How it works

Thursday, January 13, 2022 3:34 PM

General overview:

This algorithm takes in cell cycle parameters and spits out genome copies and gene copies at the time points of interest.

By using a few assumptions detailed at the beginning of the next section, we simply count the number of ongoing C+D periods, which of these have ongoing C periods, and sum their respective growths.

Detailed overview:

This algorithm assumes:
- The gene replication events in the current cell cycle are the same as the events in the next cell cycle.
- During the cell cycle you have a **C** period of linear genome doubling which is followed by a **D** period, at the end of which there is cell division and the start of the next cell cycle

The algorithm begins by accepting a few cell cycle parameters:
- The time points of interest
- The cell doubling time
- The gene of interest location on the genome
- The location of oriC on the genome
- The length of the genome
- The C period and the D period

1

We then calculate:
- The number of cell cycles that a full C+D period will be in: **generations_crossed**
  - **generations_crossed = floor((C+D) / tau) + 1;**
- The amount of time left in a C+D period that does not cross a full cell cycle: **hanging_previous**
  - **hanging_previous = (C+D) - floor((C+D) / tau) * tau;**
- The amount of time left in a C period that does not cross a full cell cycle NOT including hanging_previous: **hanging_next**
  - **hanging_next = (C - min(C, hanging_previous)) - floor((C - min(C, hanging_previous)) / tau) * tau;**
- The number of cell cycles that a C period fully covers: **completely_covered**
  - **completely_covered = (C - min(C, hanging_previous) - hanging_next) / tau;**

2

Focusing on the growth of the genome, we first calculate the maximum initial rate of growth during the cell cycle:
- To reminder the reader of the assumptions of the algorithm:
  - Because a cell cycle ends with a D period, there must be a start of a D period within that same cycle to conserve D period number. An end of a C period is a start of a D period. Conserving C period number means there should be a single start of a C period. After a C period ends the number of genome copies should double. We approximate this by having linear growth.
- Doubling during the C period => base slope of 1 / C, then 2/C, then 4/C, etc. increasing for each concurrent C period:
  - **Initial slope = sum(2.^[max(generations_crossed - 2 - completely_covered, 0) : generations_crossed - 2] ./ C);**
  - This basically says that the maximum initial slope should be equal to the sum of slopes going from all the ongoing C periods to the current C+D cycle.
  - Note that something like **[0:-1]** returns an empty vector, making the initial slope = 0
  - To understand the above line of code, the reader should convince themselves that if the generations crossed is

    only 1, then there is never an ongoing C period when the cell cycle starts.
  - Note that if we have a case where **(generations_crossed - 2 - completely_covered) < 0**, it will be dealt with later

Next we want to determine if and how much the hanging C periods overlap (from the hanging previous and hanging next):

```
        overlap = min(tau - (hanging_previous + hanging_next), 0);
```

3

Now we start to calculate the times at which a C period starts and ends and the slopes for each. In other words we will calculate the three sections of growth. We do this for 3 cases:

- No overlap:
  - **t_1 = hanging_next;**
  - **t_2 = tau - hanging_previous:**
  - **if 0 <= (generations_crossed - 2 - completely_covered)**
    - The slope will go from maximum, to having hanging_next end, to having hanging_previous begin.
    - **Slope = [slope, …**
      ```
                      slope - 2^(generations_crossed-2-completely_covered)/C, ...
                      slope - 2^(generations_crossed-2-completely_covered)/C + 2^(generations_crossed-1)/C];
      ```
    - The time divisions for each section should be as follows where our genome will be determined by:
      - genome = Slope * t_div + initial amount of genome
      - **t_div = [min(hanging_next, t); …**
        ```
                    min(max(0, tau-hanging_previous-hanging_next), max(0, t-hanging_next)); ...
                    min(hanging_previous, max(0, t-(tau-hanging_previous)))];
        ```
    - The intercept should be calculated based on the amount of genome created by the end of the cell cycle:
      - initial amount of genome = Slope * t_max:
      - **t_max = [min(hanging_next, tau);**
        ```
              min(max(0, tau-hanging_previous-hanging_next), max(0, tau-hanging_next)); ...
              min(hanging_previous, max(0, tau-(tau-hanging_previous)))];
        ```
  - **ELSE:**
    - The slope will go from the starting value, to having hanging_next end (here hanging_next cannot be present), to having hanging_previous begin. If the C period begins and ends in the same cell cycle, the whole period is considered hanging_previous. Time periods are the same
    - **Slope = [slope, slope, slope + 2^(generations_crossed-1)/C];**
- If there is overlap:
  - **t_1 = tau-hanging_previous;**
  - **t_2 = hanging_next;**
  - The slope should simply go from the starting value, to having hanging previous begin, and to having hanging next end
  - **Slope = [slope, …**
    ```
                slope + 2^(generations_crossed-1)/C, ...
                slope - 2^(generations_crossed-2-completely_covered)/C + 2^(generations_crossed-1)/C];
    ```
  - The time divisions are as follows and written above:
  - **t_div = [min(tau-hanging_previous, t); …**
    ```
               -max(overlap, min(0, -t+(hanging_next+overlap))); ...
               min(hanging_previous, max(0, t-(tau-hanging_previous-overlap)))];
    ```
  - **t_max = [min(tau-hanging_previous, tau); …**
    ```
                -max(overlap, min(0, -tau+(hanging_next+overlap))); ...
                min(hanging_previous, max(0, tau-(tau-hanging_previous-overlap)))];
    ```

Genome copy number can now be calculated as:

```
        Geno = (Slope * t_div) + (Slope * t_max);
```

4

Moving onto calculating the gene copy number we do the same exact process but count the number of ongoing C+D periods who have finished replicating our gene of interest.

```
speed = (L/2)/C;
distance = min(abs(OC - X), abs(L-OC+X));
time = distance/speed;
C_prime = C - time+D;
```

5

We can effectively ignore C period time before our gene copy replicates, and so we work with C_prime now and count the number of ongoing C_prime+D periods.

```
hanging_previous = (C_prime) - floor((C_prime)/tau)*tau;
hanging_next = (C_prime-hanging_previous) - floor((C_prime-hanging_previous)/tau)*tau;
completely_covered = (C_prime - hanging_previous - hanging_next)/tau;
gene_time = tau-hanging_previous;
gene = 2.^(completely_covered + ceil(min(1, max(0, t-tau+hanging_previous))));
```