Fairness - An Optimization Problem

Final Project
CAS 781

April 27, 2017

Michael Liut
1132938
McMaster University

Prepared for:
Dr. Douglas Down
Issues in Data Centre Design
Winter 2017

A Brief Introduction

This report investigates the topic of fairness; which can be fundamentally categorized as either *temporal fairness* or *proportional fairness*. Temporal fairness is when a job's seniority is a factor used in determining the order of which jobs are processed.

e.g. given two jobs J_1 and J_2, if J_1 arrives prior to J_2, then J_1 is deemed to have seniority over J_2, thus, J_1 should be completed before J_2.

Proportional fairness articulates the concept of a job's response time being proportional to its job's size.

e.g. given two jobs J_1 and J_2, where J_1 is a large job and J_2 is a small job, if J_2 arrives for processing shortly after J_1 it may be fair for the smaller job to complete before the larger. Despite J_1 arriving first, the larger job will take more time to serve, therefore, it is only fair that it waits in queue for a 'proportional' amount of time.

Prominent metrics utilized in assessing fairness are measures of slowdown. Slowdown is defined as being R/S, where R is the response time of a job and S is the job's size. The maximum slowdown determines the worst-case scenario, while more often than not the best-case scenario would have the minimum slowdown.

Hereinafter, when we speak of proportional fairness we utilize *Slowdown Variance* (SDV).

$$SDV = Var(\frac{R}{S}) = Avg((\frac{R}{S})^2) - Avg(\frac{R}{S})^2, \text{ where}$$

$$Avg((\frac{R_i}{S_i})^2) = ((\sum_{i=1}^{N}(\frac{R_i}{S_i})^2)/N) \quad \text{and} \quad Avg(\frac{R_i}{S_i})^2 = ((\sum_{i=1}^{N}\frac{R_i}{S_i})/N)^2.$$

N represents the total number of jobs in the system at a specific time.

In general, we infer that a truly fair policy would guarantee that the ratio between a job's response time and size remains constant. Consequentially, the closer to zero SDV approaches the more 'fair' the policy is deemed to be.

Definition 1. Fair: A system is said to be perfectly fair if, and only if, for all jobs J_i and J_k, $R_{J_i}/S_{J_i} = R_{J_k}/S_{J_k}$. In other words, a system is perfectly fair if the slowdown is constant.

It should be noted that a system of jobs can be perfectly fair while having poor performance.

e.g. a system which never processes jobs.

As a result of the stochastic nature of these systems, a perfectly fair and stable policy does not exist. However, by employing fairness, as defined above, we aim to strive towards this.

This report investigates potential avenues to minimize the SDV by modeling it in terms of an optimization problem.

The Optimization Problem

Given all jobs at a time t, where only one job can be processed at a time, we are looking to *minimize* SDV. Recall, $SDV = Var(\frac{R}{S})$, where R_i is the response time of the i^{th} largest job and response time of the i^{th} job to depart, and S_i is the size of the i^{th} largest job. N denotes the total number of jobs in the system and S_i is C_i's respective job size, which is an input. From this, intuitively, it can be seen that every smaller job would depart before the largest job in our system.

Wherefrom the following optimization problem can be modeled:

$$\begin{aligned}
\min \quad & ((\sum_{i=1}^{N}(\tfrac{R_i}{S_i})^2)/N) \quad - \quad ((\sum_{i=1}^{N}\tfrac{R_i}{S_i})/N)^2 \\
\text{s.t.} \quad & R_N = \sum_{j=1}^{N} S_j \\
& R_i \geq \sum_{j=1}^{i} S_j \qquad i = 1, \ldots, N-1 \\
& R_i \leq R_{i+1} \qquad i = 1, \ldots, N-1 \\
& R_i \geq 0 \qquad i = 1, \ldots, N
\end{aligned}$$

One key element worth mentioning, $R_N = \sum_{j=1}^{N} S_j$ ensures that the server does not idle. This is extremely important as a server which can idle may never finish processing a job. Furthermore, the size S of a job is an input (assumed knowledge when a job enters the system).

Taking this optimization model and converting it into Matlab code, utilizing YALMIP, looks like the following:

```
function [r,Obj]= hw1(S)
%S is the input,(S_1,S_2,S_3,\dots, S_n)
n = length(S);
%in case it is not in column format
if size(S,1)<=1
    S = S';
end
R = sdpvar(n,1);
F=[ R(n) == sum(S),R>=eps];
for i=1:n-1
    F = F + [ R(i) >=sum(S(1:i)) ];
    F = F + [R(i)<=R(i+1) ];
end
obj = 0 ;
for i=1:n
    obj = obj + (R(i)/S(i))^2;
end
obj = obj/n - (sum(R./S)/n)^2;
rsl = solvesdp(F,obj);
r=double(R);
Obj=double(obj);
end
```

Figure 1: Matlab Code — using YALMIP

In general, programs must be robust. A program is only as good as the validity of its results. Therefore, the following is a proposed, verifiable, solution to the optimization problem:

$$\begin{aligned}
R_N &= \sum_{i=1}^{N} S_i \\
R_i &= \tfrac{R_N}{S_N} \cdot S_i \qquad i = 1, \ldots, N-1
\end{aligned}$$

Example 1

Let's look at a simple example with two customers C_1 and C_2, where C_1's job has an unknown response time (denoted as x) and a size of 1, and C_2's job has a response time of 3 and a size of 2. Both customer's jobs are assumed to arrive at the same time.

If we are attempting to minimize SDV, then an $x = \frac{3}{2}$ would accomplish this.

Proof

Given:

C_1: $R_1 = x$ and $S_1 = 1$.
C_2: $R_2 = 3$ and $S_2 = 2$.

Solve:

$$min \ ((\sum_{i=1}^{N}(\frac{R_i}{S_i})^2)/N) - ((\sum_{i=1}^{N}\frac{R_i}{S_i})/N)^2$$

$$\frac{(\frac{x}{1} - \frac{2x+3}{4})^2}{2} + \frac{(\frac{3}{2} - \frac{2x+3}{4})^2}{2}$$

$$= (\frac{2x-3}{4})^2 + (\frac{3-2x}{4})^2$$

$$\frac{x}{1} \cdot \frac{1}{2} + \frac{3}{2} \cdot \frac{1}{2}$$

$$= \frac{2x+3}{4}$$

$$= \frac{3}{2}, \text{ negative solution ignored.}$$

$\therefore$ the optimal x-value is $\frac{3}{2}$ as our minimized $SDV = 0$, resulting in a perfectly fair system.

Example 2

Let's look at another example with three customers C_1, C_2, and C_3. In this case we will be deciding on what the optimal R-values are based on the inputted S-values: $S_1 = 1$, $S_2 = 5$, and $S_3 = 10$.

Utilizing the Matlab code, from Figure 1, we are able to determine that the objective function $SDV = 0$ and the decision variables $R_1 = 1.60$, $R_2 = 8.00$, and $R_3 = 16.00$.

Given the results, it is easy to see that the ratio's between all jobs are equal (i.e. $\frac{R_1}{S_1} = \frac{1.60}{1}$ which is equivalent to $\frac{R_2}{S_2} = \frac{8.00}{5}$ which is also equivalent to $\frac{R_3}{S_3} = \frac{16.00}{10}$). Thus, resulting in a $SDV = 0$ and a perfectly fair system; as defined in **Definition 1**.

This can also be verified utilizing the solution shown at the bottom of Page 2. The results of which are as follows:

$$R_3 = \sum_{i=1}^{3} S_i = 16$$

$$R_1 = \frac{R_3}{S_3} \cdot S_1 = 1.60$$

$$R_2 = \frac{R_3}{S_3} \cdot S_2 = 8.00$$

$$R_3 = \frac{R_3}{S_3} \cdot S_3 = 16.00$$

$\therefore$ as the Matlab output of the modeled optimization solution and the manual solution check are equivalent, it is safe to assume that these response times truly exhibit a policy to effectively schedule the jobs to make the system fair.

Example 3

Let's look at a more complex example with ten customers $C_1, \ldots, C_{10}$. Again, like in the previous example, we will be deciding on what the optimal R-values are based on the inputted S-values: $S_1 = 1$, $S_2 = 3$, $S_3 = 5$, $S_4 = 10$, $S_5 = 25$, $S_6 = 50$, $S_7 = 100$, $S_8 = 200$, $S_9 = 250$, and $S_{10} = 500$.

Utilizing the Matlab code, from Figure 1, we are able to determine that the objective function $SDV = 0.004147$ and the decision variables $R_1, \ldots, R_{10} = 2.432014, 7.296043, 12.160075, 24.320156, 60.800370, 121.600791, 243.201756, 486.406923, 644.000000, 1144.000000$.

Once again, some pretty simple analysis can be seen by visually comparing a few of the results. Let's examine the ratios of C_1, C_5, and C_9. C_1's ratio is $\frac{2.432014}{1} = 2.432014$, C_5's ratio is $\frac{60.800370}{25} = 2.432014$, and C_9's ratio is $\frac{644.0}{250} = 2.576$. Ergo, resulting in a $SDV = 0.004147$ which is as close to a perfectly fair system as we can get.

What may stick out immediately is that C_9's ratio is $\frac{644.0}{250} = 2.576$. This is off by approximately 0.1, whereas the C_1 and C_5 were found to have the same ratio (i.e. approximately 2.4). As there is both forced rounding in the Matlab code and from the lack of precision of the CPU where the code was run on, it was pertinent to verify this with the solution shown at the bottom of Page 2. The results of which verifies the output and ratio of all results, however, we are interested more so in C_9 which gave us a ratio of $\frac{600.1315}{250} \approx 2.4$. Thus, upholding the optimization model.

Large N

Finally, let's look at an example where we have a large N with a linearly increasing set of job sizes. This case will investigate ten thousand customers $C_1, \ldots, C_{1000}$. Furthermore, as per the previous two examples, we will be deciding on what the optimal R-values are to minimize our Slowdown Variance. Given the input $S_i = i$ for $i = 1, \ldots, 1000$ and an $N = 1000$ we are able to compute the results.

Utilizing the Matlab code, from Figure 1, we are able to determine that the objective function:

$$SDV = 0.273789$$

and the decision variables:

$$\begin{aligned}
R_1, \ \ldots, \ R_{10} \quad &= \quad 503.049746, \ 1006.139842, \ 1509.174441, \ 2012.228999, \ 2515.282869, \\
&\quad\ 3018.340683, \ 3521.392658, \ 4024.448245, \ 4527.503853, \ 5030.559049 \\
\ldots \qquad &\qquad\qquad \ldots \\
R_{990}, \ \ldots, \ R_{1000} \quad &= \quad 498532.498609, \ 498830.628873, \ 499219.986481, \ 499557.119937, \\
&\quad\ 499774.762108, \ 499942.926861, \ 500075.882522, \ 500222.220285, \ 500324.327541, \\
&\quad\ 500417.729523, \ 500500.000000
\end{aligned}$$

As job i approaches N, it is clear that the rounding error proliferates, however, it is negligible in comparison to the value at hand. Hence, this ϵ–value can be disregarded and consequently appropriates the response times of each job with an equivalent ratio; and perfectly fair system.

Conclusion and Future Work

From the examples seen above, it is clear that the optimization problem is modeled well, and better still, has a verifiable solution that can easily be compared against. This is demonstrated in Examples 1 and 2 quite nicely. Furthermore, the notion of approaching a large N in a linear fashion was conveyed and proven to have a slowdown variance of 0.

Where I see this concept propagating to become a successful method for policy implementation is with the addition of arrival times into the optimization problem. Initially, the task would be to take a 'snapshot' of time and assume that we can acquire (or are given) the job's arrival time and size, as well as, the number of jobs N_t that occur at time t. We must also know the total number of jobs M in the system over the course of the 'snapshot'. This will be a static policy that will hopefully be transformed into a mechanism that can be periodically called to schedule jobs in the most fair order.

For example, assuming that we have N jobs and each job has an arrival time A_t where (A_{N+1}, S_{N+1}) $\ldots$ (A_{N+M}, S_{N+M}). We would also need to assume that $A_i \leq \sum_{j=1}^{N} S_j$ for $i = 1, \ldots, M$. Furthermore, we would need a constraint that accounts for job's not being scheduled prior to their arrival $R_{N+i} \geq A_{N+1} + S_{N+i}$.

Secondly, once we have a static policy, we can take periodic and continuous arrival distributions commonly seen by AWS, Google and Microsoft. Since, in practice, each job's arrival time would be unknown it may be advantageous to explore the avenue of developing a predictive model.

P.S. A big thanks to Dr. D. Down and Vincent Maccio for guiding me through the literature, allowing me to use your technical report for reference, and in assisting me with the formulation of the optimization model.

END

Final Project

CAS 781