Role of Transcription Bursting in Cross-Feeding Interactions

Project Report

CL 717: Evolutionary Dynamics

Date of Submission:
16th September 2018

Submitted By:

Devendra Govil
Chemical Engineering, IIT Bombay
Mumbai, India—400076
devendragovil@iitb.ac.in
Roll Number: 120020064

Contents

Role of Transcription Bursting in Cross-Feeding Interactions

Cross-Feeding Interactions [1]

Introduction [1]

In the natural environment bacteria live in a multitude of complex environments with other bacterial cells in a colony exhibiting a remarkable level of interdependence on each other and cooperating to share nutrients and metabolites. This level of interdependence sometimes doesn't sit well with evolutionary theory since a non-cooperating "cheat" can take advantage of this cooperating environment and gain relative fitness advantage over other bacterial cells. This would suggest that any cross-feeding interaction would then be unstable since the present of cheater cells would lead to a collapse of the whole colony and population. However, not only is cross-feeding interaction present in the natural environment, it is widespread.

Pande et al Paper [1]

The Paper tries to study cross-feeding interactions. They delete genes from the genome of Escherichia Coli. First Amino Acid overproducing genes were identified, and these were deleted to give three strains that were Amino Acid overproducing. These 3 strains were ΔnuoN, Δmdh, and Δppc. Then 4 genes related to the production of 4 Amino Acid were identified and these genes were deleted to give 4 Auxotrophs—ΔargH, ΔtrpB, ΔleuB, and ΔhisD.

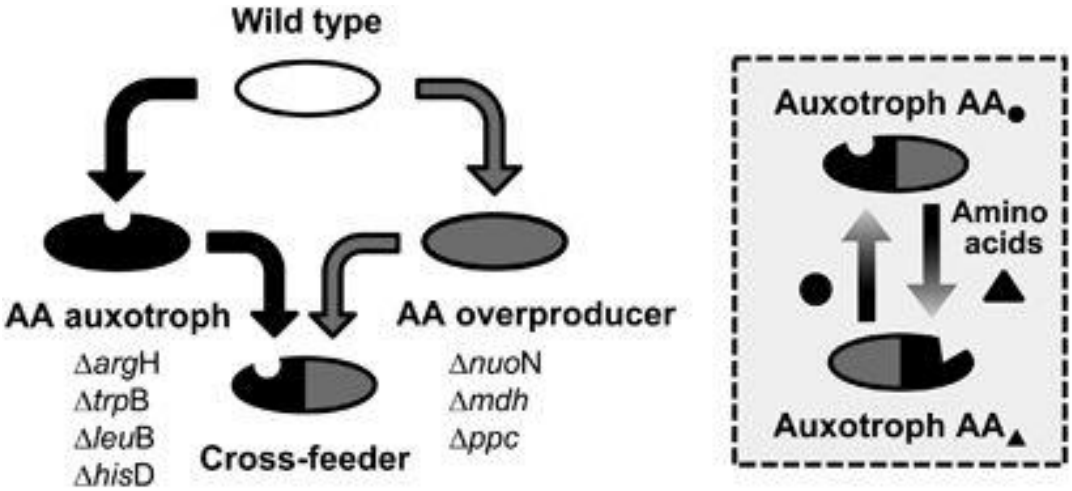


Figure 1: This image has been taken from Pande et al Paper, ISME Journal (2014), and depicts the experimental procedure implemented using the deletion of genes from the genome of E Coli. 4 Auxotrophs were hence created using single deletion of AA producing genes.

Results of the Paper [1]

It is found that when alone, both Auxotrophs and Overproducers have lower fitness compared to WT cells, however it can be seen that cross-feeding interactions of most combinations is higher than WT cells. Relative fitness here is the ratio of Malthusian parameters.

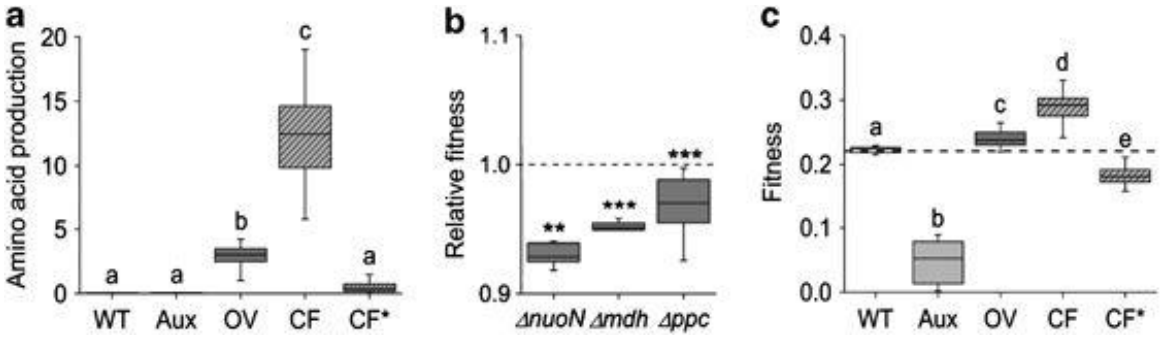


Figure 2: This image has been taken from the Pande et al (2014) Paper, and show the relative fitness of the WT cells, the Auxotrophs, as well as the Overproducers, and the Cross-feeding interactions

As the paper describes the above figure (with some editing for clarity) [1]:

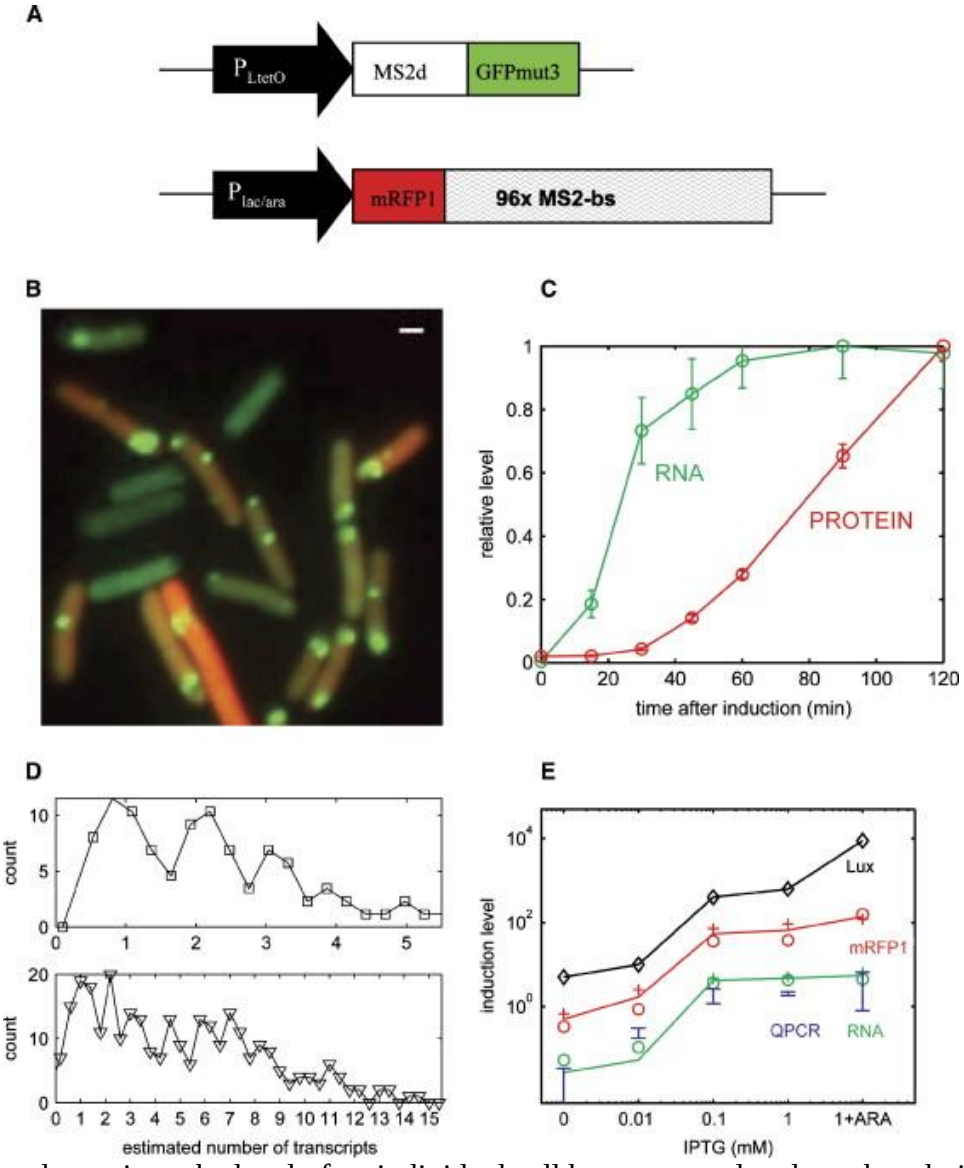
> Characterisation of mutants and coculture experiment. (**a**) Amino acid (AA) production levels (that is, arginine, tryptophan, leucine, and histidine) of WT, four auxotrophs (Aux), three overproducers (OV), eleven cross-feeders (CF) and the cross-feeding genotype Δ*leuB*Δ*nuoN* (CF*) within 24 h determined as productivity of cocultured AA auxotrophs (*n*=8 for every auxotroph-genotype combination). Combinations with matching amino acid auxotrophies were excluded. (**b**) Competitive fitness of the three AA-overproducing mutants relative to WT within 24 h in minimal medium. Relative fitness is the ratio of Malthusian parameters and the dashed line indicates equality in fitness between WT and competitor. **c**) Fitness given as the Malthusian parameter of WT, pairs of cocultured auxotrophs (Aux, 6 combinations), overproducers (OV, 3), cross-feeders (CF, 45), and cross-feeding consortia involving strain Δ*leuB*Δ*nuoN* (CF*, 9) within 24 h

The paper also demonstrates that cross-feeding interaction have higher relative fitness probably due to a division in metabolic labour. The Auxotrophs had a 20% higher fitness when they were competed against WT cells in the presence of the Amino Acid that the Auxotroph couldn't produce. While, overproducing strains had a a lower fitness compared to the WT cells.

The final answer that the paper sought was about the stability of cross-feeding interactions, since they are susceptible to cheating cells, who take advantage of the public goods without contributing to any, for example in the case of a combination of Auxotrophs and Overproducers. Such populations can be considered unstable and could potentially collapse. So invasion-from-the-rare experiments were done where a mutant invaded the space of another cell, in a cross-feeding interaction. It was however found that cheater cells, or Auxotrophs where they are cheater, had a substantial fitness advantage when rare, however the fitness advantage didn't persist, and the population stabilized at ratios around 1:1. This was true for reverse experiments as well. It is thought that "negative frequency dependent selection" is an answer to the problem by which cross-feeding interactions are stabilized and perpetuated.

Kinetics of Transcription [2]

Introduction [2]



Gene Expression is a complex and intricate process that involves many steps. These steps are various chemical and physical processes, many of which depend on probabilistic events, like a promoter finding its target etc. These all processes combine to make gene expression a dynamic and very complicated process. Even if the cellular environment were to be completely homogenous, there would still be noise in gene expression. The paper by Golding et al (2005) attempts to study this dynamical nature of gene expression by measuring mRNA and protein at the level of an individual cell by very novel and tough techniques.

Figure 3: This image has been taken from Golding et al (2005), Cell.
A. This is the basic components system of the experiment, with a tagging protein composed of MS2 and GFP fused together, and a DNA sequence, with MS2 binding sites, both inducible by promoters.
B. This is an image from the experiment where the green dots represent transcripts and the red represents proteins.
C. The protein and transcript levels after induction in the cell. The protein obviously follows behind the transcript levels.
D. This is the frequency distribution of the number of transcripts in the population of cells.
E. This is the induction levels for different concentrations of IPTG

Experimental Approach [2]

The DNA sequence belonging to a plasmid is taken and the gene sequence corresponding to a red fluorescence protein (mRFP1) is used and to it is attached a gene sequence which consists of 96 MS2 binding sites. This mRNA production is controlled by the lac/ara promoter system, using $P_{lac/ara}$ which is repressed by LacI and activated by AraC. It also responds to IPTG and is induced by IPTG. The tagging protein is really a MS2 protein which is attached to a GFP protein which is again present on a Plasmid. This is promoted by P_{LtetO} Promoter. In the experiment, this tagging protein is produced and is present in excess. When mRNA sequence is transcribed, mRNA contains the mRFP1 sequence and MS2 binding sites. The MS2 on the MS2-GFP dimer goes and binds on those sites. Since there are 96 binding sites, this leads to a saturation of the tagging protein at that location of the transcript. This leads to a local green foci which can be counted in a cell to measure the number of transcripts in a cell. If the number of transcripts is high, then the green foci fluorescence is normalized using that of a green foci of one single transcript. This is a novel way to measure and track the mRNA transcripts at a cellular level. The background fluorescence has to be discounted from the I_G levels of the foci. The transcripts produce the RFP protein which is red in colour and hence I_R levels can be used to measure the protein levels in a cell. Experimental findings and time series for transcripts and proteins are then used to model the transcription process.

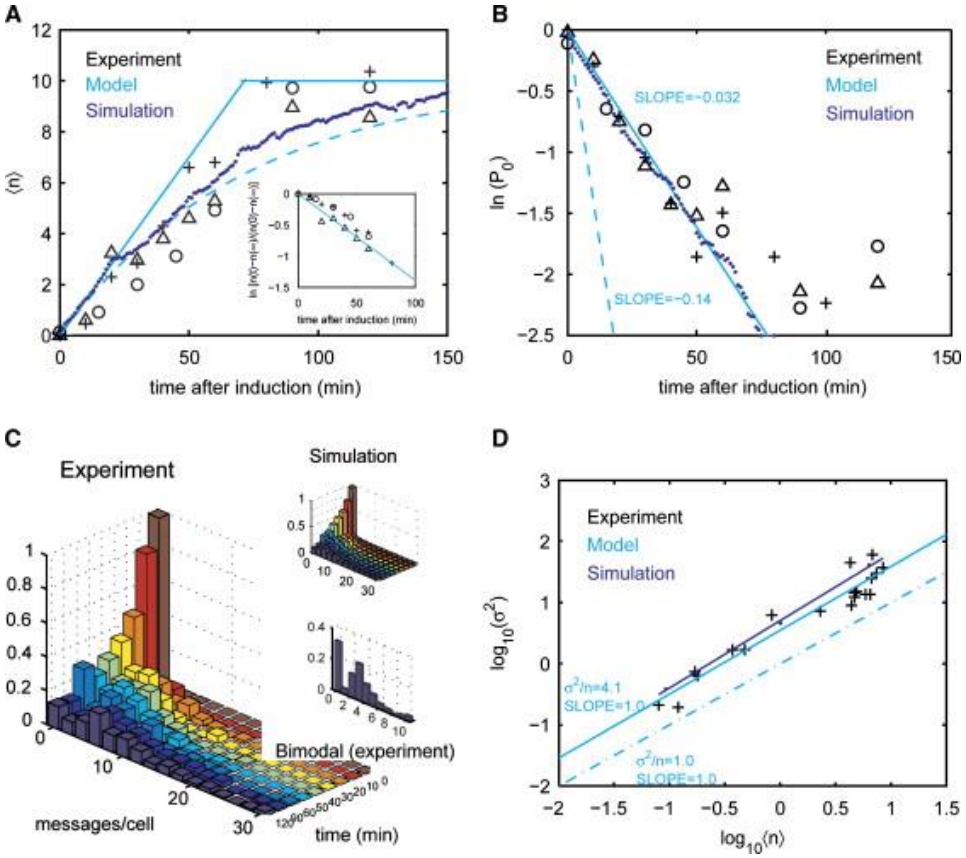
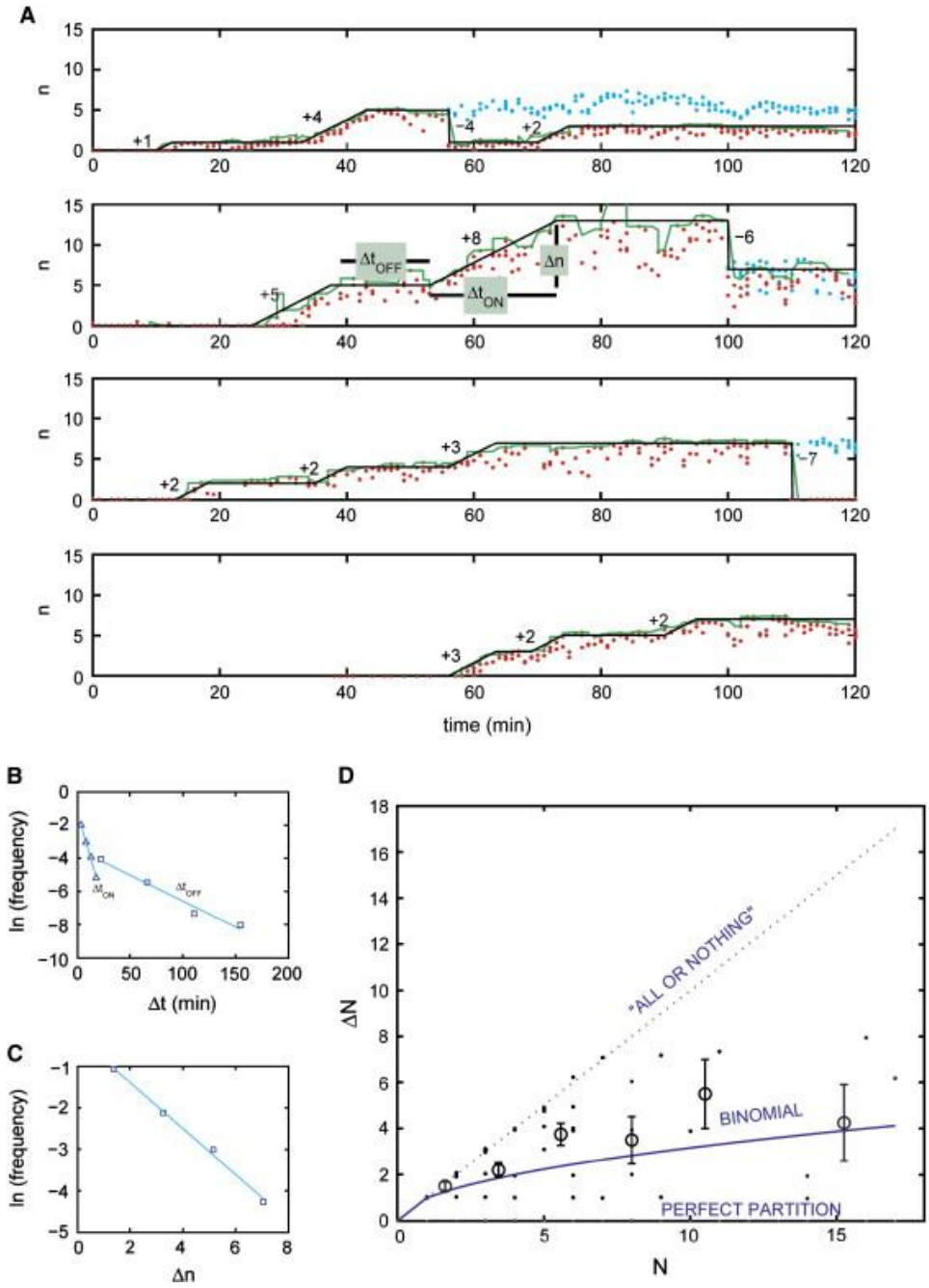


Figure 4: This figure is taken from Golding et al (2005), Cell. **A.** This is simply transcript level after induction. **B.** P_0 is the fraction of cells in population without a transcript, which reduce exponentially with time. However, the experimental parameter is off from prediction by a factor of 4. **C.** Binomial distribution of transcripts after cell division **D.** Variance vs Average: The experimental parameter is again 4 times different than Model Prediction.

Dynamic Model of Transcription [2]



The paper first considers and queries if transcription is really a Poisson process. However experimental results were inconsistent with a simple Poisson process. For one, the k_1 parameter, for the ratio of population with zero transcription events reduced at a much slower rate (4 times less) than what a Poisson Process would suggest. Similarly, while the Variance was proportional to the Average, the proportionality constant was off by a factor of 4. These things precluded a simple Poisson Process as the possible process for Transcription. The experimental method introduces

Figure 5: This image is taken from the Golding et al (2005), Cell Paper. **A.** This shows the number of transcripts as time goes along. It is clearly visible that there are sudden jumps and drops in the number of transcripts. **B.** This is the frequency distribution of Δt_{OFF} and Δt_{ON} which clearly shows that both of them are exponentially distributed random variables. Here, Δt_{OFF} has a mean of 37 minutes and Δt_{ON} has a mean of 6 minutes **C.** Frequency distribution of RNA jumps which is geometrically distributed. **D.** This shows the difference in transcripts inherited by daughter cells versus N. The experimental values are in good agreement with a Binomial Model of Cell Partition.

changes in gene sequence that makes the decay of transcripts to be a very slow process hence, transcript decay is neglected in the paper for a cell cycle.

To explain observations, an alternate model is proposed which consists of transcriptional bursts. That is transcription for a long time exists in an OFF state with no transcripts forming. After some time, the transcription state changes to ON. In this ON state, the gene sequence can either produce a transcript or revert back to OFF state. Experimental results suggest that Δt_{ON} and Δt_{OFF} are random variables with probability density functions that are exponential. Exponential distributions have only one parameter. When the gene sequence is in the ON state, the transcription process happens as a Poisson Counting Process, where the arrival times are all independent random variables with each of them having an exponential probability density function. Hence, transcription is composed of two random processes—one of them being the exponentially distributed ON/OFF states of the gene sequence and the other being the Poisson Process of the actual transcription when in ON state. These two random processes combine so that the number of transcripts that are produced in each transcriptional burst is a discrete integer-valued random variable that is geometrically distributed. The cells also show a binomial partition of transcripts and proteins after cell division, which explains why cells early on in their life cycles have protein levels and transcripts that are very weakly correlated.

Protein Levels [2]

Protein Level can be measured by the red fluorescence produced by the mRFP1 protein which is a product of translation of mRNA. We can estimate the total number of protein molecules produced by a transcript by finding out the ratio of protein molecules at steady state to the number of mRNA transcripts at steady state. This is can be done by observing the fluorescence of Green I_G and Red I_R proteins in the cell. We need to find the individual photon fluxes of each proteins of GFP and RFP that is done in the paper. In the end the paper concludes that the overall ratio is around 60-110, which means around 100 molecules of proteins are translated by each transcript of mRNA.

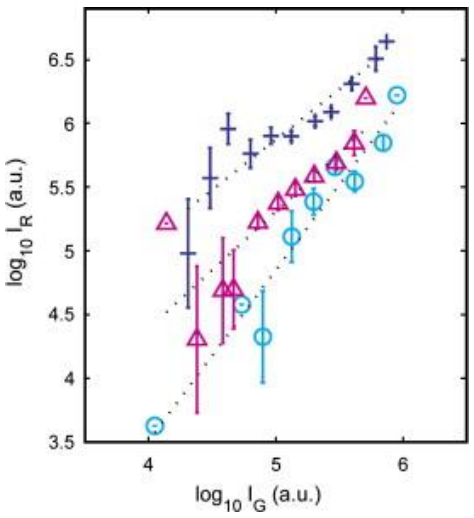


Figure 6: This image has been taken from the Golding et al Paper published in Cell, 2005. This image shows the scatter plot of logarithmic intensities of Red and Green Fluorescent Proteins. Three experiments were performed separately, and they were let to grow exponentially. The data on the left displays those three experiments.

Project Aim

Here, an attempt is made to explain phenomenon observed in Cross-Feeding Interactions using the dynamics of cellular transcription and translation. As mentioned previously Auxotrophs seem to gain a relative fitness over their Wild Type strains when in presence of nutrients that they are unable to produce. The relative fitness can often be significant. This fact needs to be explored and explained because it is essential to study and understand cross-feeding interactions.

Hypothesis

We hypothesize that Auxotrophs gain relative fitness advantage over WT cells in a nutrient rich medium because they can use all their cellular machinery to produce just one cellular metabolite. This is manifested in a way where the mean transcription levels in the ON states of transcription leads to pronounced and higher levels of transcript bursts for a particular protein. This higher level of burst than leads to a level of production of protein that is substantially higher. For example: Let's say a WT cell has to produce two proteins A and B, and there are two Auxotrophs who produce proteins A and B exclusively. So the Auxtotroph producing protein A alone is able to produce more than twice what WT cell is able to produce, hence even while sharing protein A with the other Auxotroph, Auxotroph A is able to get more protein A than what a WT cell would be able to get. This hypothesis is tested using computer simulations.

Single Cell Analysis

First, we try to simulate a single cell's transcription events and protein levels. We use a MATLAB code to do this. There are certain simplifying assumptions taken to simulate the problem. The Δt_{ON} and Δt_{OFF} time intervals are taken to be exponentially distributed, while transcription within an ON period is considered to be a Poison Counting Process with arrival times that are themselves exponentially distributed independent of each other. Here for a single cell case, we have taken average times (μ) to be 600 seconds for Δt_{OFF} and 100 seconds for $\Delta t_{ON.}$ The arrival times also have a μ of 10 seconds. The transcripts are assumed to not degrade and stay active throughout the course of the cell life cycle. The proteins are assumed to be made uniformly without stochasticity as number of transcripts times 0.3 per second. Protein levels are calculated for every second based on number of transcripts. The proteins are also assumed to accumulate without degradation and other usage. Cell Division is assumed to have happened after 10^5 molecules of proteins have been produced for this single cell. The MATLAB code for this part is given in Appendix A.

One Simulation Instance

In one of the instances of simulation, the division time for the cell came out to be 6964 seconds.

Transcription Events

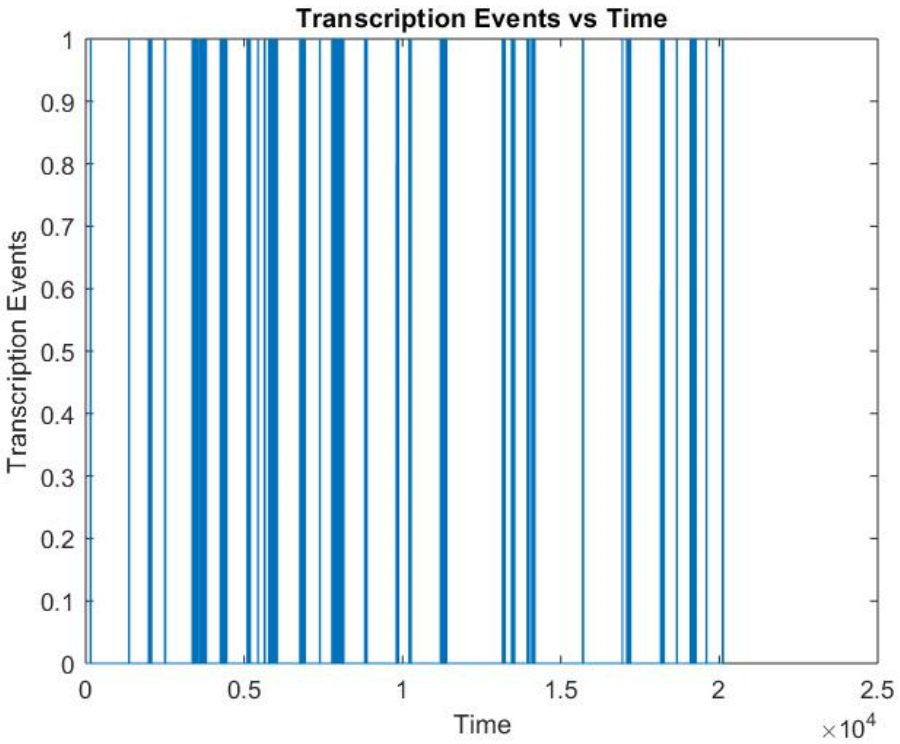


Figure 7: This figure shows the transcription events as they happen. A transcription event is a transcript mRNA being produced and is displayed on the Y Axis by 1. A default state of 0 on the Y Axis means no mRNA being produced.

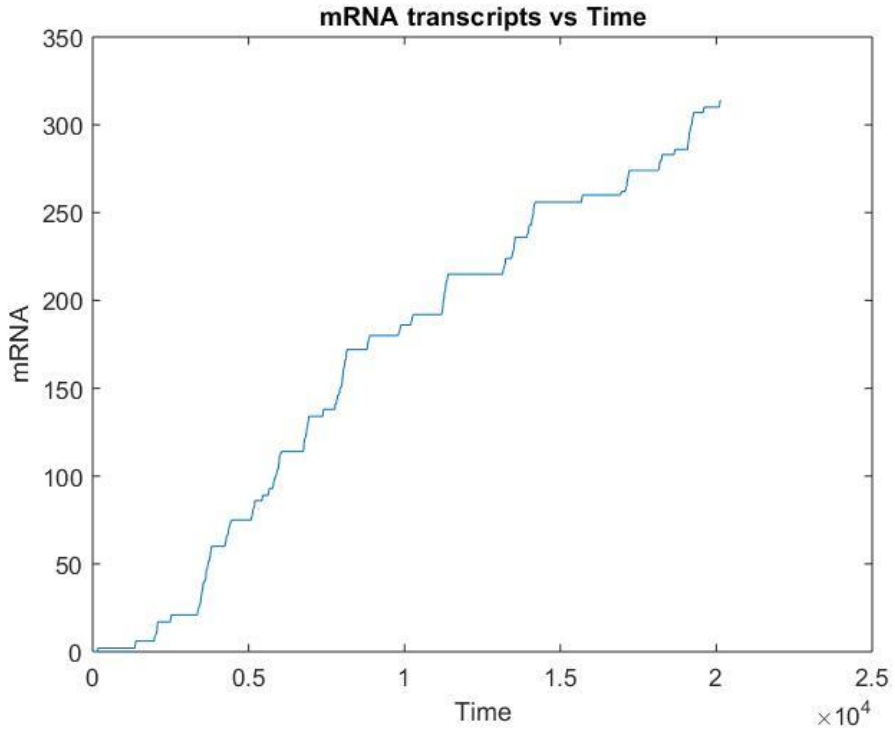


Figure 8: This figure is the cumulative transcripts produced and the net transcripts in the cell at a time. These show the mRNA levels of the cell.

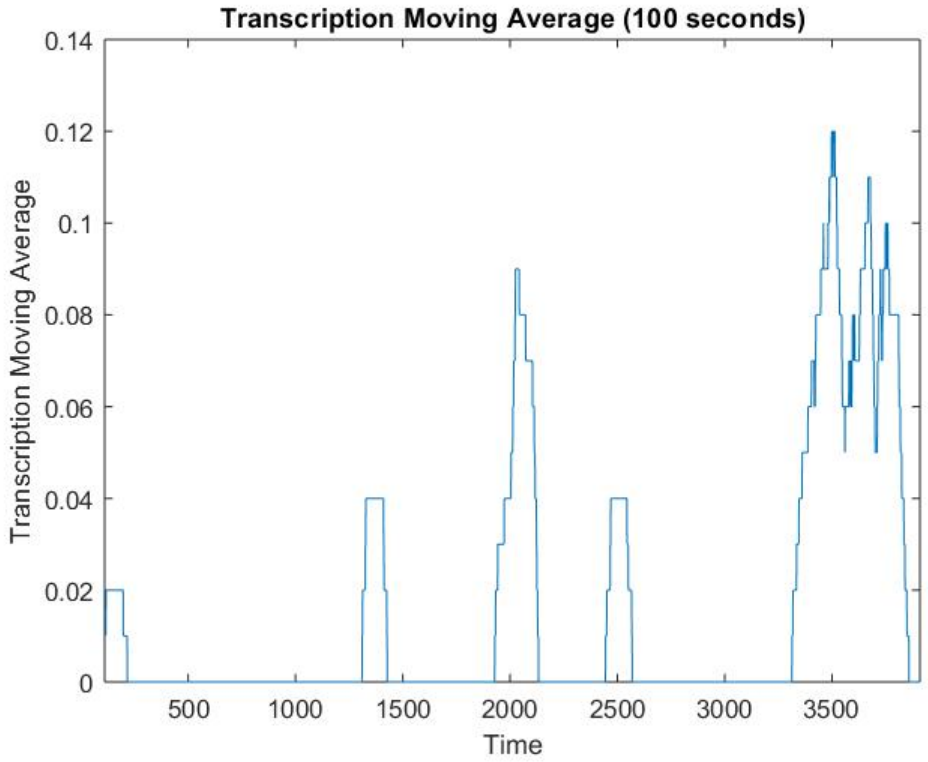


Figure 9: This figure shows the moving average transcriptions in the cell averaged over 100 seconds. This helps visualize the transcription process. The figure shows only the transcriptional activity for a part of the cell cycle.

Proteins Formation

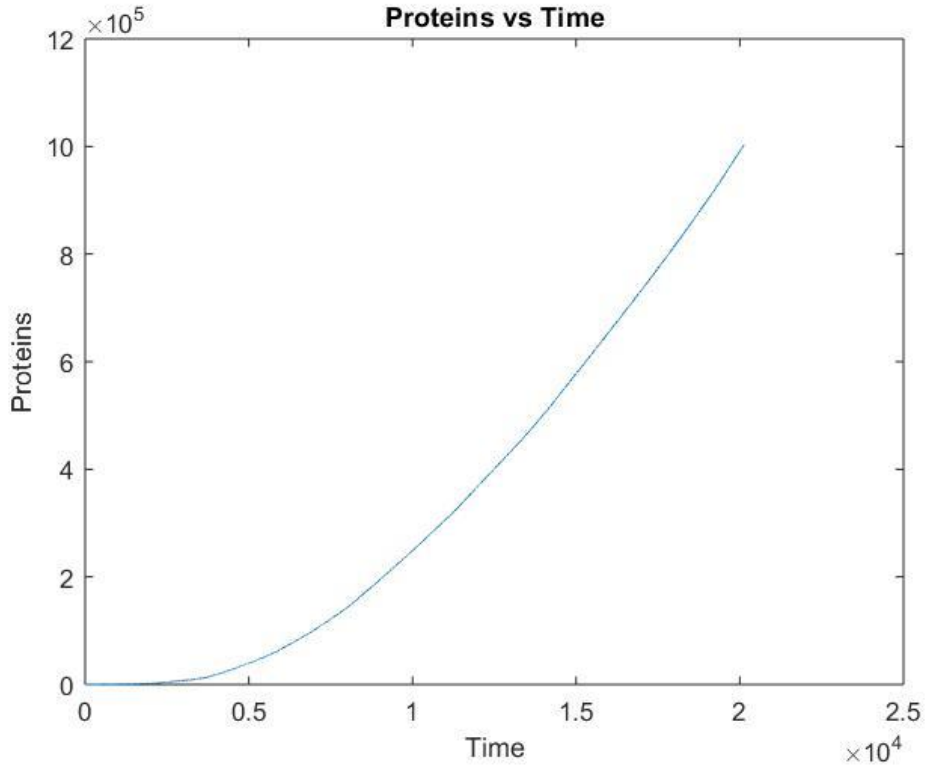


Figure 10: This figure demonstrates the protein formation with time. The proteins produced cross over the 10^5 threshold at around 7000 seconds.

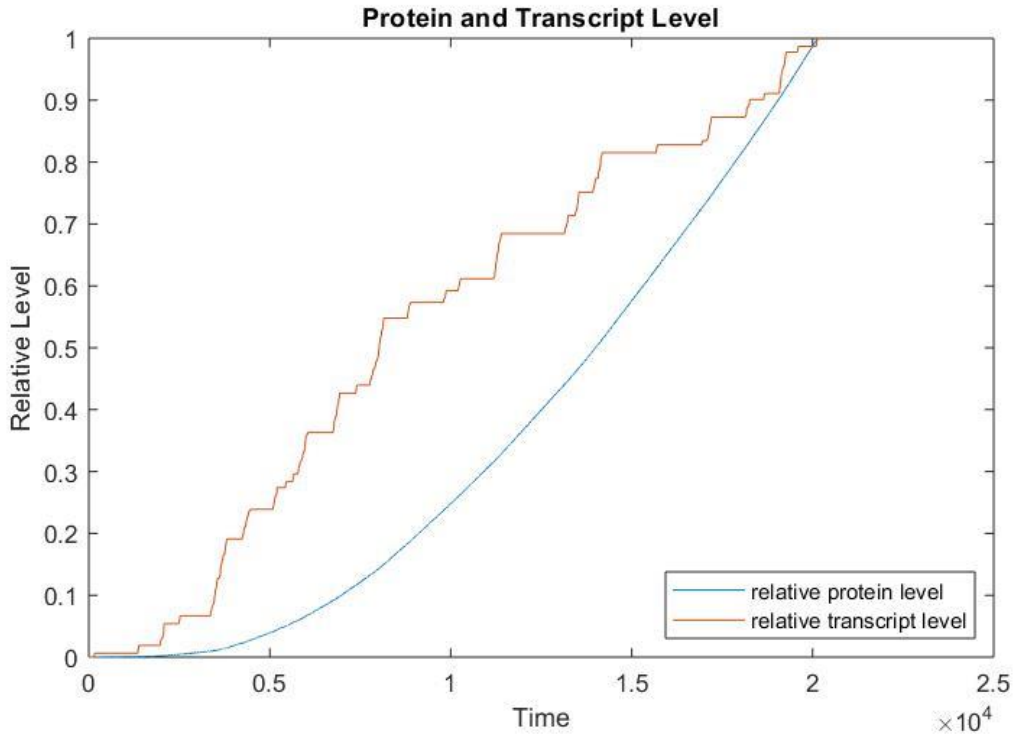


Figure 11: This figure above shows the relative levels of Proteins and Transcripts as compared to their respective maximum levels.

Cellular Division Time Distribution

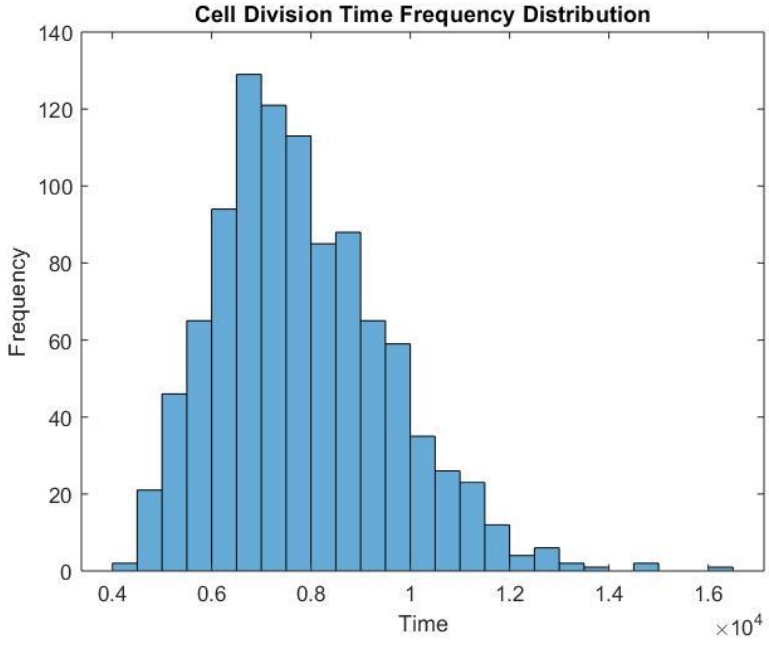


Figure 12: Cell Division Frequency Histogram

Now, using the above MATLAB code as mentioned in the Appendix A, and using the same parameters, we do 1000 runs of the simulation to get 1000 instances of cellular division times. This MATLAB code is mentioned in Appendix B. This cellular division time data is used to construct a frequency distribution curve. The maximum division time that we obtained was 16074 seconds while the minimum division time that we obtained was 4365 seconds. We used the histogram function in MATLAB to create frequency histograms, once without specifying the number of bins and later by specifying the use of 100 bins by using an input parameter in the histogram function. The mean division time of the cell came out to be 7838 seconds.

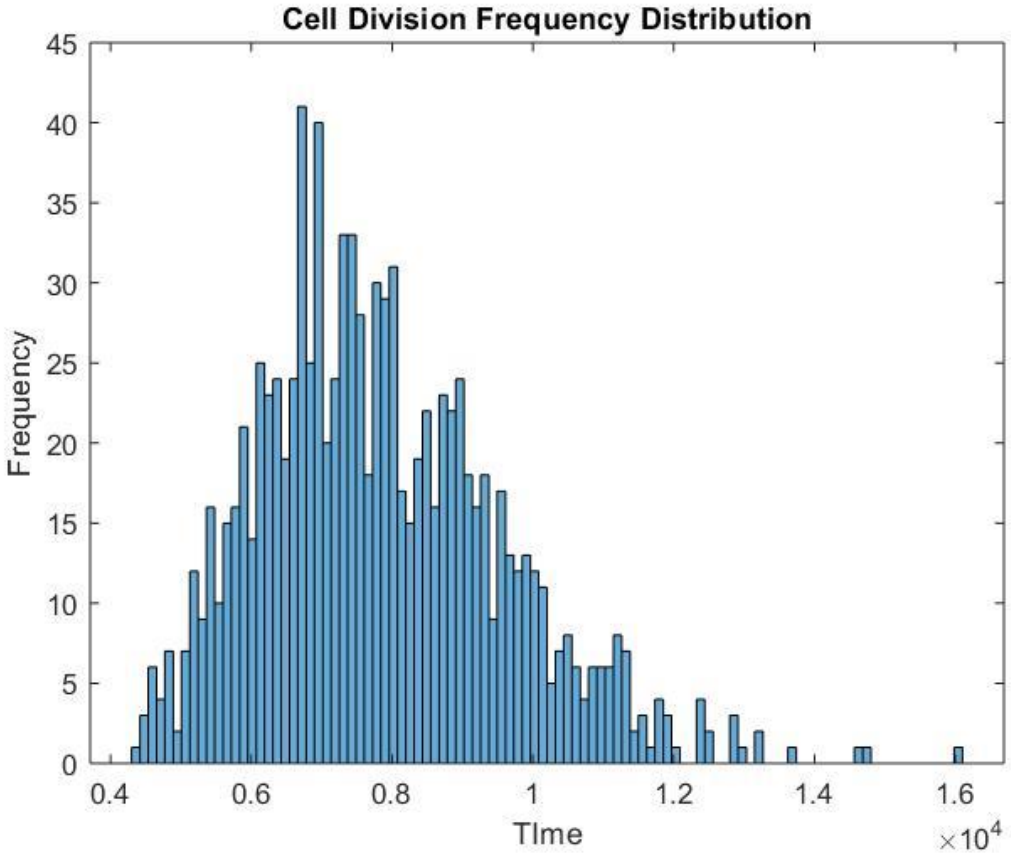


Figure 13: This figure is a histogram with custom specified 100 bins for frequency distribution to get more detail from the graph.

Analysis for Cross-Feeding Interaction

Wild Type Cell

For a Wild Type Cell, we take two proteins A and B. Protein A has average times (μ) to be 600 seconds for Δt_{OFF} and 100 seconds for Δt_{ON}. The arrival times also have a μ of 10 seconds for protein A. Protein B on the other hand has average times (μ) to be 700 seconds for Δt_{OFF} and 150 seconds for Δt_{ON}. The arrival times have a μ of 20 seconds for protein B. Both proteins have a translation rate as number of transcripts times 0.3 per second. Both of the proteins are produced by the Wild Type Cell. The cell division only happens when the protein levels of both A and B cross the threshold of 10^5 proteins. This only happens when the cell division times as per individual proteins A and B are computed using Single Cell Technique listed before and then the maximum of them is taken as the cell division time. 1000 instances of division times are computed to find out the average division time for the wild type cell in these circumstances. The code for this section is mentioned in Appendix C.

Result for Wild Type Cell

The coding simulation for the problem for 1000 instances gives us a mean cell division time for Wild Type Cells of 10069 seconds. The maximum cell division time in the 1000 instances was 20672 seconds, while the minimum time for the cell division in these instances was 5614 seconds. Fitting a Normal Distribution into the given data using the Matlab Statistics and Machine Learning Toolbox utility fitdist, we get a normal distribution with $\mu = 10069$ seconds and a $\sigma = 2062$ seconds. Fitting it with a Poisson Distribution gives a curve with $\lambda = 10069$ seconds.

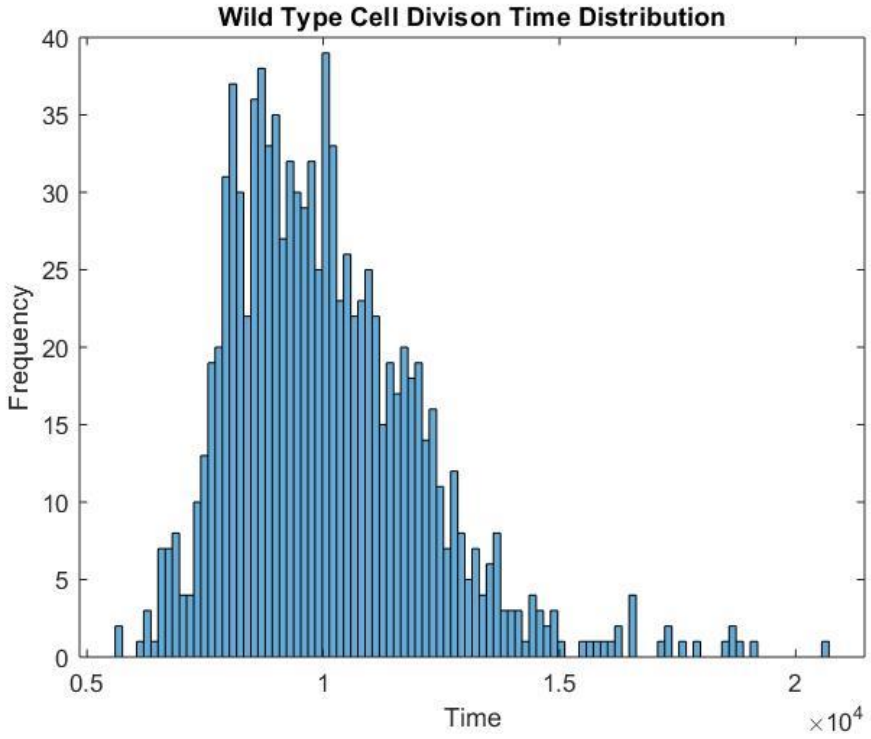


Figure 14: This figure represents the cell division distribution represented by using histograms for frequency distribution curve. The average division time was 10069 seconds.

Auxotrophic Cells

For Auxotrophic Cells we have two kinds of Auxotrophs—AuxA and AuxB. AuxA is capable of only producing protein A, and depends on AuxB for protein B. While AuxB is capable of producing only protein B and depends on Aux A for protein A. We assume that the transfer of proteins from one cell to the another is free, instantaneous, and without any time lag. Since we assume that there is no time lag in sharing and transfer of proteins, instead of solving for two separate cell cases and then doing the analysis, we solve for a single "large" cell, composed of AuxA and AuxB, where each of them produces their respective proteins, and need double the amount of protein required for a "normal" cell to reproduce. We here face the issue of considering what to change and tinker with. There are essentially 4 different variables that can be changed—$\mu(\Delta t_{OFF})$, $\mu(\Delta t_{ON})$, μ (waiting time), and average mRNA translation rates.

Case 1: Doubling the mRNA translation rates

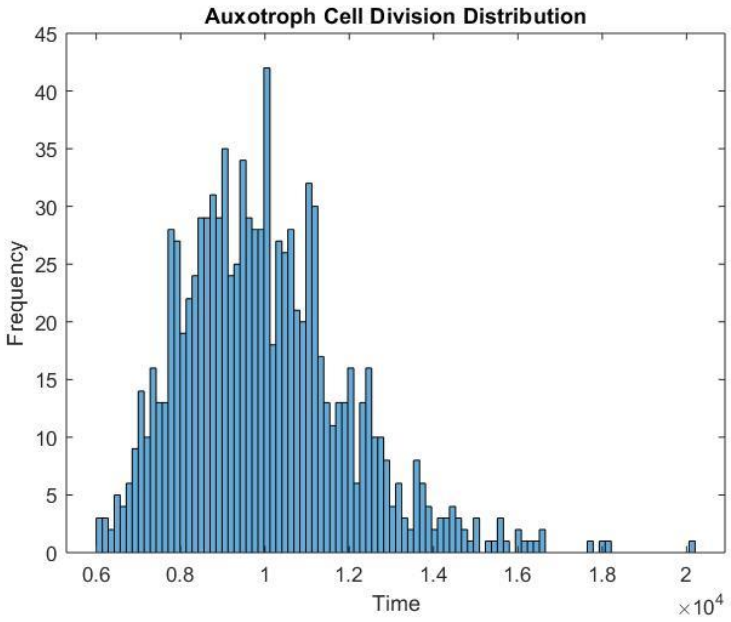


Figure 15: Auxotroph Cell Division Time Frequency Distribution. The average time of division for the cells was 9986 seconds with a relative fitness of 1.0083 over the WT cells.

If ribosomes for a given cell are limited and what were earlier used for production of two proteins are now used for production of only one protein, we double the protein production rates for the cells. This was done by doubling the protein production rates in the MATLAB code, and keeping everything else the same. The protein requirements for cell division were also doubled keeping in with our construct. This simulation was run for 1000 instances which gave a mean value for cell division in this case to be 9986 seconds. A normal distribution fit into data gave a distribution with μ= 9986 seconds and σ= 1982 seconds. This means that in this case the Auxotrophs do have a cell division time that is both less than the WT cells as well as have a less spread (Standard Deviation) than the WT cells. The relative fitness advantage to Auxotrophs compared to WT cells is calculated as the ratio of Malthusian Parameters of the cells. Using, this the Auxotrophs have a relative fitness which is 1.0083. This relative fitness increase seems to be too small.

Case 2: μ(waiting time) halves

Here we assume that transcription waiting times are reduced by half, but translation rates are the same for the Auxotrophs, and all the rest also being the same. We can rationalize this by thinking that ribosomes are already present in excess and hence Auxotrophs won't see any major increase by additional available ribosomes. But transcription, when in the ON state, has waiting times reduced to half. Let's try this hypothesis, by using the same code, and running it for 1000 instances. This gives us a mean cell division time of 10373 seconds, which is actually worse than the WT case.

Case 3: $\mu(\Delta t_{ON})$ doubles and μ(waiting time) halves

Here we assume that $\mu(\Delta t_{ON})$ doubles and waiting times are half that in the case of WT Cells. Other parameters stay the same. The simulation is run for 1000 instances. Finally, we get a mean cell division time that is 8034 seconds. The Normal Distribution that is fitted to the data gives a distribution with μ= 8034 seconds and σ= 1589 seconds. In this case the Auxotrophs have a relative fitness of 1.25 over the WT cells.

Case4: $\mu(\Delta t_{ON})$ doubles

Here we assume that $\mu(\Delta t_{ON})$ doubles, while all the other parameters stay the same. The simulation is run for 1000 instances. The mean cell division time here is 10729 seconds.

Case5: $\mu(\Delta t_{ON})$ doubles and $\mu(\Delta t_{OFF})$ halves

Here we assume that $\mu(\Delta t_{ON})$ doubles while $\mu(\Delta t_{OFF})$ halves. The mean cell division time here is 8251 seconds, which gives us a relative fitness that is 1.22 over and above the WT cells.

Results and Discussion

Transcription happens in bursts, where the ON state and OFF state are both exponentially distributed. The ON state usually is a smaller duration, where a burst of transcripts are made for a short while. While in the ON state, the transcripts arrive as a Poisson Counting Process. The waiting times between arrivals are exponentially distributed. This fact is used to model this Poisson Process in our MATLAB code.

We can clearly see that the cell can exhibit cross-feeding interactions with relative fitness increasing in many cases. In this study some cases were considered with varying the different parameters. The four parameters in our model that can be varied are the mean times of ON and OFF states of transcription which are exponentially distributed. The other parameters are the waiting times of arrival of transcripts in the Poisson Counting Process that happens in the ON state of transcription. The final parameter is the rate of translation of mRNA. We can clearly see that in many cases Auxotrophs gain substantial fitness advantage compared to the WT cells. All of this was concluded using computer simulations done using MATLAB codes given in the Appendices. The model assumes that there is no decay of mRNA, as well as protein. It also assumes a constant rate of protein production from mRNA, without considering the Poisson Process that translation can take place through. This assumption is however bolstered by the fact that there are multiple transcripts producing proteins, and the number of proteins is substantially high. More analysis needs to be done to find out about the reasons and mechanism behind cross-feeding interaction.

References

[1] Pande et al, Fitness and stability of obligate cross-feeding interactions that emerge upon gene loss in bacteria, The ISME Journal volume 8, pages 953–962 (2014)

[2] Real-Time Kinetics of Gene Activity in Individual Bacteria, Golding et al, Cell, Volume 123, Issue 6, 16 December 2005, Pages 1025-1036

Appendix A

Code for transcription events in a single cell:

```
t = 1; %timecounter
tm= 1:10000; %timematrix
transstate= 0; %transcriptionstate
transon=zeros(1,10000);  %To visualize trascription state
transcripts = zeros(1,10000);
transcriptevent= zeros(1,10000);
tramovav= zeros(1,10000);
proteins= zeros(1,10000);
proteinsevent= zeros(1,10000);
while t<20000
    if transstate == 0
        toff = t + random('Exponential',600);
        while t< toff
            transon(t)=0;
            transcriptevent(t)=0;
            t = t+1;
        end
        transstate = 1;
    end
    if transstate == 1
        ton = t + random('Exponential',100);
        while t< ton
            c = t;
            waitint= random('Exponential',10);
            d = c + waitint;
            while t<d
                if t > ton -1
                    break
                end
                transcriptevent(t)=0;
                transon(t)=1;
                t = t+1;
            end
            if t< ton-1
                transcriptevent(t)=1;
            end
            transon(t)=1;
            t = t+1;
        end
        transstate = 0;

    end

end
```

```
for k=1:length(transcriptevent)
    if k==1
        transcripts(k)=transcriptevent(k);
    else
        transcripts(k) = transcriptevent(k) + transcripts(k-1);
    end
end

tramovav = movmean(transcriptevent,100);

for k=1:length(transcripts)
    if k==1
        proteins(k)=transcripts(k)*0.3;
    else
        proteins(k) = (transcripts(k)*0.3) + proteins(k-1);
    end
    if proteins(k) < 10^5
        division = k;
    end
end

if proteins(length(transcripts)) > 10^5
    divided = 1;
else
    divided = 0;
end
```

Appendix B

Code for 1000 instances of a single cell division based on one protein:

```
celldivision = zeros(1,1000);
for f=1:1000

t = 1; %timecounter
tm= 1:10000; %timematrix
transstate= 0; %transcriptionstate
transon=zeros(1,10000);  %To visualize trascription state
transcripts = zeros(1,10000);
transcriptevent= zeros(1,10000);
tramovav= zeros(1,10000);
proteins= zeros(1,10000);
proteinsevent= zeros(1,10000);
while t<20000
    if transstate == 0
        toff = t + random('Exponential',600);
        while t< toff
            transon(t)=0;
            transcriptevent(t)=0;
            t = t+1;
        end
        transstate = 1;
    end
    if transstate == 1
        ton = t + random('Exponential',100);
        while t< ton
            c = t;
            waitint= random('Exponential',10);
            d = c + waitint;
            while t<d
                if t > ton -1
                    break
                end
                transcriptevent(t)=0;
                transon(t)=1;
                t = t+1;
            end
            if t< ton-1
                transcriptevent(t)=1;
            end
            transon(t)=1;
            t = t+1;
        end
        transstate = 0;

    end
```

```
end

for k=1:length(transcriptevent)
    if k==1
        transcripts(k)=transcriptevent(k);
    else
        transcripts(k) = transcriptevent(k) + transcripts(k-1);
    end
end

tramovav = movmean(transcriptevent,100);

for k=1:length(transcripts)
    if k==1
        proteins(k)=transcripts(k)*0.3;
    else
        proteins(k) = (transcripts(k)*0.3) + proteins(k-1);
    end
    if proteins(k) < 10^5
        division = k;
        celldivision(f)= division;
    end
end

if proteins(length(transcripts)) > 10^5
    divided = 1;
else
    divided = 0;
end

end
```

Appendix C

Code for Wild Type Cell for two protein dependent cell division:

```
celldivision1= zeros(1, 1000);
for f1=1:1000
t1 = 1; %timecounter
tm1= 1:10000; %timematrix
transstate1= 0; %transcriptionstate
transon1=zeros(1,10000);  %To visualize trascription state
transcripts1 = zeros(1,10000);
transcriptevent1= zeros(1,10000);
proteins1= zeros(1,10000);
proteinsevent1= zeros(1,10000);
while t1<20000
    if transstate1 == 0
        toff = t1 + random('Exponential',600);
        while t1< toff
            transon1(t1)=0;
            transcriptevent1(t1)=0;
            t1 = t1+1;
        end
        transstate1 = 1;
    end
    if transstate1 == 1
        ton = t1 + random('Exponential',100);
        while t1< ton
            c = t1;
            waitint= random('Exponential',10);
            d = c + waitint;
            while t1<d
                if t1 > ton -1
                    break
                end
                transcriptevent1(t1)=0;
                transon1(t1)=1;
                t1 = t1+1;
            end
            if t1< ton-1
                transcriptevent1(t1)=1;
            end
            transon1(t1)=1;
            t1 = t1+1;
        end
        transstate1 = 0;

    end

end
```

```
for k1=1:length(transcriptevent1)
    if k1==1
        transcripts1(k1)=transcriptevent1(k1);
    else
        transcripts1(k1) = transcriptevent1(k1) +
transcripts1(k1-1);
    end
end

tramovav1 = movmean(transcriptevent1,100);

for k1=1:length(transcripts1)
    if k1==1
        proteins1(k1)=transcripts1(k1)*0.3;
    else
        proteins1(k1) = (transcripts1(k1)*0.3) + proteins1(k1-
1);
    end
    if proteins1(k1) < 10^5
        celldivision1(f1)=k1;
        division1 = k1;
    end
end

if proteins1(length(transcripts1)) > 10^5
    divided1 = 1;
else
    divided1 = 0;
end
end

celldivision2= zeros(1, 1000);
for f2=1:1000
t2 = 1; %timecounter
tm2= 1:10000; %timematrix
transstate2= 0; %transcriptionstate
transon2=zeros(1,10000);  %To visualize trascription state
transcripts2 = zeros(1,10000);
transcriptevent2= zeros(1,10000);
proteins2= zeros(1,10000);
proteinsevent2= zeros(1,10000);
while t2<20000
    if transstate2 == 0
        toff = t2 + random('Exponential',700);
        while t2< toff
            transon2(t2)=0;
            transcriptevent2(t2)=0;
            t2 = t2+1;
        end
        transstate2 = 1;
    end
```

```
    if transstate2 == 1
        ton = t2 + random('Exponential',150);
        while t2< ton
            c = t2;
            waitint= random('Exponential',20);
            d = c + waitint;
            while t2<d
                if t2 > ton -1
                    break
                end
                transcriptevent2(t2)=0;
                transon2(t2)=1;
                t2 = t2+1;
            end
            if t2< ton-1
                transcriptevent2(t2)=1;
            end
            transon2(t2)=1;
            t2 = t2+1;
        end
        transstate2 = 0;

    end

end

for k2=1:length(transcriptevent2)
    if k2==1
        transcripts2(k2)=transcriptevent2(k2);
    else
        transcripts2(k2) = transcriptevent2(k2) +
transcripts2(k2-1);
    end
end

tramovav2 = movmean(transcriptevent2,100);

for k2=1:length(transcripts2)
    if k2==1
        proteins2(k2)=transcripts2(k2)*0.3;
    else
        proteins2(k2) = (transcripts2(k2)*0.3) + proteins2(k2-
1);
    end
    if proteins2(k2) < 10^5
        division2 = k2;
        celldivision2(f2)=division2;
    end
end

if proteins2(length(transcripts2)) > 10^5
```

```
    divided2 = 1;
else
    divided2 = 0;
end
end

celldivision= zeros(1, 1000);
for f= 1:1000
    celldivision(f)= max(celldivision1(f),celldivision2(f));
end
```