Maximizing Screen Element Visibility

William Braga

Objective

Within a circular screen (graph) of radius r, n lines intersect each other. No two lines can be parallel, and each line must intersect all other lines within the screen boundary (i.e. r must be large enough to encompass all points of intersection). No more than two lines can intersect at the same point.
Each line l_i is defined by an angle of entry and an angle of exit from the circle relative to its center:

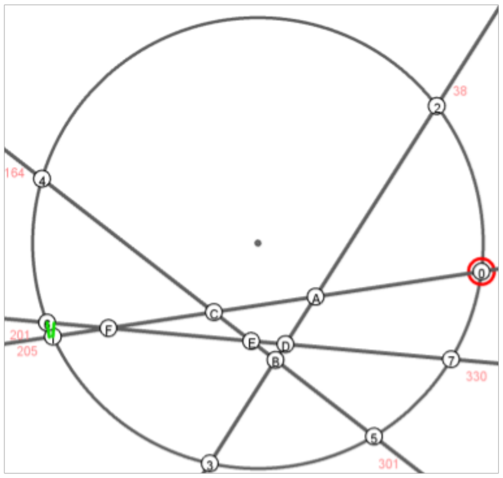

Figure 1: A valid arrangement of 4 lines
Face EDB is small and FCE is skinny

$l_i = (\theta_{2i}, \theta_{2i+1})\ s.t.\ i\ \epsilon\ [0, n-1],\ \theta_{2i}\ <\ \theta_{2i+1}$. The screen and line edges bound a set of interior faces, each denoted $f_j\ s.t.\ j\ \epsilon\ [0, \dfrac{n\,(n+1)}{2}]$.

Given n, return the arrangement of lines that produce the most optimal set of faces. The optimality of a face set is a function to be defined. One set is more optimal than another if its faces are more clearly visible within the screen. A face is not clearly visible when it is too ‘small’ or too ‘skinny’ (Fig 1).

Optimality

Our proposed function for face visibility is *thickness*. The *thickness* of a polygonal face is defined by the minimum perpendicular distance from each vertex to each of its nonadjacent edges for which that distance exists. For a face bounded by a circular arc, *thickness* is the smaller of the minimum perpendicular distance of each vertex to each of its straight, nonadjacent edges for which that distance exists and the distance from each interior vertex (i.e. vertices not on the circle boundary) to the circle's edge along the vector formed by the circle center and vertex if it projects onto the face's arc.

The optimality function for a face set is the *thickness* value for its least visible face, *min thickness*. An arrangement of lines is more optimal if its *min thickness* value is larger. Therefore, the objective is as follows:

$V(a, b)$is the vector between points a and b,$U(a, b)$is the unit vector between a and b
$c$is the point at the circle's center
e is a non-adjacent edge to a vertex v s.t. a perpendicular distance exists from v to e,
$e'$is a non-adjacent circular arc to an interior vertex v's.t. $U(c, v')$projects onto e':

$$argmax_{\theta}\{min_f\{min(min_{v,\, e=(p,q)}\{\,|\, U(p, q)\, x\, V(p, v)\,|\,\},\ min_{v',\, e'}\{\,|V(v', c + r * U(c, v'))|\,\})\}\}$$
$$= argmax_{\theta}\,\{\ min_f\,\{\ thickness(f)\ \}\ \}$$
$$= argmax_{\theta}\,\{\ minthickness(\theta)\ \}$$

Validity Theorem

Let θ' = $argsort(\theta)$. An arrangement of lines is valid iff $\forall i\ \theta'_i = \theta'_{i+n} + 1, i \epsilon [0, n]$ and no more than two lines intersect. In other words, all lines will intersect all other lines iff the order of the entry points is the same as the order of the exit points. A valid arrangement can be made from a random set of angles θ^R by sorting them, setting each line to $l_i = (\theta_i^R, \theta_{i+n}^R)$, and checking each intersection point is only between two lines.

Bruteforce Optimization

We developed an online algorithm that tests the *min thickness* for every combination of $\theta^t$where each angle θ_i^t is kept constant, increased by a small value ε, or decreased by ε at an iteration step t. All combinations are created using a recursive stack, and the $\theta^{t+1}$with the largest *min thickness* is returned each time. For a small enough ε, this algorithm will converge on the most optimal angle set for a given initial configuration. We have not yet been able to prove whether optimality is independent of the starting angles, but for dozens of trials with random starts, the same *min thickness* has been reached.

Odd Heuristic

We propose a heuristic to find the most optimal configuration in a graph with an odd number of lines, optimizing on one degree of freedom. Here, we define $\theta$to be sorted by angle value instead of entry and exit tuples for simplicity.

Let $\sigma^0 = 0$ and $\theta_i^0 = \frac{i*\pi}{n}$(i.e. equidistant points) at iteration $t = 0$. At each iteration, set $\sigma^t = \sigma^{t-1} + \varepsilon$ and $\theta_i^t = \frac{i*\pi}{n} + (-1)^i \frac{\sigma^t}{2}$. This heuristic brings each pair of nonoverlapping points closer together by εunits each iteration. When the *min thickness* is at a local maximum for a value of σ^t, the best configuration for that graph is returned as θ^t. This heuristic has not been proven, but it has matched the output of the bruteforce optimizer on several trials.

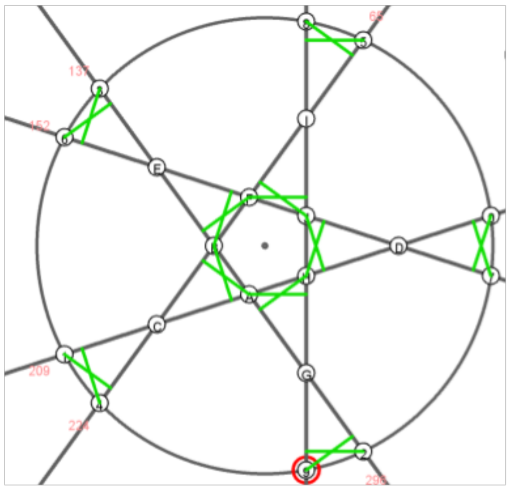
Figure 2: Heuristic Arrangement of 5 lines
σ = 0.37

Even Heuristic

We attempted to create a heuristic for graphs with an even number of lines, but it was proven incorrect by counterexample from the bruteforce optimizer.
Let $\sigma^0 = 0$ and $\theta_i^0 = \frac{i*\pi}{n}$(i.e. equidistant points) at iteration $t = 0$. At each iteration, set $\sigma^t = \sigma^{t-1} + \varepsilon$ and $\theta_i^t = \frac{i*\pi}{n} + (-1)^{\lfloor \frac{n}{2} \rfloor + i} \frac{\sigma^t}{2}$(i.e. for the second half of the points, instead of bringing pairs closer together, push them farther apart).

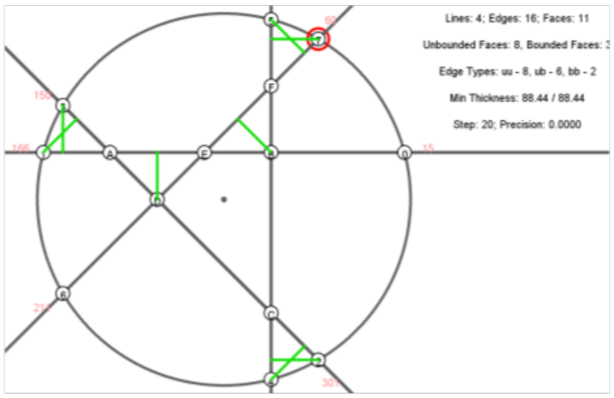
```
Lines: 4; Edges: 16; Faces: 11
Unbounded Faces: 8, Bounded Faces: 3
Edge Types: uu - 8, ub - 6, bb - 2
Min Thickness: 88.44 / 88.44
Step: 20; Precision: 0.0000
```

Figure 3: Even heuristic results (88.44)

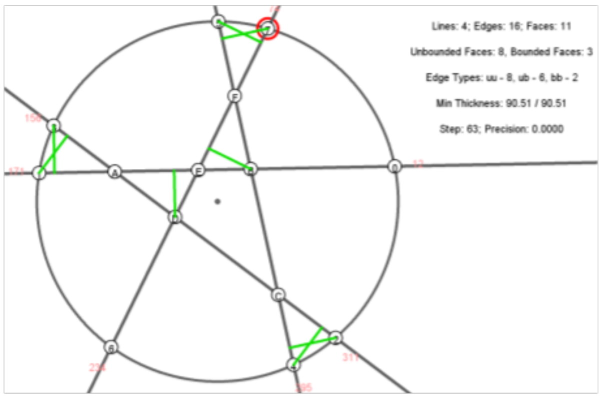
```
Lines: 4; Edges: 16; Faces: 11
Unbounded Faces: 8, Bounded Faces: 3
Edge Types: uu - 8, ub - 6, bb - 2
Min Thickness: 90.51 / 90.51
Step: 63; Precision: 0.0000
```

Figure 4: Bruteforce results (90.51)

Topology Theorems

The following formulas hold true for valid graphs. ‘UU,’ ‘UB,’ and ‘BB’ stand for edges between two unbounded faces, edges between one unbounded and one bounded face, and edges between two bounded faces. ‘U’ stands for unbounded faces and ‘B’ stands for bounded faces. An unbounded face is any face with a circular arc edge - it is not bounded within the screen.

Lines	Edges	Faces	UU	UB	BB	U	B
1	n^2	$\frac{n(n+1)}{2}+1$	n	0	0	$n+1$	0
2	n^2	$\frac{n(n+1)}{2}+1$	n	0	0	$2n$	0
3	n^2	$\frac{n(n+1)}{2}+1$	$2n$	0	0	$2n$	0
4	n^2	$\frac{n(n+1)}{2}+1$	$2n$	n	0	$2n$	$\frac{(n-1)(n-2)}{2}$
5	n^2	$\frac{n(n+1)}{2}+1$	$2n$	$n+2$	$n-2$	$2n$	$\frac{(n-1)(n-2)}{2}$
6	n^2	$\frac{n(n+1)}{2}+1$	$2n$	$2n$	$n(n-4)$	$2n$	$\frac{(n-1)(n-2)}{2}$
n	…	…	…	…	…	…	…

Next Steps

Future work would be useful in creating a faster optimizer, determining a correct even heuristic, and finding proofs for the heuristics.

The bruteforce optimizer currently recreates the full graph mesh on each test; however, a full recreation is only needed if the topology of the graph changes. Therefore, as long as the topology is constant, the only change that needs to be done is updating the coordinate points of intersections; this may potentially save significant time.