Molecular and Cellular Computing

Lecture series at Universidad Politécnica de Madrid

Martyn Amos
Department of Computing and Mathematics
Manchester Metropolitan University
United Kingdom

http://www.martynamos.com

Day 1: Molecular Computing
3. Subsequent Work

Manchester Metropolitan University

Post-Adleman

- Richard Lipton followed up on Adleman's work by showing how the *Satisfiability* problem may, in principle, be solved using a similar approach
- SAT is the “gold standard” of NP-complete problems
- Decide if there exists a combination of assignments to the terms in a propositional formula such that the overall formula is true

Richard J. Lipton. DNA solution of hard computational problems. *Science*, **268**:542-545, 1995.

Manchester Metropolitan University

Example

You are chief of protocol for the embassy ball. The crown prince instructs you either to invite Peru or to exclude Qatar. The queen asks you to invite either Qatar or Romania or both. The king, in a spiteful mood, wants to snub either Romania or Peru, or both. Is there are guest list that will satisfy the whims of the entire royal family? (Due to Pat Hayes)

Manchester Metropolitan University

Satisfiability

- We might begin by representing the different requirements of the royal family as simple Boolean expressions. We start off by denoting each country by their initial letter, so that Peru, Qatar and Romania become P, Q and R respectively
- The crown prince's instruction to either invite Peru or exclude Qatar may be written as P OR (NOT Q)
- The queen's instruction to invite either Qata or Romania or both may be written as Q OR R
- The king's desire to snub either Romania or Peru or both may be written as (NOT R) OR (NOT P)
- Every one of these expressions must be satisfied, so we connect them with ANDs, bracket off each expression in order to keep it separate from the others, and thus write the whole expression as (P OR (NOT Q)) AND (Q OR R) AND ((NOT R) OR (NOT P))

Manchester Metropolitan University

SAT

- We may now construct a truth table for this expression
- We run through every possible combination of values for P, Q, and R, with 1 meaning “invited" and 0 meaning “not invited"
- For example, the combination P=1, Q=0, R=1 would mean that Peru and Romania get to go the the ball, with Qatar left out in the cold
- Each member of the royal family's set of instructions is known as a *clause*, so we show, for each possible combination of invite/not invite, the result of evaluating each clause individually, followed by the final result obtained by ANDing them all together (we denote this by F)

Manchester Metropolitan University

Truth table

P	Q	R	P OR (NOT Q)	Q OR R	(NOT R) OR (NOT P)	F
0	0	0	1	0	1	0
0	0	1	1	1	1	1
0	1	0	0	1	1	0
0	1	1	0	1	1	0
1	0	0	1	0	1	0
1	0	1	1	1	0	0
1	1	0	1	1	1	1
1	1	1	1	1	0	0

Manchester Metropolitan University

Lipton's DNA-based SAT

- “Suddenly I didn't know what a computer was anymore.” Richard Lipton, 1995
- Represent each possible combination of assignments as a single strand of DNA, then work through them
- *Making* each one in turn is obviously difficult!
- How do we encode (construct) *arbitrary* binary sequences as strands of DNA?

Manchester Metropolitan University

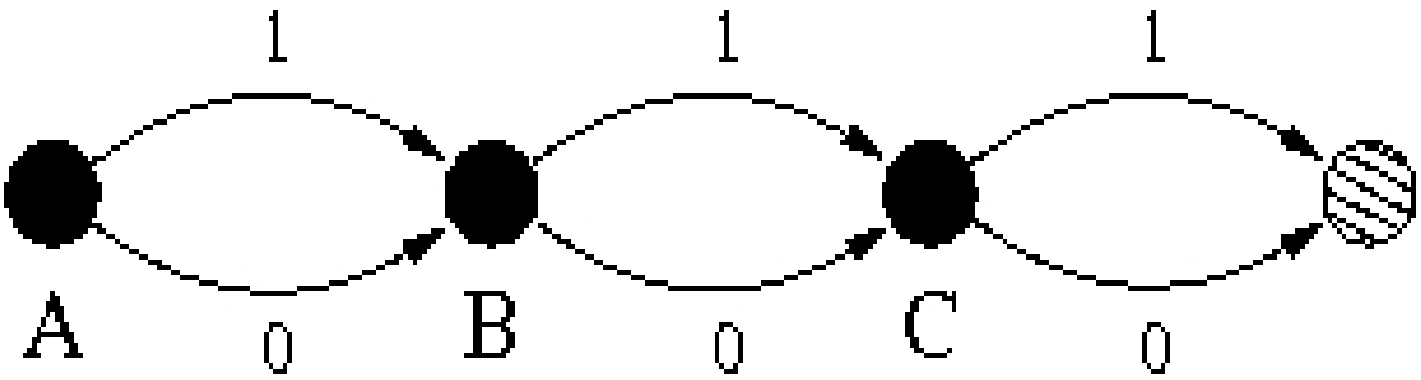
Construction problem

- Again, we need to generate an exponential-sized solution space
- Quickly becomes infeasible for even small instances
- A problem with 10 variables requires 2^{10}=1,024 individual strands
- If each solution strand is 100 bases long, initial synthesis would cost c. Euro 50,000

Manchester Metropolitan University

Lipton's solution

- Use a graph, where each vertex represents a bit
- Choice of edge dictates value of bit



Manchester Metropolitan University

Lipton's algorithm

- Generate all possible paths through the graph, a la Adleman
- Work through each clause in turn
- Since each clause *must* be satisfied (true, since they are connected by ANDs), and only one variable per clause needs to be true (since they are connected by Ors), the algorithm is very straightforward

Manchester Metropolitan University

The algorithm

create initial set, T

for each clause

for each literal v_i (variable or its complement)

if v_i = "variable", extract strands encoding v_i == 1

else extract strands encoding v_i == 0

create new set T by merging extracted strings

if T non-empty, then formula is satisfiable

Manchester Metropolitan University

An example

- Starting pool of solution strands:

P	Q	R
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Manchester Metropolitan University

First clause

- P OR (NOT Q)
- Keep *only* strands that encode P==1 or Q==0
- Lose PQR=010 and PQR=011

P	Q	R
0	0	0
0	0	1
1	0	0
1	0	1
1	1	0
1	1	1

Manchester Metropolitan University

Second clause

- Q OR R
- Retain only strands that encode 1 for Q or R
- Lose PQR=000 and PQR=100

P	Q	R
0	0	**1**
1	0	**1**
1	**1**	0
1	**1**	**1**

Final clause

- (NOT R) OR (NOT P)
- Retain only strands that encode 0 for R or P
- Lose PQR=101 and PQR=111

P	Q	R
0	0	1
1	1	**0**

Confirmation

P	Q	R	P OR (NOT Q)	Q OR R	(NOT R) OR (NOT P)	F
0	0	0	1	0	1	0
0	0	1	1	1	1	1
0	1	0	0	1	1	0
0	1	1	0	1	1	0
1	0	0	1	0	1	0
1	0	1	1	1	0	0
1	1	0	1	1	1	1
1	1	1	1	1	0	0

P	Q	R
0	0	1
1	1	0

Manchester Metropolitan University

Assessment

- Adleman's experiment (was a seminal demonstration of the feasibility of computing at the molecular level
- However, specific to HPP – not a general model (same with Lipton)
- We showed how *any* problem in NP may be solved by molecular means
- The *parallel filtering model* provided for the first time a way of elegantly expressing molecular algorithms

Martyn Amos, Alan Gibbons and David Hodgson, Error-resistant Implementation of DNA Computations, Proc. *DNA Based Computers II*, pp. 151-162, American Mathematical Society, 1996

Parallel filtering

- Entire solution space present in initial tube, as with Adleman
- "*Mark and destroy*" operation applied repeatedly in parallel to remove strands (as opposed to using mag. bead. separation)
- Algorithms for HPP, 3-vertex-colouring, subgraph isomorphism, maximum clique and maximum independent set described

Martyn Amos, DNA Computation, Ph.D. thesis, University of Warwick, 1997

Manchester Metropolitan University

The model

- Given a set of strings, a computation consists of a series of applications of the following operations:
- *remove* (U, $\{S_i\}$) Remove from set U, in parallel, any string that contains at least one occurrence of any of the substrings S_i
- *union* ($\{U_i\}$, U) Create, in parallel, the set U, which is the set union of the sets U_i
- *copy* (U, $\{U_i\}$) Create, in parallel, a number of copies of set U
- *select* (U) Select an element of U uniformly at random; return *null* if U is the empty set
- The result of the computation is (encoded in) the resultant set of strings

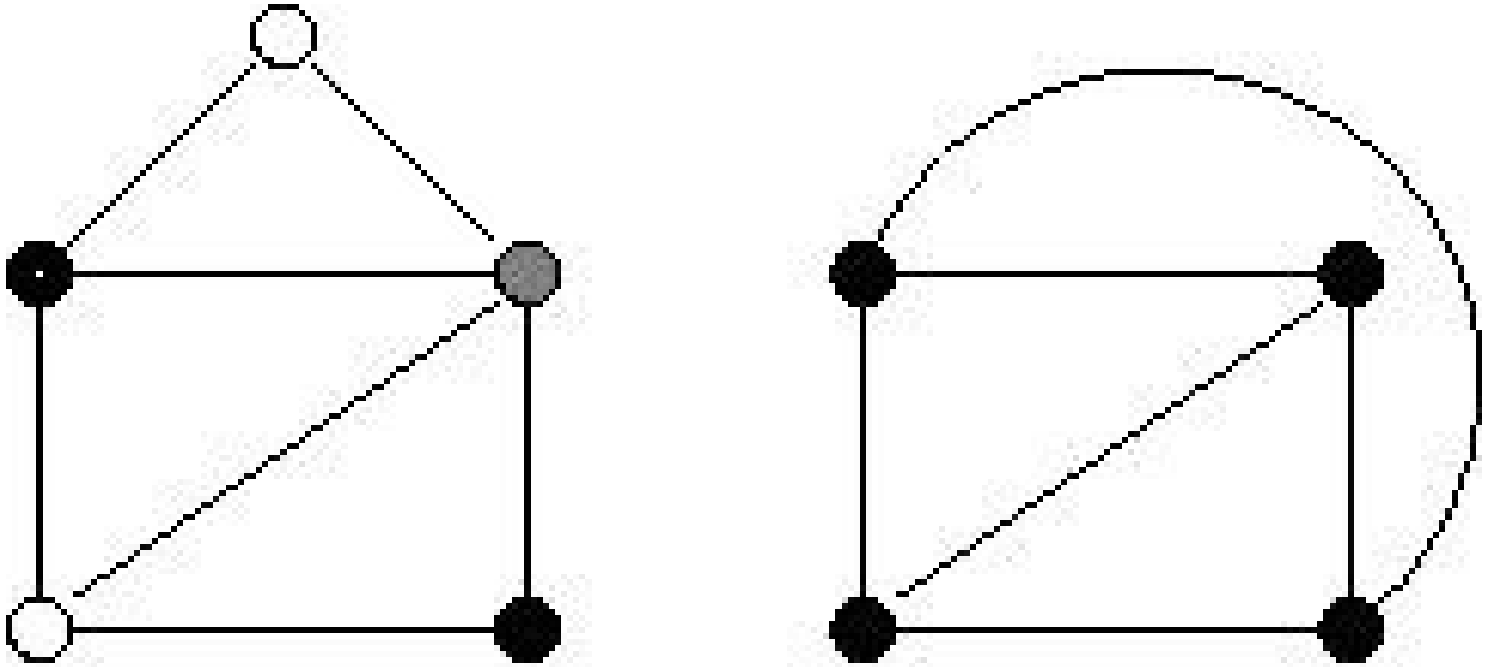
Manchester Metropolitan University

Example: 3-colouring

- Given a map, can we colour each country, using at most three “crayons”, such that no two countries sharing a border are coloured the same?
- Any planar map can be coloured with at most 4 crayons (Hakken and Appel)
- 3 crayons is NP-complete
- 3^n possible colouring for an n-vertex graph

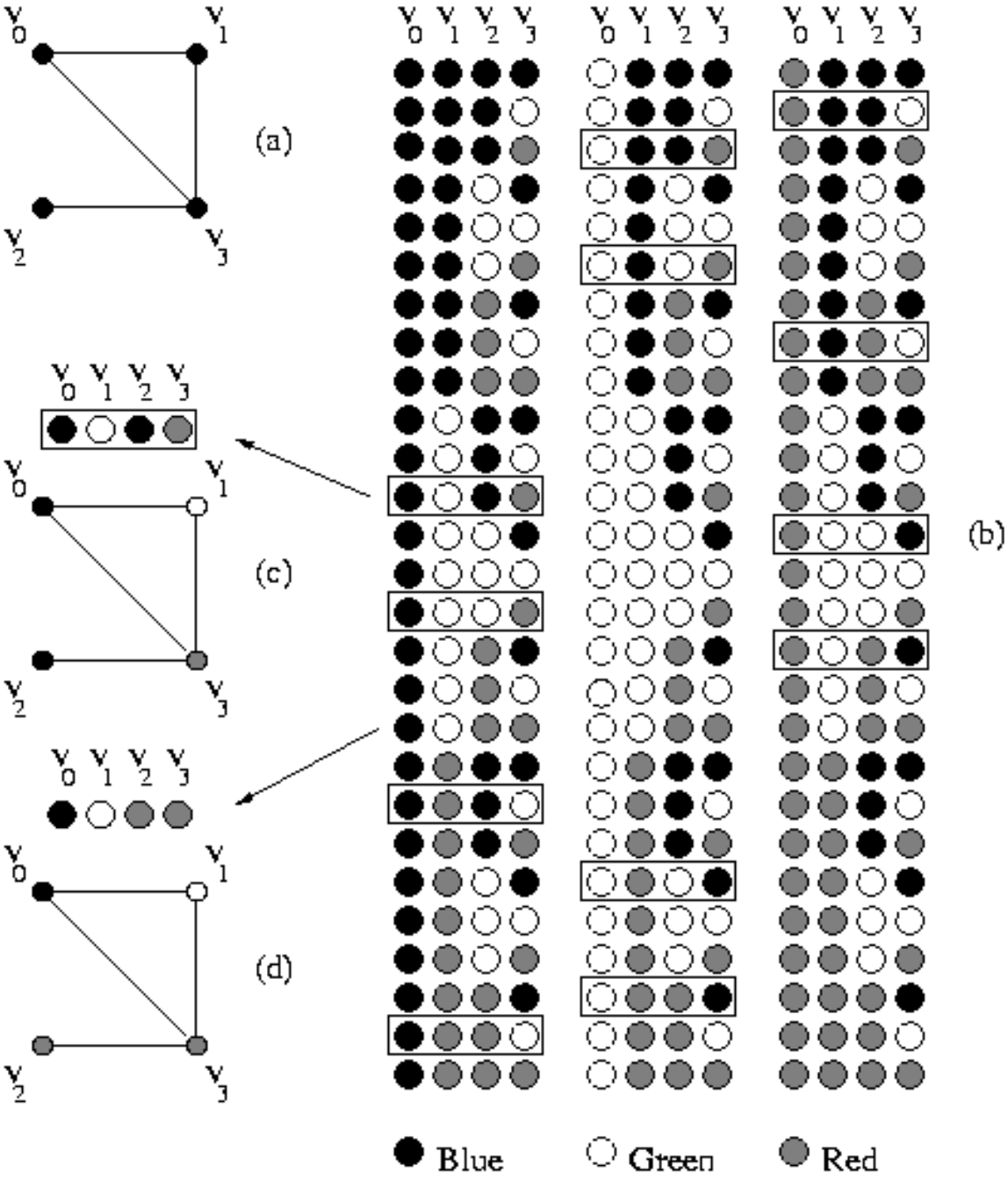
Manchester Metropolitan University

3-colourability



Manchester Metropolitan University

3-colourability



Manchester Metropolitan University

Our algorithm

Problem: Three colouring

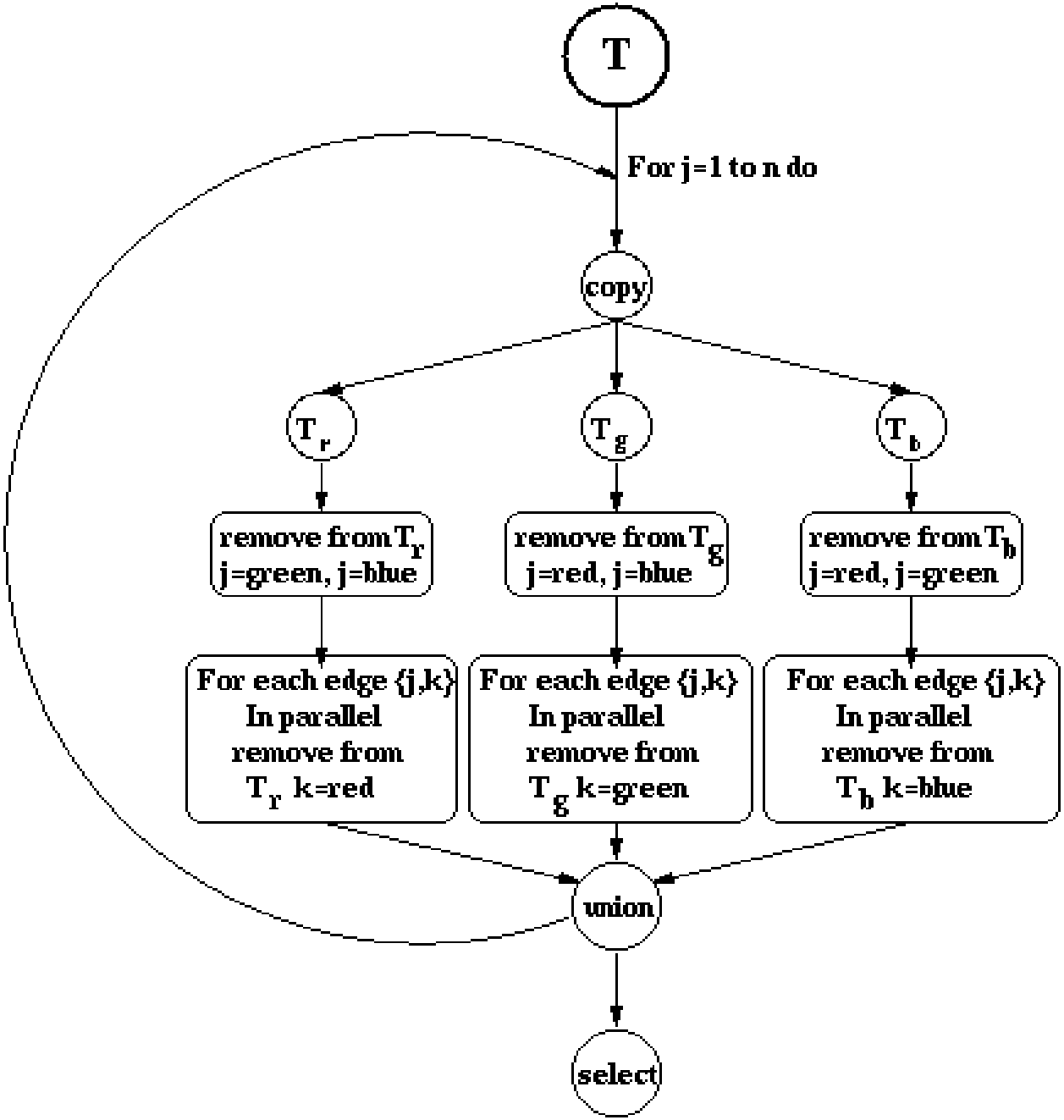
Given a graph $G = (V, E)$, find a 3-vertex-colouring if one exists, otherwise return the value $empty$.

Solution

Input:
The input set U consists of all strings of the form $p_1c_1p_2c_2 \ldots p_nc_n$ where $n = |V|$ is the number of vertices in the graph. Here, for all i, p_i uniquely encodes "position i" and each c_i is any one of the "colours" 1, 2 or 3. Each such string represents one possible assignment of colours to the vertices of the graph in which, for each i, colour c_i is assigned to vertex i.

Manchester Metropolitan University

Example: 3-vertex-colouring



Our approach

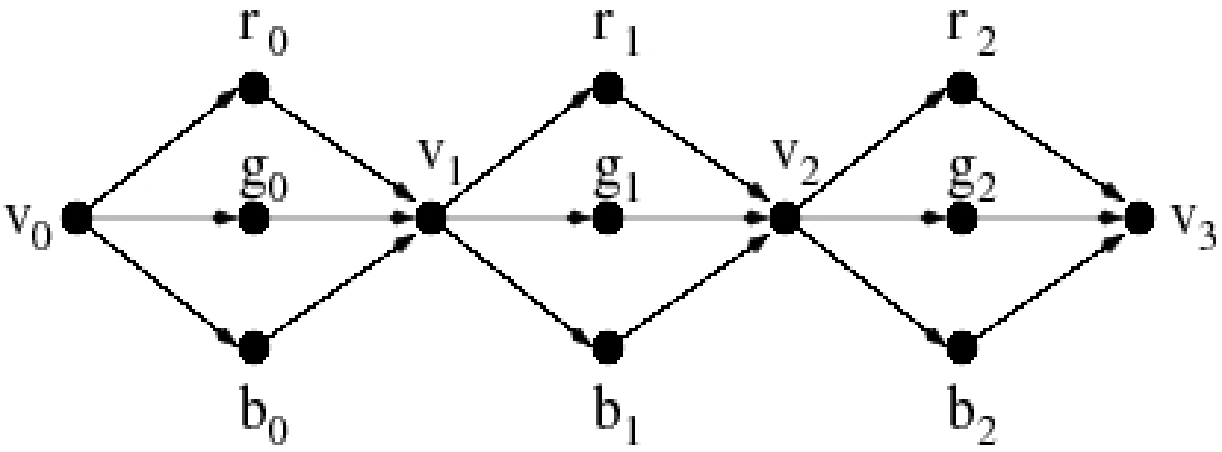
- Create a library of strands to represent all possible colourings of the graph
- Use polymerase and restriction enzymes to destroy particular strands in a systematic fashion

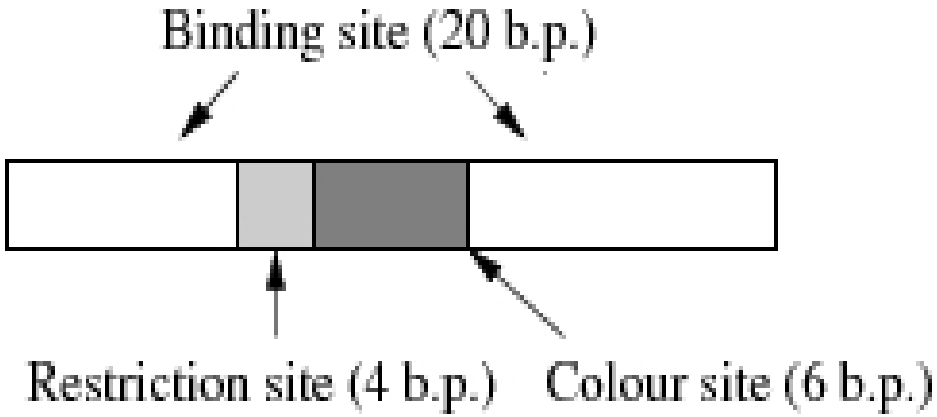
V_1 = red	V_2 = blue	V_3 = green	V_4 = red

Example strand

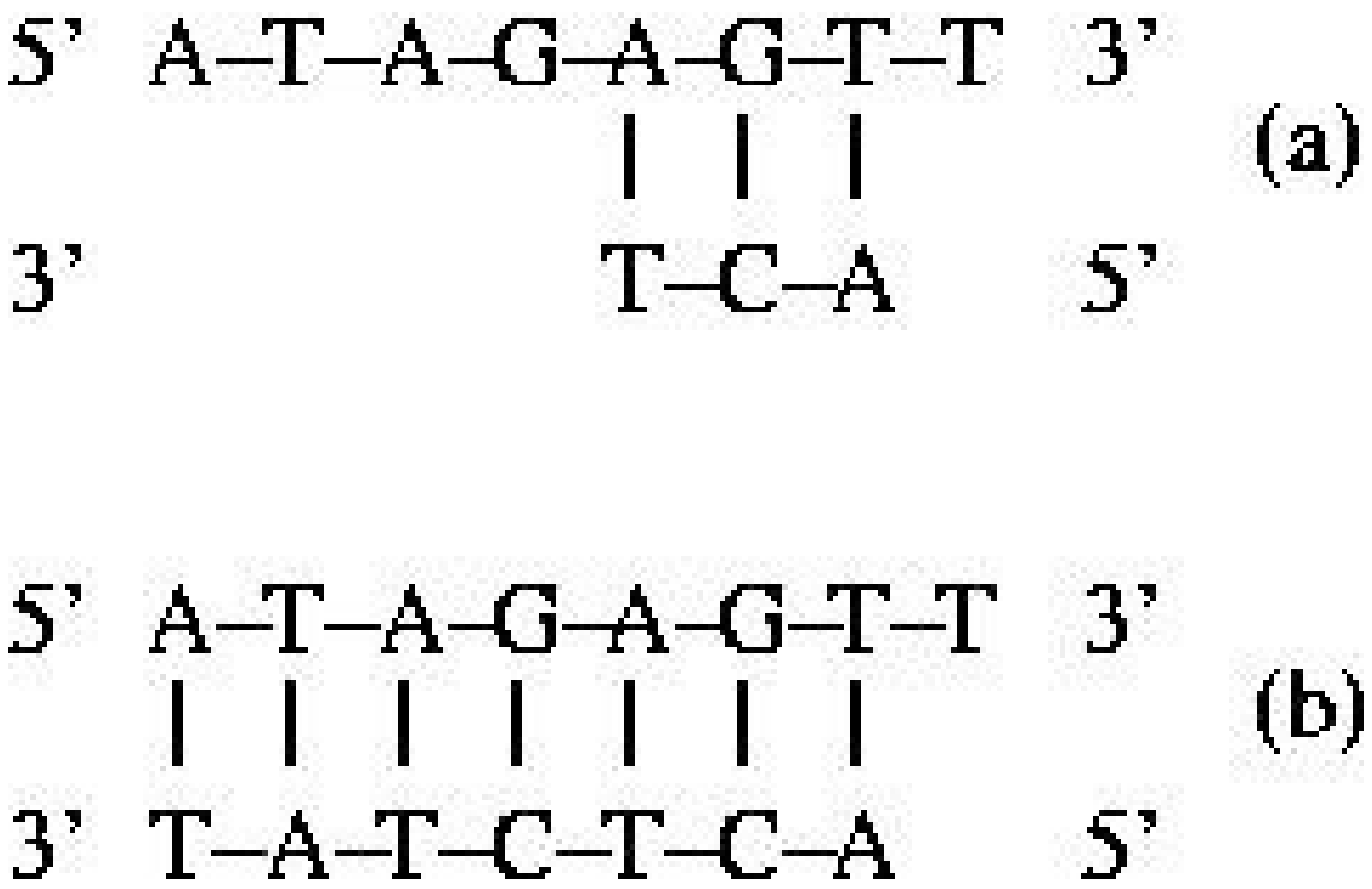
Strand creation

Similar approach to that of Lipton:

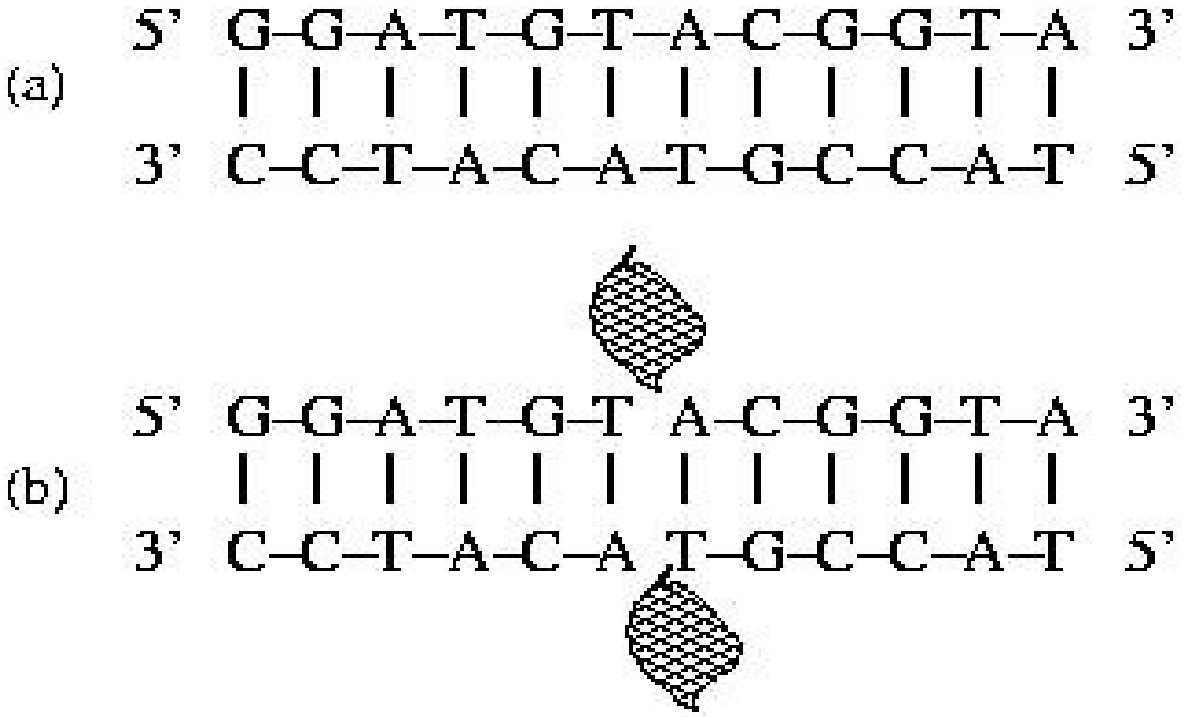




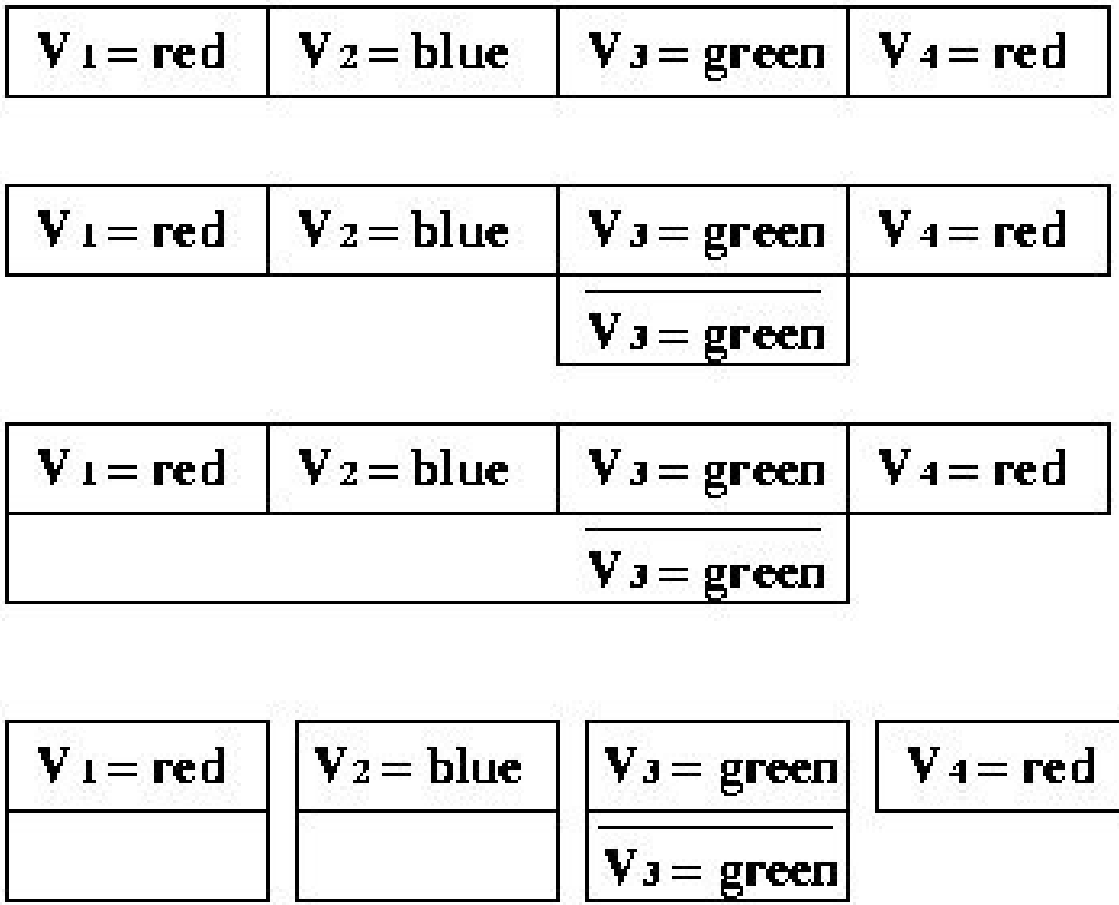
Polymerase



Restriction enzymes



Mark and destroy



Manchester Metropolitan University

Problem: Permutations

Generate the set P_n of all permutations of the integers$\{1, 2, \ldots, n\}$.

Solution

Input:
The input set U consists of all strings of the form$p_1 i_1 p_2 i_2 \ldots p_n i_n$ where, for all j, p_j uniquely encodes "position j" and each i_j is in $\{1, 2, \ldots, n\}$. Thus each string consists of n integers with (possibly) many occurences of the same integer.

Manchester Metropolitan University

Permutations Algorithm

for $j = 1$ to n **do**
begin
copy$(U, \{U_1, U_2, \ldots, U_n\})$
for i=1, 2, . . . , n and all $k > j$
    in parallel do remove$(U_i, \{p_j \neg i, p_k i\})$
union$(\{U_1, U_2, \ldots, U_n\}, U)$
end
$P_n \leftarrow U$

After the jth iteration of the **for** loop, the computation ensures that in the surviving strings the integer i_j is not duplicated at positions $k > j$ in the string.

Manchester Metropolitan University

Problem: Hamiltonian Path

Given a graph $G = (V, E)$ with n vertices, determine whether G contains a Hamiltonian path.

Solution

Input:
The input set U is the set P_n of all permutations of the integers from 1 to n as output from **Problem: Permutations**. An integer i at position p_k in such a permutation is interpreted as follows: the string represents a candidate solution to the problem in which vertex i is visited at step k.

Manchester Metropolitan University

Hamiltonian Path Algorithm

Algorithm

for $2 \leq i \leq n-1$ and j, k such that $(j,k) \notin E$
in parallel do remove $(U, \{jp_ik\})$
select(U)

Manchester Metropolitan University

Our algorithm for SAT

- Similar to that of Lipton, only this time we destroy strands that *do not* satisfy a clause rather than retaining those that *do*
- Potential problem: destruction is a global operator
- For example, to satisfy P OR (NOT Q), we need to first destroy all strands encoding P=0
- However, PQR=001 is a satisfying set of assigments, but we would have removed it
- After processing a few clauses, we would have no strands left at all

Manchester Metropolitan University

Solution

- We need to be able to move clause-by-clause
- Luckily, we are dealing with *multisets* of strands
- So we can *split* a tube, and effectively "clone" it (because of the numbers involved)
- This gives us a degree of *specificity*

Manchester Metropolitan University

The ball revisited

- P OR (NOT Q)
- Destroy P=0 and Q=1
- Split initial tube into P tube and Q tube, doing appropriate destruction in each
- Some sequences are destroyed in one tube, but survive in the other
- Destroyed sequences shown in bold (next slide)

Manchester Metropolitan University

P tube			Q tube		
P	Q	R	P	Q	R
0	**0**	**0**	0	0	0
0	**0**	**1**	0	0	1
0	**1**	**0**	**0**	**1**	**0**
0	**1**	**1**	**0**	**1**	**1**
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	**1**	**1**	**0**
1	1	1	**1**	**1**	**1**

Next clause

- Q OR R, so pool strands and then split again, this time into Q and R tubes, and destroy any strands encoding Q or R = 0

Q tube			R tube		
P	Q	R	P	Q	R
0	**0**	**0**	**0**	**0**	**0**
0	**0**	**1**	0	0	1
1	**0**	**0**	**1**	**0**	**0**
1	**0**	**1**	1	0	1
1	1	0	**1**	**1**	**0**
1	1	1	1	1	1

Manchester Metropolitan University

Final clause

- (NOT R) OR (NOT P)
- Combine, and then split into R and P tubes, destroying any strands that encode R or P = 1

P tube			R tube		
P	Q	R	P	Q	R
0	0	1	**0**	**0**	**1**
1	0	1	**1**	**0**	**1**
1	1	0	1	1	0
1	1	1	**1**	**1**	**1**

Manchester Metropolitan University

Mark and destroy

- The general notion of “mark and destroy” is now a useful and well-established operation in DNA computing
- Surface-based DNA computing operations: DESTROY and READOUT, Wang, L. et al, *BioSystems* **52**:1-3,189—191 (1999)
- Computation with biomolecules, Chen, J. and Wood, D.H., *Proceedings of the National Academy of Sciences* 97:4,1328 (2000)
- Timeline: The past, present and future of molecular computing, Wang, L. and Hall, J.G. and Lu, M. and Liu, Q. and Smith, L.M., *Nature Biotechnology* **19**, 1053—1059 (2001)

Manchester Metropolitan University

However…

- Polynomial time algorithms require exponential volumes of DNA
- If Adleman's experiment were to be scaled up to 200 vertices, the amount of DNA required to represent all possible paths through the graph would weigh more than the Earth

Juris Hartmanis, On the Weight of Computations, *Bulletin of the European Association For Theoretical Computer Science* **55**, pp. 136-138, 1995

In addition

- Assessments of the time complexities of molecular algorithms are often unrealistic
- Often assume that a laboratory assistant can pipette *n* tubes in parallel, when, in reality, they may only be able to deal with a single tube in any one time step
- Robots don't scale!
- We proposed a stronger model of molecular computation to take this into account

Martyn Amos, Alan Gibbons and Paul E. Dunne, The Complexity and Viability of DNA Computations, Proc. *Bio-Computing and Emergent Computation*, Lundh, Olsson and Narayanan (Eds.), pp. 165-173, World Scientific, 1997