Online Load Balancing and Machine Scheduling

– CT Network Theory Seminar –

Wenkai Dai

Communication Technologies Group
Computer Science Department
University of Vienna

November 9, 2020

Outlines

1. Introduction of Online Balancing
2. Results of Greedy Algorithm
3. Constant Competitive Ratio for Related Machines
4. Competitive Ratio $O(\log n)$ for Unrelated Machines

Online Problems

- **Input:** *A request sequence* $\sigma = \sigma(1), \sigma(2), \ldots,;$
- **Online:** *An algorithm M* has to serve each request *online* without knowledge of future requests. Each request must be served before serving the next requests.
- **Objective:** The goal is to serve the entire request sequence so that the objective function is minimized.

The Online Load-Balancing Problem

Input:

- The n related/unrelated machines
- *A sequence of jobs* $\sigma = (\vec{p}(1), \vec{p}(2), \ldots, \vec{p}(k))$
- Each job $j \in \sigma$ denoted by its *"load vector"*:

$$\vec{p}(j) = (p_1(j), p_2(j), \ldots, p_n(j)), \text{ where } p_i(j) \geq 0.$$

Definition of Load:

- Assign a job j to a machine i increases the load on the machine i by $p_i(j)$
- $\ell_i(j)$ denotes the load on machine i after already assigning jobs 1 through j:

$$\ell_k(j) = \begin{cases} \ell_k(j-1) + p_k(j), & \text{if } k = i \\ \ell_k(j-1), & \text{otherwise} \end{cases}$$

The Online Load-Balancing Problem (2)

Notations of Offline Algorithm:

- Given *a sequence of jobs* $\sigma = (\vec{p}(1), \vec{p}(2), \ldots, \vec{p}(k))$
- $\mathscr{A}^*$ denotes off-line (optimal) algorithm
- $\ell_i^*(j)$ denotes the load on machine i achieved by $\mathscr{A}^*$ after assigning jobs 1 to j in σ

Objective:

- $L^*(k) = \max_i \ell_i^*(k)$: the maximum load (optimal) by $\mathscr{A}^*$
- $L(k) = \max_i \ell_i(k)$: the maximum load by online algorithms
- Online algorithms try to minimize $L(k)$
- Objective: minimize the competitive ratio:

  $$\sup\{L(k)/L^*(k)\} \text{ for any length } k \text{ and all possible sequences } \sigma$$

- k may be omitted for $L^*(k)$ and $L(k)$

Three Machine Models

Each job $j \in \sigma$ denoted by its *"load vector"*:

$$\vec{p}(j) = (p_1(j), p_2(j), \ldots, p_n(j))\,, \text{ where } p_i(j) \geq 0.$$

- Identical machimes: each job j: $p_i(j) = p_{i'}(j)$ for any two machines: i and i'
- Related machimes: for any jobs j and j' and machines i and i':

$$\frac{p_i(j)}{p_{i'}(j)} = \frac{p_i(j')}{p_{i'}(j')} = \frac{v_{i'}}{v_i}, \text{ where } v_i : \text{speed of machine } i$$

- Identical is a special case of related machines:
- Unrelated machimes $(1/\infty)$: all other cases
  - e.g., $p_i(j) = +\infty$, i.e., a job j not runnable on some machines i

Greedy Algorithm for Load Balancing Problem

Greedy Algorithm:

- Algorithm: for each job j, assign it to a machine k s.t., $k = \text{argmin}\{\ell_i(k-1) + p_i(j)\}$

Identical Machines:

- competitive ratio: $2 - \epsilon$

Related Machines:

- Competitive ratio: $O(\log n)$, lower bound: $\Omega(\log n)$

Unrelated Machines:

- Competitive ratio: $O(n)$, lower bound: $\Omega(n)$

Competitive Ratio of Greedy for Unrelated Machines

Theorem 2.1

The competitive ratio of the greedy algorithm is at most n for unrelated machines.

Proof.

- Each job j has a minimum load $\min_i p_i(j)$
- $\mathscr{S}_i^*$: jobs assigned to machine i by offline algorithms

$$nL^* \geq \sum_i \ell_i^* = \sum_i \sum_{j \in \mathscr{S}_i^*} p_i(j) \geq \sum_j \min_i p_i(j)$$

- Claim: the maximum load by greedy $\leq \sum_j \min_i p_i(j)$
- Assume: after assigning $j-1$, $L(j-1) \leq \sum_{j' \leq j-1} \min_i p_i(j')$
- When j arrives, greedy chooses machine m, the resulting load at most $L(j-1) + p_m(j) \leq \sum_{j' \leq j} \min_i p_i(j')$
- $L(j) \leq \sum_{j' \leq j} \min_i p_i(j')$ implies $L \leq \sum_j \min_i p_i(j') \leq nL^*$ $\square$

Algorithm Assign-R for Related Machines

procedure ASSIGN-R($\vec{p}, \vec{\ell}, \Lambda$);
  /* Λ — current estimate of L^*.
  Let $S := \{i | \ell_i + p_i \leq 2\Lambda\}$;
  if $S = \emptyset$
    then b :=fail
    else begin
      $k := \min\{i | i \in S\}$;
      $\ell_k := \ell_k + p_k$;
      b :=success
    end;
  return($\vec{\ell}, b$).
end.

FIG. 3. Algorithm ASSIGN-R.

Algorithm Assign-R for Related Machines

Theorem 3.1

If $\lambda^* \leq \Lambda$, then Algorithm ASSIGN-R *never fails. Thus, the load on a machine never exceeds* 2Λ

Proof.

- By contradiction, assumes it fails first on a job j.
- r: the fastest machine whose load $\leq L^*$, if no $r = 0$

$$r = \max\{i \mid \ell_i\,(j-1) \leq L^*\}$$

- Clearly, $r \neq n$, otherwise j is on n $(\ell_n(j-1) + p_n(j) \leq 2L^*)$
- Define $\Gamma = \{i \mid i > r\}$: overloaded machines faster than r
- Clearly, $\Gamma \neq \emptyset$ since $r < n$
- $\mathscr{S}_i/\mathscr{S}_i^*$: jobs assigned to machine i by online/offline algo.

Algorithm Assign-R for Related Machines

Proof (Cont.)

- Now, to show: it exists a job $s \in \bigcup_{i\in\Gamma} \mathscr{S}_i$ but $s \notin \bigcup_{i\in\Gamma} \mathscr{S}_i^*$

$$\sum_{i\in\Gamma, s\in\mathscr{S}_i} p_n(s) = \sum_{i\in\Gamma s, \in\mathscr{S}_i} \frac{p_n(s)}{p_i(s)} p_i(s)$$
$$= \sum_{i\in\Gamma} \frac{v_i}{v_n} \sum_{s\in\mathscr{S}_i} p_i(s) > \sum_{i\in\Gamma} \frac{v_i}{v_n} L^*$$
$$\geq \sum_{i\in\Gamma} \frac{v_i}{v_n} \sum_{s\in\mathscr{S}_i^*} p_i(s) = \sum_{i\in\Gamma, s\in\mathscr{S}_i^*} p_n(s)$$

- We know a job s assigned to $i \in \Gamma$ by online but to $i' \notin \Gamma$ by offline
- Our assumption: $p_{i'}(s) \leq L^* \leq \Lambda$
- Since $r \geq i'$, it implies r at least as fast as i'

Algorithm Assign-R for Related Machines

Theorem 3.1

If $\lambda^* \leq \Lambda$, then Algorithm ASSIGN-R *never fails. Thus, the load on a machine never exceeds* 2Λ

Proof (Cont.)

- We know a job s assigned to $i \in \Gamma$ by online but to $i' \notin \Gamma$ by offline
- Our assumption: $p_{i'}(s) \leq L^* \leq \Lambda$
- Since $r \geq i'$, it implies r at least as fast as i'
- Since job s assigned before j then $\ell_r(s-1) \leq \ell_r(j-1) \leq L^*$
- But, this means that online algo should have placed job s on r or a slower machine instead of i, which is a contradiction. □

Doubling Approach

Theorem 3.2

For any load balancing problem, let ALG_Λ be a parameterized online algorithm satisfying $\mathrm{OPT}(\sigma) \leq \Lambda \implies \mathrm{ALG}_\Lambda(\sigma) \leq c \cdot \Lambda$ (e.g., $\textsc{Assign-R}$ with $c = 2$ for related machines). Then there is an algorithm ALG s.t., for all σ, $\mathrm{ALG}(\sigma) \leq 4c \cdot \mathrm{OPT}(\sigma)$.

- ALG executes in stages, each stage correspond to the mos recent estimate of Λ.
- Stage 0, $\Lambda_0 = \mathrm{OPT}(j = 0)$, easy to compute the optimal for the first job.
- Each stage j, ALG uses ALG_Λ to assign jobs until it fails and start a new stage by doubling Λ
- Stage k, $\Lambda = 2^k \Lambda_0$

Proof of Doubling Approach

Proof.

- To prove $\mathrm{ALG}(\sigma) \leq 4c \cdot \mathrm{OPT}(\sigma)$ for any sequence σ
- Suppose ALG terminates at the stage h.
- If $h = 0$, it is clear $\mathrm{ALG}(\sigma) \leq c \cdot \mathrm{OPT}(\sigma)$
- Let r be the first job for the stage h, and σ_j denotes the subsequence processed in stage j
- Clearly, stage $h-1$ failed on $\sigma_{h-1}r$, while ALG_Λ has $\Lambda = 2^{h-1}\Lambda_0$
- Thus, $\mathrm{OPT}(\sigma) \geq \mathrm{OPT}(\sigma_{h-1}r) > 2^{h-1}\Lambda_0$
- Moreover
$$\mathrm{ALG}(\sigma) = \sum_{j=0}^{h} \mathrm{ALG}(\sigma_j) \leq \sum_{j=0}^{h} c \cdot 2^j\Lambda_0 = c\left(2^{h+1} - 1\right)\Lambda_0$$

$\square$

Competitive Ratio for Assign-R for Related Machines

Corollary 3.3

Assign-R *Algorithm can be modified to achieve a competitive ratio of* 8.

Algorithm Assign-U for Unelated Machines

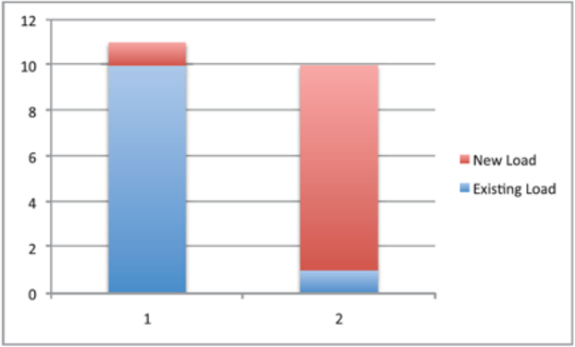
- Normalization: $\tilde{\ell}_i = \ell_i/\Lambda \ \tilde{p}_i = p_i/\Lambda$
- Find a machine s minimizing $\Delta_i = \alpha^{(\tilde{\ell}_i(j-1)+\tilde{p}_i(j))} - \alpha^{\tilde{\ell}_i(j-1)}$

procedure ASSIGN-U($\vec{p}, \vec{\ell}, \Lambda, \beta$);
  /* Λ — current estimate of L^*.
  /* β — designed performance guarantee of the algorithm.
  Let s be the index minimizing $\Delta_i = a^{(\bar{\ell}_i+\bar{p}_i)} - a^{\bar{\ell}_i}$;
  if $\ell_s + p_s > \beta\Lambda$
    then b :=fail
    else begin
      $\ell_s := \ell_s + p_s$;
      b :=success
    end;
  return($\vec{\ell}, b$).
end.

FIG. 2. Algorithm ASSIGN-U.

Intuitive Understanding of Assign-U Algorithm

- Assign-U minimizing $\alpha^{\tilde{\ell}_i(j-1)}\left(\alpha^{\tilde{p}_i(j)} - 1\right)$
- Greedy minimizing $\left(\tilde{\ell}_i\,(j-1) + \tilde{p}_i\,(j)\right)$
- Greedy choose machine 2
- But Assign-U chooses machine 1 since $1.5^{10+1} - 1.5^{10} \approx 22.83 < 51.16 \approx 1.5^{10} - 1.5^{1}$ $(\alpha = 1.5)$



Algorithm Assign-R for Related Machines

Theorem 4.1

If $\lambda^* \leq \Lambda$, then there exits $\beta = O(\log n)$ s.t., Algorithm $\textsc{Assign-U}$ *never fails. Thus, the load on a machine never exceeds $\beta\Lambda$*

Proof.

- State after $j-1$ jobs, $\alpha, \gamma > 1$ ($\alpha \approx 2, \gamma \approx 1$)
- assumption: $L^*(j) \leq L^* \leq \Lambda$
- potential function:

$$\Phi(j) = \sum_{i=1}^{n} \alpha^{\tilde{\ell}_i(j)} \left(\gamma - \tilde{\ell}_i^*(j)\right)$$

Algorithm Assign-R for Related Machines

Proof (Cont.)

- Let the job j assigned to machine i' (i) by online (offline)

$$\Phi(j) = \sum_{i=1}^{n} \alpha^{\tilde{\ell}_i(j)} \left(\gamma - \tilde{\ell}_i^*(j)\right)$$

- Compute $\Phi(j) - \Phi(j-1)$ as follows:

$$\begin{aligned}
&= \left(\gamma - \tilde{\ell}_{i'}^*(j-1)\right)\left(\alpha^{\tilde{\ell}_{i'}^*(j)} - \alpha^{\tilde{\ell}_{i'}^*(j-1)}\right) - \alpha^{\tilde{\ell}_i(j)}\tilde{p}_i(j) \\
&\le \gamma\left(\alpha^{\tilde{\ell}_{i'}^*(j-1)+\tilde{p}_{i'}(j)} - \alpha^{\tilde{\ell}_i(j-1)}\right) - \alpha^{\tilde{\ell}_i(j-1)}\tilde{p}_i(j) \\
&\le \gamma\left(\alpha^{\tilde{\ell}_i^*(j-1)+\tilde{p}_i(j)} - \alpha^{\tilde{\ell}_i(j-1)}\right) - \alpha^{\tilde{\ell}_i(j-1)}\tilde{p}_i(j) \\
&= \alpha^{\tilde{\ell}_i(j-1)}\left(\gamma\left(\alpha^{\tilde{p}_i(j)} - 1\right) - \tilde{p}_i(j)\right).
\end{aligned}$$

Algorithm Assign-R for Related Machines

Proof (Cont.)

- $\Phi(j) - \Phi(j-1) \leq \alpha^{\tilde{\ell}_i(j-1)} \left(\gamma \left(\alpha^{\tilde{p}_i(j)} - 1\right) - \tilde{p}_i(j)\right)$
- As offline assigns job j to machine i, $0 \leq \tilde{p}_i(j) \leq L^*/\Lambda \leq 1$
- To prove $\Phi(j) - \Phi(j-1) \leq 0$, we need to show $\gamma(\alpha^x - 1) \leq x$ for $x \in [0,1]$. It is true for $\alpha = 1 + 1/\gamma$
- Clearly, $\Phi(0) = \gamma n$
- Recall $\Phi(j) = \sum_{i=1}^{n} \alpha^{\tilde{\ell}_i(j)} \left(\gamma - \tilde{\ell}_i^*(j)\right)$
- Thus, $\sum_{i=1}^{n} \alpha^{\tilde{\ell}_i(j)} (\gamma - 1) \leq \gamma n$

$$\begin{aligned} L = \max_i \ell_i(k) &= \Lambda \max \tilde{\ell}_i(k) \\ &\leq \Lambda \cdot \log_a \left(\frac{\gamma}{\gamma - 1} n\right) = O(\Lambda \log n) \qquad \square \end{aligned}$$

Questions & Answers

Thanks for your Listening, and welcome your questions!