https://RikHeurter.github.io/LeanBscThesisFormalisation https://github.com/RikHeurter/LeanBscThesis
https://RikHeurter.github.io/LeanBscThesisFormalisation/docs
statedgreenGreen can$_s$*tateblueBluenot*$_r$*eadyFFAA*33*Orangeproved*9*CEC*8*BGreencan*$_p$*roveA*3*D*6*FFBlu*

BscThesisFormalisation

Rik Heurter

April 24, 2025

Chapter 1

General Roadmap

1.1 The goal

Theorem 1. *EQUIOptimal Within our model, assuming jobs are malleable, have exponentially distributed sizes, and all follow the same speedup function, s,* $\mathbb{E}[T]^{EQUI} \leq \mathbb{E}[T]^{P}$ *for any scheduling policy* P*.*

Proof. def:lemma$2_1, def : lemma2_3 Let$Pbeamappingpolicythatprocessesmalleablejobsthatcanbeswitchedbetw
$coresandlE-coresamongtheijobs.Hencefor some, therateofdeparturesfromstateiunderPisatmost\mu \sum_{m=1}^{j} s($
$\alpha_m, l \cdot \beta_m) with 0 <$ j $\leq \min(i, k+l), \alpha_m + \beta_m > 0, \forall\ 1 \leq m \leq j$ and $||\alpha||_1, \leq$
$1, ||\beta||_1 \leq 1$. However $||\alpha||_1 < 1$ always has unused cores or core time, which is clearly suboptimal. Therefore we will assume that $||\alpha||_1 = 1 = ||\beta||_1$.
Now we will upperbound the previous summation using the concavity of s.

$$\begin{aligned}
\frac{1}{j}\sum_{m=1}^{j} s(k\alpha_m, l\beta_m) &= \frac{1}{j}s(k\alpha_1, l\beta_1) + \frac{1}{j}s(k\alpha_2, l\beta_2) + \frac{1}{j}\sum_{m=3}^{j} s(k\alpha_m, l\beta_m) \\
&= \frac{2}{j}\left(\frac{1}{2}s(k\alpha_1, l\beta_1) + \frac{1}{2}s(k\alpha_2, l\beta_2)\right) + \frac{1}{j}\sum_{m=3}^{j} s(k\alpha_m, l\beta_m) \\
&\leq \frac{2}{j}\left(s(\frac{k}{2}(\alpha_1+\alpha_2), \frac{l}{2}(\beta_1+\beta_2))\right) + \frac{1}{j}\sum_{m=3}^{j} s(k\alpha_m, l\beta_m) \\
&\leq \cdots \leq \frac{j}{j}s(\frac{1}{j}\sum_{m=1}^{j} k\alpha_m, \frac{1}{j}\sum_{m=1}^{j} l\beta_m) = s(\frac{k}{j}, \frac{l}{j})
\end{aligned}$$

Thus we conclude that

$$\sum_{m=1}^{j} s(k\alpha_m, l\beta_m) \leq j \cdot s(\frac{k}{j}, \frac{l}{j}) \qquad \forall 0 < j \leq i, \forall \vec{\alpha}, \vec{\beta}$$

and thus an upperbound on P's rate of departures is

$$j \cdot s\left(\frac{k}{j}, \frac{l}{j}\right)\mu.$$

From lemma 2.1 it follows that $j \cdot s\left(\frac{k}{j}, \frac{l}{j}\right)\mu$ is non-decreasing in j. And thus to obtain the optimum rate we might as well use i to get an upper-bound of the form:

$$i \cdot s\left(\frac{k}{i}, \frac{l}{i}\right)\mu,$$

Which is exactly what is achieved by the policy EQUI. Since both markov chains only differ in their departure rates we can conclude that

$$E[N]^{EQUI} \leq E[N]^{P}$$

And now we can use Little's law, which states that the mean response time is the mean number in the system / mean throughput. The mean throughput stays the same, by the stability and constant arrival rate. From this we can conclude that

$$E[T]^{EQUI} \leq E[T]^{P}$$

□

Chapter 2

definitions

2.1 General Idea

Before we can start proving properties about our actual mathematical object we first need to actually start writing down what our definitions are. Thus what we aim to do here is write down all the definitions we require.

2.2 Speedup function

Definition 2. coreSpace*coreSpace* is defined to be the actual allowed allotment of cores i.e. if given a vector $c \in \mathbb{R}^n$ describing how many cores we have then our corespace becomes: $x \in \mathbb{R}^n$ with the requirement that: $0 \leq x_i \leq s_i$ forall $1 \leq i \leq n$. After this we shall abbreviate this space as C_c.

The Mathlib library already defines concavity more generally. We redefine our own instance to make it easier to work with:

Definition 3. def:corespacemyConcaveA function f is said to be *concave* on $c \in \mathbb{R}^{\kappa}$ the core vector if $\forall x, y \in C_s$ and all $a, b \in \mathbb{R}$ with the property that $a + b = 1$ implies that:

$$a \cdot f(x) + b \cdot f(y) \leq f(a \cdot x + b \cdot y)$$

Definition 4. def:coreSpaceSublinearA function f is said to be *sublinear* on C_c, with c the core vector if $\forall x \in C_S \ ||f(x)|| \leq ||x||$.

This brings us to the definition of the speedup function:

Definition 5. def:Sublinear, def:myConcaveSpeedUpFunctionA function $s \colon C_c \to \mathbb{R}$ is said to be a *speedup function* if s is concave and sublinear on C_c.

2.3 Markov Chain definitions

Since our queue will be one-dimensional of infinite size anyway we define our RateMatrix to be the following:

Definition 6. RateMatrixA function $Q\colon \mathbb{N}\times\mathbb{N}\to\mathbb{R}$ is said to be a rate matrix if the following holds:

1. $Q(0,1) = -Q(0,0)$
2. $\forall n \in \mathbb{N} \neq 0, Q(n,n+1) + Q(n,n-1) = -Q(n,n)$
3. $\forall n,k \in \mathbb{N}, |k-n| \geq 2, Q(n,k) = 0$

Now the invariant distribution of our queue is simply:

Definition 7. def:RateMatrixInvariantDistributionA infinite-dimensional vector lambda described by a function $\mathbb{N}\to\mathbb{R}$ is said to be an *invariant distribution* for a queue Q if the following holds:

1. $\forall n \in \mathbb{N}, n \neq 0, \lambda(n-1) * Q(n-1,n) + \lambda(n+1) * Q(n+1,n) = \lambda(n)$
2. $\lambda(1) * Q(1,0) = \lambda(0)$
3. $||\lambda||_1 = 1.$

And then the mean number in the system is:

Definition 8. def:InvariantDistributionMRTSequenceThe *mean number in the system* is defined as the following equation:

$$\mathbb{E}[\lambda] = \sum_{n=0}^{\infty} \lambda(n) * n$$

We will define the mean response time of our queue directly dependent on Little's law, since the only way we evaluate our mean response time is by invoking Little's law anyway.

Definition 9. def:MeanLambdaMeanResponseTimePolicyThe *mean response time*, denoted as $\mathbb{E}[T]$ of a queue is defined to be the mean number in the system / mean throughput i.e. (mean throughput is equal to Λ since this doesn't vary per state):

$$\mathbb{E}[T] = \frac{\mathbb{E}[\lambda]}{\Lambda}$$

2.4 Scheduling Policy

Now we have everything we need to state what a policy is. For this we first define what a policy is.

Definition 10. def:SpeedUpFunctionPolicyA *policy* is a function $p\colon \mathbb{N} \times \mathbb{N} \to \mathbb{R}^n$ describing per state n how to allocate cores in C_c over j, $0 \leq j \leq n$, jobs. Thus if we write this out, p is subject to the following conditions:

1. $p(n, j) = 0 \forall j > n$.
2. $\left(\sum_{k=0}^{i} p(i, k)\right) \in C_c$
3. $\forall 0 \leq i \leq n, p(n, i) \geq 0$

Now we can define the magnum opus of our actual definitions: the schedule policy.

Definition 11. def:Policy, def:SpeedUpFunction, def:RateMatrixSchedulePolicyA *schedule policy* is a function describing for a specific queue Q and a specific speedup function s the departurerates per state n.

Chapter 3

lemma 2.1

3.1 General Idea

What we would like to do is mostly just copy the proof from the thesis, since almost nothing happens in it anyway. For this we need the following definitions:

1. concave
2. sublinear
3. speedup.

Lemma 12. *def:SpeedUpFunction, def:Sublinear, def:myConcavelemma2*$_1$*Foranyconcave, sublinearspeedupfu* $\mathbb{R}^n$*, the function* $i \cdot s(\frac{\alpha}{i})$ *is increasing in* i *for all* $i < ||\alpha||_1$*, and is non-decreasing in* i *for all* $i \geq ||\alpha||_1$*.*

Proof. We will proof the following two different cases:

Case 1 $i < ||\alpha||_1$: We need to prove in this case that $i \cdot s(\frac{\alpha}{i})$ is increasing. We will look at the following difference for any $\delta > 0$:

$$i(1+\delta)s\left(\frac{\alpha}{i\cdot(1+\delta)}\right) - i\cdot s(\frac{\alpha}{i}) = i\left((1+\delta)s\left(\frac{\alpha}{i\cdot(1+\delta)}\right) - s(\frac{\alpha}{i})\right)$$

Since $||\frac{\alpha}{i}|| > 1$, s increases sublinearly, and $(1+\delta)s\left(\frac{\alpha}{i\cdot(1+\delta)}\right) > s(\frac{\alpha}{i})$. This yields

$$i\cdot(1+\delta)s\left(\frac{\alpha}{i\cdot(1+\delta)}\right) - i\cdot s\left(\frac{\alpha}{i}\right) > 0$$

From which we can conclude that $i \cdot s(\frac{\alpha}{i})$ is increasing in i.

Case 2 $i \geq ||\alpha||_1$: We need to prove in this case that $i \cdot s(\frac{\alpha}{i})$ is non-decreasing. This follows directly from the assumption that $s\left(\frac{\alpha}{i}\right) = \frac{\alpha\cdot\beta}{i}$. From which we conclude that

$$i\cdot s\left(\frac{\alpha}{i}\right) = \alpha\cdot\beta \qquad \forall i \geq ||\alpha||_1$$

which is non-decreasing in i.

□

Chapter 4

lemma 2.3

4.1 Change in difficulty

In the proof of theorem 2.2 we really quickly state that:

$$E[N]^{EQUI} \leq E[N]^{P},$$

intuitively this is rather clear, since one of the departurerates being higher clearly means the queue goes quicker at some point and thus all else being equal we would expect less people to be waiting in the queue. However this is really cumbersome to actually show via invariant distributions. Since we now need to prove that a higher throughput at one node needs more input from lower nodes and thus we need to scale the lower portion somewhat.

Specifically we can aim to show the following: if k is the index, whose departure rate increased and λ is the old situation and μ is the new situation then the following should hold for all $i > k$:

$$\frac{\mu_i}{\sum_{j=k}^{\infty} \mu_j} = \frac{\lambda_i}{\sum_{j=k}^{\infty} \lambda_j}$$

We can do a similar trick with $i < k - 1$, and then we can conclude that it is only rescaling on two sides and can prove that $\mu_k \leq \lambda_k$.

4.2 Step-by-step idea

4.2.1 General expression of invariant distribution

We will first describe the invariant distribution in terms of λ_0 if there are no rates which equal zero:

Lemma 13. *The invariant distribution of a scheduling policy whose rate is never* 0 *is of the form:*

$$\left(\prod_{i=0}^{n} \frac{\Lambda}{a_i}\right) \lambda_0$$

at index n

Proof. This comes down to just using the detailed balance property of the invariant distribution and using induction specifically: **Base case**: $n = 0$ clearly $\lambda_0 = \lambda_0$. **Induction case**: The induction hypothesis is that

$$\left(\prod_{i=0}^{n-1} \frac{\Lambda}{a_i}\right) \lambda_0$$

We now need a case distribution again: $n = 1$ or $n \neq 1$. If $n = 1$ then:

$$\Lambda\lambda_0 = a_1\lambda_1 \implies \lambda_1 = \frac{\Lambda}{a_1}\lambda_0$$

q.e.d. Otherwise:

$$\begin{aligned}
(\Lambda + a_{n-1})\lambda_{n-1} &= a_n\lambda_n + \Lambda\lambda_{n-2} \\
\implies \lambda_n &= \frac{(\Lambda + a_{n-1})\lambda_{n-1} - \Lambda\lambda_{n-2}}{a_n} \\
&\overset{IH}{=} \frac{(\Lambda + a_{n-1})\left(\prod_{i=0}^{n-1} \frac{\Lambda}{a_i}\right)\lambda_0 - \Lambda\left(\prod_{i=0}^{n-2} \frac{\Lambda}{a_i}\right)\lambda_0}{a_n} \\
&= \left((\frac{\Lambda}{a_n} + \frac{a_{n-1}}{a_n})\left(\prod_{i=0}^{n-1} \frac{\Lambda}{a_i}\right) - \frac{\Lambda}{a_n}\left(\prod_{i=0}^{n-2} \frac{\Lambda}{a_i}\right)\right)\lambda_0 \\
&= \left((\frac{\Lambda}{a_n} + \frac{a_{n-1}}{a_n})\Lambda^{n-1}\left(\prod_{i=0}^{n-1} \frac{1}{a_i}\right) - \frac{\Lambda}{a_n}\Lambda^{n-2}\left(\prod_{i=0}^{n-2} \frac{1}{a_i}\right)\right)\lambda_0 \\
&= \left(\left(\prod_{i=0}^{n} \frac{\Lambda}{a_i}\right) + \frac{a_{n-1}}{a_n}\Lambda^{n-1}\left(\prod_{i=0}^{n-1} \frac{1}{a_i}\right) - \frac{\Lambda}{a_n}\Lambda^{n-2}\left(\prod_{i=0}^{n-2} \frac{1}{a_i}\right)\right)\lambda_0
\end{aligned}$$

□

Then we can describe how it is normalised.

Lemma 14. *The invariant distribution of a scheduling policy whose rate is never* 0 *is of the form:*

$$\frac{\prod_{i=0}^{n} \frac{\Lambda}{a_i}}{\sum_{n=0}^{\infty} \prod_{i=0}^{n} \frac{\Lambda}{a_i}}$$

at index n.

Proof. lem:InvariantDistributionValue, def:InvariantDistribution First we will prove that the invariant distribution at index n is of the form:

$$\prod_{i=1}^{n} \frac{\Lambda}{a_i}\lambda_0$$

And then prove that only this distribution will sum to 1. □

4.2.2 Non-negative distribution values

We first will need to show that if there are unreachable states in the queue that the invariant distribution is zero there, or more concretely:

Lemma 15. *def:SchedulePolicy, def:InvariantDistribution If there exists unique $n \in \mathbb{N}$ such that the departure rate is 0 at state n then $\forall i < n$ the invariant distribution has value 0 for state i.*

Proof. This follows rather quickly from **??**. Simply state the invariant distribution definition at index $n-1$ and then simple rewriting yields an equality of the form:

$$\frac{a}{b+c} = \frac{a}{b}$$

Where we already know that $c \neq 0$ and thus $a = 0$ is the only correct option. □

Then we get the highest such values to allow for nice induction.

Lemma 16. *lem:UnreachableInvariantDistribution If there exists finite $A \subsetneq \mathbb{N}$ such that for all $n \in A$ the departure rate at state n is zero and there exists no $n \notin A$ with departurerate 0 then the invariant distribution has value 0 at $i < \max(A)$.*

Proof. We will need to do induction over i. base case $i = 0$:
then use previous lemma. induction case i:
we know that the invariant distribution of $i-1$ is zero. If the departurerate of i is not zero i.e. $i \notin A$ then simply apply induction hypothesis. Otherwise copy previous proof where we let $n = \min(\{n : i < n\})$, but now started from a different spot. □

4.2.3 Calculating the changed invariant distribution

Lemma 17. *lem:BeforeCutScaled For any two scheduling policies whose departure rates differ at exactly one index n, $\forall i \in \mathbb{N}$ with $i \leq n-1$ the invariant distribution at index i differs only a constant $c \in \mathbb{R}$.*

Proof. We will need to do a case distinction: either the rate is 0 or is isn't.
If it is: Apply previous lemma.
If it isn't: Need to go through □

Lemma 18. *def:SchedulePolicy, def:InvariantDistribution For any two scheduling policies whose departure rates differ at exactly one index n, $\forall i \in \mathbb{N}$ with $i \geq n$ the invariant distribution differs only a constant $c \in \mathbb{R}$.*