Plasmid MILP old formulation

Aniket Mane

February 2019

PlasBin is an MILP-based hybrid approach for plasmid binning. Here, we aim to obtain plasmid bins as walks in the assembly graph and design an MILP formulation accordingly.

Input:

- Contigs,
- Edges,
- For each contig v, the following information is known
  - Gene density gd_v,
  - GC content gc_v,
  - Read depth of contig rd_v,
  - Length of contig len_v.

Output:

- Ordered sets of contigs defining a walk in the assembly graph

Idea:
An ideal plasmid bin will consist of a set of contigs that have similar read depth, similar GC content and high density of plasmid genes. In this formulation, the plasmid bin will be defined by the walk that optimizes an objective function based on the above-mentioned criteria.

Consider an assembly graph G. Every contig u in G has two extremeties, a head and a tail, denoted by u_h and u_t for a contig u. The edges of the assembly graph are pairs of contig extremeties. For instance, an edge between u_h and v_h is represented as (u_h, v_h). We associate a binary variable each with every vertex, extremity and edge in the assembly graph to denote if they are part of the solution. Thus, for a vertex v, $x_v = 1$ if v belongs to the solution and 0 otherwise. Similarly, for an extremity v_y, $x_{v_y} = 1$ if v belongs to the solution and 0 otherwise. For an edge e, $x_e = 1$ if e belongs to the solution and 0 otherwise.

Modified assembly graph:
For every contig v, we make $\lceil rd_v \rceil$ copies of v, $v_1, v_2, ..., v_{\lceil rd_v \rceil}$, each with read depth 1. We add all edges incident on v to these copies.

Variable	Type	Description
x_c	Binary	$x_c = 1$ if node $c \in p$
$x_{c,ext}$	Binary	$x_{c,ext} = 1$ if extremity ext of contig $c \in p$
y_e	Binary	$y_e = 1$ if edge $e \in p$
GC_c	Continuous	$GC_c = gc_c$ if $c \in p$, 0 otherwise
S_c	Boolean	$S_c = s_c$ if $c \in p$, False otherwise
L_c	Integer	$L_c = \ell_c$ if $c \in p$, 0 otherwise
L_p	Integer	$L_p = \sum_{c \in p} \ell_c$
MGC	Continuous	$\sum_{c \in p}(gc_c * \ell_c)/L_p$

Table 1: Decision variables used in the MILP; p refers to the computed path.

Objective function: We formulate the MILP as a minimization problem, that aims to find a path in the assembly graph that minimizes a linear combination of the (negated) plasmid gene density and of a term measuring the deviation of %GC content across the contigs forming the path.

- Gene density: The aim is to maximize the plasmid gene density over the whole path. The gene density of a contig is a number between 0 and 1 so, to account for the high variability of contigs length, we weight the gene density of selected contigs by their lengths and integrate the term $-\sum_c(gd_c)*L_c$ in the objective function (the sum is over all contigs).
- %GC content deviation: We aim to minimize the sum of the deviations between the %GC content of selected contigs and the mean %GC content of the whole path. This feature was shown in HyAsP to contribute greatly to the method accuracy. Similarly to the gene density, we weight the %GC of each contig in the path by its length and incorporate the term $\sum_c(|MGC - gc_c|)*L_c$ in the objective function. Here, MGC is the mean %GC content of the path.

The objective is to minimize a linear combination of the two terms above, with each term being weighted equally.

$$\sum_c(|MGC - gc_c|) * L_c - \sum_c(gd_c) * L_c$$

An important remark is that, if we were to exclude the %GC content term from the objective function, the problem to address would be a shortest path problem, with the additional constraint that the optimal path should contain a seed. This would be amenable to efficient shortest paths algorithms. However, as mentioned above, the %GC content term is important toward selecting paths

that are more likely to originate from plasmids, which makes the optimization problem we address more complicated.

Constraints: We provide now an overview of the constraints required to solve the problem of finding a path in the assembly graph that is optimal with regard to the objective function.

1. Constraint to ensure the presence of a seed in the path: This is done by requiring that the sum of the binary decision variable x_c over all seed contigs is at least 1.

2. Constraints to handle variables in the denominator: In order to minimize %GC content deviation, we also need to compute the mean %GC content of the path, MGC. The mean %GC content of a plasmid is given by:

$$MGC = \frac{\sum_c gc_c * \ell_c * x_c}{L_p}$$

   This is not a linear constraint as L_p is a variable in the denominator. However, it is possible to obtain a linear relaxation for the above constraint through the use of McCormick envelopes. This is possible since x_c is a binary variable and thus, has definite upper and lower bounds.

3. Constraints to handle absolute values: The objective function contains the absolute value of the difference between the mean %GC of the path (MGC) and individual %GC (GC_c). This in itself is not linear. So, we introduce a difference variable d_c which is the absolute value of the difference between the mean %GC and contig c %GC for a specific path. For each contig we add the following constraints:

$$d_c >= MGC - GC_c$$

$$d_c >= GC_c - MGC$$

4. Constraints to define a path: In every iteration, we want the MILP to output a path. This is the most challenging aspect of our formulation, as it is notorious that modelling the search of an arbitrary path in a graph through a polynomial number of linear constraints is not possible. To handle this issue, we start from degree constraints that enforce that every contig extremity belongs to at most one selected edge. However, this leads to the possibility that the generated solution consists of disjoint components, all but one of which are cycles. Thus, we would like to add constraints to eliminate potential cycles. To address this issue, we use the delayed constraint generation method: whenever the solution found by the MILP contains cycles, we add constraints explicitly forbidding these cycles and repeat. This results into an iterative process that goes on until we obtain a singular component defining a path as our solution. A possible

scenario in this approach is an exponential number of iterations. To avoid this, we bound the number of iterations.

5. Constraints to define the contigs and edges of the walk:
   An edge is a part of the solution only if both its endpoints are part of the solution. For an edge $e = (u_x, v_y)$

$$x_e \leq x_{u_x}$$

$$x_e \leq x_{v_y}$$

   A contig is part of the solution if at least one of its extremities is part of the solution.

$$x_u \geq x_{u_h}$$

$$x_u \geq x_{u_t}$$