Report no. 02/06

Neuro-Dynamic Programming for Radiation Treatment Planning

Michael C. Ferris and Meta M. Voelker

Oxford University Computing Laboratory

Permanent address:

Computer Sciences Department, University of Wisconsin,

1210 West Dayton Street, Madison, Wisconsin 53706, USA.

(ferris,voelker@cs.wisc.edu)

In many cases a radiotherapy treatment is delivered as a series of smaller dosages over a period of time. Currently, it is difficult to determine the actual dose that has been delivered at each stage, precluding the use of adaptive treatment plans. However, new generations of machines will give more accurate information of actual dose delivered, allowing a planner to compensate for errors in delivery. We formulate a model of the day-to-day planning problem as a stochastic linear program and exhibit the gains that can be achieved by incorporating uncertainty about errors during treatment into the planning process. Due to size and time restrictions, the model becomes intractable for realistic instances. We show how neuro-dynamic programming can be used to approximate the stochastic solution, and derive results from our models for realistic time periods. These results allow us to generate practical rules of thumb that can be immediately implemented in current planning technologies.

This material is based on research partially supported by the National Science Foundation Grants ACI-0113051 and CCR-9972372, the Air Force Office of Scientific Research Grant F49620-01-1-0040, Microsoft Corporation and the Guggenheim Foundation.

Oxford University Computing Laboratory
Numerical Analysis Group
Wolfson Building
Parks Road
Oxford, England OX1 3QD

March, 2002

1 Introduction

In radiation therapy, ionizing radiation is applied to cancerous tissue, damaging the DNA and interfering with the ability of the cancerous cells to grow and divide [22, 25]. Healthy cells are also damaged by the radiation, but they are more able to repair the damage and return to normal function. Since both cancerous and healthy cells are affected by radiation, dose distributions need to be designed that expose the tumor to enough radiation for treatment while, at the same time, avoid excessive radiation to surrounding healthy tissue and, in particular, nearby organs.

Given a particular delivery mechanism, a treatment plan corresponds to settings of the machine that facilitate the delivery of the target dose distribution. Optimization techniques can be used to design such plans [5, 6, 21, 23]. Typically these problems are complicated due to the ever increasing complexities of the delivery mechanisms [4] and the large amount of data that needs to be manipulated to get sufficient detail of the dose on the target area. While these problems remain at the forefront of cancer treatment planning and many techniques have been proposed for the large varieties of machines (for example, see [8, 9, 10, 12, 14, 17, 20]), many of which take from minutes to hours to solve, we will not focus on this aspect of the problem. Instead, as we now describe, we will look at the day-to-day planning problem and derive target distributions that hedge against errors in the delivery process and assume the aforementioned planning tools will be used on specific machines to approximate these target distributions.

The day-to-day planning problem arises since many cancer patients are treated by a course of radiation over a period of days or weeks. For example, the full dose may be delivered in 20 or so treatments, with 1/20-th of the total dose delivered at each stage. This limits burning and gives the healthy tissue time to recover. As mentioned above, particular planning tools approximate the idealized dose, leading to errors between the planned and delivered dosage. Furthermore, in dividing the radiation dose over a series of treatments, additional errors can be introduced. Many sources can contribute errors to individual treatments, including the re-registration of the patient on the machine, the movement of the patient during treatment, and machine error [11, 24].

At this time, we are unable to determine the dose actually delivered during individual treatments. Imaging devices are currently being developed, though, that can measure the dose as it is being delivered, highlighting where the delivered treatment may be inaccurate. The purpose of this paper is to exploit this knowledge to improve the overall treatment.

The paper aims to generate a deliverable plan for each treatment in the course that compensates over time for movement of the patient and error in the delivery process. We develop a control mechanism for the treatment course, leaving the implementation of the daily dosage to a specialized planning tool. To find the control, we use neuro-dynamic programming, particularly a rollout policy, to improve upon simple heuristic policies.

In the next section, we describe the mathematical framework in which we will be working and techniques for solving the day-to-day planning problem. These techniques include neuro-dynamic programming (NDP) ideas and heuristic policies, one of which is currently in use. We next present examples and discuss their results, showing how the

NDP ideas can improve upon the heuristic policies. Finally, we define rules of thumb, which allow for immediate practical implementations of solutions suggested by NDP while still maintaining most of the improvements.

2 Model Formulations

To describe the problem more precisely, we introduce some notation and a simplified model that captures the salient features of the process. Let $\mathcal{I}$ be a collection of voxels (pixels, points) and let $T(i)$, $i \in \mathcal{I}$ represent the required final dosage (target). Suppose the course lasts N periods (stages), and the actual dose delivered (the state) after k days is x_k. This state evolves as a stationary discrete-time dynamic system:

$$x_{k+1} = f(x_k, u_k, w_k), \quad k = 0, 1, \ldots.$$

Here u_k is the control to be selected from a collection $U(x_k)$, and w_k is a random disturbance drawn from a set W. In the application, we assume that these random disturbances come from errors in the delivery process (such as patient movement) or errors in the setup (such as patient registration errors). For this reason, we assume that w_k corresponds to a shift to u_k. Further, since each treatment is delivered separately, the errors that arise pertain only to a particular treatment and time stage, and so w_k is independent over stages. A key issue to note is that the controls are nonnegative since dose cannot be removed from the patient.

At the end of N stages, the state x_N should minimize a terminal cost G. For ease of exposition we assume that $G(x)$ is a linear combination of the differences between the current dose and the target at each voxel, that is

$$G(x) = \sum_{i \in \mathcal{I}} c(i) \left| x_N(i) - T(i) \right|.$$

Here, the vector c weights the importance of hitting the target value for each voxel. We typically use similar values of c for distinct areas in the target, such as the location(s) of the tumor, sensitive structures like organs, and normal tissue. In practice, larger values of c correspond to tumor areas and/or sensitive structures. This gives us the following mathematical model:

$$\begin{array}{ll} \min_u & E(G(x_N)) \\ \text{subject to} & x_{k+1}(i) = x_k(i) + u_k(i + w_k), \quad \forall i \in \mathcal{I}, k = 0, 1, \ldots, N-1 \\ & u_k \in U(x_k), u_k \geq 0, w_k \in W, \end{array} \tag{2.1}$$

with x_0 given.

2.1 Stochastic Linear Programming

If W is a finite set, then the problem at hand can be formulated as a stochastic linear program whenever the constraints $u_k \in U(x_k)$ are linear relationships. To explain this,

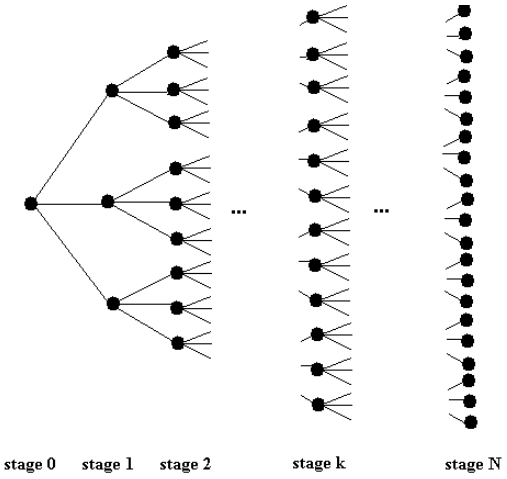


Figure 1: Scenario tree S for the application.

assume that at each time stage k, one of $|W|$ scenarios occurs. If we align the time stages on a horizontal axis, the resulting scenario tree (S) can be depicted as in Figure 1, where as an example we have taken $|W| = 3$. Let $V(k)$ represent the nodes n in this tree S belonging to time stage k, while $p(n)$ denotes the predecessor node of n, i.e., if $n \in V(k)$, then $p(n)$ is the unique element of $V(k-1)$ such that $(p(n), n) \in S$. For this section of the paper, we introduce a slight abuse of notation. We use $x(n, \cdot)$ and $u(n, \cdot)$ to denote the state and control at a node n, whereas $x_k(\cdot)$ and $u_k(\cdot)$ denote the state and control at time stage k.

Using this notation, at each node n in the scenario tree,

$$x(n, i) = x(p(n), i) + u(p(n), i + w(p(n), n))$$

where $w(p(n), n)$ is the shift that is applied to u as we move from $p(n)$ to n. Then, model (2.1) can be reformulated as the following stochastic program:

$$\begin{array}{lrcl} \min & \displaystyle\sum_{n \in V(N)} \Pr(n) \sum_{i \in \mathcal{I}} c(i)\, |x(n,i) - T(i)| & & \\ \text{subject to} & x(n,i) & = & x(p(n),i) + u(p(n), i + w(p(n),n)), \\ & & & \forall n \in V(k), k = 1, \ldots, N, i \in \mathcal{I} \\ & u(n,i) & \geq & 0, \quad \forall n \in S, i \in \mathcal{I} \end{array} \tag{2.2}$$

with $\Pr(n)$ depicting the probability of being at node n. Here we have taken just simple nonnegativity on u; more complex linear relationships are also feasible. Standard techniques can be used to reformulate (2.2) as a linear program. When N is small and the number of possible shifts are small, then a wealth of techniques for such problems can be applied [3, 16]. However, when $\mathcal{I}$ or N becomes large, the size of the problem soon becomes too great, and we have to resort to approximation schemes for the solution.

For the prototype examples of Section 3 where we have $|I| = 9$ and $N = 4, 5, 6, 10, 14$ and 20, we attempted to solve model (2.2) exactly. We formulated the model in the

GAMS [7] modeling language and used the CPLEX [15] barrier method to obtain the solutions (since this significantly outperformed all simplex options). Even with small numbers of time stages, the problems became quite large. With 4 time stages, the model consisted of 18271 equations, 14059 variables and 48250 nonzeros; solution times were around 25 seconds. With 5 time stages, the model consisted of 91396 equations, 70309 variables and 241375 nonzeros; solution times ranged from about two-and-a-half minutes to a little over 3 minutes. With 6 time stages, the model consisted of 457021 equations, 351559 variables and 1207000 nonzeros; solution times ranged from about one-half hour to around 70 minutes. Due to the storage requirements, this became intractable for more than 6 time stages. While more powerful machines would extend the number of stages somewhat, the exponential growth precludes solutions for $N = 10, 14$ or 20. It may be possible to apply scenario reduction or sampling techniques to this model [13, 18], but this was not explored here.

2.2 Dynamic Programming

Dynamic programming is another method that can be used to solve model (2.1). Since each decision impacts later decisions, the choice of control at each time stage contributes to the final cost. Thus, to find the optimal control at each stage, we must consider future states. Under dynamic programming, we apply backward recursion: start at the last stage and determine the optimal control to apply for each state; then consider the second-to-last stage, and so on, working backward through the stages.

As a means of determining the optimal control for a state, we consider the costs-to-go of various policies from the current state. The cost-to-go for a particular policy is the optimal cost over all remaining stages, starting from the current state and applying the given policy. Starting in stage k from state x_k and using the policy $\pi = \{u_0, u_1, \ldots, u_{N-1}\}$, the cost-to-go is [1]:

$$J_k(x_k) = E[G(x_N) + \sum_{i=k}^{N-1} g(x_i, u_i(x_i), w_i)],$$

where $g(x_i, u_i(x_i), w_i)$ represents the immediate cost of applying control $u_i(x_i)$. As noted in [1, 2], the cost-to-go functions satisfy the dynamic programming recursion

$$J_k(x) = E[g(x, u_k(x), w) + J_{k+1}(f(x, u_k(x), w))] \tag{2.3}$$

with initial condition

$$J_N(x) = G(x). \tag{2.4}$$

However, since we are attempting to construct the optimal policy $\bar{\pi}$, the individual controls $\bar{u}_k$ are not known a priori. These controls can be found by backward recursion by considering the costs-to-go over all possible controls:

$$\bar{u}_k(x) = \arg \min_{u \in U(x)} E[g(x, u, w) + J_{k+1}(f(x, u, w))]. \tag{2.5}$$

If we wanted to use equations (2.3), (2.4), and (2.5) in the radiation treatment application, we would need to calculate $J_k(x)$ and $\bar{u}_k(x)$ for every possible state x at each stage k. A standard technique for doing this is to discretize the state space for x and form a lookup table for J_k and u_k over this discretization. For each voxel in the target, we would need a minimum of N discretizations to allow for a simple heuristic policy, the constant policy (described later in Section 2.4), to be implemented in the lookup table. Even for the simple examples that we describe in Section 3, this becomes unmanageable.

2.3 Neuro-Dynamic Programming (NDP)

Another technique to approximately solve (2.1) is to apply a rollout policy [1, 2]. Rather than starting at stage N, we start at stage 0 and work our way forward, determining the policy $\bar{\pi}$ one stage at a time (we "roll" out the policy). This is much closer to the way a decision maker would work in practice: only today's decision is needed precisely; the remaining decisions can be approximated. The rollout policy uses equation (2.5) to find the control $\bar{u}_k$ to apply at each stage k, but using an approximation to J_{k+1} instead of the actual function. This approximation is built by applying the particular control u at stage k and a control from some base (heuristic) policy at all future stages. Then, J_{k+1} is calculated using these controls and methods such as simulation (which we use) or neural networks.

In effect, this is an example of an on-line policy choice. In the practical setting, we approximate the future by simulation and choose the policy to apply right now by optimization. After applying this policy, we wait for time to elapse and repeat the same process at the next stage. For a particular radiation therapy, the choice of the current control may itself involve a lengthy optimization, and the error produced in delivery will be provided to the decision maker automatically by the machines that are currently under development.

In the radiation treatment application that is expressed in model (2.1), we assume no immediate costs in applying individual controls and so $J_k(x_k) = E[G(x_N)]$. To apply the rollout policy, we begin by choosing the base policy for the calculation of the costs-to-go J_{k+1}. As the base policy is applied at all later stages, it should be a heuristic policy that performs well (we use the reactive policy, described in Section 2.4). Then we simulate the effects on the system as time advances. In principle, we calculate for each one-stage control $u_k \in U(x_k)$ the Q-factor,

$$Q_k(x_k, u_k) := E[g(x, u, w) + J_{k+1}(f(x, u, w))]$$

using simulation by applying u_k at stage k and the reactive policy at all later stages. The (approximated) Q-factor for each control is then the expected terminal cost from the simulation using that control at the current time stage.

To choose between controls, we need to evaluate differences between $Q_k(x_k, u)$ for each $u \in U(x_k)$. Since simulation is involved, this will be prone to errors. These errors can be alleviated somewhat using the same realization of w when calculating all Q-factor

differences [1]. Thus, we calculate the average differences in the expectations for every $u \in U(x_k)$. Note that all the controls can be compared directly like this as individual controls are only applied at the current time stage. By applying the base policy at all future time stages, we can test the effectiveness of each control against the others for the current stage. In addition, since the controls are applied simultaneously, they are applied under the same shifts and so the comparison is done for the same realization of w.

2.4 Heuristic Policies

One approach to overcome the computational burden outlined above is to apply heuristic policies. Besides offering an alternative to the techniques described above, we also use heuristic policies to define the (finite) set $U(x_k)$ for the NDP rollout approach.

Several heuristic control techniques immediately spring to mind. First, there is the simple plan to deliver

$$u_k := T/N$$

at each stage and not account at all for disturbances in the delivery. This plan can be used when treatment errors cannot be measured directly, and is currently the method of choice. We refer to this plan as the constant policy. Note that the implementation on a particular machine of this policy only needs one optimization to be performed at the start of the process. However, even when a voxel has been overdosed at stage k, the constant policy continues to add dose at subsequent time stages. An alternative is to only add dose if the current dose is less than the target dose. We refer to this modification as the constant-plus policy. Surprisingly, the simulations show that this has little effect on overall error.

An alternative to the constant policies is to attempt to compensate for the error delivered in the previous time by spreading the error over the remaining time stages. At each time stage, we divide the residual over the remaining time stages:

$$u_k := \max(0, T - x_k)/(N - k).$$

We refer to this plan as the reactive policy. Since the reactive policy takes into consideration the residual at each time stage, we expect that the reactive policy will perform better than the constant policies. Note, though, that the reactive policy requires knowledge of x_k and replanning at every stage k.

We show later in this paper how the constant and reactive heuristic policies perform on a variety of examples. We also show how the NDP rollout approach improves upon these results.

To apply the NDP rollout approach, we require a rich collection of heuristics for the finite set $U(x_k)$. We use the constant, constant-plus, and reactive policies, but we also use what we refer to as categorical policies. For these policies at stage k, we calculate the residual target for each voxel i by $\max\{0, T(i) - x_k(i)\}$. Then, the voxels are divided into three categories by comparing their residual target to the maximum residual:

$$\max_{i \in \mathcal{I}} \max\{0, T(i) - x_k(i)\}.$$

The three categories correspond to voxels whose residual target is less than 40% (low residual), between 40% and 70% (medium residual), and greater than 70% (high residual) of this maximum value. In each category, we apply one of three controls. Either we apply 0 dosage, 0.4 of the residual target, or $1/(N-k)$ of the residual target. This yields an additional 26 policies (as the reactive policy is the categorical policy with $1/(N-k)$ applied in each category).

Note that the practical implementation of a policy generated by NDP using these controls for $U(x_k)$ is exactly the same as that of the reactive policy. First of all, knowledge of x_k is required. Given this information we can calculate the actual dose that should be delivered at each voxel $i \in \mathcal{I}$ by determining which category the voxel resides in, and then multiplying the residual $T(i) - x_k(i)$ by the categorical multiplier. Knowing the dose at every voxel i is all the data that is required to specify a plan optimization that determines how to implement that particular dose on a specific machine. As mentioned in the introduction, we allow existing planning tools to perform this step, and we believe this is a key advantage of our approach.

In actual treatment plans, individual doses are subject to an upper bound, applied in order to limit burning and allow for healthy tissue to recover between treatments. For this reason, we assign a cutoff value that restricts the dose prescribed by each control to such an upper bound. For testing purposes, we set the cutoff to be $2T_{\max}/N$, which is double the dose prescribed by the constant policy in the worst case. Such a value allows for a large dose to be prescribed, while still ensuring that the dose is not unreasonably large. Although this is chosen for the application, very little changes if this upper bound is not applied.

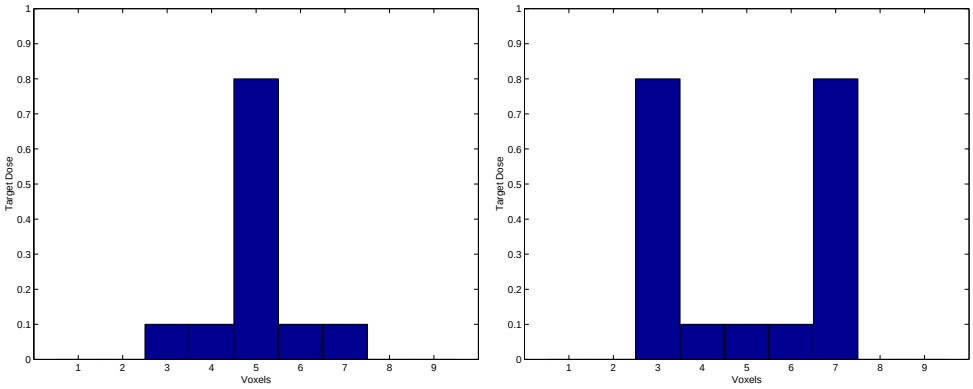
3 Examples and Results

We consider two simple, one-dimensional targets under different weighting and probability distributions, pictured in Figure 2.

For both targets, $\mathcal{I} = \{1, 2, \ldots, 9\}$ and we allow a maximum shift of 2 voxels. In both targets, the "spikes" of dose 0.8 represent tumor locations. Thus, it is important that these areas receive as much of the 0.8-prescribed dose as possible, and so these areas will have a relatively high weighting in the objective. The 0.1 areas can represent sensitive structures (which should be exposed to a minimum amount of radiation) or normal tissue, depending upon the particular weighting scheme employed. We apply 3 different weighting schemes to the spike target. Moving from easiest to hardest, these schemes are:

- the smooth weighting:
$$c = [1, 1, 1, 1, 10, 1, 1, 1, 1],$$
which only enforces the 0.8-dosage, allowing for more variation in the other voxels (including a "building" up to the spike);

- the nonsymmetric weighting:
$$c = [1, 1, 1, 1, 10, 5, 5, 1, 1],$$



(a) Spike target (b) Double spike target

Figure 2: Example targets

which allows for a build-up to the spike on the left-hand-side, but enforces the spike structure rigidly on the right-hand-side; and

- the spike weighting:

$$c = [1, 1, 5, 5, 10, 5, 5, 1, 1],$$

which enforces the spike structure rather rigidly.

For the double spike target, we apply the double spike weighting scheme:

$$c = [1, 1, 10, 5, 5, 5, 10, 1, 1],$$

which enforces high dosage on the target edges and the low dosage in the center. The examples have been chosen to simulate practical cases of interest in the application area.

For the targets above, we also consider three different probability distributions for the shifts. The low volatility examples have

$$w_k = \begin{cases} -2 & \text{with probability } 0.02 \\ -1 & \text{with probability } 0.08 \\ 0 & \text{with probability } 0.8 \\ 1 & \text{with probability } 0.08 \\ 2 & \text{with probability } 0.02. \end{cases}$$

for every stage k. The medium volatility examples have

$$w_k = \begin{cases} -2 & \text{with probability } 0.05 \\ -1 & \text{with probability } 0.15 \\ 0 & \text{with probability } 0.6 \\ 1 & \text{with probability } 0.15 \\ 2 & \text{with probability } 0.05. \end{cases}$$

for every stage k. The high volatility examples have

$$w_k = \begin{cases} -2 & \text{with probability } 0.05 \\ -1 & \text{with probability } 0.25 \\ 0 & \text{with probability } 0.4 \\ 1 & \text{with probability } 0.25 \\ 2 & \text{with probability } 0.05. \end{cases}$$

for every stage k. While it is hard to estimate the volatilities present in the given application, our results of Section 4 are fairly insensitive to these choices.

As described in Section 2.3, we use a simulation code at every stage to determine the optimal $\bar{u}_k \in U(x_k)$ by calculating differences in Q-factors under the same realizations w. The simulation code we use generates 10000 paths through the simulation tree between stage k and stage N. For each path, the Q-factor differences are calculated; at the end, the average of these differences determines $\bar{u}_k$. The same code can be used to simulate the costs of the individual heuristic policies. Essentially, for this we ensure that $U(x_k)$ is the appropriate singleton.

While we described how to develop the rollout policy in an on-line fashion in the previous section, we also need to evaluate the effectiveness of our procedure. To effect this, we apply an outer simulation that simulates paths through the scenario tree. The outer simulation evolves one stage at a time, therefore assuming that x_k is known at each stage k. We use the inner simulation (for Q-factor differences) to determine $\bar{u}_k$. The outer simulation then generates w_k and thus forms x_{k+1}. After N stages, x_N is known and the terminal cost can be evaluated for this particular path through the scenario tree. The outer simulation generates 20000 paths to form an expected value for the terminal cost.

Running the outer simulation (with repeated inner simulations needed inside), results in a great deal of computation and long running times. To deal with this efficiently, we submitted the outer simulations to Condor [19], a network resource manager. Once a job is submitted to the Condor queue, Condor searches for idle network machines. If one is found, then the simulation starts executing on that machine; otherwise the simulation is held until sufficient resources are freed. In addition, Condor migrates jobs or checkpoints them (for later continuation) when the machine's owner returns or resources become scarce.

Figures 3, 4, and 5 display simulated results for each example under the three probability distributions. For each graph, the constant policy, reactive policy, and NDP rollout policy results are displayed, as well as the optimal results for time stages 4, 5, and 6. The optimal results come from solving model (2.2) exactly, as explained in Section 2.1. Note the change of vertical scale between the three figures.

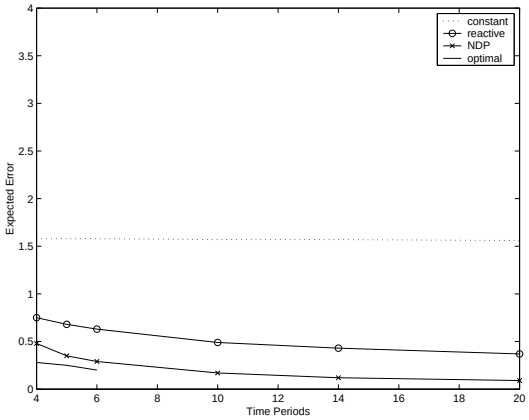
Comparing Figures 3, 4, and 5, we note the remarkable similarities. While the vertical scales are larger as the volatility increases, general conclusions are easy to draw. Firstly, the alphabetic ordering of targets (a) to (d) are increasingly difficult and lead to larger errors, independent of the optimization scheme chosen. Secondly, in all cases and for all optimization schemes, as volatility increases, so does the error.

Common to all examples is the poor performance of the constant policy. The reactive policy performs better than the constant policy, but not as well as the NDP rollout policy. The level of improvement, though, depends upon the difficulty of the target and the volatility. In the low and medium volatility spike examples, the NDP results are much closer to the optimal results than in the high volatility examples, particularly in the double spike example (Figure 5(d)). However, the NDP results decrease at a faster rate than the optimal results. Thus, more time periods are beneficial. These decreases level off at later time stages, exhibiting decreasing returns for more time stages.

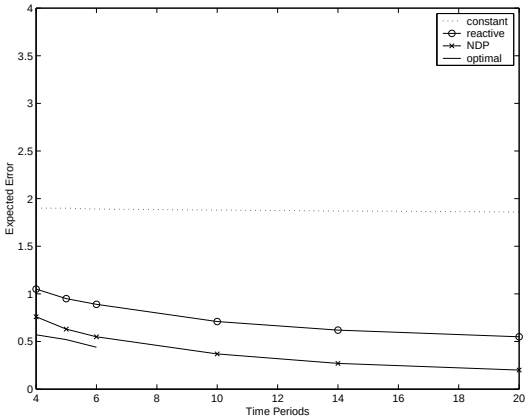
Focusing on the low volatility examples of Figure 3, we see that the reactive policy gives a large improvement over the constant policy — the error is nearly halved. NDP does even better, yielding about a 50% drop in the reactive policy error at larger time stages (the exact improvements over the constant policy are given in Table 7), and achieving near-optimal results at smaller time stages. As time advances, the improvement for both the NDP and reactive policies becomes greater: where constant remains almost level, reactive and NDP continue to drop as we move to later time stages. Further, NDP decreases faster than the optimal results do, suggesting that it may become optimal at later time stages.

The medium volatility examples of Figure 4 show less improvement in the NDP results. Although the constant policy appears level, the reactive policy gives slightly less improvement — not quite 50%. We also see slightly less than a 50% drop in the NDP results over the reactive policy results at later time stages, due to its faster rate of decrease. Although the NDP results are not as close to the optimal results as in the low volatility examples, again we see that the NDP error decreases faster than the optimal error, suggesting that the NDP policy may be close to the optimal error at some later time stage.

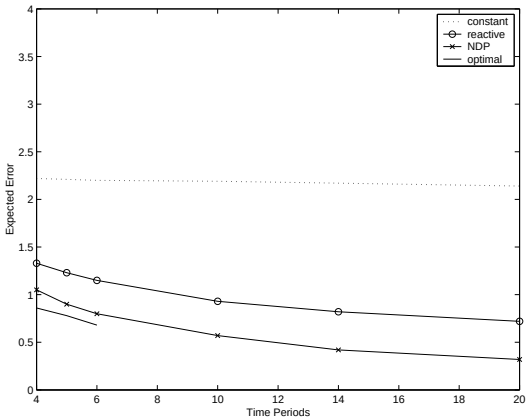
The high volatility examples of Figure 5 show the greatest errors of all of the examples. Again, the constant policy appears to be level and is much larger than in the previous examples. We see less improvement in the NDP results: in Figures 5(b) and 5(d), the NDP improvement is approximately one-third over the constant policy; Figure 5(a) is better (about one-half improvement), but Figure 5(c) is worse (the constant results, though, are much closer to optimal here). Apart from this latter case, the NDP results are far from the optimal results, lying closer to the reactive policy than to the optimal policy. We see that the NDP results do improve faster than the reactive results so this may change at later time stages.



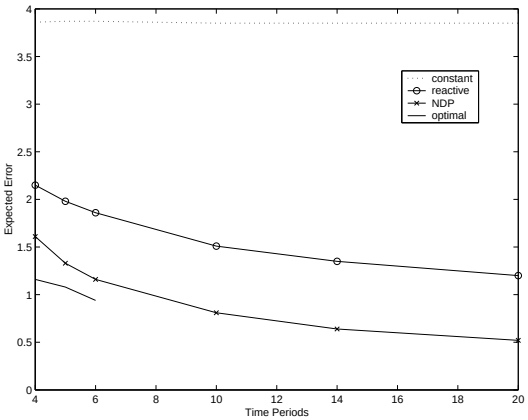
(a) Spike target with smooth weighting.



(b) Spike target with nonsymmetric weighting.

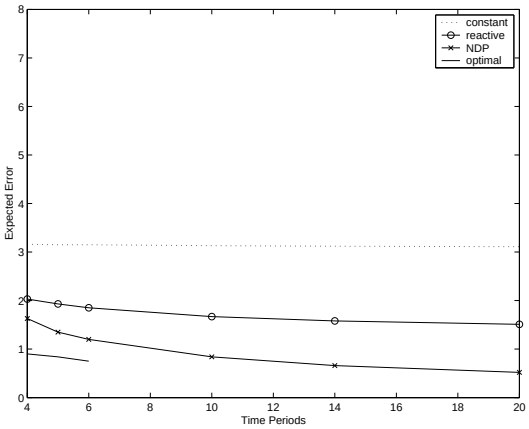


(c) Spike target with spike weighting.

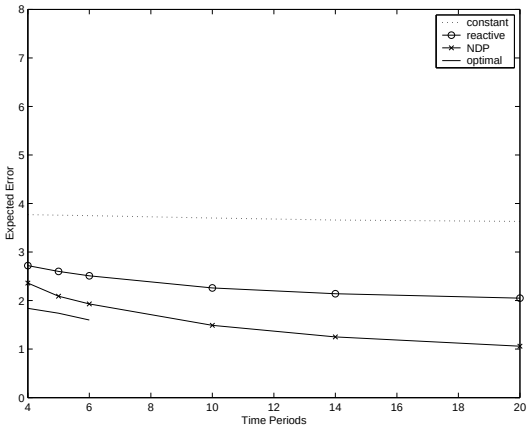


(d) Double spike target with double spike weighting.

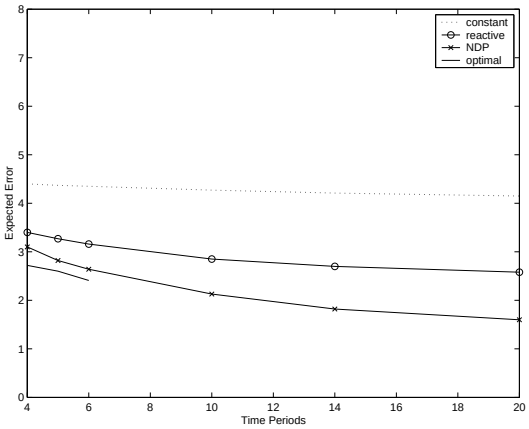
Figure 3: Examples under low volatility.



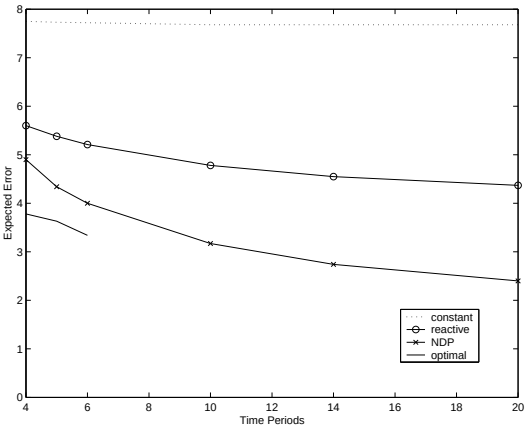
(a) Spike target with smooth weighting.



(b) Spike target with nonsymmetric weighting.

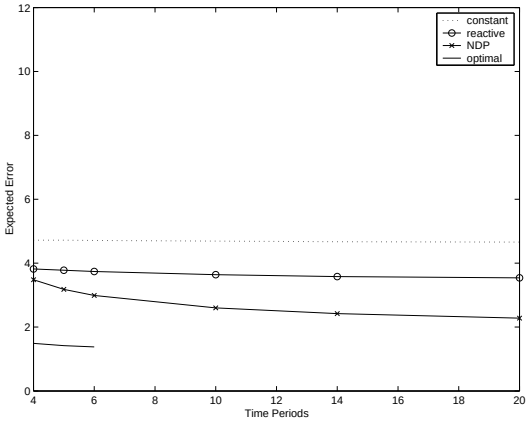


(c) Spike target with spike weighting.

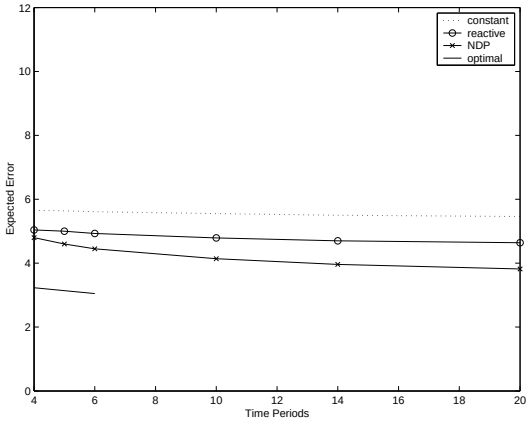


(d) Double spike target with double spike weighting.

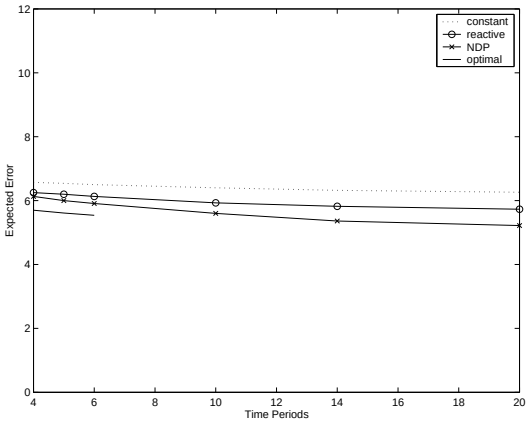
Figure 4: Examples under medium volatility.



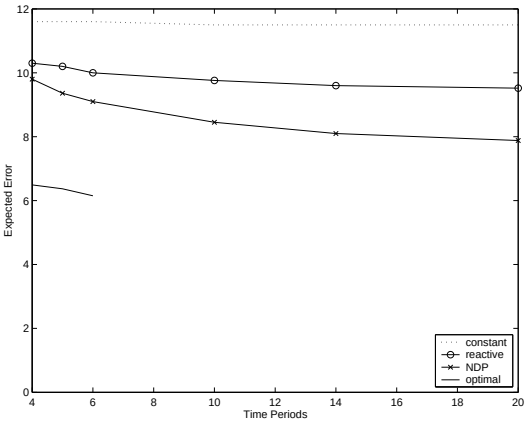
(a) Spike target with smooth weighting.



(b) Spike target with nonsymmetric weighting.



(c) Spike target with spike weighting.



(d) Double spike target with double spike weighting.

Figure 5: Examples under high volatility.

The high volatility examples are the most difficult of the examples; in these examples, we are more likely to see an error shift than not. As a result, although we use information regarding earlier errors, it is difficult to account for future errors. The optimal results show that, given unlimited possibilities for policy choices, we can often improve greatly upon the currently-used constant policy. However, as NDP is limited by a finite number of policy choices — in particular, a choice between 0, $1/(N-k)$ or 0.4 of the current target residual — it is more difficult for the NDP policy to achieve such substantial improvements.

In addition to three-category policy choices, we also experimented with two-category policy choices. Under these policies, the voxels were classified as either less than 50% of the maximum residual or more than 50% of the maximum residual. To maintain approximately the same number of policies, we allowed five choices for each category (resulting in a total of 27 policies, including the constant policies). In one experiment, we allowed for small multiples: 0, $1/(N-k)$, 0.01, 0.1, or 0.4. In another experiment, we allowed for large multiples: 0, $1/(N-k)$, 0.1, 0.4, or 0.6. Applying these policies to the double spike example (the hardest example), we found very little change in the NDP results. The small-multiple category choices returned approximately the same results as the three-category choices, while the large-multiple category choices returned slightly better results but nothing visually significant on the plot.

These results suggest that significant improvements over the presented NDP results cannot be achieved while choosing from among approximately 30 policies. Enriching the policy set by combining the two-category policies with the three-category policies, or moving to five-category policies (for example) seems to be the only way to improve upon the NDP results. Other policies may come from previous real-life plans or other planning systems.

Note that in addition, we also experimented with many more examples, including different targets, different weighting schemes, and larger targets. The results from these other examples were qualitatively the same. We did find, though, that for high volatility examples, constant weighting ($c(i) = 1, \forall i \in \mathcal{I}$) resulted in significantly underdosing the target. This strongly suggests that the use of an appropriate weighting scheme to focus the treatment is imperative.

4 Off-Line Planning

While building simple models and analyzing their properties can lead to great insight into the application at hand, it is important to draw definitive conclusions that are applicable to the real problem. In this section, we endeavor to derive policies that are directly implementable in the radiation treatment planning arena.

Besides testing the NDP model, the real-life (outer) simulation is also useful for off-line planning. In Section 3, we compared the on-line planning schemes, that is, we assumed that the controls were determined in between treatments as we moved to the next stage. Only at the end of the treatment period would we have a complete policy. Off-line planning, on the other hand, assumes that a policy is pre-defined, prior to the

Stage	Low Volatility	Med. Volatility	High Volatility	Simple Rule
1–2	$(0, 0, 0.4)$	$(0, 0, 0.4)$	$(0.4, 0.4, 0.4)$	$(0, 0, 0.4)$
3	constant-plus	$(0.5, 0.5, 0.5)$	$(0.5, 0.5, 0.5)$	$(0.5, 0.5, 0.5)$
4	$(1, 1, 1)$	$(1, 1, 1)$	$(1, 1, 1)$	$(1, 1, 1)$

Table 1: Rules of thumb for 4 time period examples.

beginning of treatment.

To find pre-defined policies, we look for policies that are good for most, if not all, of the examples. For a particular example, the outer simulation gives a series of possible policies to apply. By counting the number of times each control is chosen at each time stage, we have an idea of how important that control is for that example at that time stage. Averaging these control counts across examples with the same volatility and choosing the ones with the largest counts at each time stage, we determine a generalized policy for each volatility. We refer to this generalized policy as the "rule of thumb" policy for that volatility. These rules of thumb allow us to remove the target dependence from the simulation and also provide us with a pre-defined plan to use for a particular volatility.

We can take the generalization further and remove dependence on the volatility by averaging the control counts across volatilities as well. We refer to the resulting policy as the "simple rule of thumb". Since the total number of time stages N affects which controls are chosen and when, we define rules of thumb and simple rules of thumb for each N. The rules of thumb and simple rules of thumb for $N = 4, 5, 6, 10, 14$ and 20 are given, respectively, in Tables 1, 2, 3, 4, 5 and 6. In these tables, the categorical policies (including the reactive policy) are given as triplets. In these triplets, the first entry corresponds to the low residual areas; the second entry corresponds to medium residual areas; and the third entry corresponds to the high residual areas. These entries correspond to the multiplier of the residual that is used at all voxels in that area.

Note that if the policy pool U is changed, the simulations must be rerun and this process must be repeated on the new results in order to determine appropriate rules of thumb and simple rules of thumb.

Examining the tables, we notice some general trends in the control choices. First of all, within each table, the controls become more aggressive as we near the final time stages, generally moving from controls in which only the high residual areas are dosed, to controls in which all areas are dosed. Typically, we use the first half of the time periods to work aggressively on the high residual areas and ignore the other areas. Exceptions to this are the high volatility rules for small time stages (4,5,6); in these cases, we apply dose to all areas. This probably comes from the fact that we are likely to make an error and we have very little time to correct it.

Controls in the middle stages vary but tend to focus on both the medium and high residual areas first. Later stages focus on all three categories, ending in every case

Stage	Low Volatility	Med. Volatility	High Volatility	Simple Rule
1–2	$(0, 0, 0.4)$	$(0, 0, 0.4)$	$(0.4, 0.4, 0.4)$	$(0, 0, 0.4)$
3	constant-plus	$(0, 0, 0.4)$	$(0.4, 0.4, 0.4)$	constant
4	constant-plus	$(0.5, 0.4, 0.5)$	$(0.5, 0.5, 0.5)$	$(0.5, 0.4, 0.5)$
5	$(1, 1, 1)$	$(1, 1, 1)$	$(1, 1, 1)$	$(1, 1, 1)$

Table 2: Rules of thumb for 5 time period examples.

Stage	Low Volatility	Med. Volatility	High Volatility	Simple Rule
1–4	$(0, 0, 0.4)$	$(0, 0, 0.4)$	$(0.4, 0.4, 0.4)$	$(0, 0, 0.4)$
5	$(0.5, 0, 0.5)$	$(0.5, 0.5, 0.5)$	$(0.5, 0.5, 0.5)$	$(0.5, 0, 0.5)$
6	$(1, 1, 1)$	$(1, 1, 1)$	$(1, 1, 1)$	$(1, 1, 1)$

Table 3: Rules of thumb for 6 time period examples.

Stage	Low Volatility	Med. Volatility	High Volatility	Simple Rule
1–5	$(0, 0, 0.4)$	$(0, 0, 0.4)$	$(0, 0, 0.4)$	$(0, 0, 0.4)$
6	$(0, 0, 0.4)$	$(0, 0.2, 0.4)$	$(0, 0.2, 0.4)$	$(0, 0.2, 0.4)$
7	$(0, 0.25, 0.4)$	$(0, 0.25, 0.4)$	$(0, 0.25, 0.4)$	$(0, 0.25, 0.4)$
8	$(0, 0.33, 0.4)$	$(0, 0.4, 0.4)$	$(0.4, 0.4, 0.4)$	$(0.4, 0.33, 0.4)$
9	$(0, 0.4, 0.5)$	$(0.5, 0.4, 0.5)$	$(0.5, 0.5, 0.5)$	$(0.5, 0.4, 0.5)$
10	$(1, 1, 1)$	$(1, 1, 1)$	$(1, 1, 1)$	$(1, 1, 1)$

Table 4: Rules of thumb for 10 time period examples.

Stage	Low Volatility	Med. Volatility	High Volatility	Simple Rule
1–9	$(0, 0, 0.4)$	$(0, 0, 0.4)$	$(0, 0, 0.4)$	$(0, 0, 0.4)$
10	$(0, 0.4, 0.4)$	$(0, 0, 0.4)$	$(0, 0.4, 0.4)$	$(0, 0, 0.4)$
11	$(0, 0.4, 0.4)$	$(0, 0.4, 0.4)$	$(0, 0.4, 0.4)$	$(0, 0.4, 0.4)$
12	$(0, 0.4, 0.4)$	$(0.4, 0.4, 0.4)$	$(0.4, 0.4, 0.4)$	$(0.4, 0.4, 0.4)$
13	constant-plus	$(0.5, 0.5, 0.5)$	$(0.5, 0.5, 0.5)$	$(0.5, 0.5, 0.5)$
14	$(1, 1, 1)$	$(1, 1, 1)$	$(1, 1, 1)$	$(1, 1, 1)$

Table 5: Rules of thumb for 14 time period examples.

Stage	Low Volatility	Med. Volatility	High Volatility	Simple Rule
1–9	$(0, 0, 0.4)$	$(0, 0, 0.4)$	$(0, 0, 0.4)$	$(0, 0, 0.4)$
10	$(0, 0.09, 0.4)$	$(0, 0.09, 0.4)$	$(0, 0.09, 0.4)$	$(0, 0.09, 0.4)$
11	$(0, 0.1, 0.4)$	$(0, 0.1, 0.4)$	$(0, 0.1, 0.4)$	$(0, 0.1, 0.4)$
12	$(0, 0.11, 0.4)$	$(0, 0.11, 0.4)$	$(0, 0.4, 0.4)$	$(0, 0.11, 0.4)$
13	$(0, 0.4, 0.4)$	$(0, 0.125, 0.4)$	$(0, 0.4, 0.4)$	$(0, 0.4, 0.4)$
14–17	$(0, 0.4, 0.4)$	$(0, 0.4, 0.4)$	$(0, 0.4, 0.4)$	$(0, 0.4, 0.4)$
18	constant-plus	constant-plus	$(0.4, 0.4, 0.4)$	$(0.4, 0.4, 0.4)$
19	constant-plus	constant-plus	$(0.5, 0.5, 0.5)$	$(0.5, 0.5, 0.5)$
20	$(1, 1, 1)$	$(1, 1, 1)$	$(1, 1, 1)$	$(1, 1, 1)$

Table 6: Rules of thumb for 20 time period examples.

aggressively with the reactive policy (to attempt to apply all of the remaining dose). An interesting question arises as to whether the low volatility rules for $N = 4, 5, 14$ and 20 follow this general trend. In these three cases, the rule of thumb makes use of the constant-plus policy in the middle and/or later time stages. While $1/N$-th of the original target dose is a seemingly rather small amount, we claim that it is very likely to be an aggressive control at the later time stages. This is because most of the earlier controls will have hit the target correctly (because of the low volatility), resulting in the remaining residual being small, and hence the small fraction of the original dose is in fact a large dose in comparison to the required residual. Clearly, in this case, the removal of overdosing (the difference between constant and constant-plus) is important.

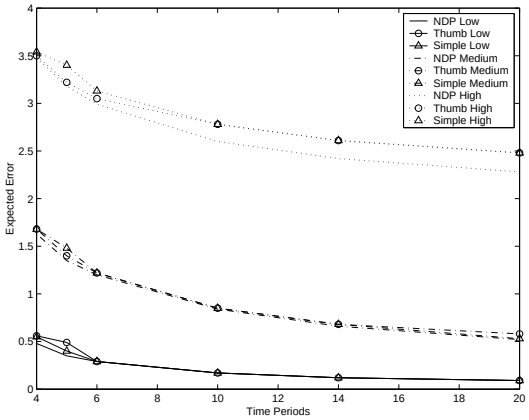
Focusing on the rules of thumb, the policies either use the same or more aggressive controls for the same time stages as the volatility increases. The simple rule policies use either these same controls or combinations of the controls. An exception occurs in the simple rule for $N = 5$, at stage 3. In this case, we find a rather strange choice, the constant policy, which does not even adjust for overdosing. In this case, the controls used in the rules of thumb varied so greatly that the only control the three had in common was the constant policy and we therefore believe this is a statistical anomaly.

Since they are generalizations, we expect that the rules of thumb and the simple rule of thumb for each N will give perform worse than the (on-line) NDP rollout policy. This is the case, although the differences tend to be so small that they are not noticeable. Figure 6 compares the NDP rollout results to the rules of thumb and simple rules of thumb for each example. In addition Table 7 shows the percentage decrease achieved by the reactive policy, the NDP rollout policy, rules of thumb and simple rules of thumb over the currently-used constant policy at 20 time stages.

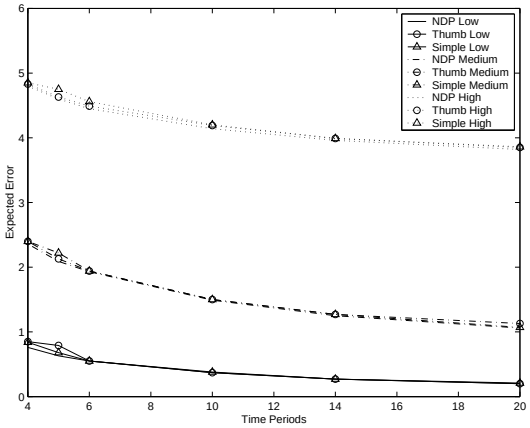
The rules of thumb and simple rules of thumb are almost always indistinguishable from one another. Exceptions occur with small N, where we do not have enough time periods to make up for error from the generalization. We also see a difference in the high volatility smooth weighting example of Figure 6(a). It is not surprising that one target suffers under a generalization built from considering all targets. The smooth weighting, in being the easiest target, probably does not require the same controls as the other targets, and so suffers particularly in the high volatility case where things are more likely to go wrong.

Table 7 verifies the results shown in Figure 6. In all cases, the NDP, rules of thumb and simple rules of thumb improved upon the reactive policy results significantly. However, their percentage decrease over the constant policy varies very little between the three of them. This suggests that very little is sacrificed in moving to the generalized simple rules of thumb.

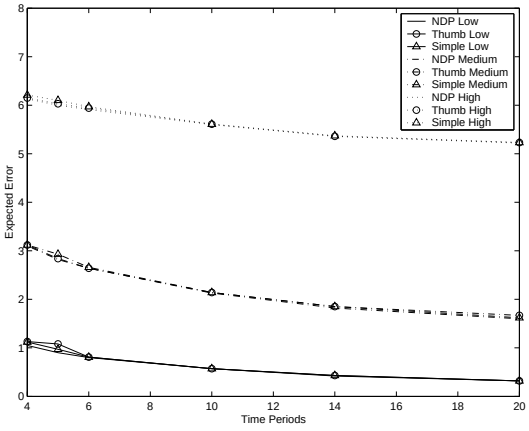
An interesting case arises for the $N = 20$, medium volatility rule of thumb. Perhaps not noticeable in Figure 6, but seen from Table 7, is the fact that the simple rule of thumb slightly outperforms the medium volatility rule of thumb. Since the simple rule of thumb is a generalization of the rules of thumb for all of the volatilities, we would expect just the opposite (in the other examples, the two actually are tied). Upon investigation, we found that this comes from the difference between the magnitudes in the errors for the rule of thumb and the simple rule of thumb. At stage 18, the constant-plus policy



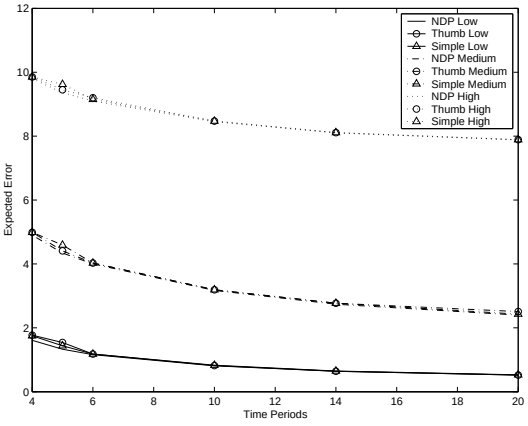
(a) Spike target with smooth weighting.



(b) Spike target with nonsymmetric weighting.



(c) Spike target with spike weighting.



(d) Double spike target with double spike weighting.

Figure 6: Rules of thumb and simple rules of thumb results for the examples.

Target	Volatility	Reactive	NDP	RoT	SRoT
Smooth Spike	Low	76%	94%	94%	94%
Smooth Spike	Medium	51%	83%	81%	83%
Smooth Spike	High	24%	51%	47%	47%
Nonsymmetric Spike	Low	70%	89%	89%	89%
Nonsymmetric Spike	Medium	44%	71%	69%	71%
Nonsymmetric Spike	High	15%	30%	29%	29%
Spike Spike	Low	66%	85%	85%	85%
Spike Spike	Medium	38%	61%	60%	61%
Spike Spike	High	8%	17%	16%	16%
Double Spike	Low	68%	86%	86%	86%
Double Spike	Medium	43%	69%	67%	68%
Double Spike	High	17%	31%	31%	31%

Table 7: Percentage decrease over the constant policy at 20 time stages, calculated for the reactive policy, the NDP rollout policy (NDP), the rules of thumb (RoT) and the simple rules of thumb (SRoT).

was chosen most often; however, when the $(0.4, 0.4, 0.4)$ policy was chosen, its error was much smaller than that given for the constant-plus policy. So, although the constant-plus policy appeared more often, its average Q-factor over the outer simulation was actually worse.

This result suggests that we should base the rules of thumb on the magnitude of error over the outer simulation, rather than on the number of times a policy is chosen. Doing so would require obtaining additional information from the inner simulation during the calculation of the NDP rollout policy, namely the Q-factors for every policy. We believe that if this additional information is provided, then the resulting rules of thumb could provide more uniform generalized policies.

5 Conclusion

Day-to-day treatment planning is a complex problem that can significantly benefit from knowledge of the errors that occur during the delivery process. While dynamic programming and stochastic optimization would undoubtedly lead to better plans, they are currently intractable for application problems of realistic size and complexity.

This paper proposes a solution based on neuro-dynamic programming, coupled with heuristic policies that are based on the particular application.

We found that the NDP approach offers significant improvement over the currently-used (constant) policy. For most examples, we saw a significant improvement, with the error generally being cut at least in half. Further, this improvement was maintained when we removed the dependence upon the target structure and applied a simple rule of thumb. Because of this, the NDP approach can be useful for both on-line planning, where the plan is reoptimized between each treatment (NDP rollout), and for off-line planning, where the plan is determined in advance (rule of thumb or simple rule of thumb).

The results show that the simple rules of thumb, once determined, are almost as favorable as the NDP results. In practice, we believe that the simple rule of thumb will be effective for large complex target shapes, and we strongly recommend its usage over both the constant and reactive policies. Certainly, the simple rule of thumb is no more costly than the reactive policy to implement and is shown to be much more effective at dealing with the errors that can arise in the planning process. If the resulting improvements are not sufficient for the treatment planning problem, then two further policies are suggested by the results of this paper. The first technique chooses a particular control structure and simulates to determine a simple rule of thumb, which can then be applied during the treatment process. A second more costly (but even more effective) approach is to generate the control policy using optimization within the on-line procedure of Section 2.3. In this setting, the treatment planner is also able to choose a particular control structure. Under the second approach, the reoptimization need not be done at every time stage; for those time stages at which the reoptimization is not done, we can use the simple rule of thumb.

For immediate use, we suggest the simple rule of thumb as given for example in

Table 6. On a day-to-day basis, the treatment planner, knowing x_k, can calculate the dose required at each voxel in the manner outlined in Section 2.4, accounting for the stochastic errors that have occurred. Once this dose distribution is known, existing planning tools can be used to implement this on particular machines. We believe our results show this will significantly improve the final dose distribution that is delivered to the patient.

References

[1] Dimitri P. Bertsekas, 1997. Differential training of rollout policies. In *Proceedings of the 35th Allerton Conference on Communication, Control, and Computing.* Available as PDF document from `http://www.mit.edu:8001//people/dmitrib/Diftrain.pdf`.

[2] Dimitri P. Bertsekas and John N. Tsitsiklis, 1996. *Neuro-Dynamic Programming.* Athena Scientific, Belmont, Massachusetts.

[3] J. R. Birge and R. Louveaux, 1997. *Introduction to Stochastic Programming.* Springer, New York.

[4] T. Bortfeld, 2001. Current status of IMRT: physical and technological aspects. *Radiotherapy and Oncology*, **61**(2): 291–304.

[5] T. Bortfeld and W. Schlegel, 1993. Optimization of beam orientations in radiation therapy: some theoretical considerations. *Physics in Medicine and Biology*, **38**(2): 291–304.

[6] A. Brahme, 1995. Treatment optimization: Using physical and radiobiological objective functions. *In* A. R. Smith, editor, *Radiation Therapy Physics*, pages 209–246. Springer-Verlag, Berlin.

[7] A. Brooke, D. Kendrick and A. Meeraus, 1988. *GAMS: A User's Guide.* The Scientific Press, South San Francisco, California.

[8] P. S. Cho, S. Lee, R. J. Marks, S. Oh, S. Sutlief and H. Phillips, 1998. Optimization of intensity modulated beams with volume constraints using two methods: cost function minimization and projection onto convex sets. *Medical Physics*, **25**(4): 435–443.

[9] M. C. Ferris, J.-H. Lim and D. M. Shepard, 2001. Optimization approaches for treatment planning on a gamma knife. Data Mining Institute Technical Report 01-12, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin.

[10] M. C. Ferris, J.-H. Lim and D. M. Shepard, 2001. Radiosurgery treatment planning via nonlinear programming. Data Mining Institute Technical Report 01-01, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin.

[11] E. E. Fitchard, J. S. Aldridge, P. J. Reckwerdt and T. R. Mackie, 1998. Registration of synthetic tomographic projection data sets using cross-correlation. *Physics in Medicine and Biology*, **43**: 1645–1657.

[12] H. W. Hamacher and K.-H. Küfer, 1999. Inverse radiation therapy planning — a multiple objective optimisation approach. Technical Report 12, Institute for Techno- and Econo-Mathematics (ITWM) and Department of Mathematics, University of Kaiserslautern, Kaiserslautern, Germany. Available as PDF document from `http://www.itwm.uni-kl.de/zentral/download/berichte/bericht12.pdf`.

[13] H. Heitsch and W. Römisch, 2001. Scenario reduction algorithms in stochastic programming. Preprint 01-8, Institut für Mathematik, Humboldt-Universität, Berlin.

[14] T. Holmes and T. R. Mackie, 1994. A filtered backprojection dose calculation method for inverse treatment planning. *Medical Physics*, **21**(2): 303–313.

[15] ILOG CPLEX Division, 889 Alder Avenue, Incline Village, Nevada. *CPLEX Optimizer*. `http://www.cplex.com/`.

[16] P. Kall and S. W. Wallace, 1994. *Stochastic Programming*. John Wiley & Sons, Chichester.

[17] K.-H. Küfer, H. W. Hamacher and T. Bortfeld. A multicriteria optimization approach for inverse radiotherapy planning. Symposium on Operations Research (OR 2000). Available as PDF document from `http://www.uni-duisburg.de/FB5/BWL/WI/or2000/sektion01/kuefer.pdf`.

[18] J. T. Linderoth, A. Shapiro and S. J. Wright, 2002. The empirical behavior of sampling methods for stochastic programming. Optimization Technical Report 02-01, Computer Science Department, University of Wisconsin, Madison, Wisconsin.

[19] M. Livny. PI, the Condor project, high throughput computing. `http://www.cs.wisc.edu/condor`.

[20] T. R. Mackie, T. Holmes, S. Swerdloff, P. Reckwerdt, J. O. Deasy, J. Yang, B. Paliwal and T. Kinsella, 1993. Tomotherapy: a new concept for the delivery of dynamic conformal radiotherapy. *Medical Physics*, **20**(6): 1709–1719.

[21] A. Niemierko, 1992. Optimization of 3D radiation therapy with both physical and biological end points and constraints. *International Journal of Radiation Oncology, Biology and Physics*, **23**: 99–108.

[22] W. Schlegel and A. Mahr, editors, 2001. *3D Conformal Radiation Therapy - A Multimedia Introduction to Methods and Techniques*. Springer-Verlag, Berlin.

[23] D. M. Shepard, M. C. Ferris, G. Olivera and T. R. Mackie, 1999. Optimizing the delivery of radiation to cancer patients. *SIAM Review*, **41**: 721–744.

[24] L. J. Verhey, 1995. Immobilizing and positioning patients for radiotherapy. *Seminars in Radiation Oncology*, **5**(2): 100–113.

[25] S. Webb, 1997. *The Physics of Conformal Radiotherapy: Advances in Technology.* Institute of Physics Publishing Ltd.