Coarse-to-Fine Dynamic Programming*

Christopher Raphael†

August 28, 2000
revised March 30, 2001

Abstract

We introduce an extension of dynamic programming (DP) we call "Coarse-to-Fine Dynamic Programming" (CFDP), ideally suited to DP problems with large state space. CFDP uses dynamic programming to solve a sequence of coarse approximations which are lower bounds to the original DP problem. These approximations are developed by merging states in the original graph into "superstates" in a coarser graph which uses an optimistic arc cost between superstates. The approximations are designed so that when CFDP terminates the optimal path through the original state graph has been found. CFDP leads to significant decreases in the amount of computation necessary to solve many DP problems and can, in some instances, make otherwise infeasible computations possible. CFDP generalizes to DP problems with continuous state space and we offer a convergence result for this extension. The computation of the approximations requires that we bound the arc cost over all possible arcs associated with an adjacent pair of superstates; thus the feasibility of our proposed method requires the identification of such a lower bound. We demonstrate applications of this technique to optimization of functions and boundary estimation in mine recognition.

Index Terms: dynamic programming, A-star, mine recognition, brachistochrone, iterated complete path, coarse to fine, global optimization

1 Introduction

Many optimization problems can be recast as searches for the minimum cost path through a trellis graph where the cost of a path is given by the sum of the costs of the arcs traversed in the path. It is well-known that in such problems dynamic programming (DP) leads to a computationally efficient identification of the globally optimal path. Examples of applications of DP to recognition problems are numerous and include speech recognition [1], [2], [3]; character recognition [4], [5], [6]; deformable template matching [7],[8]; soft decoding [9],[10],[11],[12],[13]; and road tracking [14]. Such problems often lead to enormous state spaces, however, and the computations can be infeasible, even with DP. To overcome this obstacle, we propose a variation on DP we call *coarse-to-fine dynamic programming* (CFDP). We demonstrate two applications of CFDP that emphasize the generality and utility of this technique.

*This research was partly supported by a National Science Foundation Postdoctoral Fellowship, Award 92-06369.

†Department of Mathematics and Statistics, University of Massachusetts at Amherst, Amherst, MA 01003-4515, raphael@math.umass.edu

The essential idea of our algorithm is to form a series of coarse approximations to the original DP trellis by aggregating trellis states into "superstates." For each coarse approximation, the optimal path is found using DP with "optimistic" arc costs between the superstates. The superstates along this optimal path are refined and the process is iterated until a demonstrably globally optimal path is found. In many cases this global optimum is achieved with considerably less computational expenditure than straight DP.

Our CFDP algorithm is similar to the A^* or "Branch and Bound" algorithm familiar from the AI literature [15], [16], [17]. In the A^* algorithm, one maintains a tree structure of "prefixes" which are partial paths through the graph beginning with the start state, along with the costs of these prefixes. The leaf nodes of this tree can be thought of as the "frontier" of exploration, and for each frontier state one computes a lower bound on the cost of all paths connecting the frontier state to the final state. The sum of these two costs, the lower bound and the prefix cost, is a lower bound on all complete paths beginning with the prefix, and the prefixes partition the collection of complete paths. The algorithm proceeds by "expanding" the frontier state with the lowest estimated complete cost, thereby advancing the frontier, and updating the prefix tree to retain the best prefix paths found so far. The process of estimating optimal costs and advancing the frontier continues until the frontier reaches the final node. At this point the optimal path has been found.

Kam has built upon the idea of A^* in his "Iterated Complete Path" (ICP) algorithm [18], [19]. In an attempt to compute as few arc costs as possible, one begins by substituting a cheap lower bound for each arc cost. Next the optimal path through the trellis is found using DP and along this optimal path the true arc costs are computed. This procedure is iterated until the optimal path is composed entirely of true arc costs. At this point we have found the optimal path through the original graph.

While CFDP, A^*, and ICP are all similar in spirit, their domains of useful application differ. ICP is intended for a dynamic programming problem with a small trellis in which arc costs are expensive to compute; thus, it is not surprising that ICP offers nothing in particular to problems with large state space. A^* is also not particularly well-suited for large state space since large state spaces lead to large prefix trees. Our CFDP algorithm is particularly well-suited to DP problems with large state spaces. In fact, CFDP generalizes naturally to *continuous* state space as our examples illustrate.

Our CFDP bears a resemblance to hierarchical motion planning strategies popular in the robotics literature [22],[23],[24], [25],[26]. In this work a large state space of possible robot *configurations* is represented through a recursive partitioning into cells analogous to our superstates. Successive refinements of the state space into smaller and smaller cells are examined until one produces a realizable path. Our work differs from algorithms presented for hierarchical motion planning in that CFDP produces a provably optimal configuration on the original state space. In contrast, the hierarchical motion planning algorithms are generally suboptimal, or optimal only with respect to the approximation of the original objective function associated with the state space partition. We believe CFDP could make a significant contribution to robot motion planning in problems where an optimal solution is of paramount importance.

The following contains a careful development of our CFDP algorithms along with two applications. Section 2 gives a precise description of the algorithms with finite state space, demonstrates the correctness of the algorithm, and generalizes of CFDP to continuous state space. Sections 3 and 4 demonstrate applications of the algorithm to optimization of functions and mine detection with the hope of suggesting the wide range of problems that can benefit from CFDP. The Appendix provides a convergence proof for our CFDP algorithm in the continuous state space case.

2 The CFDP Algorithm

We begin by sketching our algorithm; a more precise description follows. By *trellis*, we mean a graph in which each node has an associated level, and arcs can can only connect nodes at adjacent levels. Consider the trellis diagram in the upper-left panel of Figure 1 in which we seek the minimum cost path from the left-most "state" to the right-most "state." We solve a sequence of approximations to this problem which is guaranteed to result in the optimal path.

In the first approximation we partition the states, at each trellis level, into a small collection of subsets or *superstates*. We define an "optimistic" graph on the superstates using the following two rules: Two superstates are connected if any of the pairs of states in their cross product are connected; the cost of an arc between two superstates is the *minimal* cost over all arcs connecting the superstates. The optimistic graph obtained using these rules is shown in the upper-right panel of Figure 1. The optimal path through this graph can be found using DP; this path is shown in solid black lines.

Our second approximation is formed by refining the superstates along the optimal path as shown in the middle-left panel of Figure 1. The graph is then redrawn, using the same two rules as before, and we find the optimal path through this somewhat less optimistic graph, again using DP. This optimal path is also shown in solid black.

The process of finding the optimal path and refining the superstates along the optimal path continues until we find an optimal path composed entirely of singleton superstates as in the bottom-right panel of Figure 1. This path must be the optimal path through the original graph as we will argue presently.

More formally, let $\mathcal{G} = (\mathcal{S}, \mathcal{E}, C)$ be a weighted trellis graph where $\mathcal{S}$ is a set of nodes, $\mathcal{E}$ is a set of edges, and C is a cost function defined on the edges. Thus every node $s \in \mathcal{S}$ has a "level" $\nu(s) \in \{0, \ldots, N\}$ and we assume that $\mathcal{E} \subseteq \{(s,t) : s,t \in \mathcal{S}, \nu(s) + 1 = \nu(t)\}$. The cost of an edge $(s,t) \in \mathcal{E}$ is given by $C(s,t)$. For simplicity's sake we assume with no loss of generality that s_0 and s_N are the unique nodes having $\nu(s_0) = 0$ and $\nu(s_N) = N$ and we define the collection of paths through $\mathcal{G}$ as

$$\pi(\mathcal{G}) = \{(s_0, s_1, \ldots, s_N) : (s_n, s_{n+1}) \in \mathcal{E}, n = 0, \ldots, N-1\}$$

The cost of a path is then given by $C(s_0, s_1, \ldots, s_N) = \sum_{n=0}^{N-1} C(s_n, s_{n+1})$ and we seek the globally minimal cost path through the trellis graph. It is well known that this minimal cost path can be found through DP by recursively defining

$$C^*(s) = \begin{cases} 0 & \text{if } s = s_0 \\ \min_{\{t \in \mathcal{S} : (t,s) \in \mathcal{E}\}} C^*(t) + C(t,s) & \text{otherwise} \end{cases}$$

for $s \in \mathcal{S}$ and

$$B(s) = \arg \min_{\{t \in \mathcal{S} : (t,s) \in \mathcal{E}\}} C^*(t) + C(t,s)$$

for $s \in \mathcal{S}$ with $\nu(s) > 0$. The optimal path is then $(s_0^*, \ldots, s_N^*)$ where $s_N^* = s_N$ and $s_n^* = B(s_{n+1}^*)$ for $n = 0, \ldots, N-1$.

Our CFDP is now defined. In what follows, by a partition of a set R ($|R| > 1$) we mean a collection of at least two nonempty disjoint subsets whose union is R. If $|R| = 1$ the only possible partition is $\{R\}$.

CFDP Algorithm

Initialization: Let P_n be a partition of $\{s \in \mathcal{S} : \nu(s) = n\}$. Let $\mathcal{G}^0 = (\mathcal{S}^0, \mathcal{E}^0, C^0)$ where

$$\mathcal{S}^0 = \bigcup_{n=0}^{N} P_n$$

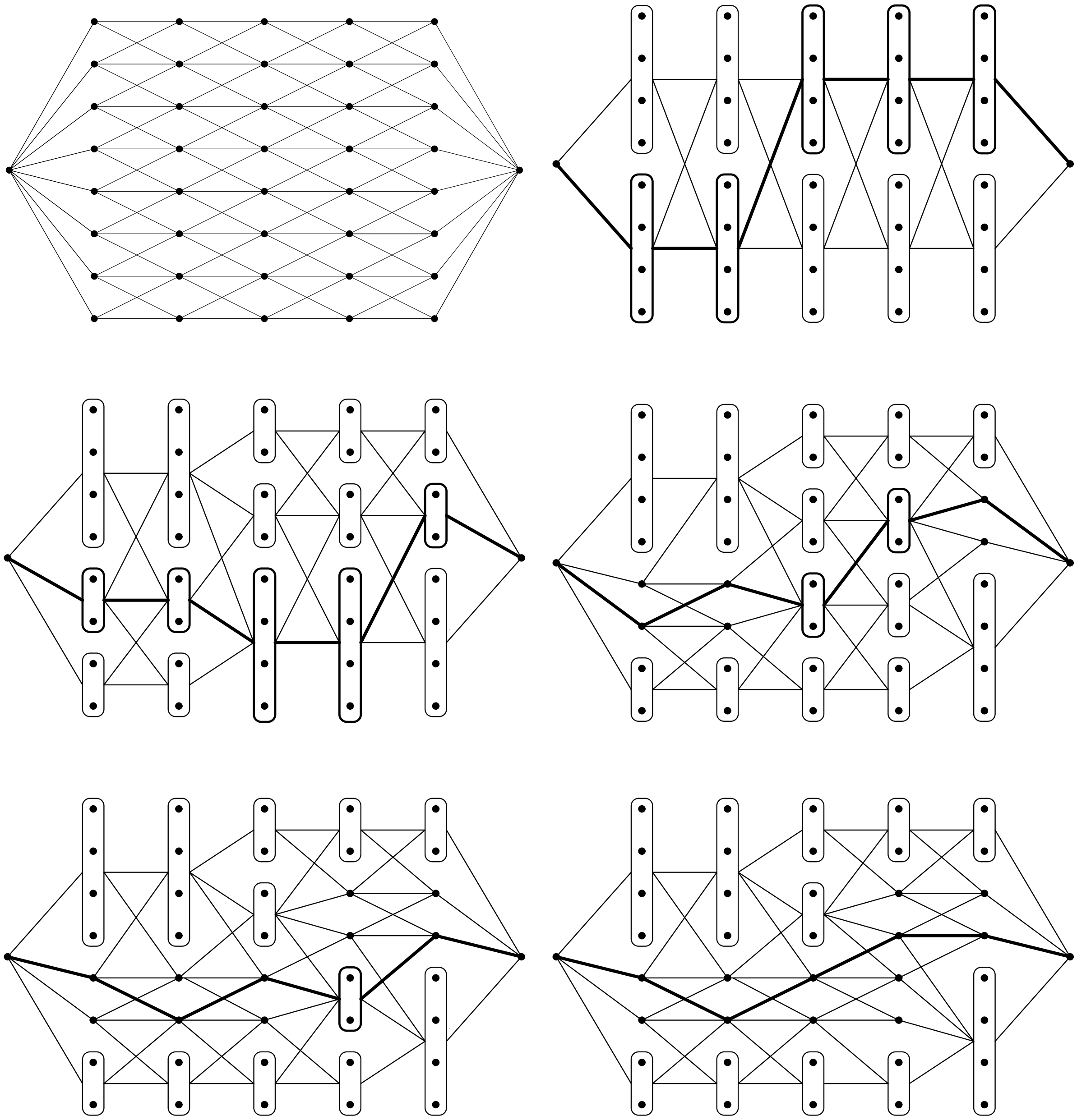

Figure 1: The upper-left figure is the original dynamic programming setup. The five remaining figures show a possible progression of our coarse-to-fine dynamic programming algorithm. The optimal path (shown in bold lines) in the final figure in the lower-right must also be the optimal path in the original trellis.

$$
\begin{aligned}
\mathcal{E}^0 &= \{(S,T) : S,T \in \mathcal{S}^0 \text{ and there exists } (s,t) \in \mathcal{E}, s \in S, t \in T\} \\
C^0(S,T) &= \min_{(\{s,t\}: s \in S, t \in T, (s,t) \in \mathcal{E}\}} C(s,t) \\
(S_0^0, \ldots, S_N^0) &= \arg \min_{\{(S_0, \ldots, S_N) \in \pi(\mathcal{G}^0)\}} C^0(S_0, \ldots, S_N)
\end{aligned}
$$

Iteration: For $k = 1, 2, \ldots$ define $\mathcal{G}^k = (\mathcal{S}^k, \mathcal{E}^k, C^k)$ where

$$
\begin{aligned}
\mathcal{S}^k &= \mathcal{S}^{k-1} \cup \left(\bigcup_{n=0}^{N} P(S_n^{k-1})\right) \setminus \{S_0^{k-1}, \ldots, S_N^{k-1}\} \\
\mathcal{E}^k &= \{(S,T) : S,T \in \mathcal{S}^k \text{ and there exists } (s,t) \in \mathcal{E}, s \in S, t \in T\} \\
C^k(S,T) &= \min_{\{(s,t): s \in S, t \in T, (s,t) \in \mathcal{E}\}} C(s,t) \\
(S_0^k, \ldots, S_N^k) &= \arg \min_{\{(S_0, \ldots, S_N) \in \pi(\mathcal{G}^k)\}} C^k(S_0, \ldots, S_N)
\end{aligned}
$$

where $P(S_n^k)$ is any partition of S_n^k. Note that $(S_0^k, \ldots, S_N^k)$ defined above can be computed through DP.

Proposition: If $|\mathcal{S}| < \infty$ let

$$k^* = \min\{k : |S_n^k| = 1, n = 0, \ldots, N\}$$

and let $S_n^{k^*} = \{s_n^{k^*}\}$, $n = 0, \ldots, N$. Then

$$C(s_0^{k^*}, \ldots, s_N^{k^*}) = \min_{\{(s_0, \ldots, s_N) \in \pi(\mathcal{G})\}} C(s_0, \ldots, s_N)$$

Proof: Since $|\mathcal{S}| < \infty$ only a finite number of iterations are possible before $\mathcal{S}^k$ is composed entirely of singleton sets so k^* is well-defined.

Let $(s_0, \ldots, s_N) \in \pi(\mathcal{G})$. Then there exist $S_n \in \mathcal{S}^{k^*}$, $n = 0, \ldots, N$ such that $s_n \in S_n$ since $\mathcal{S}^{k^*}$ partitions $\mathcal{S}$. Also note that $(s_n, s_{n+1}) \in \mathcal{E}$ for $n = 0, \ldots, N-1$ implies $(S_n, S_{n+1}) \in \mathcal{E}^{k^*}$ and $(S_0, \ldots, S_N) \in \pi(\mathcal{G}^{k^*})$. Then

$$C(s_0, \ldots, s_N) \geq C^{k^*}(S_0, \ldots, S_N) \geq C^{k^*}(S_0^{k^*}, \ldots, S_N^{k^*}) = C(s_0^{k^*}, \ldots, s_N^{k^*}) \quad \square$$

It is worth noting that one can substitute a *lower bound*, $\tilde{C}^k(S,T)$, for the true minimum arc cost, $C^k(S,T)$, between two superstates without affecting the correctness of the algorithm, as long as $\tilde{C}^k(S,T) = C^k(S,T)$ when $|S| = |T| = 1$. In fact, the proof is identical to the above with $\tilde{C}^k$ taking the place of C^k. In many applications, lower bounds are easy to compute from problem-specific information, and often they come at considerably less computational cost than the true minimum. Also we note that the sequence of DP problems generated by our CFDP algorithm are quite similar to one another, so they needn't be solved from scratch. In fact, each solution can be obtained by updating a data structure common to all of the DP problems.

For trellis graphs, the computational complexity of any dynamic programming iteration is $O(|\mathcal{E}|)$. In CFDP, with typical superstate partitioning rules, the number of edges grows linearly with the iteration. Thus the computational complexity of performing K iterations of CFDP is $O(K^2)$. The actual number of iterations before the algorithm terminates is highly problem-dependent and not amenable to simple complexity analysis. In the worst case, CFDP refines the entire trellis graph before solving the DP problem that was originally presented, thus requiring considerably more computation ($O(|\mathcal{E}|^2)$) than straight DP ($O(|\mathcal{E}|)$). In the best case, CFDP continually refines the optimal path found on the previous iteration, resulting in a computation of $O(\log|\mathcal{E}|)$. While the efficiency of CFDP varies in practice, we have found that judicious application of CFDP can lead to considerable savings as demonstrated in our examples.

The essential idea of CFDP can be extended to a particular class of optimization problems on functions of real variables. Suppose we want to minimize

$$f(x_0,\ldots,x_N)=\sum_{n=0}^{N-1} f_n(x_n,x_{n+1}) \tag{1}$$

where $x_n \in R_n$ for $n=0,\ldots,N$ with R_n some d-dimensional rectangle of finite volume: $[a_n^1,b_n^1]\times\ldots\times[a_n^d,b_n^d]$. We can create a corresponding graph so that each possible solution $(x_0,\ldots,x_N)$ is identified with a path through the graph and $f(x_0,\ldots,x_N)$ is the cost of the path. We then can express the optimization problem as a search for the minimal cost path through the graph.

Let $\mathcal{G}=(\mathcal{S},\mathcal{E},C)$ where

$$\begin{aligned}
\mathcal{S} &= \bigcup_{n=0}^{N}\{(x,n):x\in R_n\}\\
\mathcal{E} &= \{(s,t):s,t\in\mathcal{S},\nu(s)+1=\nu(t)\}\\
C(s,t) &= f_{\nu(s)}(x(s),x(t))
\end{aligned}$$

where the components of $s\in\mathcal{S}$ are given by $s=(x(s),\nu(s))$. The CFDP algorithm can be applied directly to this situation. For example, for each $n=0,\ldots,N$ we could define the initial partition at the n^{th} level to be P_n where $S\in P_n$ has components $S=(X(S),n)$ with $\cup_{S\in P_n}X(S)=R_n$ and the $\{X(S):S\in P_n\}$ are disjoint rectangles. The algorithm would proceed exactly as given above: For each iteration k we find the optimal sequence of superstates $(S_0^k,\ldots,S_N^k)$ through dynamic programming and then partition these superstates into smaller superstates. For example, each partition $P(S_n^k)$ could be such that $\cup_{S\in P(S_n^k)}X(S)=X(S_n^k)$ with $\{X(S):S\in P(S_n^k)\}$ being disjoint subrectangles of $X(S_n^k)$. For definiteness one could imagine forming $P(S_n^k)$ out of 2^d superstates obtained by splitting $X(S_n^k)$ in two across all d dimensions.

In this continuous version of CFDP while the volume of the $\{X(S_n^k)\}$ might shrink to 0, the $\{X(S_n^k)\}$ will never degenerate into singleton sets; thus there is no obvious stopping criterion for continuous CFDP. We do show in the Appendix, however, that in a suitably defined sense the approximations given by the $(X(S_0^k),\ldots X(S_N^k))$ collapse around the optimal solution, thus showing that CFDP converges to the globally optimal solution.

3 Optimization of Functions

In a famous calculus problem, the Brachistochrone Problem, one seeks the minimum time path between two points on a vertical plane (x_0,y_0), (x_N,y_N), with (x_N,y_N) lower than (x_0,y_0). It is assumed that gravity is the only force acting on an object as it traverses the path. The solution, a cycloid (Figure 2 left), can be derived using the Calculus of Variations [20] — thus our interest in this problem is for illustrative purposes.

We consider a discretization of the problem in which we seek a piecewise-linear path with knots at (x_n,y_n) where $n=0,1,\ldots,N$. The values of the $\{x_n\}$ are fixed by the design of the problem, as well as the initial and final heights, y_0,y_N, and we wish to minimize the time to traverse the path defined by the remaining heights, $y_1,y_2,\ldots,y_{N-1}$ (Figure 2, right).

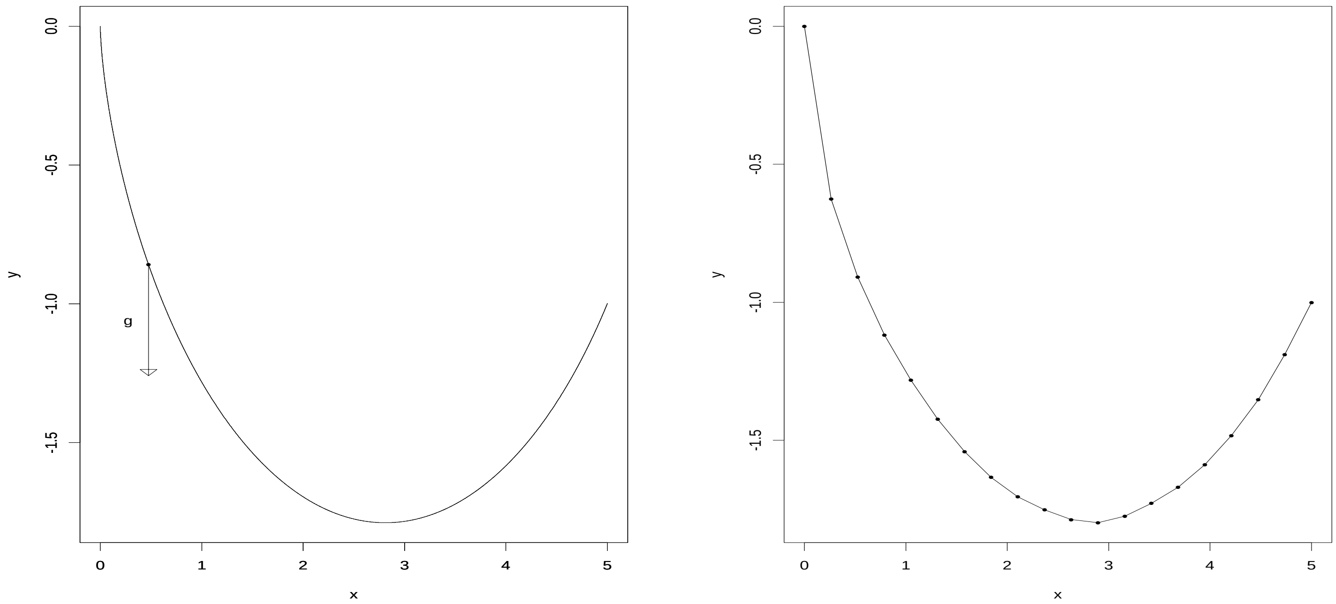

Figure 2: **Left:** For each panel the leftmost point of the curve is (x_0, y_0) and the rightmost point is (x_N, y_N). The optimal solution to the Brachistochrone problem — a cycloid. **Right:** The approximate optimal solution to a piecewise linear approximation to the Brachistochrone problem.

Assuming the object begins with velocity 0 at (x_0, y_0) , the velocity at any point, (x, y), can be calculated by the principle of conservation of energy, regardless of the path leading to (x, y). This velocity is given by $v(x, y) = \sqrt{2g(y_0 - y)}$ where g is the acceleration due to gravity. Thus the time it takes to travel from, say, (x_1, y_1) to (x_2, y_2) along a straight line is given by

$$C(x_1, y_1, x_2, y_2) = \sqrt{2/g}\frac{\sqrt{(y_2 - y_1)^2 + (x_2 - x_1)^2}}{\sqrt{y_0 - y_1} + \sqrt{y_0 - y_2}} \tag{2}$$

and the total time needed to traverse the path is given by summing the times over the individual line segments:

$$C(x_0, y_0, \ldots, x_N, y_N) = \sum_{n=0}^{N-1} C(x_n, y_n, x_{n+1}, y_{n+1})$$

as in Eqn. 1.

With a suitable discretization of the possible heights this problem would be well-suited to DP, however, the discretization might need to be very fine before we achieve sufficient accuracy. Also, we might wish to achieve the maximum accuracy for fixed computing time and fixing the discretization *a priori* makes this impossible. Instead we apply our continuous CFDP algorithm.

We begin by defining the initial discretization into superstates which in this case are intervals. Specifically, we let $y_{\min}$ be the minimum path height we are willing to consider and for each $n = 1, \ldots, N-1$ we partition the interval $[y_{\min}, y_0)$ into I intervals,

$$P_n = \{[y_{\min}, y_{\min} + \Delta), [y_{\min} + \Delta, y_{\min} + 2\Delta), \ldots, [y_{\min} + (I-1)\Delta, y_{\min} + I\Delta)\}$$

where $\Delta = (y_0 - y_{\min})/I$. Since the initial and final heights are fixed, for $n = 0$ and $n = N$ we define $P_0 = \{y_0\}$ and $P_N = \{y_N\}$. We take our initial collection of superstates to be $\mathcal{S}^0 = \cup_{n=0}^{N} P_n$ where, for the sake of clarity of notation, we drop the more formal notion of a superstate as a rectangle and level pair and simply refer to the rectangle as the superstate.

We define the graph on these superstates by forming all possible arcs between superstates at levels n and $n+1$ for $n = 0, \ldots, N-1$. For an arc between superstates $S_n = [l_n, u_n)$ and $S_{n+1} = [l_{n+1}, u_{n+1})$ at levels n and $n+1$, we define the optimal cost

$$C^*(S_n, S_{n+1}) = \min_{(y_n, y_{n+1}) \in S_n \times S_{n+1}} C(x_n, y_n, x_{n+1}, y_{n+1})$$

This cost can be computed in closed form as follows. First note that the minimal cost must be achieved at a point $(y_n, y_{n+1}) \in S_n \times S_{n+1}$ with either $y_n = l_n$ or $y_{n+1} = l_{n+1}$, thus reducing the problem to one of minimizing a function of one variable. Consider the case in which we assume $y_n = l_n$; the case in which $y_{n+1} = l_{n+1}$ is treated analogously. Then

$$\begin{aligned} C(x_n, y_n, x_{n+1}, y_{n+1}) &= \sqrt{2/g}\frac{\sqrt{(y_{n+1} - l_n)^2 + (x_{n+1} - x_n)^2}}{\sqrt{y_0 - l_n} + \sqrt{y_0 - y_{n+1}}} \\ &= \sqrt{2/g}\frac{\sqrt{s^4 + r^4 - 2s^2r^2 + d^2}}{r + s} \end{aligned}$$

by substituting $r^2 = y_0 - l_n$, $s^2 = y_0 - y_{n+1}$, and $d^2 = (x_{n+1} - x_n)^2$. Differentiating with respect to s and setting the derivative to zero gives

$$s_4 + 2s^3r - 2sr^3 - r^4 - d^2 = 0$$

and the minimizing root is given by

$$y_{\text{root}} = y_0 - \left(\frac{r}{2} + \frac{B}{6} + \frac{\sqrt{3}}{6} \sqrt{\frac{6r^2 A^{1/3}\sqrt{B} - 3\sqrt{B}A^{2/3} + 4\sqrt{B}d^2 + 18r^3 A^{1/3}}{A^{1/3}\sqrt{B}}} \right)^2$$

where

$$A = -2r^2 d^2 + \frac{2}{9}\sqrt{48d^6 + 81r^4 d^4}$$

and

$$B = \frac{9r^2 A^{1/3} + 9A^{2/3} - 12d^2}{A^{1/3}}$$

with the help of symbolic computation. Thus, for the case when $y_n = l_n$ we must have

$$C^*(S_n, S_{n+1}) = \begin{cases} \min(C(x_n, l_n, x_{n+1}, l_{n+1}), C(x_n, l_n, x_{n+1}, u_{n+1})) & y_{\text{root}} \notin [l_{n+1}, u_{n+1}) \\ C(x_n, l_n, x_{n+1}, y_{\text{root}}) & \text{otherwise} \end{cases}$$

Finding the path through this trellis graph using the arc cost function C^* can be accomplished using dynamic programming. Suppose that the optimal sequence of superstates in the k^{th} iteration of the algorithm is found to be $S_1^k, \ldots, S_{N-1}^k$ with $S_n^k = [l_n^k, u_n^k)$ for $n = 1, \ldots, N-1$. We then create the superstates at iteration $k+1$, $\mathcal{S}^{k+1}$ by letting the partition at level $n = 1, \ldots, N-1$ be

$$P_n^{k+1} = P_n^k \cup \{[l_n^k, \frac{l_n^k + u_n^k}{2}), [\frac{l_n^k + u_n^k}{2}, u_n^k)\} \setminus \{[l_n^k, u_n^k)\}$$

and letting $\mathcal{S}^{k+1} = P_0 \cup P_N \cup (\cup_{n=1}^{N-1} P_n^{k+1})$

The graph is then redrawn connecting each superstate at level $n = 0, \ldots, N-1$ with each superstate at level $n+1$, again using C^* to compute arc costs. The algorithm iterates the process of finding the optimal path using dynamic programming and refining the partition along the optimal path and redrawing the graph. The proof given in the Appendix guarantees that the optimal superstates found at each iteration of the algorithm will eventually collapse around the optimal path, provided one is willing to assume a priori that the optimal path always lies above some number $y_{\min}$.

We performed experiments on this problem with 20 evenly spaced knots, $x_0, x_1, \ldots, x_{19}$, and with $(x_0, y_0) = (0, 0)$ and $(x_{19}, y_{19}) = (5, -1)$ and a graph of the final result is given in the right panel of Figure 2. We compare the relative efficiency of continuous CFDP to straight DP as follows. For straight DP we consider discretizations $d = 3, 4, \ldots, 12$ where the dth discretization has 2^d states. For each discretization we compute the number of arc cost calculations, $\text{DP}(d)$, necessary to obtain a DP solution since the arc costs dominate the total computing time. To facilitate comparisons we ran the CFDP with the same target discretizations as used above. In seeking a CFDP solution at level d we do not allow any state to divide beyond the level d discretization, thus producing exactly the same solution as would be obtained with straight DP. We denote the number of arc cost computations in this case by $\text{CFDP}(d)$. Figure 3 shows $\log_{10}(\text{CFDP}(d)/\text{DP}(d))$ for various values of d demonstrating the increasing value of CFDP at progressively finer discretizations. At the finest discretization the figure demonstrates a 100-fold decrease in the number of arc cost computations. Since the arc cost computations require a constant amount of time for both straight DP and CFDP, the figure suggests an increasingly bigger win for CFDP as one seeks a solutions at progressively finer discretizations.

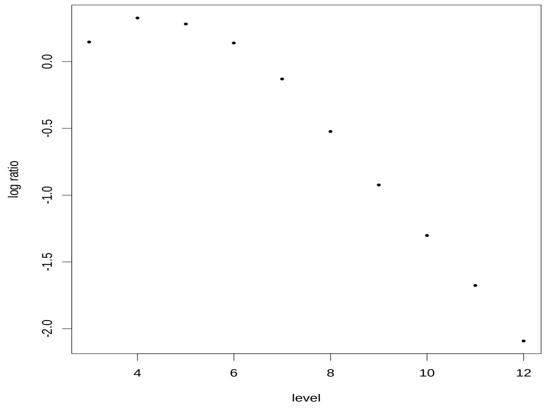

Figure 3: The figure shows $\log_{10}(\mathrm{CFDP}(d)/\mathrm{DP}(d))$ at levels $d = 3, 4, \ldots, 12$. The index d represents a discretization into 2^d states.

4 Boundary Estimation in Mine Recognition

In our formulation of a mine recognition problem we are given a collection of images containing mine-like objects. These objects can be recognized as areas of relatively lighter pixel values coming from a known library of possible shapes — parallelograms and ellipses in the data presented here. Our approach has four stages: finding candidate locations, estimating mine boundaries, optimizing over the boundary estimates using the shape library information, and accepting only the mine hypotheses whose likelihood exceeds a threshold. We focus here on boundary estimation which we believe to be the most difficult stage. Since all of the shapes in our library are convex, we formulate the boundary estimation problem as one of finding the mostly likely convex set containing the candidate location. This is the problem we discuss here; for a complete discussion see [21].

We begin by developing a representation of the class of convex sets containing a fixed reference point x. Such a set can be thought of as a pair of 2π-periodic functions $(r(\theta), t(\theta))$ representing the radius and tangent direction at every angle in a coordinate system with origin x (see Figure 4). Thus $r(\theta) > 0$ and $-\pi/2 \leq t(\theta) \leq \pi/2$ for $\theta \in [0, 2\pi]$. In practice, we will only specify $(r(\theta), t(\theta))$ at a finite number of angles $\theta_0, \theta_1, \ldots, \theta_{N-1}$, typically $N = 16$ or $N = 32$ in our experiments. Additionally we will restrict our attention to convex sets whose boundaries are contained in an annulus centered around x. Thus we seek $(r(\theta_n), t(\theta_n)) \in [r^{\min}, r^{\max}] \times [-\pi/2, \pi/2]$ for $n = 0, \ldots, N-1$.

Consider the problem of specifying values $(r(\theta_n), t(\theta_n))$, $n = 0, 1, \ldots, N-1$ in a manner consistent with convexity. Our choice of $(r(\theta_0), t(\theta_0))$ is unconstrained, however, convexity imposes constraints on the choices at other angles, as follows. Consider any adjacent pair of angles, say θ_1 and θ_2. In a convex set the tangent angle must continue to rotate counter-clockwise as we move counter-clockwise through the possible angles. As a result, the maximum possible value for $r(\theta_2)$, as dictated by our choice of $(r(\theta_1), t(\theta_1))$, is the length at which the ray emanating from $(\theta_1, r(\theta_1))$ in the $t(\theta_1)$ direction intersects the line containing x and $(\theta_2, r(\theta_2))$ (see Figure 4, right). For

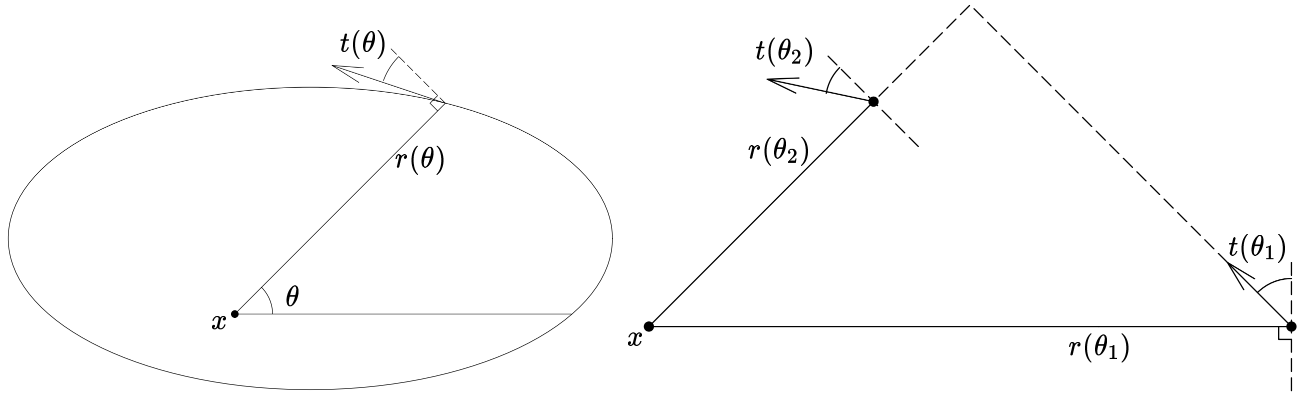

Figure 4: **Left:** A convex set containing x can be described in terms of the function $r(\theta)$ giving the distance to the boundary as a function of θ. $t(\theta)$ is the relative orientation of the tangent direction to the convex set at the boundary point $(\theta, r(\theta))$. **Right:** The first of the two conditions for a legal arc is described pictorially. For convexity, $t(\theta)$ must rotate counter-clockwise as θ moves counter-clockwise. This places a constraint on $r(\theta_1)$, $r(\theta_2)$, and $t(\theta_1)$.

instance, the configuration depicted in Figure 4 *is* consistent with convexity. A similar relation obtained by observing the way $(r(\theta_2), t(\theta_2))$ constrains $r(\theta_1)$ leads to another inequality. So, for any $n = 0, 1, \ldots, N-1$, a "legal" configuration of $r(\theta_n), t(\theta_n), r(\theta_{n+1}), t(\theta_{n+1})$ satisfies

$$\frac{\cos(t(\theta_n) - (\theta_{n+1} - \theta_n))}{\cos(t(\theta_n))} \leq \frac{r(\theta_n)}{r(\theta_{n+1})} \leq \frac{\cos(t(\theta_{n+1}))}{\cos(t(\theta_{n+1}) + (\theta_{n+1} - \theta_n))} \tag{3}$$

where the subscript addition is always taken to be modulo N. If we define the "states" $s_n = (r(\theta_n), t(\theta_n))$, then the convexity constraints operate on adjacent pairs of states so the possible convex sets can be identified with paths through a trellis. The last state choice, at θ_{N-1}, must satisfy two pairs of constraints, one for s_{N-2}, s_{N-1} and one for s_{N-1}, s_0, so our trellis "wraps around" on itself.

Our data model is defined for a circular window of pixel data centered around x as follows (Figure 5). The pixel data $\mathbf{y}$ are partitioned into N "pie-slices" $y_0, y_1, \ldots, y_{N-1}$ bounded by the $\{\theta_n\}$ as in Figure 5. We then assume, given a convex set $\mathbf{s} = (s_0, s_1, \ldots, s_{N-1})$, the $\{y_n\}$ are generated independently and with only local dependence on the $\{s_n\}$. That is,

$$P(\mathbf{y}|\mathbf{s}) = \prod_{n=0}^{N-1} P(y_n|s_n, s_{n+1})$$

We have experimented with a number of different data models, some corresponding to generative models with straightforward probabilistic assumptions, and others simply being quantities that seem reasonable to optimize. The actual experiments were performed with this latter kind of "data model" defined by

$$\log(P(y_n|s_n, s_{n+1})) = \frac{\hat{\mu}_{\text{in}} - \hat{\mu}_{\text{out}}}{\hat{\sigma}}$$

where $\hat{\mu}_{\text{in}}$ and $\hat{\mu}_{\text{out}}$ are the empirical mean values for the "inside" and "outside" regions defined by the triangle associated with s_n and s_{n+1} and $\hat{\sigma}$ is a pooled standard deviation estimate assuming

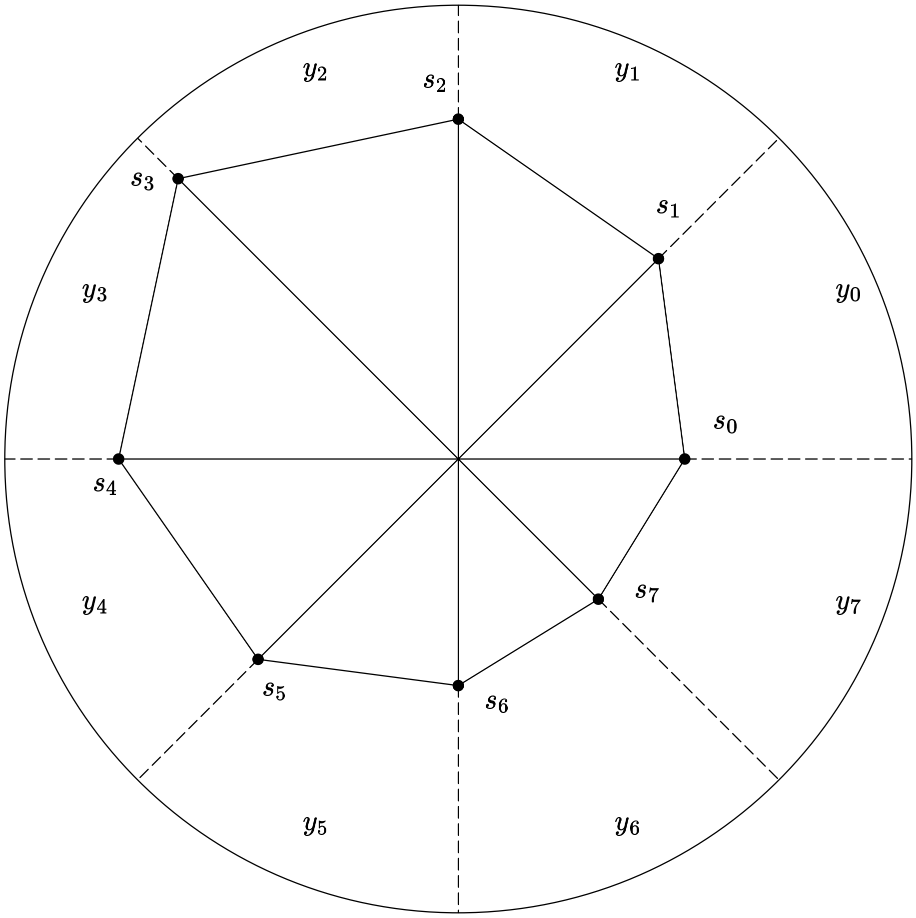

Figure 5: The circular data window, $\mathbf{y}$, is divided into N "pie-slices" $y_0, y_1, \ldots, y_{N-1}$. Given a convex set hypothesis: $\mathbf{s} = s_0, s_1, \ldots, s_{N-1}$, we assume the $y_0, y_1, \ldots, y_{N-1}$ are conditionally independent.

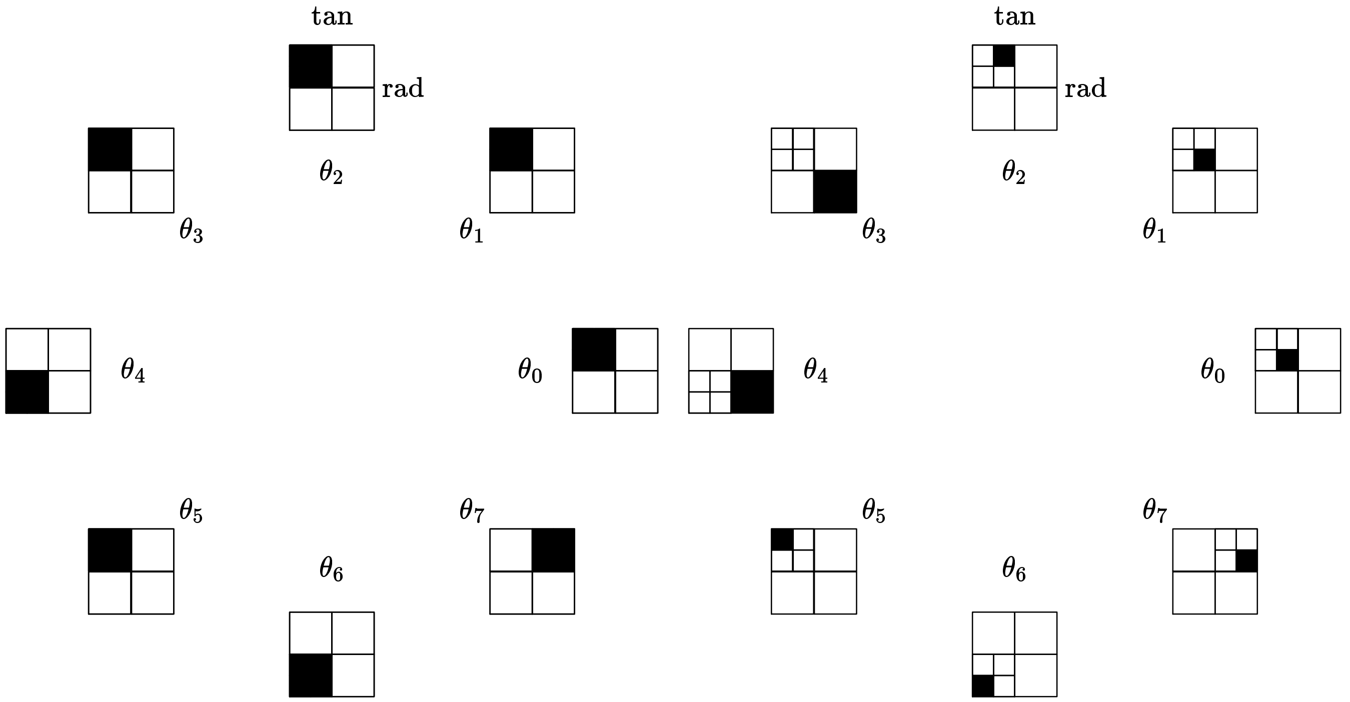

Figure 6: **Left:** The first iteration of CFDP for convex sets. **Right:** The second iteration of CFDP.

the inside and outside regions contain homogeneous pixel values of differing means but constant variance. Here we have assumed that mines are regions of high pixel values on a lower pixel value background. The log likelihood of a convex set can then be expressed as a sum of arc scores over the associated trellis path.

If we discretize the state space *a priori*, then fixing s_0, the maximum likelihood convex set containing s_0 can be found by traditional DP. We, of course, must consider all possible starting points for s_0, so the computation increases by an order of magnitude. See [21] for a more computationally elegant solution to this problem.

To apply continuous CFDP, we partition the radius-tangent space, $[r^{\min}, r^{\max}] \times [-\pi/2, +\pi/2]$, into a collection of rectangles as in the left panel of Figure 6. Each subrectangle in this figure represents a range of possible radii and tangent values. We call this initial partition P^0 and let $P_n = P^0$ for $n = 0, \ldots, N-1$. Our initial collection of superstates is then $\mathcal{S}^0 = \cup_{n=0}^{N-1} P_n$.

In creating the k^{th} superstate graph, $\mathcal{G}^k = (\mathcal{S}^k, \mathcal{E}^k, C^k)$, $k = 0, 1, \ldots$

$$\begin{aligned}\mathcal{E}^k \quad &= \quad \{(S_n, S_{n+1}) : S_n, S_{n+1} \in \mathcal{S}^k; \nu(S_n) = n, \nu(S_{n+1}) = n+1; \\ &\qquad \text{there exists } (s_n, s_{n+1}) \in S_n \times S_{n+1} \text{ with } L(s_n, s_{n+1}) = 1\}\end{aligned} \tag{4}$$

where $L(s_n, s_{n+1})$ is 1 or 0 as Eqn. 3 is satisfied or not. If $S_n = [r_n^l, r_n^u] \times [t_n^l, t_n^u]$ and $S_{n+1} = [r_{n+1}^l, r_{n+1}^u] \times [t_{n+1}^l, t_{n+1}^u]$, then this condition can be described by the constraints

$$\frac{\cos(t_n^l - 2\pi/N)}{\cos(t_n^l)} \leq \frac{r_n^u}{r_{n+1}^l} \tag{5}$$

$$\frac{\cos(t_{n+1}^u)}{\cos(t_n^u + 2\pi/N)} \geq \frac{r_n^l}{r_{n+1}^u} \tag{6}$$

$$t_{n+1}^l + 2\pi/N \quad \geq \quad t_n^u \tag{7}$$

The first two constraints above are simply restatements of Eqn. 3 while the last constraint forces the tangent angle to be a nondecreasing function of the angle. When dealing with isolated states the monotonicity of the tangent angle is a direct byproduct of Eqn. 3, however, when dealing with superstates we must explicitly include this constraint.

Ideally, the arc score between two superstates S_n and S_{n+1}, $(S_n, S_{n+1}) \in \mathcal{E}^k$, $k = 0, 1, \ldots$, would be

$$C(S_n, S_{n+1}) = \max_{(s_n, s_{n+1}) \in S_n \times S_{n+1}} \log(L(s_n, s_{n+1}) P(y_n | s_n, s_{n+1})) \tag{8}$$

The effort necessary to perform this computation would defeat the purpose of CFDP, however. Instead we use the upper bound for $C(S_n, S_{n+1})$:

$$\tilde{C}(S_n, S_{n+1}) = \max_{(s_n, s_{n+1}) \in S_n \times S_{n+1}} \log(L(s_n, s_{n+1})) + \max_{(s_n, s_{n+1}) \in S_n \times S_{n+1}} \log(P(y_n | s_n, s_{n+1})) \tag{9}$$

The left term on the right hand side is just the computation of Eqns. 5 – 7, while the right term, depending only on the two radius ranges, is readily approximated by maximizing over a discretization of the possible values. More specifically, we approximate the right term of Eqn. 9 by maximizing $P(y_n | s_n, s_{n+1})$ over the states corresponding to triangles $(x, r(\theta_n), r(\theta_{n+1}))$ where $r(\theta_n) = r(\theta_{n+1} = \rho \in r_n^l, \ldots, r_n^h$ (c. f. Figure 5).

For the k^{th} iteration of our algorithm, the legal paths, $\pi(\mathcal{G}^k)$, are the sequences $(S_0, \ldots, S_{N-1})$ such that $(S_n, S_{n+1}) \in \mathcal{E}^k$ and $S_0 = S_{N-1}$ — a convex set must end in the same state in which it begins. The cost of a path is then

$$\tilde{C}(S_0, \ldots, S_{N-1}) = \sum_{n=0}^{N-2} \tilde{C}(S_n, S_{n+1})$$

As usual, the optimal path in the k^{th} iteration is defined to be

$$(S_0^k, \ldots, S_{N-1}^k) = \arg \max_{\{(S_0, \ldots, S_{N-1}) \in \pi(\mathcal{G}^k)\}} \tilde{C}(S_0, \ldots, S_{N-1})$$

which can be is found by solving a dynamic programming problem for each $S_0 = S_{N-1}$ and taking the maximizing path.

The algorithm then proceeds as follows. We define the *depth* of a superstate $S = [r^l, r^u] \times [t^l, t^u]$ by

$$\text{depth}(S) = \log_2 \frac{r^{\max} - r^{\min}}{r^u - r^l}$$

and let L be our "target" depth. For a superstate $S = [r^l, r^u] \times [t^l, t^u]$. define

$$P(S) = \begin{cases} \{S\} & \text{if depth}(S) = L \\ \{[r^l, (r^l + r^u)/2] \times [t^l, (t^l + t^u)/2]\} \quad \cup & \\ \{[(r^l + r^u)/2, r^u] \times [t^l, (t^l + t^u)/2]\} \quad \cup & \\ \{[r^l, (r^l + r^u)/2] \times [(t^l + t^u)/2, t^u]\} \quad \cup & \\ \{[(r^l + r^u)/2, r^u] \times [(t^l + t^u)/2, t^u]\} & \text{otherwise} \end{cases}$$

After having found the optimal path at the k^{th} iteration we let

$$\mathcal{S}^{k+1} = \mathcal{S}^k \ \cup \ (\bigcup_{n=0}^{N-1} P(S_n^k)) \ \setminus \ \{S_0^k, \ldots, S_{N-1}^k\}$$

and then repeat the process of redrawing the graph according to Eqn. 4 and computing the edge costs according to Eqn. 8. The algorithm terminates when $\text{depth}(S_n^k) = L$ for $n = 0, \ldots, N-1$.

For illustrative purposes, the left panel of Figure 6 shows a possible optimal sequence of superstates found in the $\mathcal{G}^0$. As described above, we refine the superstates along this optimal path and construct $\mathcal{G}^1$. A possible optimal path through $\mathcal{G}^1$ is shown in the right panel of Figure 6.

Although the convergence of CFDP is not guaranteed by the result in the Appendix, since the objective function is discontinuous, the algorithm still remains plausible and seems to work well in practice. In our experiments we chose the $\{\theta_n\}$ to be 16 equally spaced locations and iterated the CFDP algorithm until all superstates along the optimal path had reached depth 6; superstates were not allowed to divide beyond depth 6. At this point the superstates are about 1 pixel by $\pi/64$ radians in size. The top image of Figure 7 shows the candidate mine locations that we consider and the bottom image shows the optimal convex sets located by CFDP.

Figure 8 shows results from our boundary estimation experiments describing the savings in computation due to using CFDP. The top panel gives $\log_{10}(\text{CFDP}(l)/\text{DP}(l))$ for levels $l = 2\ldots, 6$ for three different boundary estimation problems where $\text{CFDP}(l)$ and $\text{DP}(l)$ are the total number of DP comparisons using CFDP and DP. This is the calculation that dominates the total computation of our algorithm. We could not compute $\text{DP}(5)$ and $\text{DP}(6)$ directly, so in each of the three examples the last two points represent *estimates* of the total log speedup. These estimates are based on the observation that $\text{DP}(l)/\text{DP}(l+1)$ should converge to a constant as l increases. The bottom panel of Figure 8 shows the discretization of the superstate space for a particular angle after finding the optimal boundary at level 6. Note that the majority of the superstate space remains unexplored (c. f. Figure 6).

5 Discussion

The basic idea of CFDP is simple, however, as the preceding might suggest, its application is not always straightforward. For CFDP to work effectively one must identify superstates with the following properties

1. The superstates must be easily partitioned into smaller superstates.
2. One must be able to efficiently compute an optimum over all possible arc costs between two connected superstates, or a good bound for that optimum.
3. The superstate arc scores should be homogeneous. That is, for connected superstates, S_1, S_2, $C(S_1, S_2)$ should be close to $C(s_1, s_2)$ for any $(s_1, s_2) \in \mathcal{E}$.

The necessity of the first two properties is clear; the third property, homogeneity of arc costs, ensures that the superstate arc costs are reasonable proxies for the constituencies they represent. When arc costs are not homogeneous the superstate arc score will be significantly different from some of the constituent arc scores and can make the arc appear to be unrealistically attractive. Thus time is wasted in refining superstates before they can be verified to be suboptimal.

Often the above objectives compete with one another to some extent. Typically simple superstates "shapes" such as rectangles are the easiest to to represent and split, however such superstates do not necessarily give the most homogeneous arc scores. On the other hand, superstates defined with homogeneity in mind might have complex shapes and therefore be difficult to represent and split.

Occasionally it is possible to find a representation of superstates for which one can efficiently compute a optimum for the cost between superstates as in Section 3. More often there is a tradeoff

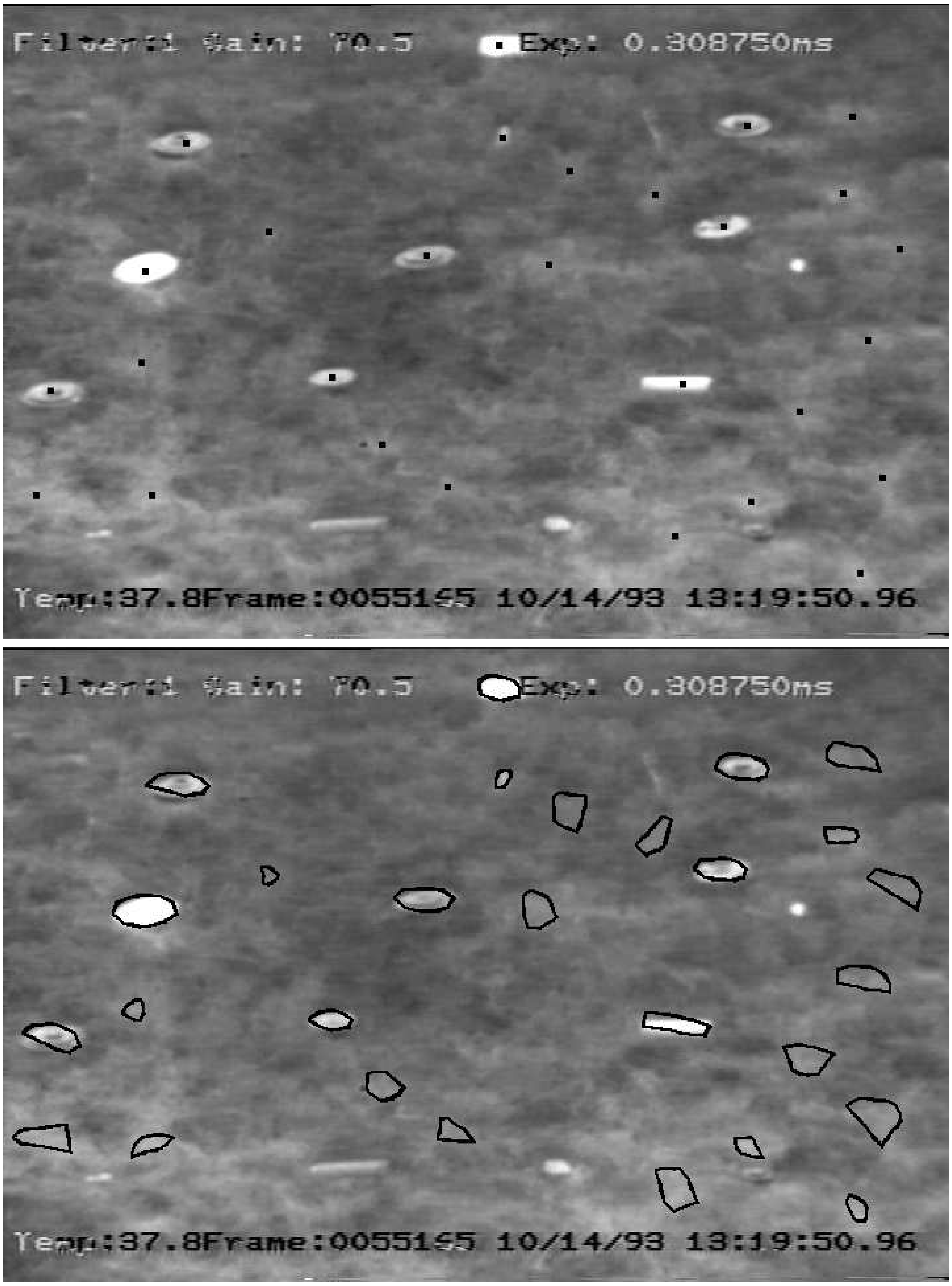

Figure 7: **Top:** The candidate locations. **Bottom:** Optimal convex sets.

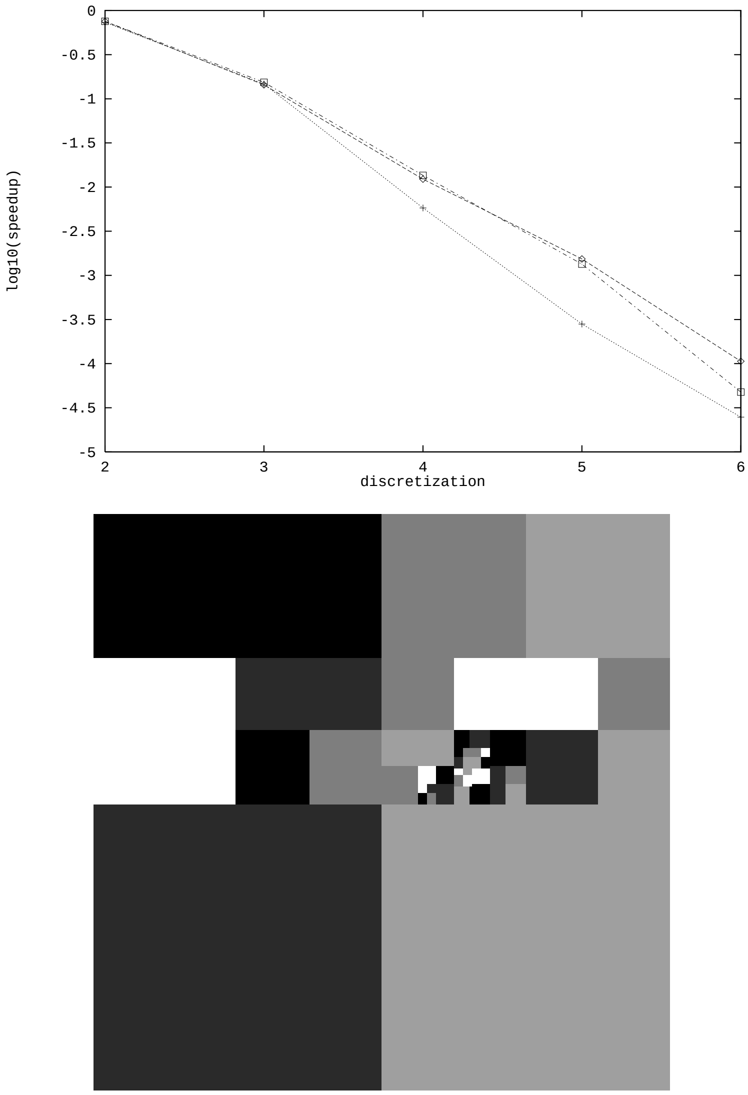

Figure 8: **Top:** $\log_{10}(\mathrm{CFDP}(l)/\mathrm{DP}(l))$ for levels $l = 2\ldots, 6$. **Bottom:** The final discretization into superstates for a typical angle at level 6.

between the quality of the bound on arc costs and the cheapness with which it can be computed. The "art" of using CFDP lies in navigating these tradeoffs effectively. We can offer no prescription for identifying the right choice of superstate for any given problem, or any guarantee that such a choice exists. However, we hope that the examples demonstrate that situations exist that do lend themselves to CFDP. In fact, we believe them to be plentiful.

Successful results with CFDP depend also on aspects of the application domain having nothing to do with the choice of superstates. Specifically, the ideal domain is one in which there are relatively few paths that give nearly optimal scores. In such a problem, CFDP wastes little time refining superstates that do not lie on the eventual optimal path and proceeds directly to the optimal solution thereby realizing a significant savings in computation.

The correctness of our CFDP algorithm relies on bounding the cost of all possible arcs between superstates. In each of the applications we have presented, a genuine upper or lower bound for this arc cost was computed. As a final comment, we remark that good results might be obtained by relaxing this constraint and using a near upper or lower bound, or perhaps even an average arc cost. With this relaxation, we lose the guarantee of eventually identifying an optimal path, however, good paths can be found this way in some problems in which "true" CFDP is impractical.

6 Appendix

The continuous version of CFDP introduced in Section 2 and exemplified in Sections 3 and 4 generates a sequence of approximations to the optimal path through a trellis with a finite number of levels and continuous state space. The approximations are obtained by solving a finite state distillation of the continuous state problem with an "optimistic" version of the objective function. Each approximate solution gives a rectangle for each trellis level that, we hope, contains the optimal state. We show here that, under suitable assumptions, the rectangles collapse around the optimal solution, thus continuous CFDP *converges* to the optimal solution.

Suppose we have a closed subset $\Omega \subset \Re^N$ with diameter $D(\Omega) < \infty$. Let $H : \Omega \to \Re$ be a continuous function we wish to maximize and assume that x^* is the unique maximizer in Ω. We assume the existence of an extension of H to rectangular subsets $X \subset \Omega$ such that $H(x) \leq H(X)$ for all $x \in X$ and such that for every $\epsilon > 0$ there is a $\delta > 0$ with

$$D(X) < \delta \Rightarrow |H(X) - H(x)| < \epsilon$$

for all $x \in X \subset \Omega$. Consider the following process which describes the essential behavior of CFDP. We begin with some initial partition of Ω into a finite number of subsets we denote by $\mathcal{P}_1$ and we generate the subsequent partitions, $\{\mathcal{P}_m\}$, by letting

$$X_m = \arg\max_{X \in \mathcal{P}_m} H(X)$$

and recursively defining $\mathcal{P}_m$ from $\mathcal{P}_{m-1}$ by subdividing X_{m-1} into sets of half the diameter of X_{m-1}. Other elements of $\mathcal{P}_{m-1}$ may also be subdivided to obtain $\mathcal{P}_m$ but it is not required.

Proposition: If $\{x_m\}$ is a sequence of points chosen so that $x_m \in X_m$ for $m = 1, 2, \ldots$ where the $\{X_m\}$ are defined as above, then $x_m \to x^*$.

Before we give the proof of the proposition we make two remarks about the way the proposition relates to CFDP. First of all, the proposition makes no reference to how the $\{X_m\}$ are found. Of course we do this in CFDP with dynamic programming but this is not relevant to the argument presented here. Secondly, we clarify the relationship between the $\{\mathcal{P}_m\}$ of our proposition and the partitions of CFDP. In CFDP, at each iteration, m, we have a partition on the state space for each of the N levels of the trellis. Any set that can be formed as an N-fold cross product of these partitions will be an element of $\mathcal{P}_m$. Thus H is our upper bound on *path* scores.

Proof: Since for any $\Delta > 0$ we can only have $D(X_m) > \Delta$ finitely many times, we must have $D(X_m) \to 0$ and hence

$$H(X_m) - H(x_m) \to 0$$

Also $H(x_m) \leq H(x^*) \leq H(X_m)$ so $H(x_m) \to H(x^*)$. Fix ϵ and observe that, since x^* is the unique maximizer of the continuous function H,

$$\max_{x \notin B(x^*, \epsilon)} H(x) \stackrel{\text{def}}{=} H_\epsilon < H(x^*)$$

where $B(x, \epsilon)$ is the open ϵ-radius ball around x. Choose m such that for $k > m$,

$$H(x^*) - H(x_k) < H(x^*) - H_\epsilon$$

Then $H(x_k) > H_\epsilon$ so $x_k \in B(x^*, \epsilon)$.

References

[1] L. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE,* 77, 257–286.

[2] L. Bahl, F. Jelinek, P. Mercer, "A Maximum Likelihood Approach to Continuous Speech Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI–(52), 179–90, 1983.

[3] K. F. Lee, "Large-Vocabulary Speaker-Independent Continuous Speech Recognition: The Sphinx System," Ph.D. Thesis, Computer Science Dept. Carnegie Mellon Univ. Pittsburgh, PA, 1988.

[4] Bazzi I, Schwartz R., Makhoul J., (1999), "An Omnifont Open-Vocabulary OCR System for English and Arabic," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. 21, No. 6, 495–504.

[5] Mohamed M. and Gader P. (1996), "Handwritten Word Recognition Using Segmentation-Free Hidden Markov Modeling and Segmentation-Based Dynamic Programming Techniques," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. 18, No. 5, 294–302.

[6] G. Kopec and P. Chou, "Document Image Decoding Using Markov Source Models," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 16, 602–617.

[7] Geiger D., Gupta A., Costa L., and Vlontzos J. (1995), "Dynamic Programming for Detecting, Tracking, and Matching Deformable Contours," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. 17, No. 3, 294–302.

[8] Khaneja N., Miller M. and Grenander U., (1998), "Dynamic Programming Generation of Curves on Brain Surfaces," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. 20, No. 11, 294–302.

[9] Muder D. (1988), "Minimal Trellises for Block Codes," *IEEE Transactions on Information Theory,* Vol IT-34, No. 5, 1049–1053.

[10] Wolf J. (1978), "Efficient Maximum Likelihood Decoding of Linear Block Codes Using a Trellis," *IEEE Transactions on Information Theory,* Vol IT-24, No. 1, 76–80.

[11] Fujiwara T., Yamamoto H, Kasami T., Lin S. (1998), "A Trellis-Based Recursive Maximum-Likelihood Decoding Algorithm for Binary Linear Block Codes," *IEEE Transactions on Information Theory,* Vol IT-44, No. 2, 714–728.

[12] Kockanek K. (1998), "Dynamic Programming Algorithms for Maximum Likelihood decoding," Ph.D. Dissertation, Division of Applied Mathematics, Brown University, 1998.

[13] Kasami T., Takata T., Fujiwara T., and Lin S. (1993), "On Complexity of Trellis Structure of Linear Block Codes," *IEEE Transactions on Information Theory,* Vol IT-39, No. 3, 1057–1064.

[14] Merlet N. and Zerubia J. (1996), "New Prospects in Line Detection by Dynamic Programming," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. 18, No. 4, 426–431.

[15] Nilsson N. (1980), "Principles of Artificial Intelligence," Tioga Publishing Co., Palo Alto, CA, pp. 74–84.

[16] Hart P., Nilsson N., and Raphael B. (1968), "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions of Systems Science and Cybernetics,* Vol. SSC-4, No. 2, 98–107.

[17] Pearl J., (1984), "Heuristics: Intelligent Search Strategies for Computer Problem Solving," Addison-Wesley Publishing Co., pp. 61–65

[18] Kam A., (1993), "Heuristic Document Image Decoding Using Separable Markov Models," *M.S. Thesis*, Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science.

[19] Kam A. and Kopec G. (1994), "Heuristic Image Decoding Using Separable Models," *IEEE International Conference on Acoustics, Speech and Signal Processing* Vol. 5, 145–8.

[20] Akhiezer N. (1962), "The Calculus of Variations," Blaisdell Publishing Co., New York, pp. 180–182.

[21] Raphael C. and Geman S. (1997), "A Grammatical Approach to Mine Detection," Proceedings of SPIE, April, 1997, Orlando, FL, *SPIE Vol. 3079, Detection and Remediation Technologies for Mines and Minelike Targets II, 1997*, 316–332.

[22] Barbehenn M., Hutchinson S. (1995), "Efficient Search and Hierarchical Motion Planning by Dynamically Maintaining Single-Source Shortest Paths Trees," *IEEE Transactions on Robotics and Automation,* Vol. 11, num 2, 198–214.

[23] Kambhampati S., Davis L. (1986), "Multiresolution Path Planning for Mobile Robots," *IEEE Transactions on Robotics and Automation,* Vol. RA-2, num 3, 135–145.

[24] Chen P. C., Hwang Y. K. (1998), "SANDROS: A Dynamic Graph Search Algorithm for Motion Planning," *IEEE Transactions on Robotics and Automation,* Vol. 14, num 3, 390–403.

[25] Zhu D., Latombe J.-C. (1991), "New Heuristic Algorithms for Efficient Hierarchical Path Planning," *IEEE Transactions on Robotics and Automation,* Vol. 7, num 1, 9–20.

[26] Fujimura K., Samet H. (1989), "A Hierarchical Strategy for Path Planning Among Moving Obstacles," *IEEE Transactions on Robotics and Automation,* Vol. 5, num 1, 61–69.

Biography

Christopher Raphael was born in Hayward, CA in 1960. He received his MS in Computer and Information Science from the University of CA at Santa Cruz in 1984 and his Ph.D. in Applied Mathematics from Brown University in 1991. He has been an NSF Postdoctoral Research fellowship and is currently on the faculty in the Mathematics and Statistics Department at the University of Massachusetts at Amherst.