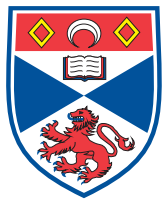
Exact Closest String as a Constraint Satisfaction Problem

4th Workshop on Biomedical & Bioinformatics Challenges to Computer Science

Tom Kelsey & Lars Kotthoff

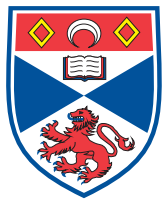
School of Computer Science
University of St Andrews

www.cs.st-andrews.ac.uk/~tom/
twk@st-andrews.ac.uk



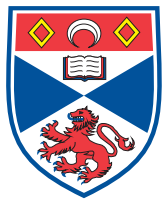
Overview

- The Problem
  - Restriction to nucleotide & amino acid sequences
  - Existing methods
- New developments
  - Exact closest string as a CSP
  - Search heuristics
  - Computational tools and techniques
  - Distributed methods
  - Hybrid Numeric/CSP approach
- Implications



Closest String

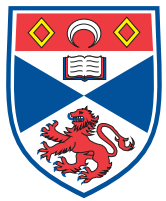
- Measure of self-similarity within a set of strings of equal length
- Any metric can be used, but Hamming distance is standard
  - number of positions at which the corresponding symbols are different
- A solution is a string with the smallest possible maximum Hamming distance from any input string
- Known to be be NP-complete for alphabets of arbitrary size
- Believed to be NP-complete for fixed alphabets
  - $|\Sigma|^L$ candidate solutions, where Σ and L are the alphabet and string length



Bioinformatics Applications

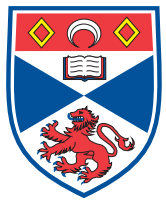
- DNA sequences
- $\Sigma = \{C, G, A, T\}$
- Consensus patterns
- Motif representation
  - See the paper for citations
- Homo sapiens or Neanderthal?

- Peptide sequences
- $|\Sigma| = 20$ (or 22)
- Too complex for meaningful calculations
- (currently)



State of the Art

- An approximate solution (to within $4/3 + \epsilon$) of the minimal distance d can be obtained in polynomial time, using Genetic Algorithms
- Excellent exact results – provided that close bounds have already been identified – have been obtained by modelling and solving as an IP (Integer Programming Problem)
- IP is useless for finding all solutions
  - IP branch and bound is optimised for optimisation
  - No exhaustive search is performed, subject to sub-trees being ruled out by logical impossibility
- Our aim is to find the exact d, together with **all** witnesses having that distance



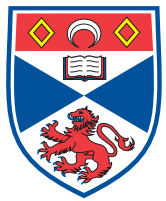
Constraint Satisfaction Problems

Consist of

- A set of constraints $\mathcal{C}$
- Acting on a finite set of variables $\Delta := \{A_1, A_2, \ldots, A_n\}$
- Each of which has a finite domain of possible values $D_i := D(A_i) \subseteq \Lambda$
- A *solution* to Y is an instantiation of all of the variables in Δ such that no constraint in $\mathcal{C}$ is violated.
- There may – or may not – be an objective function to be minimised

Are solved by

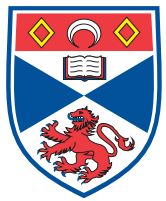
- AI backtrack search
- Heuristics for search order, consistency, ...



Exact Closest String as a CSP

Given S, a set of N strings of length L over alphabet Σ, we first compute the Hamming Diameter $HD(S)$ and use this to provide a lower bound, d_{min}, for the optimal distance d.
$\Upsilon(S, d_{min}, HD(S))$ denotes the Closest String instance in which the set of variables is $\Delta := \Delta_1 \cup \Delta_2 \cup \Delta_3 \cup \Delta_4$, where

1. Δ_1 is the array $[CS_1, CS_2, \ldots, CS_L]$ of variables representing the closest string, each such variable having domain $1 \ldots 4$
2. Δ_2 is an $N \times L$ array of binary variables used to calculate Hamming Distances from Δ_1 to the input strings S
3. Δ_3 is the array $[D_1, D_2, \ldots, D_N]$ of variables representing the distance of each string in S to the current CS candidate, each such variable having domain d_{min} through $HD(S)$
4. Δ_4 is the single distance variable d with domain d_{min} through $HD(S)$.

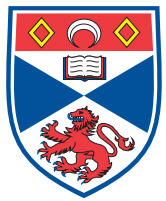


Exact Closest String as a CSP

The constraints are:

1. $\Delta_2(i,j) = 0$ iff $S_i(j) = \Delta_1(j)$
2. $\Delta_3(k)$ is the sum of row k of Δ_2
3. Δ_4 is the maximum value appearing in Δ_3
4. Δ_4 is minimised: if a solution is found with $\Delta_4 = d$, search for another solution with $\Delta_4 = d - 1$ (unless $d = d_{min}$).

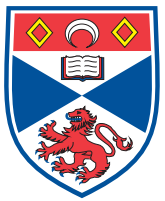
Once an optimal d is found, we can carry on exploring the tree to find all the solutions for that distance



Position Weight Matrices

Position	1	2	3	4	5	6
A	1.4	2.4	5.8	4.8	2.1	6.7
C	2.4	0	6.5	4.0	4.3	6.9
G	0	3.7	0	0.8	0	0
T	3.6	1.7	7.1	0	4.8	6.8

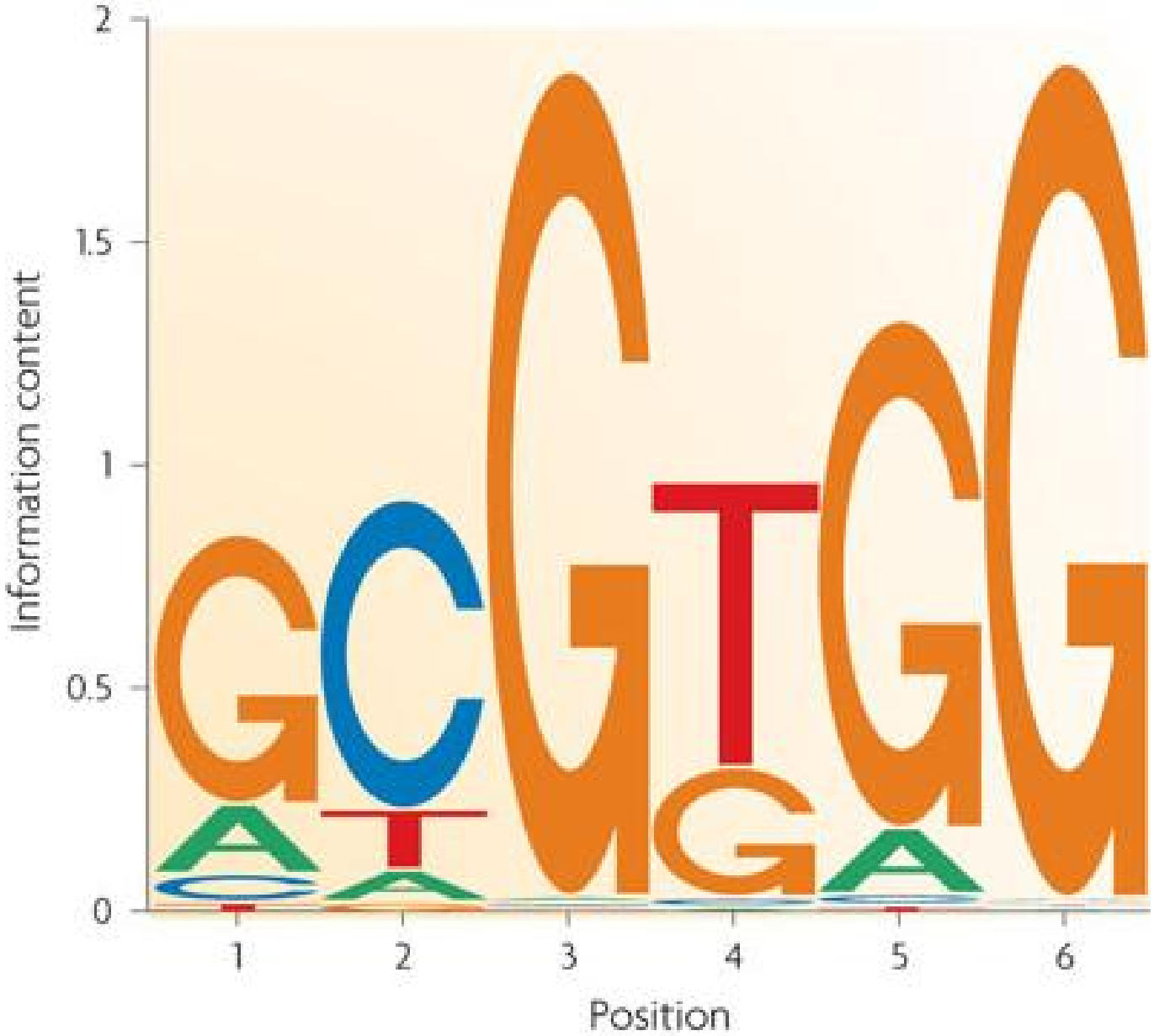
Stormo and Zhao; “Determining the specificity of protein – DNA interactions"; Nature Reviews Genetics 11, 751–760 (2010)



Position Weight Matrices

GCGTGG	0
GCGGGG	0.8
ACGTGG	1.4
GCGTGG	1.7
GCGTAG	2.1
ACGGGG	2.2
GAGTGG	2.4
GTGGGG	2.5
GCGGAG	2.9
...	...

Position Weight Matrices

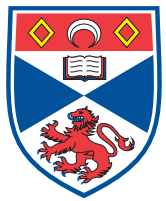


Base Camp

- Our CSP solution works
- We use the MINION solver that has solved several large-scale problems
- It is the first framework for obtaining **all** exact closest strings
- Using search heuristics from bioinformatics gives orders of magnitude speedup (in general)
- Showing that an optimal d **is** optimal is still difficult

- Distribute the problem
- Sending kill signals to jobs when a new d is found
- Work up from a lower bound as well as down

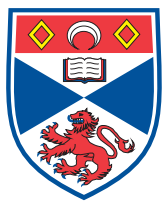
- Hybrid methods
- IP is known to work well within tight bounds
- CSP method is good at finding tight bounds



A recursive distributed algorithm to solve any CSP

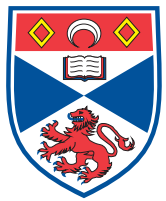
Input : A CSP Y, a cutoff period T_{max} and a branching factor K
Output: Either 1st solution, or a guarantee that there are none

while *not* `Solved?(Y)` **do**
  Send Y to a node
  Solve(Y, T_{max},0)
  if `Solved?(Y)` **then** Return solution
  else
    Y $\leftarrow$ Y with new constraints ruling out previous search
    Split Y into K subproblems $Y_1, Y_2, \ldots, Y_K$
    do in parallel
      for $1 \leq k \leq K$ **do**
        Solve(Y_k, T_{max},K)
      end
    end
  end
end



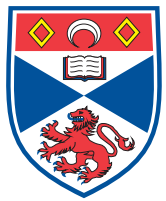
Distributed Exact Closest String

- We first run Minion on the original problem with the PWM ordering heuristic as a single process
  - we have shown that this will quickly lower the upper bound, in general
- From above, we carry on optimising as before, but using the recursive distributed algorithm
- From below we create instances each having a fixed distance, the idea being to exhaustively rule out any closest strings at these distances
  - these instances are run on the compute nodes at the same time as the optimisation sub-problems
- If at any stage we obtain a candidate closest string at a distance for which all lower distances have been ruled out, then this is our solution
  - this can happen both from above or below



A hybrid numeric/CSP algorithm

Input : $\Upsilon^0(S, d_{min}, HD(S))$
TOL, a limit for the gap between the bounds
Output: The optimal distance & a closest string to S
Seek closer distance bounds for $\Upsilon^0(S, d_{min}, HD(S))$ *using CSP alone;*
while $|d_{high} - d_{low}| < TOL$ **do**
  Run CSP on $\Upsilon^0(S, d_{min}, HD(S))$
  Output d_{low} and d_{high} when updated
end
Once bounds are close enough, send to numeric IP;
if $TOL > 1 \wedge |d_{high} - d_{low}| \leq TOL$ **then**
  Formulate the remaining problem as an IP problem
  Search for solution using numeric branch and bound
end
if $|d_{high} - d_{low}| = TOL = 1$ **then**
  Formulate the remaining problem as a fixed d instance
end



Implications

- The hybrid algorithm can also be distributed
- The aim is to combine the best features of each approach
- Web scale facilities are needed for large problems
  - Google exacyle: projects that can consume at least 100 million core-hours
  - Announced April, 2011
- We have been approached by a research group from the Friedrich Schiller Universität, Jena, Germany
- They have new numeric results, and we hope to combine our work with theirs later this year