[28] Parametric and Inverse-Parametric Sequence Alignment with XPARAL

By D. GUSFIELD and P. STELLING

Introduction

When aligning DNA or amino acid sequences using numerical-based optimization, there is often considerable disagreement about how to weight matches, mismatches, insertions and deletions (indels), and gaps. Most alignment methods require the user to specify fixed values for those parameters, and it is widely observed that the quality of the resulting alignment can be greatly affected by the choice of parameter settings.

Parametric alignment attempts to avoid the problem of choosing fixed parameter settings by computing the optimal alignment as a function of variable parameters for weights and penalties. The goal is to partition the parameter space into regions (which are necessarily convex) such that in each region one alignment is optimal throughout and such that each region is maximal for this property. Thus parametric alignment allows one to see explicitly, and completely, the effect of parameter choices on the optimal alignment. Parametric sequence alignment was first used in a paper by Fitch and Smith[1] and was studied more extensively in papers by Gusfield *et al.*,[2,3] Waterman *et al.*,[4] and Vingron and Waterman.[5] The last four papers concern the number, shape, pattern, and interpretation of the regions. The basic algorithmic ideas for computing two-dimensional parametric decompositions were first developed in contexts other than sequence alignment.[6]

In this chapter we first describe a publicly available, user-friendly interactive software package, XPARAL, that solves the parametric alignment problem; we emphasize newer features in XPARAL. We next illustrate the use of XPARAL by reexamining a study done by Barton and Sternberg[7] on gap weights used in aligning protein secondary structure. Finally, we discuss the empirical and theoretical efficiency of XPARAL. We use stan-

[1] W. Fitch and T. Smith, *Proc. Natl. Acad. Sci. U.S.A.* **80,** 1382 (1983).
[2] D. Gusfield, K. Balasubramonian, and D. Naor, "Proceedings of the Third Annual Symposium on Discrete Algorithms," p. 432. Orlando, FL. 1992.
[3] D. Gusfield, K. Balasubramonian, and D. Naor, *Algorithmica* **12,** 312 (1994).
[4] M. Waterman, M. Eggert, and E. Lander, *Proc. Natl. Acad. Sci. U.S.A.* **89,** 6090 (1992).
[5] M. Vingron and M. Waterman, *J. Mol. Biol.* **235,** 1 (1994).
[6] D. Gusfield, *J. ACM* **30,** 551 (1983).
[7] G. Barton and M. Sternberg, *Protein Eng.* **1,** 89 (1987).

dard terminology for sequence alignment, but note that the term gap refers to a maximal contiguous run of spaces (possibly only a single space).

Parametric Alignment and XPARAL Features

The XPARAL package allows the user to specify alignment objective functions with or without character-specific scoring matrices (such as PAM matrices). The treatment (mathematical and algorithmic) of these two cases is somewhat different and is discussed separately.

Any alignment $\mathcal{A}$ of two strings contains a specific number of matches, mismatches, indels, and gaps. We denote these numbers by $mt_{\mathcal{A}}$, $ms_{\mathcal{A}}$, $id_{\mathcal{A}}$, and $gp_{\mathcal{A}}$, respectively. Without the use of character-specific scoring matrices, the value of alignment $\mathcal{A}$ is therefore $v_{\mathcal{A}}(\alpha, \beta, \gamma, \delta) \equiv \alpha mt_{\mathcal{A}} - \beta ms_{\mathcal{A}} - \gamma id_{\mathcal{A}} - \delta gp_{\mathcal{A}}$, where α, β, γ, and δ are variables that adjust the relative importance of matches, mismatches, indels, and gaps. Note that the value of the alignment is a linear function of the four parameters. When these four parameters have fixed values α_0, β_0, γ_0, and δ_0, then the fixed-parameter problem is to find an alignment $\mathcal{A}$ maximizing the objective function: $\alpha_0 mt_{\mathcal{A}} - \beta_0 ms_{\mathcal{A}} - \gamma_0 id_{\mathcal{A}} - \delta_0 gp_{\mathcal{A}}$.

The XPARAL package allows the user to select two of the parameters α, β, γ, and δ to be variable, and to choose constant settings for the other two parameters. For illustration, suppose γ and δ are chosen to be variable, while α and β are fixed at one. This is a choice that is typical for aligning amino acid sequences. As a function of two variable parameters, the value of alignment $\mathcal{A}$ specifies a plane in three-dimensional space. Thus,[2-4] for any pair of input sequences, the γ, δ parameter space decomposes into convex polygons such that any alignment which is optimal for some α, γ point in the interior of a polygon $\mathcal{P}$ is optimal for all points in $\mathcal{P}$ and nowhere else. XPARAL computes and displays this polygonal decomposition. The user can then select any particular polygon $\mathcal{P}$ to see an alignment that is optimal for all points in $\mathcal{P}$, and to display the number of matches, mismatches, indels, and gaps in that alignment. There may be alternative alignments that are optimal throughout $\mathcal{P}$ (and nowhere else), but all those cooptimal alignments have exactly the same number of matches, mismatches, indels, and gaps. XPARAL can count the number of cooptimal alignments in polygon $\mathcal{P}$ and display each one in turn.

Figure 1 shows the XPARAL display of one such polygonal decomposition of the γ, δ space. The grid points are superimposed and will be explained later. The topmost windows display menu buttons. The two horizontal windows below the menu buttons display the input sequences (which must be scrolled to see the complete sequences); the main window shows the polygonal decomposition and contains one dark polygon; the horizontal

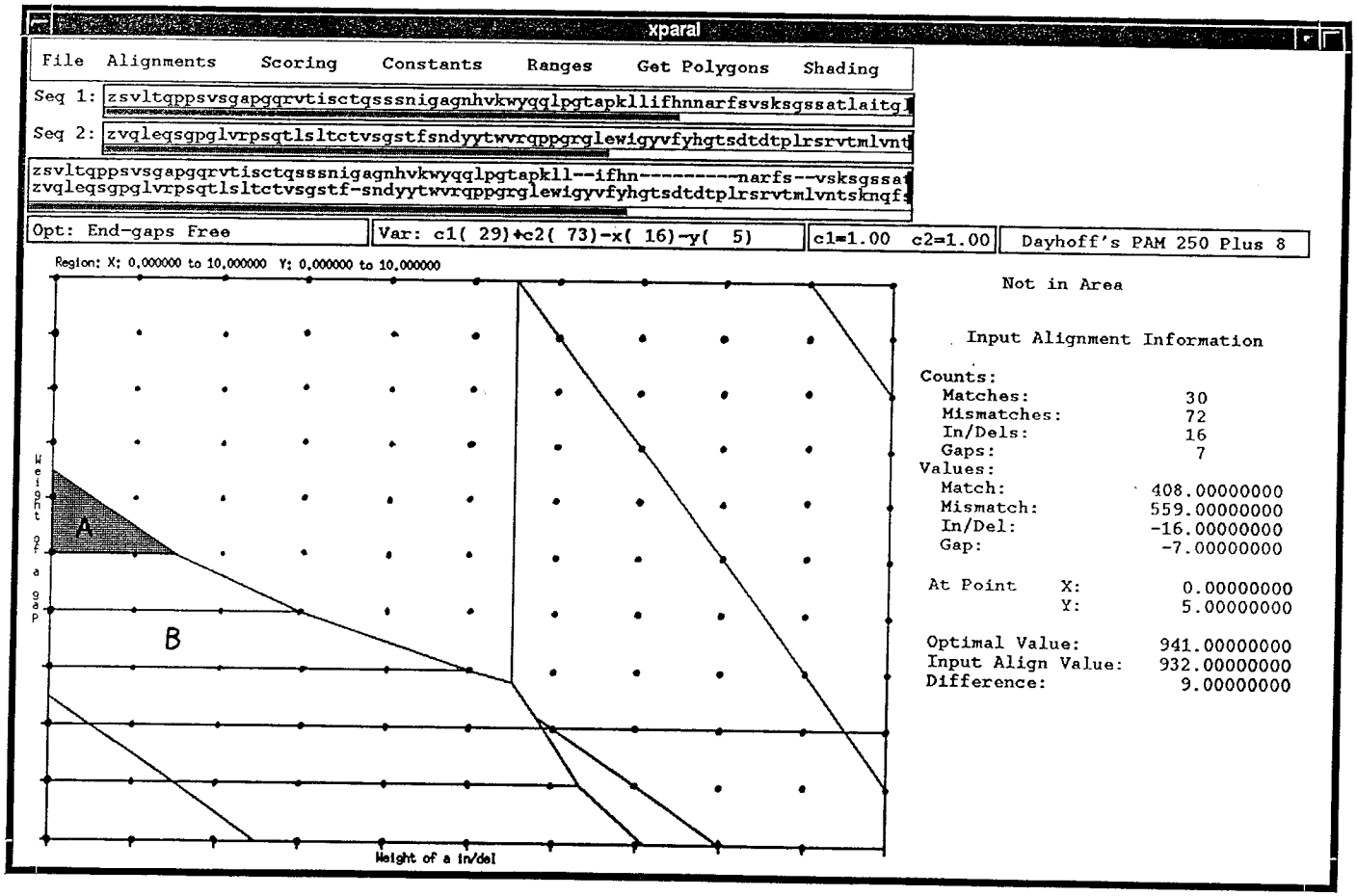

FIG. 1. XPARAL display for indels versus gap weights.

window below the input windows shows an alignment that is optimal for all the γ, δ points in the dark polygon. The smaller horizontal windows below the alignment window give information on the settings the user has selected as well as on the number of matches, mismatches, spaces, and gaps in the displayed alignment. Those numbers remain the same throughout the entire polygon, but the value of the alignment changes throughout the polygon. When the user selects a particular point, the value of the optimal alignment at that point is displayed to the right of the main window.

Use of Scoring Matrices

The XPARAL package allows the user to define and use any character-specific scoring matrix, such as a PAM matrix, to assign a weight to each possible pair of characters (either a match or a mismatch). With the use of scoring matrices we let $smt_{\mathscr{A}}$ denote the total weight given to the matches in alignment $\mathscr{A}$ and let $sms_{\mathscr{A}}$ denote the total weight given to the mismatches in $\mathscr{A}$. Then the value of $\mathscr{A}$ is $v_{\mathscr{A}}(\alpha, \beta, \gamma, \delta) \equiv \alpha smt_{\mathscr{A}} + \beta sms_{\mathscr{A}} - \gamma id_{\mathscr{A}} - \delta gp_{\mathscr{A}}$. The term $\beta sms_{\mathscr{A}}$ is added, rather than subtracted, because the scoring matrices already incorporate the appropriate sign of the mismatch penalty. When selecting two parameters to be variable, the value of alignment $\mathscr{A}$ is again a function of those two variable parameters and defines a plane in three-space.

Choice of Alignment Model and Input Mode

Using XPARAL, the user can specify whether the alignment should be global (Needleman–Wunsch), global with end gaps free, local (Smith–Waterman), or internal, where end spaces in the longer string are penalized but end spaces in the shorter string are ignored. Sequences are input to XPARAL either from files or at the input window. XPARAL also allows the input sequences to be taken from a file that holds a previously determined reference alignment $\mathscr{A}$. Then, when the user selects a point p in the decomposition, XPARAL displays the value of the input alignment at p, the value of the optimal alignment at p, and the difference between the two values. In this mode, XPARAL can also solve the inverse-parametric problem: For what points in the γ, δ space does the optimal alignment (computed by dynamic programming) have a value which is closest to the value of the reference alignment $\mathscr{A}$? That is, if we let $v(\alpha, \beta, \gamma, \delta)$ denote the value of the optimal alignment at point $(\alpha, \beta, \gamma, \delta)$, then we seek the γ, δ points where $v(\alpha_0, \beta_0, \gamma, \delta) - v_{\mathscr{A}}(\alpha_0, \beta_0, \gamma, \delta)$ is minimum. These points are called inverse-optimal.

By convexity, the inverse-optimal point(s) must occur either at a single vertex, at a single edge (line segment between two vertices), or at a single complete polygon of the polygonal decomposition. The latter case happens when the plane representing the value of $\mathcal{A}$ is parallel (or identical) to a plane in the decomposition. For efficiency, the inverse parametric problem is solved using gradient-descent rather than by first finding the complete decomposition.

Inverse-parametric computation is useful for trying to deduce parameter settings where the optimal alignment (with respect to a chosen objective function) might likely reconstruct correct alignments that have been determined by other methods. Of course, a parameter setting that minimizes the numerical difference between the value of the optimal alignment and the value of the reference alignment is not necessarily a parameter setting where the optimal alignment is most similar in form (an undefined concept) to the reference alignment. In experiments we have done, however (see the next section), we see a good correlation between this numerical distance and the ability to recapture significant features of the reference alignment. Inverse-parametric computations also allow an efficient, but rough, test of the validity of both the alignment model and the reference alignment. For if a numerical-based alignment model is valid in some biological setting, then a known alignment that is correct in that setting should have a numerical value that is close to optimal, at least for some points in the parameter space.

XPARAL is written in C++ and compiled for the DECstation 5000 and the DEC α. Compiled versions can be obtained at the following web site: http://wwwcsif.cs.ucdavis.edu/~gusfield/strpgms. Source code is available from the authors on request.

Using XPARAL to Study Gap Models for Secondary Structure

We illustrate one use of XPARAL by reexamining a study done by Barton and Sternberg[7] on the effectiveness of global and end-gap free sequence alignment to correctly align protein secondary structures. Their study examined sequences whose three-dimensional structure was known. Superimposing the structures for two proteins gives a reference alignment against which they evaluated alignments computed from sequence data alone. They scored a sequence alignment $\mathcal{A}$ by identifying the residues in regions of known secondary structure and counting the number of those residue pairs which align the same in $\mathcal{A}$ as in the reference alignment.

The sequence alignment objective function they used was the following: Maximize $smt_{\mathcal{A}} + sms_{\mathcal{A}} - \gamma id_{\mathcal{A}} - \delta gp_{\mathcal{A}}$ over all alignments $\mathcal{A}$, where the scoring matrix was integer-rounded PAM250 with a constant of 8 added

to every entry.[8] They considered every integer combination of γ and δ from 0 to 10 and found alignments with optimal values (using both global and end-gap free models) at each of those 121 integer γ, δ combinations. Each optimal was then scored, as above, to evaluate it against the reference alignment. They considered five such pairs of sequences, but full details were published for only one pair (immunoglobulins FABVL versus FABVH). In that case, the best of the 121 optimal sequence alignments they found received a score of 32 of a possible maximum of 41. Optimal global and end-gap free alignments at the 121 integer points were next computed using a new alignment model that penalizes gaps and indels in the known regions of secondary structure more heavily than gaps and indels outside those regions. With this differential gap model, they found several integer points where the optimal alignments had scores of 36.

One of their main conclusions[7] is that standard alignment models are not adequate for aligning secondary structure, but the differential gap model described above is effective. A similar conclusion was reached by Lesk *et al.*[9] The final recommendation, to use secondary structure information when available, is not in dispute, but such information is not always available. Moreover, since secondary structure is often predicted from sequence alone, one would like an independent alignment method to test those predictions, rather than simply incorporate them. So the effectiveness of standard alignment models remains of interest.

Vingron and Waterman[5] have also examined the immunoglobulins FABVL and FABVH using their parametric alignment program. In contrast to the previous study, they found polygons in the γ, δ space where the resulting alignment agreed with the main features of the known structural alignment. This gives a different picture of the effectiveness of standard alignment models to align secondary structure. However, Vingron and Waterman[5] used corrected sequences compared to those used by Barton and Sternberg,[7] modified PAM250 differently, used local alignment, and did not use a numerical score to evaluate alignment quality. So those differences alone might account for the differing results. Alternatively, the different results might be due to chance, since only one optimal alignment was computed at each sample point, even if several cooptimal alignments exist, or it may be that complete parametric decomposition yields a wider, more informative range of alignments than is obtained from the (grid) sampling approach.[7]

[8] For clarity, the score of an alignment $\mathscr{A}$ is the number given by the Barton and Sternberg evaluation criteria, while the value of $\mathscr{A}$ is the number given by the dynamic programming objective function. The term optimal refers to the value.

[9] A. Lesk, M. Levitt, and C. Chotia, *Protein Eng.* **1**, 77 (1986).

We used XPARAL to examine why the two studies obtained differing results. We first kept the exact alignment conditions of Barton and Sternberg[7] (using their exact sequences, using integer-rounded PAM250 plus 8, and computing global and end-gap free alignments), but we used XPARAL to completely decompose the γ, δ parameter space. We considered three questions. First, with these conditions, are there polygons where an optimal alignment has a larger score than 32? Second, if there are such alignments, were they missed[7] because of a limited choice of test points, or because the polygons are small, or because only a single optimal alignment was computed at each test point? Third, how well does the inverse-parametric feature of XPARAL find parameter settings where high-scoring alignments are obtained?

Results from Empirical Study

The polygonal decomposition obtained is shown in Fig. 1, with the 121 integer points superimposed. The striking feature is that although there are only seventeen polygons in the decomposition, the 121 integer points miss the interior of all but four of the seventeen polygons. Moreover, several of the polygons have multiple (up to 12) sample points on their boundaries but none in their interiors, so polygon size is not the reason that interior points were missed. Conversely, more than 40% of the test points fall into the three largest polygons, and so give redundant information.

Sampling at interior points gives robust alignments since any alignment that is optimal at an interior point is optimal throughout the polygon, whereas an alignment that is optimal on an edge or vertex may only be optimal on that edge or vertex. In particular, with only seventeen alignments, no mater what γ, δ point p is selected (in Fig. 1), one of those seventeen alignments will be optimal at p. The larger set of 121 alignments computed at integer sample points need not have that property. Interior points are further desirable because the number of cooptimal alignments in the interior of a polygon is always less than the number on any of its boundaries, except at borders of the parameter space. So although the integer points provide a dense, regular, sampling of the parameter space, they may only provide a biased representation of the alignment space and miss important alignments. We believe this happened.

Without using differential gap penalties, XPARAL found a polygon containing optimal alignments with score as high as the best alignments found when Barton and Sternberg[7] used their differential gap penalties. The interior of the polygon labeled A contains 24 cooptimal alignments, 18 with a score of 36 and 6 with a score of 32. The alignment XPARAL

displayed for polygon A had a score of 36. The differences among the 18 high-scoring alignments are outside of the regions of secondary structure, as are the differences among the 6 lower scoring alignments. So with respect to secondary structure, there are effectively two different optimal alignments in polygon A. Three integer points fall on the boundary of A, but none falls in the interior. The polygon bordering A from below has six cooptimal alignments, all with score 32. So the three alignments computed[7] at the grid points on the boundary of A missed the high scoring alignments, essentially by chance. However, the chance that a deterministically computed alignment is one of the high-scoring alignments is improved by sampling in the interior of A. For example, when an alignment is computed on the boundary of A and the polygon below it, the algorithm chooses a single alignment (and always the same one no matter how many points on the boundary are sampled) from among 18 with a high score and 12 with a low score. Inside A, the algorithm choses from among 18 with a high score and 6 with a low score. The way, then, to optimize the chances that a deterministic algorithm will find a high-scoring alignment, without enumerating cooptimals, is to compute one alignment in the interior of each polygon and one on each boundary and vertex.

We have no theorem that integer points miss polygon interiors with high probability, but it is not surprising that they do (especially near parameter settings where biologically reasonable alignments occur), since one should expect those boundaries to preferentially contain integer points. To see this, consider the equations for the values of two neighboring alignments $\mathscr{A}_1$ and $\mathscr{A}_2$, and let id_i and gs_i denote the number of indels and gaps in alignment $\mathscr{A}_i$. Each equation describes a plane, and every polygon boundary lies on the intersection of two such planes. A boundary therefore lies on a line

$$\delta = \frac{C}{gs_2 - gs_1} + \gamma \frac{id_1 - id_2}{gs_2 - gs_1}$$

where C is the total of the match and mismatch weights for $\mathscr{A}_2$ minus the total of the match and mismatch weights for $\mathscr{A}_1$. With an integer PAM matrix, C is an integer, and so the intercept of the line will be an integer if C is a multiple of $gs_2 - gs_1$. That is likely because $gs_2 - gs_1$ is an integer and tends to be small, often one. The reason is that the number of gaps tends to be small for real protein data, and the number of gaps must monotonically fall along any vertical line of increasing δ, whenever the line crosses a boundary. For example, in Fig. 1, on the vertical line starting at (0.25, 0) and ending at (0.25, 10), the alignments encountered have 14, 11, 10, 8, 7, 6, 5, and 3 gaps in that order. Those two facts (small gap number

an monotonic change) suggest that $gs_2 - gs_1$ will tend to be a small integer for alignments in neighboring polygons. In fact, 17 of the 27 interior boundary edges in Fig. 1 lie on lines with integer intercepts, and 5 more have half-integer intercepts. The argument for integer (or small denominator) slope is similar, and 21 of the 27 boundary edges have integer slopes. Integer (or half-integer) slopes and/or intercepts imply that the edges will preferentially contain integer points.

Results from Inverse-Parametric Alignment

We used the same immunoglobulin sequences to test the inverse-parametric feature of XPARAL. For reference alignment we used the alignment of score 36 that was obtained[7] with differential gap and indel penalties. XPARAL found that all the points in the polygon labeled B (Fig. 1) are inverse-optimal, and hence the plane for the reference alignment is parallel to the plane for polygon labeled B. The reference alignment has 30 matches, 72 mismatches, 16 indels, and 7 gaps, whereas the three optimal alignments for polygon B have 29 matches, 73 mismatches, 16 indels, and 7 gaps. It is interesting how similar the two vectors are. Moreover, the optimal alignment value (at every point in B) differs from the reference alignment value by less than 1%. So the reference alignment is very close to optimal in that polygon, although each of the three cooptimals in B have a score of only 32. This provides a rough confirmation of the validity of the standard alignment model (applied to FABVL and FABVH sequences). There are over 2^{200} alignments of these two sequences, and yet there are only 120 alignments that are (co-)optimal anywhere in the entire γ, δ parameter space, and only 110 (co-)optimal alignments inside the 17 polygons found in the bounded 10 by 10 region.

Related Results

All of the above results remain essentially the same when global alignment is used in place of end-gap free alignment. However, there are alignment models in which the results are stronger. For example, when PAM250 is used instead of PAM250 plus 8, polygon A essentially expands to contain three interior integer points, and XPARAL again returns a displayed alignment of score 36. Departing completely from these conditions, using global alignment, choosing α and γ to be variable, and setting $\beta = \delta = 0$ (as is sometimes suggested for DNA), but using the PAM250 matrix, then the decomposition contains 11 polygons, including a large one where the reference alignment[7] is optimal. In other words, the standard alignment model

without differential gap weights not only returns an alignment with a score as good as the reference alignment, it returns the reference alignment exactly.

Conclusion from Empirical Study

To study properties of particular alignment models, parametric alignment can more selectively and effectively home in on critical parameter regions than can a grid sampling of the parameter space. Cooptimal or near-optimal alignments in those critical regions can then be generated and studied in more depth. Generating a range of alignments, such as all of the optimal alignments from each polygon that neighbors the polygon which contains some initial parameter setting, also provides an alternative to the recommended practice of generating suboptimal alignments for a single fixed parameter setting.

Efficiency of XPARAL

We have put a great deal of effort into making XPARAL practical. The decomposition shown in Fig. 1 took less than 6 sec to compute using a DEC α, and 14 sec on a slower DECstation 5000/25. As another empirical measure, XPARAL computed only 160 fixed-parameter alignments to determine that decomposition. With additional programming effort, that number can be reduced by about one-half. We will now prove that for most alignment models, the number of fixed-parameter alignments that XPARAL must compute to find the decomposition is proportional to the number of polygons in the decomposition. This issue was stated as an open question by Pevzner and Waterman.[10] The result to be established here was claimed earlier[3] without a proof.

Basic Algorithm for XPARAL

We need to describe the inner workings of XPARAL at a high level. Our approach is different in some important ways from the Waterman *et al.*[4] method, and does not use infinitesimals. For illustration, we again assume that γ and δ are the variable parameters. The workhorse of XPARAL is the following problem called the ray-search problem: Given an alignment $\mathscr{A}$, a point p where $\mathscr{A}$ is optimal, and a ray h in γ, δ space starting at p, find the furthest point (call it r^*) from p on ray h where $\mathscr{A}$ remains optimal. If $\mathscr{A}$ remains optimal until h reaches a border of the

[10] P. Pevzner and M. Waterman, *Proc. Israel Symposium on Theory of Computing and Systems*, 158. (1995).

parameter space, then r^* is that border point on h. The ray-search problem is solved as follows:

Set r to the (γ, δ) point where h intersects a border of the parameter space.
While $\mathscr{A}$ is not an optimal alignment at point r do
begin
Find an optimal alignment $\mathscr{A}^*$ at point r.
Set r to be the unique point on h where the value of $\mathscr{A}$ equals the value of $\mathscr{A}^*$.
end;
Set r^* to r.

This algorithm is Newton's classic zero finding method specialized to a piecewise linear function. The following three facts will be needed in the analysis and are easy to establish: Newton's method finds r^* exactly; unless $\mathscr{A}$ is optimal at the initial setting of r, the last computed alignment $\mathscr{A}^*$ is cooptimal with $\mathscr{A}$ at r^* and yet is also optimal on h for some nonzero distance beyond r^*; and, when it computes an alignment at a point r on h, none of the alignments computed previously (in this execution of Newton's algorithm) are optimal at r.

We now explain how to find the edges of polygon $\mathscr{P}(\mathscr{A})$, given an alignment $\mathscr{A}$ that is optimal at a point p, and known to be optimal for an (unknown) polygon $\mathscr{P}(\mathscr{A})$. First pick any ray h from p and solve the ray-search problem along h. There are two degenerate cases that can occur: one is that r^* lies on a border of the parameter space, and the other is that r^* is a vertex of the decomposition. We will consider those degenerate cases later and assume for now that they do not occur. Therefore, the ray search along h will find a point r^* that lies on an edge e of polygon $\mathscr{P}(\mathscr{A})$. By Newton's second fact, the ray search will also return an alignment $\mathscr{A}^*$ that is optimal in the interior of the polygon bordering edge e. The intersection of the two planes for $\mathscr{A}$ and $\mathscr{A}^*$ describes a line l^* that contains edge e, so the full extent of e can be found by solving two more ray-search problems using $\mathscr{A}$. In one problem, ray h is the half-line of l^* starting at r^* and running in one direction along l^*, and in the other problem ray h is the remaining half-line of l^* in the other direction. These two ray searches find the opposite endpoints of edge e. Once edge e is fully described, we select another ray h from p that does not intersect edge e, and find a second edge of $\mathscr{P}(\mathscr{A})$. By linking identical endpoints of edges of $\mathscr{P}(\mathscr{A})$ that have been found, it is easy to continue selecting rays from p that do not intersect previously discovered edges or vertices of $\mathscr{P}(\mathscr{A})$. In this way, we find all the edges of $\mathscr{P}(\mathscr{A})$, stopping when the discovered edges of $\mathscr{P}(\mathscr{A})$ link together to form a closed cycle.

Consider now the two degenerate cases that may occur when trying to find an edge of $\mathscr{P}(\mathscr{A})$. In the case that r^* is on a border of the parameter space, then that border line is used in place of l^*. In the other case, when r^* is a vertex, the algorithm will realize this because $\mathscr{A}$ will not be optimal past r^* on at least one of the two rays on l^* from r^*. When this occurs, the algorithm simply begins a new ray search from p using a ray that avoids r^* and all other previously discovered vertices and edges.

To compute a full polygonal decomposition, one first finds an alignment that is sure to be optimal for some (unknown) polygon. This is easy to do with a constant number of ray searches, and we omit details. Now we explain how the algorithm finds successive polygons. When finding the first polygon $\mathscr{P}$ (and for each additional polygon it finds), the algorithm inserts into a list, L, one distinct vector $(smt_{\mathscr{A}^*}, sms_{\mathscr{A}^*}, id_{\mathscr{A}^*}, gp_{\mathscr{A}^*})$ for each alignment $\mathscr{A}^*$ found to be optimal in the interior of a polygon bordering $\mathscr{P}$. When $\mathscr{P}$ is finished, the algorithm finds and marks one of the unmarked vectors from L, say for $\mathscr{A}'$, and then finds the polygon $\mathscr{P}(\mathscr{A}')$ where $\mathscr{A}'$ is optimal. The parameter space will be fully decomposed when all vectors in L are marked. Since the algorithm never chooses a marked vector, nor inserts two equal vectors, and since when a polygon is found the algorithm learns one alignment optimal at the interior or each neighboring polygon, each polygon in the full decomposition is found exactly once.

Time Analysis and New Idea

The above details lead to the following time analysis. Let R, E, and V be the number of polygons, edges, and vertices, respectively, in a decomposition, and let $O(nm)$ be the time to compute a single fixed-parameter alignment for sequence of lengths n and $m > n$. How many ray searches are executed to find a polygon $\mathscr{P}(\mathscr{A})$, given $\mathscr{A}$ and p? Let d be the number of edges of $\mathscr{P}(\mathscr{A})$. Then $3d$ ray searches are done to find the edges of $\mathscr{P}(\mathscr{A})$, and, in the highly degenerate case that selected rays from p intersect all the vertices of $\mathscr{P}(\mathscr{A})$, then another $3d$ (wasted) ray searches may be done as well. Hence at most $6d$ ray searches suffice to describe $\mathscr{P}(\mathscr{A})$. Each edge lies on at most two polygons, so the algorithm does at most $12E$ ray searches to find the complete decomposition. Further, from Newton's third fact each ray search requires at most R fixed-parameter alignment computations, so the complete decomposition requires at most $12RE$ fixed-parameter alignments which can be done in $O(ERnm)$ time. This unsatisfactory bound will be improved with one additional idea.

The new idea is to modify the Newton algorithm (given $\mathscr{A}$) to pick the initial point r far enough on h to be at or beyond the (unknown) point r^*, yet as close to r^* as present information allows. Consider any alignment

$\mathcal{A}'$ computed before the present execution of Newton's method. Compute the intersection of the planes for $\mathcal{A}$ and $\mathcal{A}'$ and project that line onto the γ, δ plane. If the projection intersects h, then the initial r need not be any further from p than r', since $\mathcal{A}'$ has greater value than $\mathcal{A}$ beyond r'. If the projection misses h, then $\mathcal{A}$ has greater value than $\mathcal{A}'$ at every point on h. Any intersection and projection can be done in constant time. Repeating this for each previously computed alignment $\mathcal{A}'$, we set the initial r to the point closest on h to p among all the computed r' points.

The modified Newton method clearly reduces the number of needed alignments, but by how much, and how much added time is needed to implement it? During the entire algorithm we will keep a list L' that is a superset of list L. Whenever any alignment is computed, the vector for that alignment is placed into L', if it is not already there. When we begin any ray search, we use L' as explained above to find the initial point r. It takes constant time to compute each point r', so the added cost of using L' in a single ray search is proportional to the size of L'. We claim that size is at most $V + E + R$, that is, over the entire running of the algorithm only $V + E + R$ distinct vectors will be computed. This follows because XPARAL is a deterministic algorithm so that no matter how many times it might compute an optimal alignment at the same vertex, it always returns the same alignment (hence, same vector). Similarly, even when XPARAL computes alignments at different points on the same polygon edge, it will return the same alignment each time, and it is true for alignments inside a polygon, since all optimal alignments inside a polygon have the same vector. So the added bookkeeping time for using L' is just $O(V + E + R)$ per ray search or $O[12E(V + E + R)]$ overall.

For the analysis of the number of needed fixed-parameter alignments, call an alignment computation redundant if it returns a vector that is already in L'. We claim that in any single ray search (using modified Newton with alignment $\mathcal{A}$) only the last alignment computation in that search could be redundant. To see this, note that if $\mathcal{A}$ is optimal at the initial r, then only one alignment is computed in that ray search; otherwise, the redundant alignment, $\mathcal{A}'$, is computed at a point closer to p than the initial r. Since, by Newton's third fact, each vector computed during the ray search is distinct, $\mathcal{A}'$ must have been in L' before the present ray search. But that would contradict the choice of the initial r.

We can now analyze the time to find a complete polygonal decomposition. There are still at most $12E$ ray searches, and, in each, at most one computed alignment is redundant. Each other alignment computation finds a new vector to add to L' (which has size at most $V + E + R$), so the complete polygonal decomposition is computed using at most $V + 13E + R$ fixed-parameter alignments. This leads to an overall time bound of

$O[12E(V + E + R) + (V + 13E + R)nm]$. Now a polygonal decomposition can be viewed as a connected planar graph, and when each vertex is incident with at least three edges (as in the case of a polygonal decomposition), then $V \leq E \leq 3R$. This is easy to show using Euler's classic theorem, but is not true for general planar graphs. So the terms $12E$ and $V + E + R$ and $V + 13E + R$ are each proportional to R, that is, each are $O(R)$. Hence, the above time bound becomes $O(R^2 + Rnm)$, which is $O(R + nm)$ per polygon.

The bound of $O(R + nm)$ per polygon holds no matter what choices are made in XPARAL. However, it was shown[2,3] that when no character-specific scoring matrices are used, and two variable parameters are picked, then $R = O(nm)$. In fact,[2,3] for global alignment, when no scoring matrices are used then $R < n^{2/3}$. When scoring matrices are used, but γ and δ are the chosen variable parameters, then again $R = O(nm)$. This establishes the claim that for most of the (important) parameter choices, a full polygonal decomposition can be found in $O(nm)$ time per polygon, proportional to the time needed to compute just a single fixed-parameter alignment. It is unknown how big R can be when scoring matrices are used and α, β are the variable parameters. The algorithm description and analysis above is much cruder than what is actually implemented in XPARAL but suffices for the main result.

Acknowledgments

We thank John Nguyen for helpful efforts and insights in working with XPARAL. Research was partially supported by Grant DE-FG3-9ER6999 from the U.S. Department of Energy.