Regularization Schemes for the Real Time Spatial Management of Pelagic Longline Fisheries

Kyle Egerdal
kegerdal@pugetsound.edu

Follow this and additional works at: http://soundideas.pugetsound.edu/summer_research

Part of the Discrete Mathematics and Combinatorics Commons, Dynamical Systems Commons, Other Applied Mathematics Commons, and the Other Mathematics Commons

Recommended Citation
http://soundideas.pugetsound.edu/summer_research/140

This Article is brought to you for free and open access by Sound Ideas. It has been accepted for inclusion in Summer Research by an authorized administrator of Sound Ideas. For more information, please contact soundideas@pugetsound.edu.
Spatial Management of Pelagic Longline Fisheries
Kyle Egerdal, Adviser: Carl Toews
The University of Puget Sound
Summer 2011

Background

Bycatch from commercial fisheries often includes critically endangered species. Fisheries managers can ameliorate this problem by channeling fishing effort into regions where threatened species are scarce. In the case of Bluefin tuna, local habitat preferences correlate strongly with ocean depth and temperature profiles. Coupling satellite temperature data with a regional and seasonal depth-temperature model allows managers to make near-real time spatial estimates of Bluefin prevalence. These estimates can be used to allocate fishing zones. A key challenge is to automate this process in a way that yields intelligible boundaries while balancing economic and environmental costs.

An Optimization Problem

Consider a regular square lattice, each node associated with a “tuna suitability” score between 0 and 1. The objective is to draw 2 lines, \( \Lambda_1 \) and \( \Lambda_2 \), each with at most \( m \) segments, dividing the lattice into zones A, B, and C, where A = open fishing, B = limited fishing (permit based), and C = closed. Following [1], the \( (i,j) \)’th lattice point is assigned a classification penalty \( p(s_{ij}) \) that depends on the zone of the lattice point and its suitability score \( s_{ij} \):

<table>
<thead>
<tr>
<th>Suitability</th>
<th>Zone A</th>
<th>Zone B</th>
<th>Zone C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 ≤ ( s_{ij} ) ≤ 1/3</td>
<td>( p(s_{ij}) = 0 )</td>
<td>( p(s_{ij}) = U )</td>
<td>( p(s_{ij}) = 0 )</td>
</tr>
<tr>
<td>1/3 ≤ ( s_{ij} ) ≤ 2/3</td>
<td>( p(s_{ij}) = L_1 )</td>
<td>( p(s_{ij}) = 0 )</td>
<td>( p(s_{ij}) = L_1 )</td>
</tr>
<tr>
<td>2/3 &lt; ( s_{ij} ) ≤ 1</td>
<td>( p(s_{ij}) = L_2 )</td>
<td>( p(s_{ij}) = U )</td>
<td>( p(s_{ij}) = L_2 )</td>
</tr>
</tbody>
</table>

Note that higher values of \( U \) support “fish-friendly” policies, while higher values of \( L \) support “fisherman-friendly” policies. The total cost associated with boundary lines \( \Lambda_1 \) and \( \Lambda_2 \) is given by:

\[
\Gamma(\Lambda_1, \Lambda_2) = \sum_{i,j=1}^{n} p(s_{ij})
\]

For a given constellation of suitability scores, the optimization problem is to minimize this cost function.

Regularization

Zoning boundaries change over time in response to changing temperatures. Since large boundary movements are expensive to fishermen, we can discourage them by adding another penalty term to the cost:

\[
\Gamma(\Lambda_1^t, \Lambda_2^t) = \sum_{i,j=1}^{n} p(s_{ij}^t) + \gamma \sum_{i=1}^{2} ||\Lambda_i^t - \Lambda_i^{t-1}||
\]

Methods of Optimization

Both the regularized and the unregularized problems involve combinatorial optimization. Since this tends to be computationally intensive, solving a “relaxed” problem is often necessary in practice [2]. The following chart compares several novel relaxation methods. The best method will be an effective mediation between computational complexity and managerial efficiency.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Complexity</th>
<th>Estimated Time for a 20x20 Lattice</th>
<th>Estimated Time for a 100x100 Lattice</th>
<th>Sample Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete Search</td>
<td>Calculates every possibility and chooses the one with the lowest penalty.</td>
<td>( n^2 )</td>
<td>5 seconds</td>
<td>2 days</td>
<td>Sample</td>
</tr>
<tr>
<td>'Curves'</td>
<td>Optimizes once for every y-coordinate, but does not combine y-coordinates. Messy.</td>
<td>( n^2 )</td>
<td>2 seconds</td>
<td>30 seconds</td>
<td></td>
</tr>
<tr>
<td>'Segments'</td>
<td>Checks only segment combinations with endpoints on “Curves.”</td>
<td>( n^2 + n(n-1) )</td>
<td>1 segment: 10 seconds</td>
<td>2 segments: 5 minutes</td>
<td></td>
</tr>
<tr>
<td>'Greedy'</td>
<td>Starts at the bottom and takes the segment with the least per-height penalty until it reaches the top.</td>
<td>( n + n^2(n-1) )</td>
<td>5 seconds</td>
<td>1 hour</td>
<td></td>
</tr>
<tr>
<td>Gradient Dissent</td>
<td>Disregarding any penalty during calculation, moves from the bottom along the gradient of the desired temperature until it reaches the top.</td>
<td>Dependent on complexity of the distribution</td>
<td>0-5 seconds</td>
<td>0-5 seconds</td>
<td></td>
</tr>
</tbody>
</table>

Future Work

Questions we would like to consider include:
- How disastrously wrong can a very simple line allocation go?
- What happens if we penalize the number of line segments?
- Should we penalize lines for being “unnecessarily” long or short?

REFERENCES: