Online learning and the Metrical Task System problem

Avrim Blum (Carnegie Mellon)
Carl Burch (Carnegie Mellon)

+ (if time) recent unpublished work joint with Carl Burch and Adam Kalai

+ Background

[1999 Dimacs Workshop on Online Decision Making]

Theme of this talk

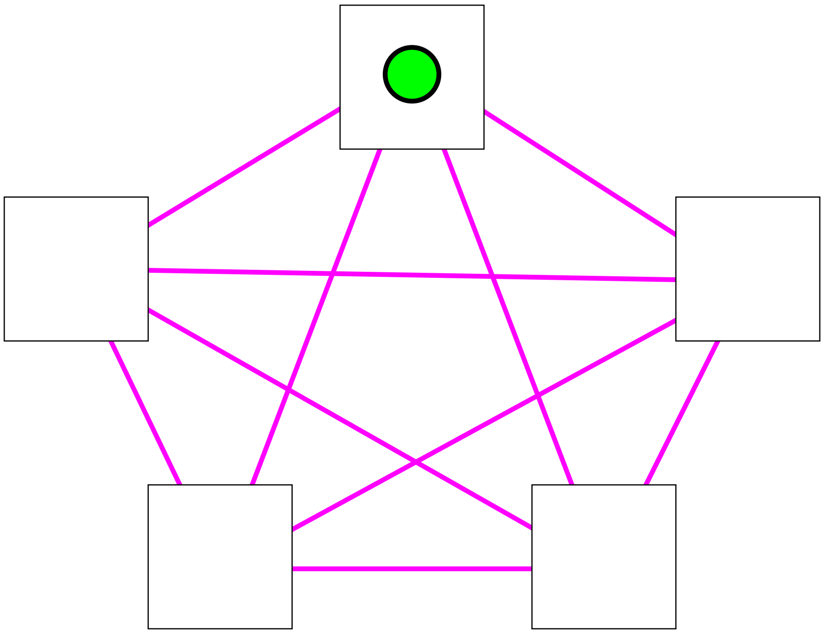
- "Online Algorithms" community has studied problems like:
  - disk paging
  - k-server problem
  - MTS problem
  - Adaptive data structures

- "Online learning" community has studied problems like:
  - combining expert advice
  - tracking the best expert

It turns out there are interesting relationships between these.

A motivating problem

Process Migration:



- Running a process in the background on some computer in a network.
- Amount of work done depends on speed of machine, whether there is a local user, load average, etc.
- So, equip process with ability to move itself. (Of course, moving wastes time too)

Goal: good strategy to decide when and where to move.

The Metrical Task System problem

[Borodin, Linial, Saks]

- Have a device that can be in one of n states.
- Can move from state i to state j at cost d_{ij}, where d is a distance metric.
- We are given, on-line, a sequence of tasks $\sigma = T_1, T_2, T_3, \ldots$. Each task has a cost vector, e.g., $T_1 = (3, 1, 2, 0)$, that gives the cost of processing the task in each of the n states of the device.

$\rightarrow$ Given a new task T, we can stay in the current state i, paying $T[i]$, or move to a new state j and perform it there, paying $d_{ij} + T[j]$.

$$\text{E.g.,}\quad \begin{array}{c} (1,0,0,0) \\ (1,1,0,0) \\ (0,0,1,0) \\ (0,1,0,1) \\ (1,1,0,1) \end{array}$$

for $d_{ij} = 1$ for all $i \neq j$.

Modeling process migration as MTS

- State = Computer
- d_{ij} = Time wasted to move from machine i to machine j.
- Cost vector = How much time would you have wasted had you been on machine i?

  E.g., (0.3, 0.5, 0, 0.9)
- Minimizing total cost = minimizing total time wasted.

Some mis-fits:

- Double-counting a little when move.
- 0-step vs 1-step lookahead.

For now on will assume: $d_{ij} = 1$ (uniform MTS).

Uniform MTS

- n states, $d_{ij} = 1$ for $i \neq j$.
- For any deterministic strategy, there is a sequence that causes it to pay cost $\geq n \cdot$ optimal.

1	0	0	0	0	0	0
0	1	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	0	0	0

Alg pays 1 per step. In hindsight could have paid 1 per n steps.

What about randomized (mixed) strategies? Assume *oblivious adversary*: knows our strategy but never sees outcome of our random choices.

Marking Algorithm [BLS],[FKLMSY]

1. Begin phase with all states unmarked.
2. When a state's cost reaches 1, mark it.
3. If current state gets marked, jump to random unmarked state. If none exist, end phase, jump to random state and go to Step 1.

0	0	0	1	0	1	0
0	0	0	0	0	1	1
1	0	0	0	0	0	1
1	1	1	0	0	0	0

→ Optimal cost in phase ≥ 1.

→ E[online cost] = O(log n).

Note: if each vector is a random [0 0 1 0 0 0 0] then *any* strategy has E[cost/task] $\geq 1/n$, but optimal in hindsight $\approx 1/(n \log n)$.

Combining "expert" advice

- Setting: each day we get a prediction from n experts about whether it is going to rain.
- Then, we use them to make our prediction.

E.g. $(n = 4)$:

CNN	Ch11	window	neighbor's dog	truth
N	N	Y	Y	Y
N	Y	Y	N	N
...	...	...	...	...

What's a good strategy for combining their opinions? Can we do nearly as well as best expert in hindsight?

Game-theoretic version

- No predictions, just losses (in $[0, 1]$).
- At each time step, alg chooses prob dist $\mathbf{p}$, gets loss vector ℓ, then pays expected cost $\mathbf{p} \cdot \ell$.

Differences compared to MTS:

- 0-step vs 1-step lookahead.
- Movement costs

For uniform metric:

- Losses $\ll$ distances $\Rightarrow$ MTS is harder.
- Losses $\gg$ distances $\Rightarrow$ Experts is harder.

If $d_{ij} = 1$, losses in $\{0, 1\}$, and alg is "reasonable" then these cancel out. E.g.,

$\mathbf{p}$	1/4	1/4	1/4	1/4
ℓ	1	1	0	0
$\mathbf{p}'$	1/8	1/8	1/4	1/2

Randomized WM alg [LW]

- Initialize weights w_i to 1.
- $p_i \propto w_i$.
- Update: $w_i \leftarrow w_i \cdot \beta^{\ell_i}$.

Guarantee: does nearly as well as best expert. As $\beta \to 1$, total loss satisfies

$$\mathbf{E}[\text{Alg loss}] \lesssim (1 + \varepsilon/2)[\text{best expert}] + \frac{1}{\epsilon}\log n.$$

In uniform MTS setting,

$$\mathbf{E}[\text{Alg cost}] \lesssim (1 + 3\varepsilon/2)[\text{best state}] + \frac{1}{\epsilon}\log n.$$

In process migration scenario, doing nearly as well as best single machine in hindsight.

Which is better: Marking or RWM?

Bad case for Marking:

$$\begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ . & . & . & \end{matrix}$$

Bad case for RWM:

$$\begin{matrix} 1 & 1 & 0 & 0 \\ . & . & . & \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ . & . & . & \\ 0 & 0 & 1 & 1 \end{matrix}$$

Experts-tracking gives nice "best of both". But first, a digression....

Combining online algorithms online

Disk-paging problem:

- Cache holds k memory pages. Disk has $\gg k$.
- *Page fault* if item requested is not in cache. Bring page in and throw something else out.
- Many different online algorithms: LRU, FIFO, Marking, etc.

View online algs as "experts" and do nearly as well as best in hindsight?

- Experts have state. Loss is f(state).
- To switch from following expert X to Y need (at least lazily) to switch state too.

Can view as changing metric space, but always diameter $\leq k$. Apply RWM to get:

$$\mathbf{E}[\text{cost}] \lesssim (1 + 3\varepsilon/2)[\text{best alg}] + \frac{1}{\epsilon} k \log n.$$

List Update Problem

- Have ℓ items in a linked list: (3 2 6 5 1)
- Get a request: "5"
- Pay based on how far down. Can also swap neighbors for cost of \$1 each.

Move-to-Front gets factor of 2 w.r.t. best strategy in hindsight. What if just want to compare to best *static* list?

→ MTS on $\ell!$ states.

→ Not uniform metric, but diameter $\leq \ell^2$.

RWM:

$$\mathbf{E}[\text{cost}] \lesssim (1 + 3\varepsilon/2)[\text{best list}] + \frac{1}{\epsilon}\ell^2 \log(\ell!).$$

Of course, computation time is horrendous...

Best of both worlds

- [Herbster & Warmuth] Weight-sharing version of RWM. [BB] analyze in this setting.
- In "experts" setting, for all k,

$$\mathbf{E}[\text{Alg loss}] \lesssim (1+\varepsilon)L_k + (\frac{1}{\epsilon}\log n)(k+1)$$

where L_k is loss of best strategy in hindsight that switches k times.

- In uniform MTS setting, can view as:

$$\mathbf{E}[\text{Alg cost}] \leq$$
$$O(1) \times \text{OPT}_{local} + O(\log n) \times \text{OPT}_{move}.$$

Simpler alg: Phased-RWM

- Run RWM until every expert has incurred total loss $\geq r$.
- Then, reset weights to be all equal again.

Per phase,

$$\mathbf{E}[\text{Alg loss}] \lesssim (1 + \varepsilon/2)r + \frac{1}{\epsilon}\log n.$$

If choose $r = \frac{1}{\alpha\varepsilon}\log n$, then in "experts" setting,

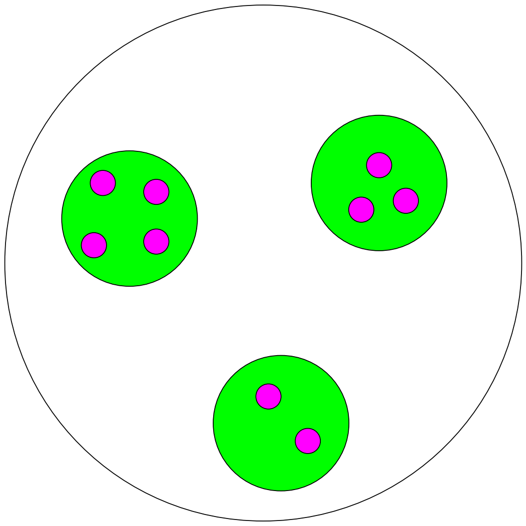
$$\mathbf{E}[\text{Alg loss}] \lesssim (1 + \alpha + \varepsilon/2)L_k + (\frac{1}{\alpha\epsilon}\log n)(k+1)$$

where L_k is loss of best strategy in hindsight that switches k times.

Not quite as good as Share bounds but simpler.

BBBT alg for general MTS problem

- For general metric space, approximate with hierarchy.



- Recursively, have ratio r in each space.
- At top level, view as "unfair MTS" problem.

  Given task vector T, alg in state i pays $T[i]$ but offline pays only $T[i]/r$.

  - Marking gets ratio $O(r \log n)$.
  - randWF gets ratio $r + O(\log n)$.

r-unfair $\Leftrightarrow$ shifting bounds.

(OK to be $O(\log n)$ competitive wrt movement cost but need only $(1+\varepsilon)$ wrt local cost)

Work-function based algorithms

- $WF[i] = OPT[i] =$ Total cost incurred by best strategy in hindsight *ending* at state i.
- Given task T, update $OPT[i]$ to:

$$\min \left[OPT[i] + T[i], \min_j \left[OPT[j] + T[j] + d_{ij}\right]\right]$$

- Work-function alg: go to state j minimizing above expression (or stay at i if that term was smallest).
- Randomized-WF alg:

$$p_i = \frac{1}{n} + \frac{1}{n}\sum_j \left[\frac{OPT[j] - OPT[i]}{r}\right]^t$$

(t is odd $\approx \log n$)

(ignore losses at states of prob 0)

Empirical results on process migration

Sampled load averages from 112 machines every 5 min for 6.5 days. Took top 32 machines as the states/experts.

Chart for $\{0, 1\}$ losses, $d_{ij} = 1$:

algorithm	parameter setting	cost ratio
Uniform		33.4
Least-Used		23.3
Greedy		17.8
Recent	$k = 5$	7.6
	$k = 29$	3.4
Marking	$r = 1.0$	3.9
	$r = 0.30$	3.3
WF		2.6
Rand-WF		2.8
WM-Share	$\beta = 0.5, \alpha = 0.01$	4.0
	$\beta = 2 \times 10^{-6}, \alpha = 10^{-10}$	2.9

Process migration demo

Open questions

- Can experts-style algorithms be used for simpler algorithm for general metric spaces (without using HST approximation)?

  How about for the case of n equally-spaced points on the line: can you achieve $O(\log n)$ competitive ratio?

- Apply to other online problems?