An Algorithm for SAT Without an Extraction Phase

Pierluigi Frisco[1], Christiaan Henkel[2], and Szabolcs Tengely[3]

[1] Dept. of Comp. Sci., School of Eng., C. S. and Math., University of Exeter, Harrison Building, North Park Road, Exeter, EX4 4QF, UK
P.Frisco@exeter.ac.uk
[2] Institute of Biology, Leiden University, Wassenaarseweg 64, 2333AL Leiden, The Netherlands
henkel@rulbim.leidenuniv.nl
[3] Mathematical Institute, Leiden University, Niels Bohrweg 1, 2333CA Leiden, The Netherlands
tengely@math.leidenuniv.nl

Abstract. An algorithm that could be implemented at a molecular level for solving the satisfiability of Boolean expressions is presented.

This algorithm, based on properties of specific sets of natural numbers, does not require an extraction phase for the read out of the solution.

1 Introduction

Adleman's solution of an instance of the direct Hamiltonian path problem with the implementation of an algorithm at a molecular level [1] has been of inspiration for many to pursue other algorithms that can be implemented in the same way to solve instances of hard computational problems.

A problem is said to be *hard* if it cannot be solved by a deterministic Turing machine with a polynomial time algorithm in function of its input [8, 14]. For many of this kind of problems the number of possible solutions increases exponentially in function to the size of the input.

The algorithm described in [1] is related to the research of all Hamiltonian paths in a graph. The algorithm proposed by Adleman can be simplified in a two-phase process: first a library of DNA molecules encoding the input of the problem is *created* and is put in a test tube such that the DNA molecules can, under appropriate conditions, anneal and ligate, then the DNA molecules encoding solutions to the problem are *extracted* from the test tube.

During annealing and ligation other, 'new', DNA molecules, different from the ones present in the input library, can be created. Because of the massive parallelism and the nondeterminism of the annealing process the creation of the 'new' DNA molecules is quite fast and can lead to DNA molecules encoding solutions for the considered instance of the problem. As the name suggests during the extraction phase the solutions are extracted from the pool.

It should be clear that this kind of algorithms does not guarantee that a solution will be created even if it could. This because the annealing between

A. Carbone and N.A. Pierce (Eds.): DNA11, LNCS 3892, pp. 67–80, 2006.

complementary single stranded DNA molecules is a genuinely nondeterministic operation. Anyhow even if present in the pool a solution could not be detected during the extraction phase. More than error prone this last phase can be quite laborious and expensive.

Algorithms based on this two-phase process are common in Molecular Computing [2, 4, 6, 11, 12, 13, 16, 17, 20].

In Section 3 we describe how a specific creation of the input library of DNA molecules can be used to implement an algorithm without an extraction phase for satisfiability of Boolean expression (SAT), a hard computational problem, stated as decision problem (a problem remains hard if it is stated as decision, enumeration or research problem [14]). The presented algorithm is based on specific sets of natural numbers defined in Section 2.

The first algorithm for DNA computing without an extraction phase has been introduced in [10]. Here the authors define LOD (Length Only Discrimination), that is the concept of not having an extraction phase, and give an experimental result on a small instance of Hamiltonian path problem (HPP). In [19] another algorithm for HPP based on LOD is presented. In Section 5.2 we indicate the elements of novelty of our algorithms compared to the ones already present in the literature.

We did not implement in a biological laboratory the algorithm presented in Section 3, anyhow the biochemical specifications related to the creation of the input library of DNA molecules and to the implementation of the presented algorithms are sketched in Section 4.

2 Unique-Sum Sets

In this section we define unique-sum sets used in the algorithm presented by us in Section 3. Moreover we give some examples, we indicate some properties and results related to these sets, and we define a family of unique-sum sets.

Let $\mathbb{N}$ be the set of natural numbers.

Definition 1 (unique-sum set, ordered unique-sum set). *Let $G = \{n_1, ..., n_p\}$ be a set of different positive integers, and $s = \sum_{i=1}^{p} n_i$ the sum of the elements of G. G is said to be a unique-sum set if the equation $\sum_{i=1}^{p} c_i n_i = s, c_i \in \mathbb{N} \cup \{0\}$, has only the solution $c_i = 1, i \in \{1, \ldots, n\}$.*

A unique-sum set $G = \{n_1, \ldots, n_p\}$ is an ordered unique-sum set if $n_i < n_{i+1}$ for $1 \leq i \leq p-1$.

In what follows we will only consider ordered unique-sum sets.

An example of a unique-sum set is $G = \{4, 6, 7\}$, $4+6+7 = 17$ and 17 cannot be written in a different way as a non-negative integer linear combination of the elements in G. An example of a set that is not a unique-sum set is $G' = \{3, 4, 5\}$, $3 + 4 + 5 = 12 = 4 + 4 + 4 = 3 + 3 + 3 + 3$.

The concept of unique-sum set resembles that of subset-sum-distinct set (see e.g. [3]), but there one requires that for any two distinct finite subsets $G_1, G_2 \subseteq G$ the sum of all elements of G_1 is distinct from the sum of all elements of G_2.

Lemma 1. *Given a unique-sum set $G = \{n_1, \ldots, n_p\}$, any proper subset of G is a unique-sum set.*

Lemma 2. *Let k be a positive integer, and $kG = \{k \cdot n_1, \ldots, k \cdot n_p\}$. If G is a unique-sum set, then kG is also a unique-sum set.*

Definition 2 (maximal unique-sum set). *Given a unique-sum set $G = \{n_1, ..., n_p\}$, it is maximal if there exists no positive integer $n_{p+1} \notin G$, such that $G \cup \{n_{p+1}\}$ is a unique-sum set.*

It is easy to check that $G = \{2, 3\}$ is a maximal unique-sum set, but $G = \{4, 6\}$ is not, since $\{4, 6, 7\}$ is a unique-sum set too.

Now we describe a method to verify if a set is a unique-sum set. It is based on generating functions (see [15]). We consider the function

$$F_G(x) = \prod_{i=1}^{p} (1 - x^{n_i})^{-1}.$$

Using the identity $(1-x)^{-1} = 1+x+x^2+x^3+\ldots, x \in \mathbb{R}, |x| < 1$, we can rewrite $F_G(x)$ as a power series, having rational integers as coefficients, in the following form: $F_G(x) = P_0 + P_1 x + P_2 x^2 + \ldots + P_k x^k + \ldots$, and, by construction, the coefficient of x^k is the number of solutions of the equation

$$\sum_{i=1}^{p} c_i n_i = k, \quad c_i \in \mathbb{N} \cup \{0\}.$$

Therefore G is unique-sum set, if and only if $P_s = 1$, where $s = \sum_{i=1}^{p} n_i$. We do not have to use infinite expansions, since we are interested in the value of P_s. The coefficient of x^s in

$$\prod_{i=1}^{p} (1 + x^{n_i} + x^{2n_i} + \ldots + x^{[\frac{s}{n_i}]n_i})$$

is exactly P_s, where $[\cdot]$ denotes the integer part of the rational number $\frac{s}{n_i}$. Let us see two examples. Let $G = \{8, 12, 14, 15\}$, thus $s = 49$ and we have to compute the coefficient of x^{49} in $F_G(x) = (1 - x^8)^{-1}(1 - x^{12})^{-1}(1 - x^{14})^{-1}(1 - x^{15})^{-1} = f_1(x)f_2(x)f_3(x)f_4(x)$, where

$$\begin{aligned} f_1(x) &= 1 + x^8 + x^{16} + x^{24} + x^{32} + x^{40} + x^{48}, \\ f_2(x) &= 1 + x^{12} + x^{24} + x^{36} + x^{48}, \\ f_3(x) &= 1 + x^{14} + x^{28} + x^{42}, \\ f_4(x) &= 1 + x^{15} + x^{30} + x^{45}. \end{aligned}$$

It turns out to be 1, thus G is a unique-sum set. Let $G = \{8, 12, 14, 15, 19\}$, thus $s = 68$ and we have to compute the coefficient of x^{68} in $F_G(x) = (1 - x^8)^{-1}(1 - x^{12})^{-1}(1 - x^{14})^{-1}(1 - x^{15})^{-1}(1 - x^{19})^{-1} = f_1(x)f_2(x)f_3(x)f_4(x)f_5(x)$, where

$$f_1(x) = 1 + x^8 + x^{16} + x^{24} + x^{32} + x^{40} + x^{48} + x^{56} + x^{64},$$
$$f_2(x) = 1 + x^{12} + x^{24} + x^{36} + x^{48} + x^{60},$$
$$f_3(x) = 1 + x^{14} + x^{28} + x^{42} + x^{56},$$
$$f_4(x) = 1 + x^{15} + x^{30} + x^{45} + x^{60},$$
$$f_5(x) = 1 + x^{19} + x^{38} + x^{57}.$$

It turns out to be 12, thus G is not a unique-sum set.

Now we will deal with the construction of unique-sum sets. Given a set of different positive integers $G = \{n_1, \ldots, n_p\}$, such that $\gcd(n_1, \ldots, n_p) = 1$, it is known (see e.g. [5]) that for suitable large integer M, the equation

$$\sum_{i=1}^{p} c_i n_i = M, c_i \in \mathbb{N} \cup \{0\}, \tag{1}$$

has at least one solution. Let us denote by Φ_G the greatest positive integer for which (1) is not solvable. Wilf [18] gave an algorithm to compute Φ_G efficiently. We can use this constant to find possible extensions of a given unique-sum set (in the case when $\gcd(n_1, \ldots, n_p) = 1$), or to prove that it is maximal. First suppose that $\gcd(n_1, \ldots, n_p) = 1$, then we can compute Φ_G using the algorithm described in [18]. By the definition of Φ_G we know that if there exists an integer n_{p+1} such that $G \cup \{n_{p+1}\}$ is a unique-sum set, then $n_{p+1} \leq \Phi_G$. Thus we have to check only finitely many sets using the method mentioned previously. We have checked that the set $G = \{8, 12, 14, 15\}$ is a unique-sum set. In this case $\Phi_G = 33$, but there is no positive integer $k \leq 33$ such that $G \cup \{k\}$ is a unique-sum set, therefore G is maximal. If $\gcd(n_1, \ldots, n_p) = d > 1$ and the new element n_{p+1} is such that $\gcd(n_1, \ldots, n_p, n_{p+1}) = d' > 1$, then we still can succeed, since $\frac{1}{d'}G$ has to be a unique-sum set. In the remaining case, when $\gcd(n_1, \ldots, n_p) = d > 1$ and $\gcd(n_1, \ldots, n_p, n_{p+1}) = 1$, we show an example. Let $G = \{4, 6\}$ and n_3 is odd, then $s = n_3 + 10$ is also odd, thus if we have a solution of $4x_1 + 6x_2 + n_3x_3 = s$, then $x_3 > 0$. We obtain that $4x_1 + 6x_2 + n_3(x_3 - 1) = 10$, that is $x_1 = x_2 = x_3 = 1$ if $n_3 > 6$. In this way we obtained infinitely many unique-sum sets in the form $\{4, 6, 2k + 1\}, k > 2$.

Now we give a family of sets. Let $G_k = \cup_{m=1}^{k} \{2^k - 2^{k-m}\}$, the sum of the elements of G_k is $s_k = (k - 1)2^k + 1$. The first sets in this family are:

$$G_1 = \{1\},$$
$$G_2 = \{2, 3\},$$
$$G_3 = \{4, 6, 7\},$$
$$G_4 = \{8, 12, 14, 15\},$$
$$G_5 = \{16, 24, 28, 30, 31\},$$
$$G_6 = \{32, 48, 56, 60, 62, 63\},$$

Theorem 1. *For all $k \in \mathbb{N}$ the set G_k is a unique-sum set.*

The proofs of Lemma 1, Lemma 2 and Theorem 1, the proof that each element in the family of sets previously given is the unique-sum set having the smallest sum

in function of the number of elements and other properties and results related to unique-sum sets can be found in [7].

3 An Algorithm for the Satisfiability of Boolean Expressions

The *satisfiability of Boolean expressions* (SAT) problem can be formulated as: given a Boolean expression ϕ with variables $X = \{x_1, \ldots, x_n\}$, is there an assignment $A : X \to \{T, F\}$ such that A satisfies ϕ?

If the Boolean expression ϕ is given by a conjunction of clauses $C_1 \wedge C_2 \wedge \ldots \wedge C_p$ (where '$\wedge$' is the logical AND operator) each being a disjunction of at most k literals (a literal is a variable x_i or its negation $\neg x_i$, for $1 \leq i \leq n$), then the problem is called k-SAT.

In [11] the author demonstrates that 3-SAT is well suited to take advantage of the massive parallelism present in molecular computation. At the present time SAT is probably the problem with the most number of algorithms implemented [12, 20, 16, 4] or implementable [11, 9, 13] at a molecular level.

Let ϕ be an instance for k-SAT having n variables and p clauses, let $L = \{l_1, l_2, \ldots, l_q\}$ ($q \leq 2n$), an ordered set of literals satisfying at least one clause of ϕ such that if $l_i, \neg l_i \in L$ for $1 \leq i \leq q$, then $l_i = l_j, \neg l_i = l_{j+1}$ for a $1 \leq j \leq q-1$. Moreover let $C = \{C_1, \ldots, C_p\}$ the set of clauses present in ϕ, and let $G = \{n_1, \ldots, n_{p+2}\}$ be a unique-sum set having sum s_G.

The input library of molecules is composed by:

edges: Each pair $(l_i, l_j), i \leq j, l_i \neq \neg l_j, 1 \leq i, j \leq q$, of literals in L is encoded by an ordered (from 5' to 3') single stranded DNA molecule composed by the 8-mer s_{l_i} (encoding l_i) followed by the 8-mer s_{l_j} (encoding l_j). It is important to notice now that these pairs define a partial order in L. The order is partial and not total as there is no pair for a literal and its negation if both literals are present in L.

Moreover there are going to be two additional 8-mer single stranded DNA molecules: s_b and s_e.

For each literal $l \in L$ there will be ordered (from 5' to 3') single stranded DNA molecules composed by the 8-mer s_b followed by the 8-mer s_l and single stranded DNA molecules composed by the 8-mer s_l followed by the 8-mer s_e.

All the $s_l, l \in L$, s_b and s_e are different sequences of nucleotides.

vertices: We associate to each clause $C_j \in C$ a unique number $n_k \in G$. We will consider C_j associated to n_{j+1} for $1 \leq j \leq p$. For each literal l in L there will be a set of ordered (from 5' to 3') partially double DNA molecules composed by: a single stranded 8-mer $\bar{s}_l$ complementary to s_l; a double stranded $(n_{j+1} - 16)$-mer for each clause $C_j, 1 \leq j \leq p$ satisfied by l; a single stranded 8-mer $\bar{s}_l$ complementary to s_l.

begin: Ordered (from 5' to 3') partially double DNA molecules composed by: a single stranded 8-mer $\bar{s}_b$ complementary to s_b followed by a double stranded $(n_1 - 8)$-mer.

end: Ordered (from 5' to 3') partially double DNA molecules composed by: a double stranded $(n_{p+2} - 8)$-mer followed by a single stranded 8-mer $\bar{s}_e$ complementary to s_e.

The following example is meant to clarify the above. Let the Boolean expression $\phi = C_1 \wedge C_2 \wedge C_3 = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$ be an instance of 3-SAT. An ordered set of literals of ϕ satisfying at least one clause is $L = \{x_1, \neg x_1, x_2, \neg x_2, x_3, \neg x_3\} = \{l_1, l_2, l_3, l_4, l_5, l_6\}$, while the set of clauses of ϕ is $C = \{C_1, C_2, C_3\}$.

The set of single stranded DNA molecules encoding **edges** is depicted in Figure 1.

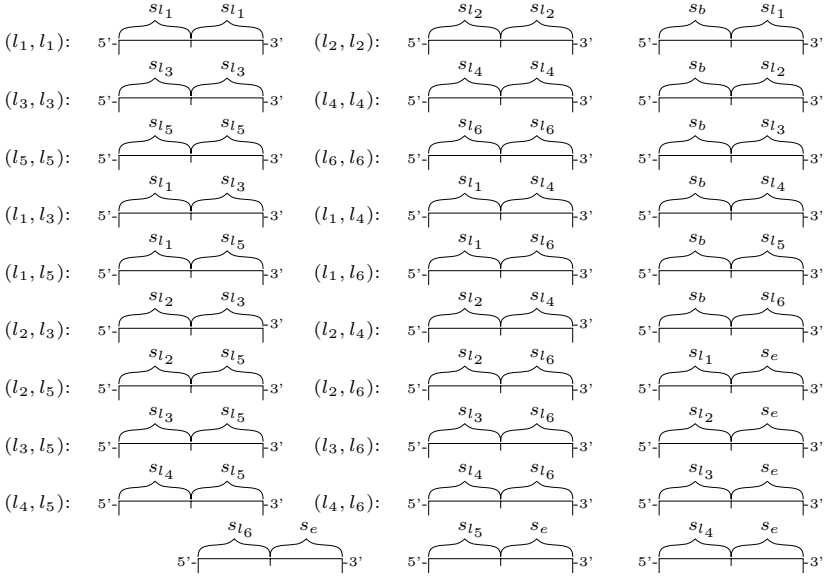

Fig. 1. Encoding of **edges** for the example of 3-SAT

Let us consider now the unique-sum set $G = \{16, 24, 28, 30, 31\}$, having sum $s_G = 129$. We associate 24 to C_1, 28 to C_2 and 30 to C_3. The literals l_1 and l_3 both satisfy only C_1; the literals l_2 and l_4 both satisfy C_2 and C_3; the literal l_5 satisfies C_1 and C_3; the literal l_6 satisfies C_2. Considering this we give now the lengths of the double stranded DNA molecules present in the encodings of **vertices**. As C_1 is associated to 24, then the double stranded DNA molecule is 8-bp (result of 24-16); as C_2 is associated to 28, then the double stranded DNA molecule is 12-bp (result of 28-16); as C_3 is associated to 30, then the double stranded DNA molecule is 14-bp (result of 30-16). The double stranded DNA molecule present in the encoding of **begin** is 8-bp (result of 16-8), while the one present in the encoding of **end** is 23-bp (result of 31-8). These molecules are depicted in Figure 2.

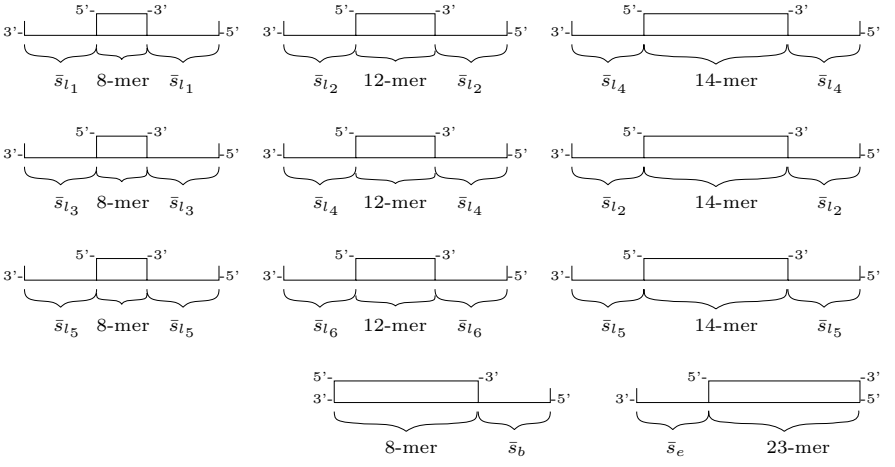

Fig. 2. Encoding of **vertices** for the example of 3-SAT

The described encoding for this example can be visualised as the graph depicted in Figure 3, where $C_i(l_j)$ indicates that the clause C_i is satisfied by the literal l_j (for $1 \leq i \leq 3, 1 \leq j \leq 6$). In this graph black dots indicate hubs: they have been introduced to decrease the number of arrows present in the graph and make it more readable, hubs have no relation with the encoding described by us.

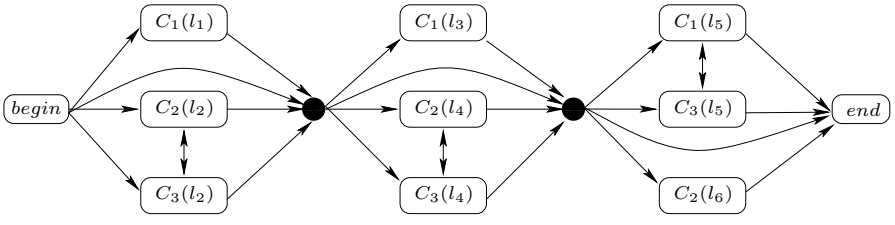

Fig. 3. Graph related to the example of 3-SAT

The annealing and ligation of the library of molecules is likely to form DNA molecules s_G-bp long only if there is an assignment A satisfying ϕ. Considering the graph depicted in Figure 3 such molecules can be visualised as paths starting at *begin* and ending at *end* and passing by nodes encoding clauses satisfied by literals where the encoding of each clause is present only once. Examples of such paths are: $begin - C_1(l_1) - C_2(l_4) - C_3(l_5) - end$, $begin - C_2(l_4) - C_3(l_4) - C_1(l_5) - end$.

If a resulting molecule is s_G-bp long, then it will start with a sequence encoding **begin** and it will end with a sequence encoding the **end**, the intermediate part will be composed by encodings of **vertices** (clauses satisfied by literals) annealed and ligated to **edges**. This intermediate part cannot contain both the

encoding of a clause satisfied by a literal l and the encoding of a clause satisfied by a literal $\neg l$, for $l \in L$. Moreover in the intermediate part the encoding of a clause can be present only once.

Any assignment A satisfying ϕ can be encoded (by the annealing and ligation of the molecules in the input library) in a double stranded DNA molecule s_G-bp long.

The presence of such a molecule can be detected by one run of gel electrophoresis independent of the size of the instance of the problem.

For a Boolean formula ϕ, instance of k-SAT, with p clauses and n variables (so at most $2n$ literals), in the worst case (all literals are present in a clause and each literal satisfies each clause) the input library of molecules is composed by:

$2n$ DNA molecules encoding **edges** of the form (l_i, l_i);
$2n \sum_{i=1}^{n-1}(2n-2i)$ DNA molecules encoding **edges** of the form $(l_i, l_j), i > j, l_i \neq \neg l_j, 1 \leq i, j \leq 2n$;
$4n$ DNA molecules encoding **edges** of the form (b, l) and (l, e) for $l \in L$;
$2np$ (each of the $2n$ literals can satisfy each of the p clauses) DNA molecules encoding **vertices**;
1 DNA molecule encoding **begin**;
1 DNA molecule encoding **end**.

In the following section we describe how the initial library of DNA molecules can be created.

4 Biochemical Specifications

As presented in the previous section unique-sum sets allow the creation of algorithms where part of the instance of the problem is encoded in the length of partially double DNA molecules. The actual sequence of the double part of these molecules is then of only minor importance. This fact can be exploited in the efficient production of these molecules.

Each element of the family of unique-sum sets presented in Section 2 can be written as $G_k = \{2^{k-1}, 2^{k-1}+2^{k-2}, 2^{k-1}+2^{k-2}+2^{k-3}, \ldots, 2^{k-1}+2^{k-2}+\cdots+2^0\}$. If moreover we consider that $2^h = 2^{h-1} + 2^{h-1}$, then it is possible to devise an efficient algorithm for the creation of long double stranded DNA molecules by controlled concatenation of two shorter ones. Only the short ($\leq$ 8-bp) DNA molecules need to be chemically synthesised.

The concatenation of two molecules requires tight control of the reaction as a simple ligation of molecules in solution will also produce many longer multimers. One way to perform controlled reactions is making the ends of the double stranded DNA molecules unavailable for ligation.

The following steps will create a specific concatenation of two generic double stranded DNA molecules A and B:

1. attach one end of A to a solid support. For example, use a 5' biotin label and streptavidin coated beads;
2. ensure the free 5' end is phosphorylated;

3. remove phosphates from B by alkaline phosphatase treatment;
4. mix and ligate;
5. remove all unbound molecules;
6. remove the molecules from the beads. This can be accomplished by simple endonuclease digestion if a DNA linker is used between the biotin label and molecule A;
7. if necessary, PCR (with or without biotynilated primers) can be used as an amplification procedure.

This procedure ensures that only one copy of molecule B can be attached to the immobilized A. However, some small chances of error still exist. For example, two molecules A can be ligated, creating a tether between two beads. Another possibility is incomplete ligation, i.e. some molecules A may not be ligated to B. Such errors are inevitable, but the chances can be minimized by optimization of laboratory protocols. If measurable quantities of erroneous molecules are formed, the correct molecules can be purified by preparative gel electrophoresis.

Very small molecules ($\leq$ 8-bp) can be added in an alternative way, using an extra sequence which is recognized by a type IIs restriction endonuclease. The sequence recognized by the restriction enzyme should be concatenated only at the two ends of the double stranded DNA molecule. The rest of the DNA molecule could be easily constructed so not to contain the restriction site. For example, one base pair can be added by ligation to 5' `NNNNNNGACTC`, and subsequent digestion with *MlyI* (New England Biolabs). This enzyme recognises the sequence 5' `GAGTC` and produces a blunt cut five bp to the 3' end. The result is 5' `N`, or any one base pair added. A similar technique can be used to produce different single stranded extensions necessary for programmable ligation. The enzyme used should then produce a staggered cut outside its recognition sequence. Using this method, the only molecules that need to be synthesized chemically are the 2 original 8 nucleotide strands and in total 6 oligonucleotides for adding 1, 2, or 4-bp.

The following example should clarify the strategy outlined above. Let us imagine that we want to create DNA molecules long as the elements in the unique-sum set $G_6 = \{32, 48, 56, 60, 62, 63\}$. Let us also consider that the two ends of each molecule have to be single stranded (each 8 bases long) while the rest of the molecule has to be double stranded. So, considering the elements in G_6, the part of the molecules that is double stranded has to be as long as the elements of the set $G'_6 = \{16, 32, 40, 44, 46, 47\} = \{8+8, 16+16, 32+8, 40+4, 44+2, 46+1\}$.

1. synthesize a molecule 8-bp long (such a molecule is stable enough and long enough to be ligated);
2. generate a molecule 16-bp long (element of G'_6) concatenating two molecules 8-bp long;
3. generate a molecule 32-bp long (element of G'_6) concatenating two molecules 16-bp long;
4. generate a molecule 40-bp long (element of G'_6) concatenating a molecule 32-bp long with one 8-bp long;
5. generate a molecule 44-bp long (element of G'_6) concatenating a molecule 40-bp long with one 4-bp long;

6. generate a molecule 46-bp long (element of G'_6) concatenating a molecule 44-bp long with one 2-bp long;
7. generate a molecule 47-bp long (element of G'_6) concatenating a molecule 46-bp long with one 1-bp long.

The single stranded molecules used in the algorithm presented in Section 3 need to be chemically synthesized and concatenated to the two sides of the double stranded DNA molecules.

5 Discussions

5.1 Biological

Experimental implementation of the algorithm presented in Section 3 is subject to some constraints. Thermodynamics dictates a certain minimum length for the DNA molecules present in the input library. DNA molecules of only a few bp do not anneal at room temperature: if, for example the unique-sum set $G_3 = \{4, 6, 7\}$ is considered for the encoding, then all members of the set should be multiplied by a constant to yield to DNA molecules long enough to be stable. The set obtained by the multiplication is ensured to be a unique-sum set by Lemma 2.

Length separation by electrophoresis imposes an upper limit on the size of the DNA molecules associated to the elements of a unique-sum set considered for encoding an instance of a problem. DNA electrophoresis has a maximum resolution of about 0.1%: discriminating between DNA fragments that have a difference in length of 1-bp per 1000 is realistic using large polyacrylamide gels or capillary electrophoresis. This limitation is due to current technology and not on DNA itself. Let us consider the set G_7, having sum $s_{G_7} = 769$, indicated in Section 2. The number $768 = 12{\cdot}64$ can be obtained as sum of elements in G_7. The difference between s_{G_7} and 768 represents the 0.13% of s_{G_7}. Similar computation for G_8 gives a value of 0.05% of its sum, already below the maximal resolution of the just described DNA electrophoresis.

We can envisage three possibilities to overcome this limit in the implementation of algorithms based on unique-sum sets:

1. other families of unique-sum sets may be found having a bigger difference between the sum of the set and the smaller or bigger number that can be obtained summing elements in the set;
2. different algorithms based on unique-sum sets can be devised;
3. the technology of DNA analysis can be improved so to increase the resolution.

The algorithm devised for the decision problem presented in Section 3 can be easily modified for research problems. If the presence of a solution is detected by gel electrophoresis, the precise sequence of it (telling in the case of SAT the sequence of clauses satisfied by a literal) can be found by DNA sequencing, multiplex PCR or restriction analysis. The analysis techniques themselves also entail some sequence design considerations.

5.2 Algorithmic

The creation of algorithms in DNA computing without an extraction phase is not new. Length-only discrimination (LOD) was introduced in [10] where the authors present experimental confirmations of this technique.

In [10] the algorithm giving the length of the molecules encoding the vertices is: "... if we need to find n different lengths, then starting with an arbitrary number for the lengths of the first vertex, we can produce the sequence of length with desired properties by making a gap between the lengths of the i^{th} and the $(i+1)^{th}$ vertices be $(n+i)$.". So, if for instance we want to find the lengths of the molecules for a graph with 9 vertices we have:

1: $k,\ k \in \mathbb{N}$
2: $(k) + 9 + 1 = k + 10$
3: $(k+10) + 9 + 2 = k + 2 \cdot 9 + 3 = k + 21$
4: $(k+21) + 9 + 3 = k + 3 \cdot 9 + 6 = k + 33$
5: $(k+33) + 9 + 4 = k + 4 \cdot 9 + 10 = k + 46$
6: $(k+46) + 9 + 5 = k + 5 \cdot 9 + 15 = k + 60$
7: $(k+60) + 9 + 6 = k + 6 \cdot 9 + 21 = k + 75$
8: $(k+75) + 9 + 7 = k + 7 \cdot 9 + 28 = k + 91$
9: $(k+91) + 9 + 8 = k + 8 \cdot 9 + 36 = k + 108$

So we obtain the set $K_9 = \{k, k+10, k+21, k+33, k+46, k+60, k+75, k+91, k+108\}$ having sum $s_{K_9} = 9k + 444$ (so we are considering the coefficients $f_1 =< 1,1,1,1,1,1,1,1,1 >$, notice that the sum of these coefficients is 9). But this sum can also be written as k+3(k+10)+(k+33)+3(k+91)+(k+108) which means that it can be obtained also by the coefficients $f_2 =< 1,3,0,1,0,0,0,3,1 >$ (notice that also the sum of these coefficients is 9). So, if in this example we consider that the initial vertex (having no incoming edges and only one outgoing edge) is associated to **1**, that the final vertex (having only one incoming edge and no outgoing edges) is associated to **9**, and that the rest of the graph is totally connected, then **1-2-8-2-8-4-2-8-9** would be interpreted as an Hamiltonian path (while it is not). This implies that the just presented algorithm to generate sets of numbers for algorithms based on LOD is not always valid.

The fact that the two sets of coefficients have both sum 9 is essential as also molecules encoding edges are present. In [10] edges are encoded such that the relative molecules are: "...longer than any vertex encoding.". This implies that any two sets of coefficients (as the ones indicated in the above) having the same sum would bring to accepted solutions (this would not be the case if the sets of coefficients had different sums as the associated DNA molecules would have different lengths). This affirmation is wrong if we consider f_2.

The other sets of coefficients for K_9 having the same properties of f_2 are: $f_3 =< 1,2,0,0,1,2,2,0,1 >$, $f_4 =< 1,0,2,2,1,0,0,2,1 >$, $f_5 =< 1,0,0,0,6,1,0,0,1 >$. These sets of coefficients can be used to find other sets of coefficients for $K_n, n \geq 10$, that is for sets obtained by the algorithm described in [10].

Let us list the elements found by the algorithm described in [10] from the second to the eighth for a set with $n \geq 9$ elements:

2: $k + n + 1$
3: $k + 2n + 3$
4: $k + 3n + 6$
5: $k + 4n + 10$
6: $k + 5n + 15$
7: $k + 6n + 21$
8: $k + 7n + 28$

The sum of these elements is $7k + 28n + 84$ but this sum can also be obtained by $2(k + 2n + 3) + 2(k + 3n + 6) + k + 4n + 10 + 2(k + 7n + 28)$ (we just used the set of coefficients f_2 but we could have used also f_3, f_4 or f_5).

This means that for $n = 10$ the set of coefficients $< 1, 0, 2, 2, 1, 0, 0, 2, 1, 1 >$ (having sum 10) gives the sum $s_{K_{10}}$; for $n = 11$ the set of coefficients $< 1, 0, 2, 2, 1, 0, 0, 2, 1, 1, 1 >$ (having sum 11) gives the sum $s_{K_{11}}$, etc..

The just given description does not render all the sets of coefficients for sets with $n \geq 10$ elements. For instance other sets of coefficients giving the sum $s_{K_{10}}$ are $< 1, 3, 0, 1, 0, 0, 1, 1, 2, 1 >, < 1, 2, 0, 1, 0, 0, 4, 1, 0, 1 >$, etc..

In [19] the authors describe algorithms based on LOD. Also in this paper sets with a unique sum are considered. The elements of such sets $G = \{n_1, \ldots, n_p\}$ are defined as follows:

$$\begin{cases} n_1 = 1 \\ n_k = kn_{k-1} + 1 - \sum_{i=1}^{k-1} n_i \end{cases}$$

The numbers in these sets grow (from n_1 to n_p) as $p!$. It is possible to see this if we express n_k as a function of n_{k-1}. We have that $n_{k-1} = (k-1)n_{k-2} + 1 - \sum_{i=1}^{k-2} n_i$, so $n_k = kn_{k-1} + 1 - \sum_{i=1}^{k-1} n_i = k(k-1)n_{k-2} + k - k\sum_{i=1}^{k-2} n_i + 1 - \sum_{i=1}^{k-1} n_i$. So $n_p = p(p-1)(p-2)\ldots 1 - x$ where x is a polynomial in n_i $(1 \leq i \leq p-1)$. This implies that the sum of a set with p elements grows as $p!$, while the sum of a set with p elements in the family of sets given in Section 2 grows as an exponential (power of 2).

As proved in [7] the family of unique-sum sets given in Section 2 is the one giving unique-sum sets with the smallest sum in relation to the number of elements in the set. So given a unique-sum set G' with n elements its sum $s_{G'}$ cannot be smaller than s_G the sum of the smallest set with n elements in the family presented in Section 2. A consequence of this is that the algorithm for SAT we presented is not of practical use because of the exponential increase in length of the DNA molecules needed to encode large instances of the considered problem.

The presented research is a starting point in creating algorithms that can be implemented at a molecular level based on properties of specific sets of numbers. Some natural continuations of this research are identified by the following questions:

Is it possible to relax the definition of unique-sum set (to, for instance, sets whose sum can be obtained with only a constant number of non-negative linear combinations of the elements in the set) and create algorithms implementable at a molecular level that can take advantage of this relaxed definition?

Are there other kind of sets that can be considered when we take in account the specific problem we want to solve and the way the algorithm is devised?

Acknowledgements

We thank J. Khodor for the interesting discussions about sets with a unique sum. The work of P. Frisco has been supported by the research grant NAL/01143/G of The Nuffield Foundation.

References

1. L. M. Adleman. Molecular computation of solutions to combinatorial problems. *Science*, 266:1021–1024, November 11, 1994.
2. L. M. Adleman. On constructing a molecular computer. volume 27 of *DIMACS: Series in Discrete Mathematics and Theoretical Computer Science*, pages 1–22. American Mathematical Society, 1995.
3. J. Bae. On generalized subset-sum-distinct sequences. *Int. J. Pure Appl. Math.*, 1(3):343–352, 2002.
4. R. S. Braich, N. Chelyapov, C. Johnson, P. W. K. Rothemund, and L. Adleman. Solution to a 20-variable 3-SAT problem on a DNA computer. *Science*, 296(5567):499–502, 2002.
5. A. Brauer. On a problem of partitions. *Amer. J. Math.*, 64:299–312, 1942.
6. D. Faulhammer, A. R. Cukras, , R. J. Lipton, and L. F. Landweber. Molecular computation: RNA solutions to chess problems. In *Proc. Nat. Acad. Sci. USA*, volume 97, pages 13690–13695, 2000.
7. P. Frisco and Sz. Tengely. On unique-sum sets. *Manuscript in preparation*, 2005.
8. M. R. Garey and D. S. Johnson. *Computers and intractability.* W. H. Freeman and Co., San Francisco, 1979.
9. N. Jonoska, S. A. Karl, and M. Saito. Three dimensional DNA structures in computing. *BioSystems*, 52:243–253, 1999.
10. Yevgenia Khodor, Julia Khodor, and T. F. Knight Jr. Experimental conformation of the basic principles of length-only discrimination. Poster at 7th International Workshop on DNA-Based Computers, DNA 2001, Tampa, U.S.A, 10-13 June 2001.
11. R. J. Lipton. Using DNA to solve NP-complete problems. *Science*, 268:542–545, April 28, 1995.
12. Q. Liu, L. Wang, A. G. Frutos, A. E. Condon, R. M. Corn, and L. M. Smith. DNA computing on surfaces. *Nature*, 403, 2000.
13. V. Manca and C. Zandron. *A DNA algorithm for 3-SAT(11,20)*, volume 2340 of *Lecture Notes in Computer Science*. Springer Verlag, Berlin, Heidelberg, New York, 2001.
14. C. H. Papadimitriou. *Computational complexity.* Addison-Wesley Pub. Co., 1994.
15. G. Pólya. On picture-writing. *Amer. Math. Monthly*, 63:689–697, 1956.

16. K. Sakamoto, H. Gounzu, K. Komiya, D. Kiga, S. Yokoyama, T. Yokomori, and M. Hagiya. Molecular computation by DNA hairpin formation. *Science*, 288: 1223–1226, May 19, 2000.
17. K. A. Schmidt, C. V. Henkel, G. Rozenberg, and H. P. Spaink. DNA computing using single-molecule hybridization detection. *Nucleic acid research*, 32:4962–4968, 2004.
18. H. S. Wilf. A circle-of-lights algorithm for the "money-changing problem". *Amer. Math. Monthly*, 85(7):562–565, 1978.
19. T. Yokomori, Y. Sakakibara, and S. Kobayashi. A magic pot : Self-assembly computation revisited. In W. Brauer, H. Ehrig, J. Karhumki, and A. Salomaa, editors, *Formal and Natural Computing: Essays Dedicated to Grzegorz Rozenberg*, volume 2300 of *Lecture Notes in Computer Science*, pages 418–429, 2002.
20. H. Yoshida and A. Suyama. Solution to 3-SAT by breadth first search. volume 54 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 9–22. American Mathematical Society, 1999.