**1. BACKGROUND**

Synthetic biologists face the challenge of specifying a cellular system behavior, designing a synthetic genetic regulatory network (sometimes called a genetic circuit) from DNA that will create that behavior and physically constructing the genetic circuit that will be inserted into cells. This often proves to be a difficult task, as the complete function of many DNA parts is not well understood and even the best network designs prove to pose difficulty in both assembly and design refinement. Currently, the synthetic biology community has placed focus upon improving DNA assembly methods – how to make the assembly of genetic circuits faster, cheaper and more reliable. One popular assembly method, BioBricks**™** [1], involves assembly of two parts into a larger composite part. Recent work has been done to develop algorithms that optimize BioBrick**™** assembly [2].

Unfortunately, these algorithms do not currently account for any biological attributes of parts used to compose a genetic circuit, which can be important in choosing the best assembly plan for a properly functioning circuit. By accounting for relevant biological data of parts, one can devise an assembly plan that maximizes the testability of newly assembled composite parts. Here, we present algorithms based upon the SDS algorithm [2], which have a scoring function that optimizes the assembly plan for structural and functional testability. The algorithms also have an additional pre-processing step in which they are trained with a large set of future goal parts to discover the most common design motifs among these parts. The final assembly plan is biased towards these motifs to maximize future reusability of intermediate parts.

**2. STRUCTURAL TESTING**

The structural model of testing in electrical engineering assumes that within a circuit, all small parts have known function and function properly [3]. Thus, as long as all parts are present and connected properly, the final circuit should function as specified. All administered 

FIGURE 1: A) A digital logic representation of a 3-input NOR gate B) An electrical transistor representation of a 3-input NOR gate. Structural testing tests that all wires are connected properly and that all CMOS transistors and wires are undamaged. C) A genetic regulatory network representation of a 3-input NOR gate. Structural testing tests that the DNA parts are in the correct order and are not mutated.

tests are to ensure that all parts are properly connected and undamaged.

In genetic engineering, structural tests are defined as tests that determine if all DNA parts have the correct sequence and are in the correct order. This information can be acquired through DNA sequencing, but for assembly with many intermediates, sequencing every intermediate during assembly is too costly and time consuming compared to other methods. Another structural testing method, restriction mapping, is more practical for this purpose because it is cheaper and faster than sequencing. Restriction mapping is a very practical structural test for intermediates, but has a constraint – two connected parts with a combined length of fewer than 200 bp is difficult to resolve on a gel. Therefore, we define any two parts with a combined length of fewer than 200 bp as structurally untestable. For any structurally testable intermediate, an assembly step that results in a correct restriction map passes the structural test.

From a practical standpoint, structural testing is the most important heuristic in DNA assembly; if two parts are out of sequence they will not work and all additions to this part also will not work. Therefore, this is the strongest constraint in the algorithm’s scoring function and composite parts that cannot be structurally tested (size fewer than 200 bp) will be scored less favorably and might not be assembled. Additionally, parts that do not experimentally pass structural testing will not be used and the assembly step will be repeated until a successful outcome of the structural test.

**3. FUNCTIONAL TESTING**

Unlike the structural model, the functional model of circuit testing does not assume that all parts have a well-defined and proper function. Instead, functional testing aims to validate that a circuit functions as specified [3]. Genetic circuit assembly intermediates are ideal targets for functional testing because they are small functional units with poorly defined or unreliable function. Since genetic circuits are often designed to produce a number of functional proteins that interact with other genetic parts, here we define a transcriptional unit that takes proteins as input and produces proteins as output as the primitive functional element to be tested. This function can be tested when a system is designed such that the protein produced is a fluorescent protein, which can be detected by a flow cytometer. If a fluorescent protein is detected as being expressed at the specified level, then it passes the functional test. If a part or series of parts fail their functional test, they will not be used and the design will be refined to include other parts.

The algorithm will therefore bias the assembly plan towards constructing transcriptional units as soon as possible. Once these testable functional units are built and pass a functional test, they will again be tested with the addition of each new part, as a sort of “bootstrap”

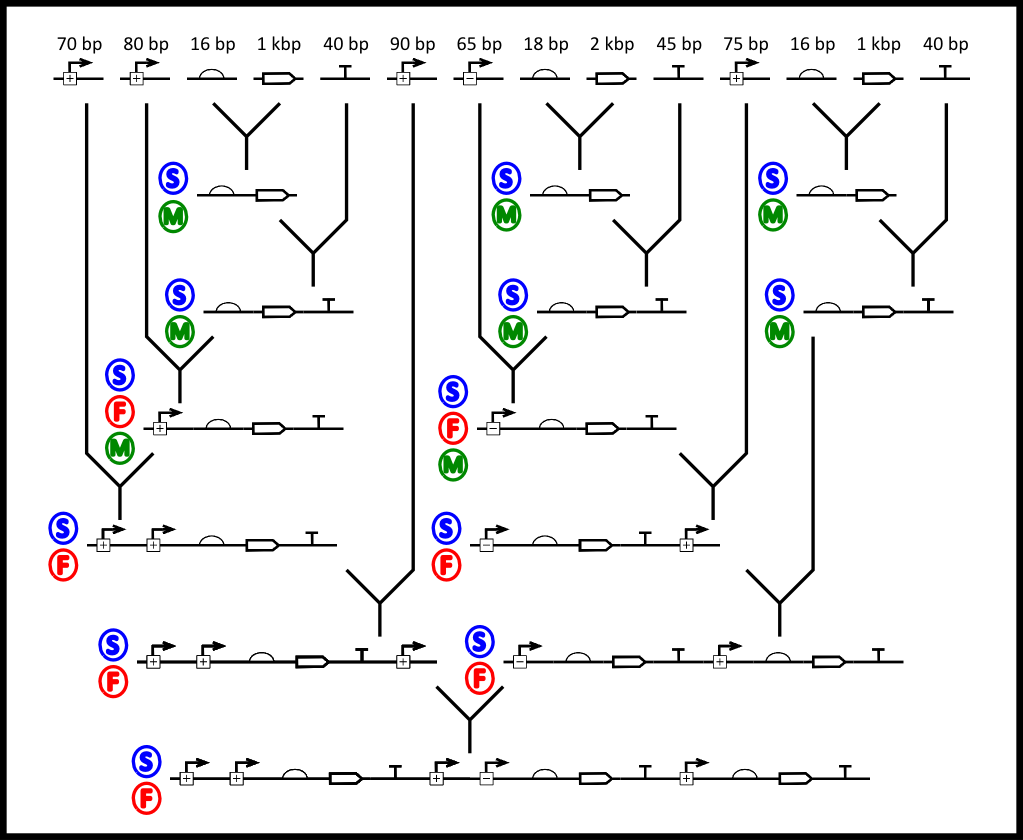


FIGURE 2: An optimal assembly plan for the circuit in Figure 1.C. A motif training step through the database finds that RBS-Gene, RBS-Gene-Term and Prom-RBS-Gene-Term are common motifs (M). All intermediates are structurally testable (S) (>200bp) and there is a 7/13 coverage of functionally testable intermediates.

method to track the composite part’s function as more parts are added. Unlike with structural testing, however, it is not mandatory that all parts be functionally tested, and so the algorithm’s scoring function has a trade off between the percent of functionally testable intermediates and assembly stages required, as a complete “bootstrap” bias would result in an

unreasonable amount of assembly stages for large parts. The overall testability of an assembly plan will be percent coverage – (number of testable steps/total steps).

**4. MOTIF TRAINING**

To minimize the number of assembly steps and maximize the reusability of intermediates, our algorithm analyzes a large database of parts to identify part type motifs (i.e. RBS-Gene). The algorithm then biases the assembly plan solution towards the most common part motifs. This extra step adds an additional element to the SDS algorithm, which already biases for part sharing. In addition to selecting intermediates that are most shared amongst the set of immediate goals parts, it will select intermediates that are common for all parts in a large database.

**5. CONTRIBUTIONS**

These additions to the SDS algorithm improve the practicality of the algorithm by taking important biological features of DNA parts into consideration. These modifications should make using algorithms for assembly planning more useful for biologists. Furthermore, we provide a systematic method for improving a biologist’s ability to test genetic circuits throughout the assembly process and refining the design in the middle of a large construct’s assembly. This is common practice for a biologist assembling genetic circuits manually, but this represents one of the first known attempts to automate this process.