1 The Trackless Server Problem

A *trackless* algorithm $\mathcal{A}$ for the 2-server problem is now allowed to remember the name of any point. It knows the distance between its two servers, but it does not know the names of the points where its servers are located. If the servers are at s_1 and s_2, and if the new request is r, $\mathcal{A}$ receives only two numbers r_1 and r_2, where $r_i = ||s_i, r||$. The algorithm can then give one of two commands: "Move my first server to the request point," or "Move my second server to the request point."

Despite the fact that $\mathcal{A}$ knows no names of points, it can remember other information, such as what the distances were at any given past step, and which server it moved at that step. Similar to the normal server problem, that information gives rise to an object which we call the *trackless work function*.

Let M be the metric space. We can define an equivalence relation on M as follows: $x \approx y$ if $||s_1, x|| = ||s_1, y||$ and $||s_2, x|| = ||s_2, y||$, since $\mathcal{A}$ cannot distinguish x from y. Without loss of generality, s_2 is the last request point, *i.e.,* $\mathcal{A}$ served the last request with its second server. Then τ, the trackless work function, is defined on all pairs $\{s_2, x\}$, and satisfies the condition that $\tau(s_2, x) = \tau(s_2, y)$ if $x \approx y$.

We can thus represent the trackless work function as a function (which, by an abuse of notation, we also call τ) on $D = \left\{(u, v) \in \mathbb{R}^2 : u, v \geq 0, u + v \geq ||s_1, s_2||\right\}$, by using the the rule $\tau(s_2, x) = (||s_1, x||, ||s_2, x||)$

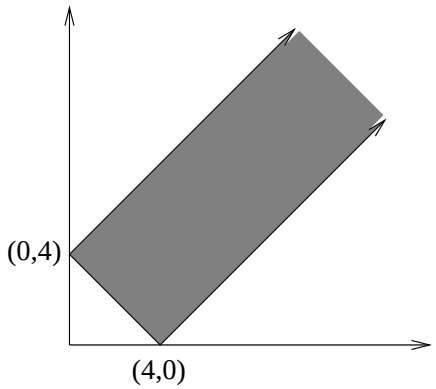


Figure 1: The Reigion D if $||s_1, s_2|| = 4$

The value of $\tau(s_2, x)$ is the minimum possible value that an optimal algorithm could have expended if its servers are at s_2 and x, as far as $\mathcal{A}$ can tell.

1.1 Updating the Trackless Work Function

\label{subsec: update twfa} computed by $\mathcal{A}$ using only the information it is allowed to know. Let τ^t be the trackless work function after t requests. Let r^t be the request at step t, and let (r_1^t, r_2^t) be the ordered pair of distances from r^t to the two servers. Let $D^t \subseteq \mathbb{R}^2$ be the domain of τ^t.

Updating the trackless work function takes two steps. Given τ^{t-1}, $\mathcal{A}$ first reads $((r_1^t, r_2^t)$, and then computes an updated function $\hat{\tau}^t$, defined on D^{t-1}. The value of $\hat{\tau}^t$ is the minimum cost of satisfying the first t requests, as computed by $\mathcal{A}$ with only the information available to $\mathcal{A}$ up to the point before it moves during step t.

The algorithm then decides which server to move. After that decision, we compute τ^t, which defined on D^t.

The following rules define the update calculations.

1. $\tau^0(u,v) = u$ for all $(u,v) \in D^0$.
2. If $t > 0$, then $\hat{\tau}^t(u,v) = \min \begin{cases} \tau^{t-1}(u,v) + r_2^t \\ \tau^{t-1}(r_1^t, r_2^t) + r_1^t + v \end{cases}$
3. If $t > 0$ and $(u,v) \in D^{t-1}$, let $\delta^t_{\min}(u,v) = \max\left\{\left|u - r_1^t\right|, \left|v - r_2^t\right|\right\}$, and let $\delta^t_{\max}(u,v) = \min\left\{u + r_1^t, v + r_2^t\right\}$.
   (a) If $\mathcal{A}$ serves request t with its first server, then
   $$\tau^t(u,v) = \min\left\{\hat{\tau}^t(\hat{u},\hat{v}) : \hat{u} = v \text{ and } \delta^t_{\min}(\hat{u},\hat{v}) \le v \le \delta^t_{\max}(\hat{u},\hat{v})\right\}$$
   (b) If $\mathcal{A}$ serves request t with its second server, then
   $$\tau^t(u,v) = \min\left\{\hat{\tau}^t(\hat{u},\hat{v}) : \hat{u} = u \text{ and } \delta^t_{\min}(\hat{u},\hat{v}) \le v \le \delta^t_{\max}(\hat{u},\hat{v})\right\}$$

1.2 The Trackless Work Function Algorithm

\label{subsec: twfa}

The *trackless work function algorithm* (TWFA) moves its first server if $r_1^t + \hat{\tau}(d^t, 0) < r_2^t + \hat{\tau}(0, d^t)$, its second server if $r_2^t + \hat{\tau}(0, d^t) < r_1^t + \hat{\tau}(d^t, 0)$. If the two quantities are equal, it moves one of its servers arbitrarily.

Theorem 1.1 \label{thm: lower bound cube} *Let H^3 be the hypercube of dimension 3 with the Hamming metric,* i.e.,*, the graph consisting of the 8 vertices and 12 edges of a cube, where distance is path length. The competitiveness of TWFA on H^3 no less than 3.*

Proof: The Figure shows a cycle consisting of 3 steps. □

Theorem 1.2 \label{thm: 3-competitive} *If all distances in M are integers, and none is larger than 6, TWFA is 3-competitive for the 2-server problem on M.*

Proof: We ran a computer program. □

By Theorems 1.2 and thm: lower bound cube, the competitiveness of TWFA on the cube is exactly 3.

Conjecture 1.3 *The competitiveness of TWFA is at most 3 for any metric space.*