Princeton University
Department of Operations Research
and Financial Engineering

ORF 201
Computer Methods in Problem Solving

Final Project:
Dynamic Programming — Optimal Sailing Strategies

Due 11:59 pm, May 9, 2000

---

1. Problem Description

In this project, we wish to find the quickest path for a sailboat to go from a starting point A to a destination B accounting explicitly for the random fluctuations of the wind, the fact that a sailboat cannot sail directly into the wind, and that changing heading in such a way that the wind changes from one side of the boat to the other takes time.

This project is a final project. It counts for 20 points. It *cannot* be used as one of the assignments whose grade is dropped. For this project you are expected to work alone. Furthermore, the TAs are instructed not to provide debugging help so you should not seek it.

2. The Model and Its Assumptions

We will model a lake by a rectangular lattice of points (x, y), $x, y = 0, 1, \ldots, N$. The south shore and the north shore correspond to the bottom and top edges of the lattice: $(x, 0)$ and (x, N), $x = 0, 1, \ldots, N$. Similarly, the west shore and the east shore correspond to the left and right edges of the lattice: $(0, y)$ and (N, y), $y = 0, 1, \ldots, N$. All other points of the lattice $x, y = 1, 2, \ldots, N-1$ will correspond to actual positions on the lake.

From a position (x, y) on the lake, we assume that the sailboat can only sail in one of eight directions:

- north to $(x, y+1)$,
- south to $(x, y-1)$,
- east to $(x+1, y)$,
- west to $(x-1, y)$,

- northeast to $(x+1, y+1)$,
- southeast to $(x+1, y-1)$,
- northwest to $(x-1, y+1)$, or
- southwest to $(x-1, y-1)$.

If the sailboat gets beached along one of the four bounding beaches, then it is stuck there forever (losing the race).

We also assume that the wind only blows from one of the eight directions listed above. We say that the sailboat is on a *starboard tack* if the wind is hitting the sailboat on the right side. If the wind is hitting the boat on the left, then the sailboat is said to be on a *port tack*. (If the boat is heading directly into the wind or directly away from the wind, then it is on neither a starboard nor a port tack.) For example, if the boat is heading northeast, then winds out of the east, southeast, and south correspond to starboard tacks, whereas winds out of the north, northwest, and west correspond to port tacks.

We assume that the wind is constant as the sailboat sails from one lattice point to the next, but that once the sailboat gets to a new lattice point the wind can change. The new wind is random, its distribution depending on the old wind according to a table of probabilities such as this:

⋱ NEW OLD ⋱	N	NE	E	SE	S	SW	W	NW
N	0.4	0.3						0.3
NE	0.4	0.3	0.3					
E		0.4	0.3	0.3				
SE			0.4	0.3	0.3			
S				0.4	0.2	0.4		
SW					0.3	0.3	0.4	
W						0.3	0.3	0.4
NW	0.4						0.3	0.3

Blank entries in the table have probability zero. (Note that this is an example of a sparse matrix, but for simplicity we'll store it just as a two-dimensional array.)

We assume further that the skipper sees the new wind direction (from the pattern of ripples it makes on the water) before the new wind actually hits the sails.

3. ENCODINGS

It will be convenient to encode the sailing headings and the wind directions according to the following scheme:

Value	Direction
0	N
1	NE
2	E
3	SE
4	S
5	SW
6	W
7	NW

If w denotes the wind direction and h denotes the current heading, then the sailboat is on a starboard tack if

$$\text{tack}(w,h) = \begin{cases} w-h & \text{if } w >= h, \\ w-h+8 & \text{otherwise} \end{cases}$$

takes the value 1, 2 or 3 and it is on a port tack if tack(w,h) is 5, 6 or 7. If the value is 0 or 4, then the boat is not on a tack.

4. Dynamic Programming

Time will be discretized so that the skipper only has to make decisions when arriving at a new lattice point. At this time, the skipper knows the current position (x,y), the current tack t (i.e., either port$=0$, starboard$=1$ or no tack$=2$) and the new wind direction w. Based on this information she must decide on a new heading. Let $v(x,y,t,w)$ denote the minimum expected amount of time to reach the destination from this point on. If the skipper (or is it skippess?) decides to head out from the current lattice point along heading h, then, given this decision, the minimum expected time to reach the destination will be

$$\text{delay}(t,t') + \text{time}(w,h) + \sum_{w' \in \{0,1,\ldots,7\}} P(w,w')v(x+\Delta x_h, y+\Delta y_h, t', w')$$

where

- t' is the new tack associated with the new wind w and the new heading h.
- delay(t,t') is the delay associated with a change in tack. delay(t,t') is zero if t and t' are the same or if either t or t' corresponds to a no-tack situation. Otherwise, delay(t,t') is a positive number `delay` (corresponding to a switch from a port tack to a starboard tack or a switch from a starboard tack to a port tack).
- time(w,h) is the time it takes to reach the next lattice point given that the sailboat is being sailed along heading h and the wind is coming from direction w. The time to travel one unit of distance (such as from (x,y) to $(x+1,y)$) will take on one of five values depending on tack(w,h):

tack(w,h)	Time
0	`into_wind_time`
1,7	`up_wind_time`
2,6	`cross_wind_time`
3,5	`down_wind_time`
4	`away_wind_time`

Values for these five times will be part of the input for the problem. Generally speaking, `into_the_wind_time` is infinite (since sailboats can't sail into the wind) and each subsequent time down the list is shorter than the one before. time(w,h) will be one of these five times if the heading h is either north, south, east or west. But if the heading h is along one of the diagonal directions, then the time must be multiplied by $\sqrt{2}$ to account for the somewhat longer distance that must be traveled.

- $P(w,w')$ is the probability that the next new wind direction will be w' given that the current new wind is w. These probabilities are taken from the table above.
- Δx_h is the change in the x coordinate given that the heading is h and Δy_h is the change in the y coordinate given that the heading is h.

Since the skipper wants to pick the new heading so as to minimize the expected time, we see that

$$v(x,y,t,w) = \min_{h\in\{0,1,\ldots,7\}} \left\{ \text{delay}(t,t') + \text{time}(w,h) + \sum_{w'\in\{0,1,\ldots,7\}} P(w,w')v(x+\Delta x_h, y+\Delta y_h, t', w') \right\}$$

The function $v(x,y,t,w)$ defined above is found by using an iterative scheme which we now describe. For all possible values of t and w, $v(x,y,t,w)$ is initialized to infinity at all points (x,y) except for the destination point B at which $v(x,y,t,w)$ is set to zero. (Of course, infinity is simply chosen to be a sufficiently large number.) Then, for a given x,y,t and w, $v(x,y,t,w)$ is "improved" by replacing its current value with the value computed as follows:

$$v(x,y,t,w) = \min_{h\in\{0,1,\ldots,7\}} \left\{ \text{delay}(t,t') + \text{time}(w,h) + \sum_{w'\in\{0,1,\ldots,7\}} P(w,w')v(x+\Delta x_h, y+\Delta y_h, t', w') \right\}$$

These improved values are computed for all x,y,t and w. The new values of $v(x,y,t,w)$ are still an overestimate of the true function. So this improvement procedure is continued several times until the change from one pass through all choices of x,y,t and w to the next produces only a very small maximum change in any of the estimates. At this point we declare that we have found the function $v(x,y,t,w)$. Given the function $v(x,y,t,w)$, it is easy to see what the optimal strategy should be. Namely, always choose the heading h that attains the minimum in the right-hand side of the above equation.

5. Computer Algorithm

The above dynamic programming equations suggest the following computer algorithm.

First, we initialize $v(x, y, t, w)$ to infinity everywhere except at the destination B where it is initialized to zero:

foreach `x` in `0,...,N` {
  foreach `y` in `0,...,N` {
    foreach `t` in `0,1,2` {
      foreach `w` in `0,1,...,7` {
        $v(x, y, t, w) = \infty$
      }
    }
  }
}
foreach `t` in `0,1,2` {
  foreach `w` in `0,1,...,7` {
    $v(x_B, y_B, t, w) = 0$
  }
}

Then we update the initial estimate at every point on the lake except at the destination B, keeping track of the largest change in the estimate in a variable called `error` and saving the best heading out of each situation x, y, t and w in bestheading(x, y, t, w). These iterations are to continue until `error` is less than some small tolerance, say `1.0e-3`.

do {
  error = 0
  foreach `x` in `1,...,N-1` {
    foreach `y` in `1,...,N-1` {
      if ($(x, y) \neq (x_B, y_B)$) {
        foreach `t` in `0,1,2` {
          foreach `w` in `0,1,...,7` {
            new_value $= \min_{h\in\{0,1,\ldots,7\}}\{$delay$(t, t') +$ time(w, h)
              $+ \sum_{w'\in\{0,1,\ldots,7\}} P(w, w')v(x + \Delta x_h, y + \Delta y_h, t', w')\}$
            if (|new_value $- v(x, y, t, w)| >$ error)
              error = |new_value $- v(x, y, t, w)|$
            bestheading$(x, y, t, w) = h$ attaining the min above
            $v(x, y, t, w) =$ new_value
          }
        }
      }
    }
  }
} while (error $>$ smalltolerance)

6. Data Structures

The functions $v(\cdot,\cdot,\cdot,\cdot)$ and bestheading$(\cdot,\cdot,\cdot,\cdot)$ should be represented in the computer as four dimensional arrays. For example, the declaration of v might be something like this:

```
double[][][][] v = new double[N+1][N+1][3][8];
```

Then, if x and y are integers between 0 and N, t is an integer equal to $0, 1$ or 2 and w is an integer between 0 and 7, then $v(x, y, t, w)$ is referred to as

```
v[x][y][t][w] = /* whatever */
```

7. Getting Started

As usual, you need to create a directory in which to work: Begin like this:

```
cd public_html/JAVA/ORF201
mkdir sail
cd sail
```

Then you need to copy over the following files:

```
cp /u/orf201/public_html/JAVA/ORF201/sail/index.html .
cp /u/orf201/public_html/JAVA/ORF201/sail/sail.html .
cp /u/orf201/public_html/JAVA/ORF201/sail/Sail.java .
```

Compile the java code that you just copied over:

```
javac Sail.java
```

Then use appletviewer to see what the code currently does:

```
appletviewer sail.html
```

An applet window should pop up with button labelled *Push here to go sailing*. Pressing that button brings up another window (the `sf` instance of a `SailFrame`) that allows you to define and then solve the sailing problem.

The `SailFrame` is divided into a left-hand part and a right-hand part. The left-hand part is an `InPanel` called `ip` that contains three buttons and a number of textfields into which the input parameters can be entered. The right-hand side is an instance `gp` of the class `GraphPad`. This is where the simulation is to be drawn.

All of the input user-inteface code is included in the file `Sail.java`. Your job is to code up the algorithm, the simulation, and the graphical output.

8. Programming Notes

Use less than 8 spaces for indentation (4 would be good) otherwise you'll find your code running off on the right.

You will find it convenient to use the absolute-value function. In JAVA, it is accessed via `Math.abs()`.

9. Input

The file `Sail.java` contains all of the code to set up text fields in which you can enter all of the relevant data.

10. Output

The functions $v(x, y, t, w)$ and bestheading(x, y, t, w) represent the "answers" that you will compute. If these functions had had only two arguments, it would be easy to display the results graphically. However, the fact that there are four arguments makes a graphical representation of them difficult. So, instead, we will simulate the random variations of the wind and display on a two dimensional grid the actual path of the sailboat for the given evolution of wind patterns. We will always assume that the boat is just arriving to point A on a starboard tack and that the wind is about to change, the new wind being out of the northeast. The skipper decides the new heading to exit point A based on bestheading$(x_A, y_A, 1, 1)$'s recommendation.

On reaching the next grid point, the wind is allowed to change randomly according to the given table of probabilities. Using the new wind direction, the skipper again refers to bestheading() to determine the direction to leave this grid point. This procedure is continued until the sailboat reaches its destination B.

You should show the path of the sailboat on the grid lake. For each leg of the path, you should indicate whether the boat is on a port or a starboard tack (or neither) by color coding the leg appropriately. You should also show the wind direction for each leg of the trip. After simulating one trip from A to B, you should display (somewhere on the lake) the time the trip took taking into account the time it takes to sail from each grid point along the path to the next grid point and also the delays incurred for each change of tack. In addition, you should display the expected time the trip should take as predicted by the value of $v(x_A, y_A, 1, 1)$.

11. HINTS ON SIMULATION

Given an old wind direction, say `old_w`, you will need to generate randomly a new wind direction, say `new_w`, using a two-dimensional array of probabilities, say `prob[][]`. The following code fragment accomplishes this task:

```
rannum  = Math.random();
cumprob = 0.0;

for (w=0; w<8; w++) {
   cumprob += prob[ old_w ][ w ];
   if (rannum < cumprob) { break; }
}
new_w = w;
```