Constraint Satisfaction Problems on DNA Strings

Luca Bortolussi\textsuperscript{1} \hspace{1cm} Andrea Sgarro\textsuperscript{2}

\textsuperscript{1}Department of Mathematics and Computer Science
University of Udine, Italy.

\textsuperscript{2}Department of Mathematics and Computer Science
University of Trieste, Italy.

WCB 2005, Sitges, 5\textsuperscript{nd} October 2005
1. Introduction
2. Symbolic Representation of Constraints
3. The Branch and Bound Algorithm
4. Results
5. Conclusions
Biological Background

- DNA stores all the information needed to build a complex living being.
- Since 1994, it has been used as a computational entity: **DNA computing**.
- The basic idea of DNA computing is to encode a given problem in a set of DNA strings, let these strings interact together and “read” the output.
- The main operation happening between those strings is the **hybridization** (formation of double helix from complementary strings). It can have errors.
- One needs good set of strings (that don’t self-hybridize): **DNA word design**.
DNA strings are oriented strings over $\Sigma = \{a, c, g, t\}$. Fixing the length $n$, and taking $x, y \in \Sigma^n$ we consider the following distances:

- $d_H(x, y)$, the usual Hamming distance (number of positions in which $x$ and $y$ differ);
- $d_{RC}(x, y) = d_H(x, y^{RC})$, the reverse complement distance, which is the distance between $x$ and the reverse complement of $y$.

A subset $C \subset \Sigma^n$ is a DNA code with threshold $D$ iff
\[ \forall x \neq y \in C, \ d_H(x, y) \geq D \text{ and } d_{RC}(x, y) \geq D, \text{ and also } d_{RC}(x, x) \geq D. \]

We look for maximal size codes.
We want to develop a constraint-based algorithm for the DNA word design problem, both for the CSP version (find a code of given length) and for the optimization version (find the optimal code).

There are several problems in trying to do this:

- **The search space is enormous**: for strings of length $n$ and codes of size $m$ it has size $4^{mn}$ (for $m = 50$ and $n = 10$ is $10^{300}$!).
- **The domain of each variable is exponential in $n$**, i.e. has initial cardinality $4^n$. We need a way to store those values, and a way to propagate new constraints.
- **The problem itself is full of symmetries**, not always obvious.
To represent effectively set of strings we use a symbolic representation of their characteristic function, by means of (a variation of) OBDD.

OBDD (Ordered Binary Decision Diagrams), are directed acyclic graphs used to represent compactly boolean functions, especially in symbolic model checking.
OBDD

- They can have exponential size in the number of variables, but on average (if the function is sufficiently structured) they perform well.

- Fixing the order of variables, they have a canonical representation, and the construction of the product (intersection) of two OBDDs can be computed in polynomial time.
For representing sets of DNA strings, we used a slightly different version (GDD), where from each node we can have 4 exiting edges, labeled with the DNA letters. Here a path from the root to terminal node 1 is labeled with a string belonging to the represented set.
Computational Issues

- We can represent all the constraints using GDD. The propagation algorithm thus becomes the construction of the product of two such GDD. This propagation is globally consistent.

- The dimension of such GDD in principle can explode, especially after combing them. But in practice we see an initial increase in complexity in size, which drops quickly after some steps (the represented subset of strings becomes inherently simple).
Search and Propagation

- Suppose we want to solve a CSP problem, looking for a code of size $m$ for strings of length $n$.
- We have $m$ variables $\{X_1, \ldots, X_n\}$, and the constraints $d_H(X_i, X_j) \geq D$ and $d_{RC}(X_i, X_j) \geq D$ for all $i < j$.
- At initial stage all variables have domain $\Sigma^n$. Suppose we instantiate the variable $X_1$ with the string $s_1$. Then all the domains of variables $\{X_2, \ldots, X_n\}$ are reduced, as they have to satisfy $d_H(s_1, X_j) \geq D$ and $d_{RC}(s_1, X_j) \geq D$.
- To represent them we construct the GDD for the set $d_H(s_1, X) \geq D$, the one for the set $d_{RC}(s_1, X_j) \geq D$, and we intersect them computing their product.
Search and Propagation

- Every time we instantiate a new variable, we do the same construction, and we intersect it with the GDD representing the intersection of all previously posted constraints.

- The selection of a new string is performed by choosing a string accepted by the current GDD. This is done by visiting the unfolding of such graph.
Search and Propagation

Heuristics

- To break symmetry under permutation of variables, we impose a lexicographic ordering of variables: $X_i < X_j$ for $i < j$.
- To reduce symmetry under isometries, we fix the minimum self-distance, and fix $X_1$ to the first of such strings (lexicographically).
- Hope: have constraints such that the maximal code is unique.
Search and Propagation

Heuristics

- The next string is chosen first among the ones with minimum distance from code constructed so far equal to $D$ (or minimal).
- These new constraints and heuristics are used in the selection phase, i.e. in the visit of the unfolded GDD, to prune parts of it.
- The bound we use is the size of the best code found. It is however difficult to estimate the size of the code under construction, so this bound is not very effective.
Experimental Results

- We implemented the whole procedure in C.
- The algorithm, when looking for maximal size codes, finds quickly a good solution, but never terminates in 1 hour limit.
- The problem is that the pruning is not very effective: we need to work out more refined estimates of the constructing code size.
- We also compared the performances of our engine with SICStus Prolog, for the easier task of constructing binary Hamming codes.
## Experimental Results

### Results for CSP on DNA strings

<table>
<thead>
<tr>
<th>string length</th>
<th>dist. threshold</th>
<th>size of the solution found</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2</td>
<td>40</td>
<td>0.18 sec</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>7</td>
<td>0.03 sec</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>59</td>
<td>99.00 sec</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>19</td>
<td>0.12 sec</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>6</td>
<td>0.03 sec</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>113</td>
<td>16.19 sec</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>30</td>
<td>4.57 sec</td>
</tr>
<tr>
<td>8</td>
<td>6</td>
<td>10</td>
<td>2.47 sec</td>
</tr>
<tr>
<td>10</td>
<td>6</td>
<td>58</td>
<td>35.10 sec</td>
</tr>
<tr>
<td>10</td>
<td>7</td>
<td>19</td>
<td>52.20 sec</td>
</tr>
<tr>
<td>10</td>
<td>8</td>
<td>9</td>
<td>105.18 sec</td>
</tr>
</tbody>
</table>
## Experimental Results

### Results for Binary Hamming code construction

<table>
<thead>
<tr>
<th>string length</th>
<th>distance threshold</th>
<th>size of the code</th>
<th>existence of a solution</th>
<th>time in SICStus</th>
<th>time in our engine</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>3</td>
<td>8</td>
<td>Y</td>
<td>0,0</td>
<td>0,0</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>9</td>
<td>N</td>
<td>562,49</td>
<td>0,5</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>4</td>
<td>Y</td>
<td>0,0</td>
<td>0,0</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>5</td>
<td>N</td>
<td>3,85</td>
<td>0,03</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>64</td>
<td>Y</td>
<td>5,71</td>
<td>0,22</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>8</td>
<td>Y</td>
<td>0,05</td>
<td>0,01</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>9</td>
<td>Y</td>
<td>668,00</td>
<td>14,40</td>
</tr>
</tbody>
</table>
Future Work

- The algorithm still fails to complete the exploration of the search tree, due to the gigantic size of the search tree even for small lengths of the strings.
- We need to impose more constraints (to break the symmetries of the problem), and to find better pruning heuristics.
- Another direction is that of integrating some stochastic ingredient in the branch and bound schema, in order to find good solutions in reasonable time, maybe finding the optimum with high probability.
The End

THANKS FOR THE ATTENTION!

QUESTIONS?