obj}(a(z')) \geq b(z)) \wedge (b(z') = b(z)) \wedge (c(z') = c(z) + 1) \\ 0 & \text{otherwise.} \end{cases}$$

Thus, π' is proper and Markovian over the space $Z \cup \{\text{END}\}$, concluding the first segment of our proof.

2. We now apply Lemma 1 to show that the best-so-far abstraction of π', denoted $\text{BSF}(\pi')$, is also proper and Markovian in the augmented state space Z. Let the subset Y be defined by

$$Y = \{z \in Z : (b(z) = \text{Obj}(a(z))) \wedge (c(z) = 0)\}.$$

Y consists of precisely the best-so-far states on any trajectory of π'. By Lemma 1, $\text{BSF}(\pi')$ is proper and Markovian over the subspace $Y \cup \{\text{END}\}$. That is, following procedure $\text{BSF}(\pi')$, the transitions from any state y are given by fixed probabilities $p(y'|y)$ independent of any past history of the procedure.

3. Finally, we note that every state $y \in Y$ has the form $\langle x, \text{Obj}(x), 0\rangle$. All the information about the state y is determined by the first component, the underlying state in the un-augmented space. It follows that the procedure $\text{BSF}(\pi')$ is Markovian in the underlying space $X \cup \{\text{END}\}$, with transition probabilities given simply by

$$\mathrm{P}(x_{i+1}^\tau = x' \mid x_i^\tau = x) = p(\langle x', \text{Obj}(x'), 0\rangle \mid \langle x, \text{Obj}(x), 0\rangle).$$

We note that the property of strict monotonicity follows trivially from the definition of BSF, completing the proof of Proposition 2. □

A.2 Least-Squares TD(1) Is Equivalent to Linear Regression

This section demonstrates that the incremental algorithm LSTD(1), which is a special case of the LSTD(λ) procedure introduced in Section 6.1.2, produces an approximate value function which is equivalent to that which would be generated by standard, non-incremental, least-squares linear regression.

To be precise, assume we are given a sample trajectory $(x_0, x_1, \ldots, x_L, \text{END})$ of a Markov chain, with one-step rewards of $R(x_j, x_{j+1})$ on each step. From this trajectory, a supervised learning system would generate the training pairs:

$$\begin{aligned}
\boldsymbol{\phi}_0 &\mapsto R(x_0, x_1) + R(x_1, x_2) + \cdots + R(x_L, \text{END}) \\
\boldsymbol{\phi}_1 &\mapsto R(x_1, x_2) + \cdots + R(x_L, \text{END}) \\
\vdots &\mapsto \vdots \\
\boldsymbol{\phi}_L &\mapsto R(x_L, \text{END})
\end{aligned}$$

where $\boldsymbol{\phi}_i$ is the vector of features representing state x_i. Performing standard least-squares linear regression on the above training set, as described for STAGE in Chapter 3 (Equation 3.8, page 67), produces the following regression matrices:

$$\mathbf{A}_{\mathrm{LR}} = \sum_{i=0}^{L} \boldsymbol{\phi}_i \boldsymbol{\phi}_i^{\mathsf{T}} \qquad\qquad \mathbf{b}_{\mathrm{LR}} = \sum_{i=0}^{L} \boldsymbol{\phi}_i y_i$$

$$\text{where} \quad y_i = \sum_{j=i}^{L} R(x_j, x_{j+1})$$

I now show that, thanks to the algebraic trick of the eligibility vectors $\mathbf{z}_t$, LSTD(1) builds the equivalent $\mathbf{A}$ and $\mathbf{b}$ fully incrementally—without having to store the trajectory while waiting to observe the eventual outcome y_i. Please refer to Table 6.1.1 (p. 140) for the definition of the algorithm.

Proof. With simple algebraic manipulations, the sums built by LSTD(1)'s $\mathbf{A}$ and $\mathbf{b}$ telescope neatly into $\mathbf{A}_{\mathrm{LR}}$ and $\mathbf{b}_{\mathrm{LR}}$, as follows:

$$\begin{aligned}
\mathbf{A} &= \sum_{i=0}^{L} \mathbf{z}_t (\boldsymbol{\phi}_i - \boldsymbol{\phi}_{i+1})^{\mathsf{T}} \\
&= \sum_{i=0}^{L} \Big(\sum_{j=0}^{i} \boldsymbol{\phi}_j\Big)(\boldsymbol{\phi}_i - \boldsymbol{\phi}_{i+1})^{\mathsf{T}} && \textit{(by definition of } \mathbf{z}_t\textit{)} \\
&= \sum_{i=0}^{L} \sum_{j=0}^{i} \boldsymbol{\phi}_j \boldsymbol{\phi}_i^{\mathsf{T}} - \sum_{i=0}^{L} \sum_{j=0}^{i} \boldsymbol{\phi}_j \boldsymbol{\phi}_{i+1}^{\mathsf{T}}
\end{aligned}$$

$$
\begin{aligned}
&= \big(\boldsymbol{\phi}_0\boldsymbol{\phi}_0^\mathsf{T} + \sum_{i=1}^{L}\sum_{j=0}^{i}\boldsymbol{\phi}_j\boldsymbol{\phi}_i^\mathsf{T}\big) - \big(\sum_{k=1}^{L+1}\sum_{j=0}^{k-1}\boldsymbol{\phi}_j\boldsymbol{\phi}_k^\mathsf{T}\big) && \textit{(substituting } k \equiv i+1\textit{)} \\
&= \boldsymbol{\phi}_0\boldsymbol{\phi}_0^\mathsf{T} + \sum_{i=1}^{L}\sum_{j=0}^{i}\boldsymbol{\phi}_j\boldsymbol{\phi}_i^\mathsf{T} - \sum_{k=1}^{L}\sum_{j=0}^{k-1}\boldsymbol{\phi}_j\boldsymbol{\phi}_k^\mathsf{T} && \textit{(since } \boldsymbol{\phi}_{L+1} \stackrel{\text{def}}{=} \boldsymbol{0}\textit{)} \\
&= \boldsymbol{\phi}_0\boldsymbol{\phi}_0^\mathsf{T} + \sum_{i=1}^{L}\big(\sum_{j=0}^{i}\boldsymbol{\phi}_j\boldsymbol{\phi}_i^\mathsf{T} - \sum_{j=0}^{i-1}\boldsymbol{\phi}_j\boldsymbol{\phi}_i^\mathsf{T}\big) && \textit{(substituting } i \equiv k\textit{)} \\
&= \sum_{i=0}^{L}\boldsymbol{\phi}_i\boldsymbol{\phi}_i^\mathsf{T} \\
&= \mathbf{A}_{\text{LR}}, \quad \text{as desired;}
\end{aligned}
$$

and

$$
\begin{aligned}
\mathbf{b} &= \sum_{i=0}^{L}\mathbf{z}_i R(x_i, x_{i+1}) \\
&= \sum_{i=0}^{L}\big(\sum_{j=0}^{i}\boldsymbol{\phi}_j\big) R(x_i, x_{i+1}) && \textit{(by definition of } \mathbf{z}_t\textit{)} \\
&= \sum_{i=0}^{L}\sum_{j=0}^{L} 1(j \le i)\boldsymbol{\phi}_j R(x_i, x_{i+1}) && \textit{(where } 1(\textsf{\textit{True}}) \stackrel{\text{def}}{=} 1,\ 1(\textsf{\textit{False}}) \stackrel{\text{def}}{=} 0\textit{)} \\
&= \sum_{j=0}^{L}\sum_{i=0}^{L} 1(j \le i)\boldsymbol{\phi}_j R(x_i, x_{i+1}) \\
&= \sum_{j=0}^{L}\boldsymbol{\phi}_j \sum_{i=j}^{L} R(x_i, x_{i+1}) \\
&= \mathbf{b}_{\text{LR}}, \quad \text{as desired.}
\end{aligned}
$$

These reductions prove that the contributions to $\mathbf{A}$ and $\mathbf{b}$ by any single trajectory are identical in LSTD(1) and least-squares linear regression. In both algorithms, contributions from multiple trajectories are simply summed into the matrices. Thus, LSTD(1) and linear regression compute the same statistics and, ultimately, the same coefficients for the approximated value function. □