Funding Scheme: THEME [ICT-2007.8.0] [FET Open]

Paving the Way for Future Emerging DNA-based Technologies: Computer-Aided Design and Manufacturing of DNA libraries

Grant Agreement number: 265505

Project acronym: CADMAD

Deliverable number: D2.4 + D2.5

Deliverable name: Report on the features of advanced construction planning algorithms & Report on the features of version 1.0 of algorithm for DNA construction planning

Contractual Date[1] of Delivery to the CEC: M24
Actual Date of Delivery to the CEC: M25
Author(s)[2]: Tuval Ben-Yehezkel
Participant(s)[3]: WEIZMANN
Work Package: WP2
Security[4]: Pub
Nature[5]: R
Version[6]: 0.0
Total number of pages:

1 As specified in Annex I

2 i.e. name of the person(s) responsible for the preparation of the document

3 Short name of partner(s) responsible for the deliverable

4 The Technical Annex of the project provides a list of deliverables to be submitted, with the following classification level:
Pub - Public document; No restrictions on access; may be given freely to any interested party or published openly on the web, provided the author and source are mentioned and the content is not altered.
Rest - Restricted circulation list (including Commission Project Officer). This circulation list will be designated in agreement with the source project. May not be given to persons or bodies not listed.
Int - Internal circulation within project (and Commission Project Officer). The deliverable cannot be disclosed to any third party outside the project.

5 **R (Report):** the deliverables consists in a document reporting the results of interest.
P (Prototype): the deliverable is actually consisting in a physical prototype, whose location and functionalities are described in the submitted document (however, the actual deliverable must be available for inspection and/or audit in the indicated place)
D (Demonstrator): the deliverable is a software program, a device or a physical set-up aimed to demonstrate a concept and described in the submitted document (however, the actual deliverable must be available for inspection and/or audit in the indicated place)
O (Other): the deliverable described in the submitted document can not be classified as one of the above (e.g. specification, tools, tests, etc.)

6 Two digits separated by a dot:
The first digit is 0 for draft, 1 for project approved document, 2 or more for further revisions (e.g. in case of non acceptance by the Commission) requiring explicit approval by the project itself;
The second digit is a number indicating minor changes to the document not requiring an explicit approval by the project.

Abstract

Most of CADMAD users' libraries consist of existing sequences as the inputs, sub-sequences of these, mutated variants and novel synthetic sequences are defined as intermediates, and the desired products of their recombination will be the outputs.

Keywords[7]:

DAWG, LCS, Suffix tree

CADMAD's Libraries construction can be optimized in two major aspects, minimizing the cost of the construction tree of given DNA fragments, and optimizing the DNA fragments themselves to optimize both the possible construction plan and the cost of required synthetic DNA.
We will present:

- Nested data model for library representation
- Optimal DAWG construction
- Pairing algorithm
- Synthetics optimization

1. Implementation

For the most part, non-synthetic DNA fragments are better left as defined by the user since they can be obtained in any form by amplifying a given NF. When DNA fragments are not supplied by the user or are too short to justify their amplification from a NF, they are regarded as synthetic and can be subjected to additional optimization.

Definitions:
Ref – a logical reference to a user defined DNA sequence.
Sub – a logical list of Refs describing a DNA sequences comprised of refs concatenations
Sequence – a DNA sequence requested by the user as a library target
DAWG - Directed Acyclic Word Graph.
Nodes Pairing - join two consecutive nodes within the DAWG-like graph

Library data model

[7] Keywords that would serve as search label for information retrieval

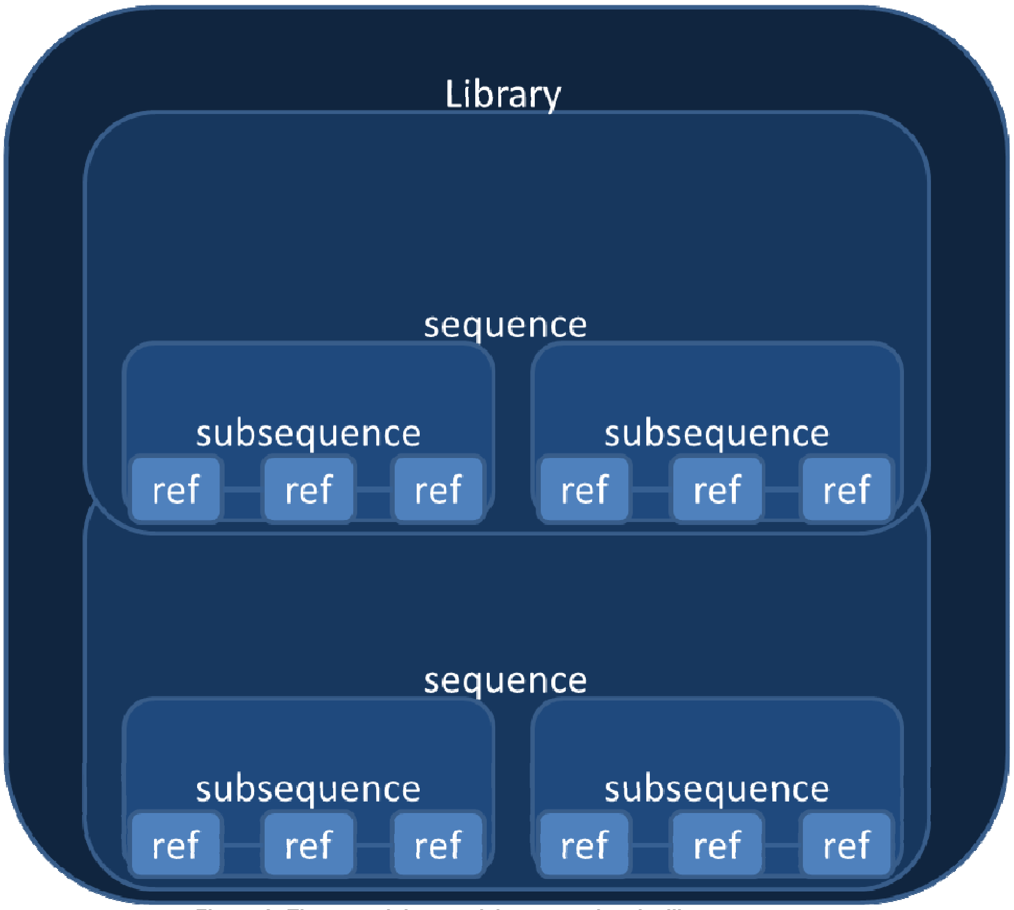

Figure A: The nested data model representing the library

Optimal DAWG insertion

The common DAWG graphical data model is perhaps the best form of textual compression that does not impair access time. To enjoy its benefits without impairing the human designer's context, an altered DAWG is proposed, using all the subsequences within the library's context as its alphabet. Using this modified DAWG, we can enjoy the best of both worlds. We have the human designer's context decorating the nested objects as well as the space efficiency and quick access times of the DAWG data model.

The DAWG like graphical library representation is constructed in linear runtime (linear to the number of refs resulting from the user's design) according to the principles described in reference [1].

Iterative LCS

When assembling the synthetic fragments of the library, the parts and objects described by the user should have no implication on the construction process. The idea is to optimize the library's synthetic fragments description for by minimizing the number and length of required oligos.

A suffix tree based LCS solving algorithm, finds the longest common substring among a set of strings in $\Theta(n_1+\ldots+n_k)$ runtime. Our proposed optimization works by extracting all the non-branching sub-sequences in the synthetics DAWG, use them as input to the LCS solver, slice the resulting LCS instances out of the nodes containing it, iterate.

The DAWG compression can be further optimized by the iterative LCS by recognizing two identical sub-sequences in two or more different nodes and logically unifying them.

Nodes pairing operation

A pairing operation will join two consecutive nodes within the DAWG-like graph. This in turn, will translate to a construction step, concatenating two DNA sequences.

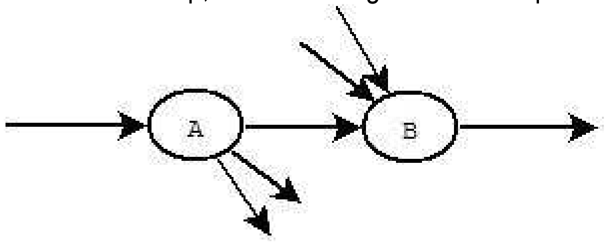

Figure B: consecutive nodes A, B

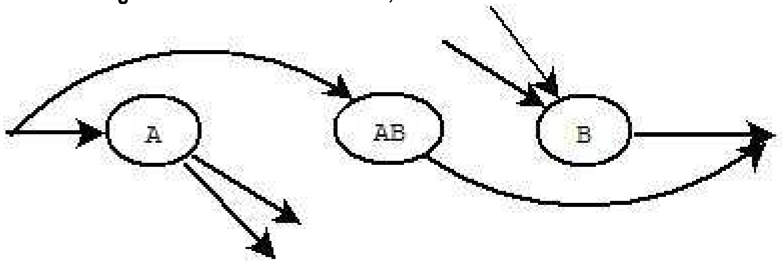

Figure C: concatenated nodes A, B

In many cases there are no additional edges coming out of A or into B so the original nodes or just one of them can be deleted and completely replaced by the new AB node.

```
pair(G, node_a, node_b):
    node_ab = create_node(node_a.name + '_cat_' + node_b.name, *node_a.refs + node_b.refs)
    G.add_node(node_ab)
    G.remove_edge(node_a,node_b)
    for pred in G.predecessors_iter(node_a): #copy all the edges coming into A into the new AB
        G.add_edge(pred, node_ab)
    for succ in G.successors_iter(node_b):  #copy all the edges coming out of B out of the new AB
        G.add_edge(node_ab, succ)
    if not G.in_degree(node_b):
        G.remove_node(node_b)
    if not G.out_degree(node_a):
        G.remove_node(node_a)
```

Assembly graph based on pairing operations:

We score a pair of nodes by the number of source to sink routes that pass through that pair. In each iteration, we pair the highest scoring pair based on that score, until we’re left with an explicit set of sequences that is the full library. Every pairing operation is documented as a Y in the emerging construction tree.

```
pairing(G):
    construction_tree = nx.DiGraph()
    while G.edges():
        node_a, node_b = get_highest_scoring_pair(G)
        pair(G, node_a, node_b)
        construction_tree.add_node(node_ab)
        construction_tree.add_edge(node_a, node_ab, direction='left')
        construction_tree.add_edge(node_b, node_ab, direction='right')
    return construction_tree
```

Optimizing assembly of combinatorial Cartesian product
When assembling a set by set type of combinatorial concatenation, finding the minimal set of primers that spans the full required combinatorial space results allows for some impressive savings in the price and number of required oligos.

Incorporating short synthetic nodes within primers:
The most expensive metric we take into account when assessing a construction plan is the depth of the concatenations tree. Each construction tree generation is error prone and can take anywhere between a day and a full week, even when manufacturing in full power.
When designing a library's construction plan, we try to incorporate as many short synthetic fragments as possible into their neighbouring nodes' primers. These small savings add up to saving full generations in the resulting construction tree, drastically reducing its duration and cost.

2. Results

Significant reduction in the number and length of required synthetic DNA fragments was achieved using the described algorithms as presented in report 2.8.

3. Conclusions

We have incorporated a few data structures and algorithms from the world of stringology with a few novel point optimization algorithms and an inclusive planning algorithm. When examining CADMAD users' libraries using our new methods, we've unravelled some major optimization possibilities, unknown to the user and very hard to find for the manual planner.
The point optimization algorithms are mostly optimal but the same is left to be proven regarding the pairing algorithm. A major issue when addressing the pairing problem is exactly defining the optimal solution while it changes wildly with different manufacturing technologies, oligos pricing and human resources.Future work will extend the described algorithm with generic construction tree assessment that will allow the required degree of freedom for this quickly changing field.

4. References

[1] Construction of minimal, deterministic, acyclic finite-state automata from sets of strings:
http://www.eti.pg.gda.pl/katedry/kiw/pracownicy/Jan.Daciuk/personal/adfa.html

5. Abbreviations

List all abbreviations used in the document arranged alphabetically.

DAWG	Directed acyclic words graph
LCS	Longest common substring
NF	Natural fragment (DNA given by the user)
Oligo	DNA oligonucleotide