DNA Recombination by XPCR

Giuditta Franco[1], Vincenzo Manca[1], Cinzia Giagulli[2], and Carlo Laudanna[2]

[1] Department of Computer Science, University of Verona, Italy
franco@sci.univr.it, vincenzo.manca@univr.it
[2] Section of General Pathology, Department of Pathology, University of Verona, Italy
cinzia.giagulli@univr.it, carlo.laudanna@univr.it

Abstract. The first step of the Adleman-Lipton extract model in DNA computing is the combinatorial generation of libraries. In this paper a new method is proposed for generating a initial pool, it is a quaternary recombination of strings via application of null context splicing rules. Its implementation, based on a kind of PCR called XPCR, results to be convenient with respect to the standard methods, in terms of efficiency, speed and feasibility. The generation algorithm we propose was tested by a lab experiment here described, since the presence of few sequences is enough for checking the completeness of the library. The simple technology of this approach is interesting in and of itself, and it can have many useful applications in biological contexts.

Keywords: SAT, PCR, XPCR, generation of DNA initial pool.

1 Introduction

DNA Computing is an emerging area where information is stored in biopolymers, and enzymes manipulate them in a massively parallel way according to strategies that can produce computationally universal operations [7, 1, 9]. The research in this field has already taken different pathways of theoretical and experimental interest, also developing into new sub-disciplines of science and engineering, for example nanotechnology and material design.

One of the ambition of DNA computation was solving NP complete problems, and since the exponential amount of DNA, typical of the Adleman-Lipton generate-and-search model, became a limit to the scale up of the procedures for problems of realistic size, then alternative approaches have been explored recently. For example, one of them is focused on building directly the solutions of the problem by means of 3D graph self-assembly [10], while another one evolves approximate solutions of instances of the NP-complete problems by means of the evolution of a population of finite state machines [13].

Nevertheless, the production of combinatorial libraries as the solution space of instances of SAT problem [9, 3, 12] is a typical recombination which is fundamental, not only to DNA computation, but to various types of in vitro selection experiments, for selecting new DNA or RNA enzymes (such as ribozymes), or for performing crossover of homologous genes [14] and mutagenesis.

A. Carbone and N.A. Pierce (Eds.): DNA11, LNCS 3892, pp. 55–66, 2006.

In [4] a special type of PCR called XPCR was introduced to implement a procedure for extracting, from an heterogeneous pool, all the strands containing a given substrand. The XPCR technique was tested in different situations and was shown working as expected [4]. Here we focus on the combinatorial generation of libraries, and propose an economic and efficient XPCR-based procedure for generating DNA solution spaces; a lab experiment is reported that confirm the correctness and the completeness of this method.

The classic two methods for initial pool generation were introduced in 1994. The *hybridization-ligation* method was introduced by Adleman in [1] for solving a Hamiltonian path problem; it links oligonucleotides hybridized by complementarity and ligase reaction. The other method, called *parallel overlap assembly (POA)* was introduced by Stemmer [14] to perform crossover between homologous sequences of genes. Its implementation was based on hybridization/polymerase extension, and it was applied successfully by Kaplan et al. in DNA computing for solving maximal clique problem [11]. The second method demands less time and is a better choice in terms of generation speed and material consumption than the first one [8].

More recently, *mix-and-split* method was introduced [6] to generate an initial RNA pool for the knight problem, that is a combinatorial library of binary numbers. It was used by Braich et al. in [2] to generate an initial pool for a 20-variable 3-SAT problem, the biggest instance solved in lab. It combines a chemical synthesis and an enzymatic extension method, in fact, two half libraries are combined by primer extension while each of them was synthesized chemically on two columns by repetitive mixing and splitting steps. Therefore this method takes the advantages of an extension method but performs a big part of the work by chemical synthesis, which can become quite expensive.

Finally, a modified version of PNA-mediated Whiplash PCR (PWPCR) was presented in [13], with an in vitro algorithm to evolve approximate solutions of Hamiltonian Path problem. The PWPCR procedure is basically implemented by the recursive polymerase extension of a mixture of DNA hairpins, and here a similar approach is pursued with an easier and cheaper technology.

Let us consider a solution space of dimension n, that is, given by all possible sequences of type $\alpha_1\alpha_2\cdots\alpha_n$, where α_i can be, for each i, one of two different strings X_i or Y_i.

Our method starts from four types of strands and, by using only polymerase extension, generates the whole DNA solution space, where all types of possible sequences of the pool are present. Therefore, with respect to the other methods it is very easy and cheap: it does not use PNA molecules and no chemical synthesis is required apart that one for the initial strands.

In next sections we give, with examples, the intuition behind our XPCR generation algorithm, and recall the XPRC procedure (for more details see [4]) which XPCR generation is based on. Then, the algorithm is described, and its implementation in laboratory is presented with an experiment which confirms the validity of the generation procedure. In a further paper [5] the mathematical correctness and completeness of the algorithm is proved in a formal setting. In

particular, in the case of solution spaces of dimension n where each element of the sequence can be of k different types, the following general proposition holds.

Proposition 1. *Starting from four sequences and for any value of n, by XPCR generation algorithm all 2^n combinations are generated in $2(n-2)$ steps.*

Moreover in [5] we prove the existence of some special types of sequences, such that if they are present in the pool, then the pool contains all the solutions of the n-dimensional DNA solution space.

Our approach takes all the advantages and the efficiency of an enzymatic elongation method. Moreover, with respect to the POA, in any step of recombination, the unwanted side products are eliminated by the electrophoresis during the XPCR procedure, and the completeness of the library can be tested very easily by checking the presence of two special strands. The existence of such strands [5] is a consequence of the null context splicing theory [7], and it holds because the procedure is completely based on splicing rules.

Finally, the recombination of strings modeled by a null context splicing rule is performed in nature by few enzymes, thus it can be done for a very limited number of restriction sites, while the XPCR scales up this kind of recombination to any site.

2 Quaternary Recombination

The main intuition underlying the quaternary recombination was inspired by [2], where the following sequences were studied to avoid mismatch phenomena. We start from the sequences I_1, I_2, I_3, I_4, each constituted by n different strings that can occur in two distinct forms.

For the sake of simplicity we describe this method for a solution space of dimension 6. From a mathematical point of view, the method could be generalized to any dimension, and to cases where the variables can assume k values with $k > 2$. Consider the sequences:

1. *Positive*: $I_1 = X_1X_2X_3X_4X_5X_6$
2. *Negative*: $I_2 = Y_1Y_2Y_3Y_4Y_5Y_6$
3. *Positive-Negative*: $I_3 = X_1Y_2X_3Y_4X_5Y_6$
4. *Negative-Positive*: $I_4 = Y_1X_2Y_3X_4Y_5X_6$

Starting from these four initial sequences, every *solution* $\alpha_1\alpha_2\alpha_3\alpha_4\alpha_5\alpha_6$ of the solution space is generated by means of *null context splicing rules* [7], that are of type

$$r_\gamma : \quad \phi\,\gamma\,\psi,\ \ \delta\,\gamma\,\eta \quad \longrightarrow \quad \phi\,\gamma\,\eta,\ \ \delta\,\gamma\,\psi$$

where ϕ, γ, ψ, δ, η are strings on the considered alphabet.

In fact, any string $\alpha_1\alpha_2\alpha_3\alpha_4\alpha_5\alpha_6$ can be seen as the concatenation of substrings of I_1, I_2, I_3, I_4 that suitable splicing rules cut and recombine along common substrings. For example, the sequence $X_1X_2Y_3X_4Y_5Y_6$, can be obtained starting from the initial ones in the following way:

1. $r_{X_2} : I_1,\ I_4 \longrightarrow X_1X_2Y_3X_4Y_5X_6,\ Y_1X_2X_3X_4X_5X_6$
2. $r_{Y_5} : I_2,\ X_1X_2Y_3X_4Y_5X_6 \longrightarrow Y_1Y_2Y_3Y_4Y_5X_6,\ \mathbf{X_1X_2Y_3X_4Y_5Y_6}$

The generation of any solution can be associated to a *generation sequence* of rules, that is, a sequence on the alphabet $R = \{r_{X_2}, r_{X_3}, r_{X_4}, r_{X_5}, r_{Y_2}, r_{Y_3}, r_{Y_4}, r_{Y_5}\}$. For example, the generation sequence of the string $X_1X_2Y_3X_4Y_5Y_6$ considered above is $r_{X_2}r_{Y_5}$. We observe that, in this context, the order of application of the rules is not relevant. For example, the same string $X_1X_2Y_3X_4Y_5Y_6$ can be also obtained by permuting the application order of the rules (from the same initial strings):

1. $r_{Y_5} : I_4,\ I_2 \longrightarrow Y_1X_2Y_3X_4Y_5Y_6,\ Y_1Y_2Y_3Y_4Y_5X_6$
2. $r_{X_2} : I_1,\ Y_1X_2Y_3X_4Y_5Y_6 \longrightarrow \mathbf{X_1X_2Y_3X_4Y_5Y_6},\ Y_1X_2X_3X_4X_5X_6$

The following *Recombination Canonic Procedure* gives a general insight of the validity of the method outlined above.

- **input:** a sequence $\alpha_1\alpha_2\alpha_3 \ldots \alpha_n$
- **put** $i = 1$ and H_1 equal to the initial sequence (among the four ones) where the subsequence $\alpha_1\alpha_2$ occurs
- **for** $i = 1, \ldots, n-2$, **do**
  begin
  - **put** L_i equal to the initial sequence where $\alpha_{i+1}\alpha_{i+2}$ occurs
  - **if** $H_i = L_i$ **then** put $H_{i+1} = H_i$
  - **else** put H_{i+1} equal to the first product of the null context splicing $r_{\alpha_{i+1}}$ applied to strings H_i, L_i.

  end
- **output:** the sequence of null context splicing rules that were applied during *for* instruction.

The sequence of rules given in output by the previous algorithm is a generation sequence for the input string. In fact, if we apply these rules to the initial pool, then we get a set of strings which surely contains the input string. In the general case of n variables, a generation sequence is at most $n-2$ long, since there can be only one occurrence of rules r_{α_i} for each i belonging to $\{2, 3, \ldots, n-1\}$.

To implement null context splicing rules r_γ we use a kind of PCR called $XPCR_\gamma$, which was introduced as tool of extraction in [4], and which we recall briefly in the next section.

2.1 XPCR Procedure

We suppose to have an heterogeneous pool of DNA double strands having the same length and sharing a common prefix α and a common suffix β. Given a specified string γ, by means of $XPCR_\gamma$, we can recombine all the strings of the pool that contain γ as substring.

The $XPCR_\gamma$ procedure is described by the following steps. We indicate by $PCR(\xi, \overline{\eta})$ a standard PCR performed by forward primer ξ and backward primer $\overline{\eta}$, where $\overline{\eta}$ is the reversed and complemented sequence of η.

- **input** a pool P of strings having α as prefix and β as suffix
- **split** P into P_1 and P_2 (with the same approximate size)
- **apply** $PCR(\alpha, \overline{\gamma})$ to P_1 **and** $PCR(\gamma, \overline{\beta})$ to P_2 (*cutting step*, see Figure 1)
- perform **electrophoresis** on P_1 **and** on P_2 to select γ-prefixed or γ-suffixed strings, that corresponds to eliminate the sequences of the initial length.
- **mix** the two pools resulting from the previous step in a new pool P
- **apply** $PCR(\alpha, \overline{\beta})$ to P (*recombination step*, see Figure 2)
- **output** the pool resulting from the previous step.

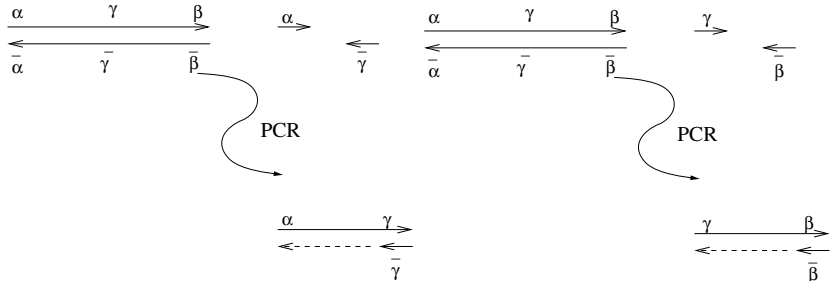

Fig. 1. Cutting step of $XPCR_\gamma$

In the recombination step, left parts $\alpha \cdots \gamma$ and right parts $\gamma \cdots \beta$ of the sequences of the pool having γ as subsequence are recombined in all possible ways, regardless to the specificity of the sequences between α and γ, or γ and β. Therefore, not only the whole sequences containing γ are restored but also new sequences are generated by recombination (see Figure 2).

Note that, if the hybridization between the two strands $\alpha \cdots \gamma$ and $\gamma \cdots \beta$ happens in a different location (other than at point γ), the 'wrong' product has not the length of the sequences in the initial pool, so it is eliminated by final electrophoresis [4].

However, in the XPCRs performed by the XPCR Recombination Algorithm described in the next section, the interactions of hybridization/polymerase extension between the strands $\alpha \cdots \gamma$ and $\gamma \cdots \beta$ can happen only along the codeword γ, because by construction they do not have more complementary regions (in fact, if γ represents the $i-th$ variable, than the strand $\alpha \cdots \gamma$ contains the codewords of the first i variables, and $\gamma \cdots \beta$ contains the complementary codewords of the last $n - i + 1$ variables).

Procedure $XPCR_\gamma$ implements a version of null context splicing rule with common substring γ, where strings are assumed to share a common prefix and a common suffix (both ϕ and δ start with a certain prefix, and both ψ and η end with a certain suffix) and strings obtained after their recombination are added to the strings available before it

$$r_\gamma : \quad \phi\,\gamma\,\psi, \;\; \delta\,\gamma\,\eta \quad \longrightarrow \quad \phi\,\gamma\,\eta, \;\; \delta\,\gamma\,\psi, \;\; \phi\,\gamma\,\psi, \;\; \delta\,\gamma\,\eta.$$

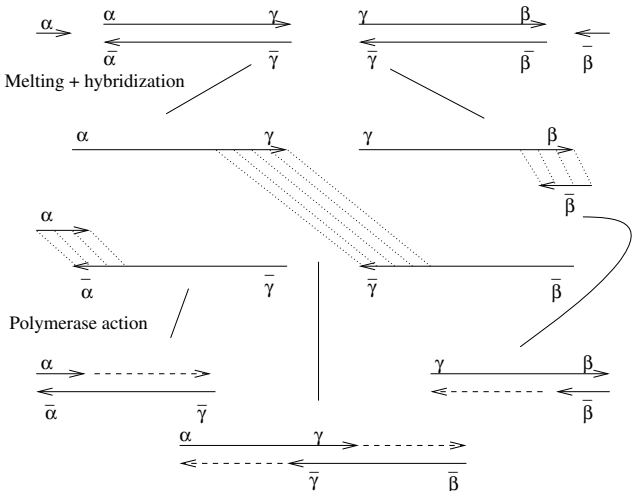

Fig. 2. Recombination step of $XPCR_\gamma$

2.2 XPCR Recombination Algorithm

We start with a pool having only the four initial sequences I_1, I_2, I_3, I_4 that are extended by a common prefix α and a common suffix β (for performing the XPCR procedure).

- **Let** P_1 and P_2 be two copies of the pool

$$\{\alpha \cdots I_1 \cdots \beta,\ \alpha \cdots I_2 \cdots \beta,\ \alpha \cdots I_3 \cdots \beta,\ \alpha \cdots I_4 \cdots \beta\}$$

- **for** $i = 2, 3, 4, 5$ **do**
  begin
  - **perform** $XPCR_{X_i}$ on P_1 and $XPCR_{Y_i}$ on P_2
  - **mix** the two pools obtained in the previous step in a unique pool P
  - **split** P randomly in two new pools P_1 and P_2 (with the same approximate size)

  end

If all steps are performed correctly we obtain a pool where all 2^6 combinations $\alpha_1\alpha_2 \cdots \alpha_6$ with $\alpha_i \in \{X_i, Y_i\}$ are present.

It can be proved that, for any dimension n of the solution space, two sequences exist such that their simultaneous presence in the pool guarantees that all the possible recombinations happened. The proof of this fact is based essentially on the following lemma (see [5] for the formal details).

Let us call *i-trio-factor* any substring $\alpha_{i-1}\alpha_i\alpha_{i+1}$ where exactly two consecutive variables are Xs or Ys, and consider the corresponding null context splicing rule r_{α_i}, then it holds that

Lemma 1. *For any $i = 2, \ldots, n-1$ the rule r_{α_i} has been applied in the pool iff in the pool there is a string including a corresponding i-trio-factor.*

A set W of strings is a recombination witness if the strings of W include all the possible i-trio-factors. In our case, we have chosen the following strings that, according to the lemma, are clearly a set of recombination witnesses.

1. *XXYY alternating*: $W_1 = X_1X_2\,Y_3Y_4\,X_5X_6$
2. *YYXX alternating*: $W_2 = Y_1Y_2\,X_3X_4\,Y_5Y_6$

In order to verify that all the expected recombinations happened, in the experiment described in the next section, it was enough to check that in the final pool the sequences W_1 and W_2 were present.

3 Experiment

The following experiment showed that, given any SAT problem with 6 variables, the solution pool with sequences encoding all 64 assignments can be generated by starting from 4 specific sequences and by using only XPCR and electrophoresis. The success of this experiment is meaningful because it can be scaled up easily to any number n of variables: starting from 4 specific sequences, all 2^n combinations can be generated by XPCR and electrophoresis in a linear number of biosteps.

The experiment started on an initial pool containing the sequences 150bp long[1] (in the following all the sequences are written with respect to the usual $5' - 3'$ orientation):

$$\begin{aligned} S_1 &= \alpha\, W_a\, X_1\, X_2\, X_3\, X_4\, X_5\, X_6\, Z_a\, \beta \\ S_2 &= \alpha\, W_b\, Y_1\, Y_2\, Y_3\, Y_4\, Y_5\, Y_6\, Z_b\, \beta \\ S_3 &= \alpha\, W_a\, X_1\, Y_2\, X_3\, Y_4\, X_5\, Y_6\, Z_b\, \beta \\ S_4 &= \alpha\, W_b\, Y_1\, X_2\, Y_3\, X_4\, Y_5\, X_6\, Z_a\, \beta \end{aligned}$$

where α and β, necessary to perform XPCR procedure, were 20b long[2], for $i = 1, 2, \ldots, 6$, X_i and Y_i, representing the boolean values of the variables of the problem, were 15b long[3], and the elongating sequences W_j and Z_j for $j = a, b$ were 10b long[4]. The steps of the experiment follow.

[1] S_1 = GCAGTCGAAGCTGTTGATGC CAAGAGATGG TCGTCTGCTAGCATG TCACGCCACGGAACG GTGAGCGCGAGTGTG ATATGCAATGATCTG ATCCGTCCCGATAAG CAAGTCAGATTGACC GCACGTAACT AGACGCTGCCGTAGTCGACG
S_2 = GCAGTCGAAGCTGTTGATGC CAAGATATGG CCCGATTAGTACAGC TACTGATAAGTTCCG TCGCTCCGACACCTA TCAGCCGGCTTGCAC AACTGATACGACTCG TATTGTCACGCATCG GTACGTAACT AGACGCTGCCGTAGTCGACG
S_3 = GCAGTCGAAGCTGTTGATGC CAAGAGATGG TCGTCTGCTAGCATG TACTGATAAGTTCCG GTGAGCGCGAGTGTG TCAGCCGGCTTGCAC ATCCGTCCCGATAAG TATTGTCACGCATCG GTACGTAACT AGACGCTGCCGTAGTCGACG
S_4 = GCAGTCGAAGCTGTTGATGC CAAGATATGG CCCGATTAGTACAGC TCACGCCACGGAACG TCGCTCCGACACCTA ATATGCAATGATCTG AACTGATACGACTCG CAAGTCAGATTGACC GCACGTAACT AGACGCTGCCGTAGTCGACG

[2] $\alpha = GCAGTCGAAGCTGTTGATGC$, $\beta = AGACGCTGCCGTAGTCGACG$

[3] X_1 = TCGTCTGCTAGCATG X_2 = TCACGCCACGGAACG X_3 = GTGAGCGCGAGTGTG X_4 = ATATGCAATGATCTG, X_5 = ATCCGTCCCGATAAG, X_6 = CAAGTCAGATTGACC. Y_1 = CCCGATTAGTACAGC, Y_2 = TACTGATAAGTTCCG, Y_3 = TCGCTCCGACACCTA, Y_4 = TCAGCCGGCTTGCAC, Y_5 = AACTGATACGACTCG, Y_6 = TATTGTCACGCATCG

[4] W_a = CAAGAGATGG, W_b = CAAGATATGG, Z_a = GCACGTAACT, Z_b =GTACGTAACT

1. The initial pool was split randomly in four test tubes (a) (b) (c) (d), and the following PCRs were performed respectively
   (a) $PCR(\alpha, \overline{X_2})$
   (b) $PCR(X_2, \overline{\beta})$
   (c) $PCR(\alpha, \overline{Y_2})$
   (d) $PCR(Y_2, \overline{\beta})$
   so obtaining amplification of four types of sequences $\alpha \dots X_2$, $X_2 \dots \beta$, $\alpha \dots Y_2$, and $Y_2 \dots \beta$, that are 60, 105, 60, 105 long respectively (see lanes 2, 3, 4, 5 of Figure 3).
2. Two $PCR(\alpha, \overline{\beta})$ were performed in parallel, one after having put together the product of (a) and (b), and the other one after having put together the product of (c) and (d) (sequences 150 long were amplified, see lanes 6 and 7 of Figure 3), then the two pools were mixed.
3. For $i = 3, 4, 5$ the analogues of the previous steps (1) and (2) were performed by replacing index 2 with $i = 3, 4, 5$, and by referring to Figure 4, Top left for $i = 3$, Top right for $i = 4$, and Bottom for $i = 5$ respectively.
4. An electrophoresis was performed to select the sequences 150 long among longer ones (generated by unspecific amplification).

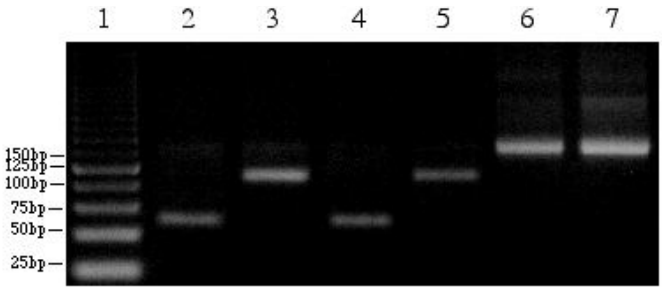

Fig. 3. Electrophoresis results. *Lane 1*: molecular size marker ladder (25bp). *Lane 2*: amplification of $\alpha \cdots X_2$ strands (60bp) and *lane 3*: amplification of $X_2 \cdots \beta$ strands (105bp), both PCRs performed at 52°C. *Lane 4*: amplification of $\alpha \cdots Y_2$ strands (60bp) and *lane 5*: amplification of $Y_2 \cdots \beta$ strands (105bp), both PCRs performed at 45°C. *Lane 6*: cross pairing amplification of $\alpha \cdots X_2$ and $X_2 \cdots \beta$ (150bp) and *lane 7*: cross pairing amplification of $\alpha \cdots Y_2$ and $Y_2 \cdots \beta$ (150bp), both XPCRs performed at 63°C.

Remark. By the electrophoresis results one can note that the sequences 150 long present in the pool were amplified (in linear or exponential manner) during each step. Although this phenomenon did not disturb the correctness of the computation, it caused noise and useless occupation of space in a test tube by increasing the unspecific matter, as one can see clearly in the figure 4. An improved experiment could be performed very easily by inserting intermediate electrophoresis steps to clean the signal from the noise caused by these amplifications.

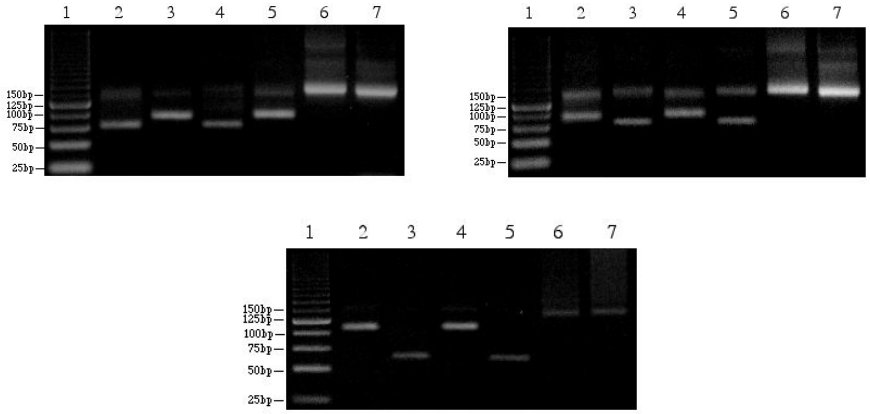

Fig. 4. Electrophoresis results. *Lane 1*: molecular size marker ladder (25bp). **Top left.** *Lane 2*: amplification of $\alpha \cdots X_3$ strands (75bp), *lane 3*: amplification of $X_3 \cdots \beta$ strands (90bp), *lane 4*: amplification of $\alpha \cdots Y_3$ strands (75bp), *lane 5*: amplification of $Y_3 \cdots \beta$ strands (90bp), all PCRs performed at 52°C. *Lane 6*: cross pairing amplification of $\alpha \cdots X_3$ and $X_3 \cdots \beta$ (150bp) and *lane 7*: cross pairing amplification of $\alpha \cdots Y_3$ and $Y_3 \cdots \beta$ (150bp), both XPCRs performed at 63°C. **Top right.** *Lane 2*: amplification of $\alpha \cdots X_4$ strands (90bp), *lane 3*: amplification of $X_4 \cdots \beta$ strands (75bp), *lane 4*: amplification of $\alpha \cdots Y_4$ strands (90bp), *lane 5*: amplification of $Y_4 \cdots \beta$ strands (75bp), all PCRs performed at 42°C. *Lane 6*: cross pairing amplification of $\alpha \cdots X_4$ and $X_4 \cdots \beta$ (150bp) and *lane 7*: cross pairing amplification of $\alpha \cdots Y_4$ and $Y_4 \cdots \beta$ (150bp), both XPCRs performed at 63°C. **Bottom.** *Lane 2*: amplification of $\alpha \cdots X_5$ strands (105bp), *lane 3*: amplification of $X_5 \cdots \beta$ strands (60bp), *lane 4*: amplification of $\alpha \cdots Y_5$ strands (105bp), *lane 5*: amplification of $Y_5 \cdots \beta$ strands (60bp), all PCRs performed at 45°C. *Lane 6*: cross pairing amplification of $\alpha \cdots X_5$ and $X_5 \cdots \beta$ (150bp) and *lane 7*: cross pairing amplification of $\alpha \cdots Y_5$ and $Y_5 \cdots \beta$ (150bp), both XPCRs performed at 63°C.

The success of the experiment was tested by the presence of the recombination witnesses $X_1X_2\ Y_3Y_4\ X_5X_6$ and $Y_1Y_2\ X_3X_4\ Y_5Y_6$ in the final pool, guaranteed by the amplification of the following PCRs[5].

We indicated with $Z_1Z_2Z_3Z_4Z_5Z_6$ a generic recombination witness, and for each of them the following procedure was executed in lab on a distinct copy of the pool P resulting from the experiment. That is, firstly the pool was split in two pools P_1 and P_2 with the same approximate size, and then on each of them the presence of a recombination witness $Z_1Z_2Z_3Z_4Z_5Z_6$ was checked by means the following steps:

1. **perform** $PCR(Z_1, \overline{Z_6})$
2. perform **electrophoresis** and select strands 90 long

[5] The authors thank an anonymous referee for his/her interesting comments and for suggesting us to find a better procedure checking the complete recombination of the final pool.

3. **perform** $PCR(Z_2, \overline{Z_5})$
4. perform **electrophoresis** and select strands 60 long
5. **perform** $PCR(Z_3, \overline{Z_4})$

output: YES if the last PCR amplifies (sequences 30 long), NO otherwise.

The procedure proved the presence of $Z_1Z_2Z_3Z_4Z_5Z_6$ in the pool P because i) after the first two steps all and only the strands $Z_1 \cdots Z_6$ of P were present in the resulting pool (lines 2 and 5 of Figure 5), ii) after the second PCR and electrophoresis all and only the strands $Z_2 \cdots Z_5$ from strands $Z_1Z_2 \cdots Z_5Z_6$ of P were present in the resulting pool (lines 3 and 6 of Figure 5), iii) and the last PCR amplified the portions Z_3Z_4 of such strands, that are found if and only if the sequence $Z_1Z_2X_3Z_4Z_5Z_6$ was present in P (lines 4 and 7 of Figure 5).

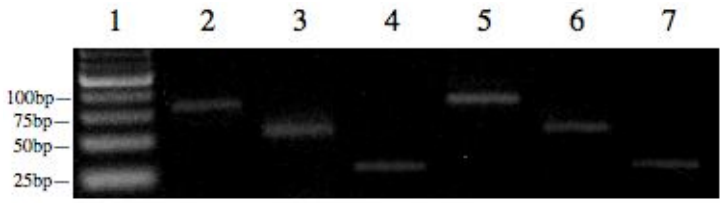

Fig. 5. Electrophoresis results. *Lane 1*: molecular size marker ladder (25bp). *Lane 2*: positive control by PCR $(X_1, \overline{X_6})$ (90bp) performed at 44°C. *Lane 3*: positive control by $PCR(X_2, \overline{X_5})$ (60bp) performed at 46°C. *Lane 4*: positive control by $PCR(Y_3, \overline{Y_4})$ (30bp) performed at 42°C. *Lane 5*: positive control by $PCR(Y_1, \overline{Y_6})$ (90bp) performed at 44°C. *Lane 6*: positive control by $PCR(Y_2, \overline{Y_5})$ (60pb) performed at 42°C. *Lane 7*: positive control by $PCR(X_3, \overline{X_4})$ (30pb) performed at 42°C.

Appendix (Experimental Protocols)

Reagents. 25 bp marker DNA ladder and agarose (Promega); PCR buffer, $MgCl_2$ and dNTP (Roche); Taq DNA Polymerase (produced in laboratory); all the synthetic DNA oligonucleotides 150 bp long and all the primers were from Primm s.r.l.(Milano, Italy).

Annealing of synthetic DNA oligonucleotides. Two complementary synthetic 150 bp long DNA oligonucleotides ($5' - 3'$ and $3' - 5'$) were incubated at 1:1 molar ratio at 90°C for 4 min in presence of 2.5 mM of $MgCl_2$ and then at 70°C for 10 min. The annealed oligos were slowly cooled to 37°C, then further cooled to 4°C until needed.

PCR amplification. PCR amplification was performed on a PE Applied Biosystems GeneAmp PCR System 9700 (Perkin Elmer, Foster City, CA) in a 50 μl final reaction volume containing 1.25U of Taq DNA Polymerase, 1.5 mM $MgCl_2$, 200 μM each dNTP, PCR buffer, 80 ng DNA template, 0.5-1 μM of

forward and reverse primers. The reaction mixture was preheated to 95∘C for 5 min. (initial denaturing), termocycled 30 times: 95°C for 30 sec (denaturing), different temperatures (see captures of the figures) for 30 sec. (annealing), 72°C for 15 sec. (elongation); final extension was performed at 72°C for 5 min.

Preparation and running of gels. Gels were prepared in 7×7 cm plastic gel cassettes with appropriate combs for well formation. Approximately 20 ml of 4% agarose solutions were poured into the cassettes and allowed to polymerize for 10 min. Agarose gels were put in the electrophoresis chamber and electrophoresis was carried out at 10 volt/cm^2, then the bands of the gels are detected by a gel scanner. The DNA bands (final PCR products) of interest were excised from the gel and the DNA was purified from the gel slices by Promega Kit (Wizard SV Gel and PCR Clean-Up System).

References

1. Adleman, L. M.: Molecular Computation of solutions to combinatorial problems. Science **266** (1994) 1021–1024
2. Braich, R. S., Chelyapov, N., Johnson, C., Rothemund, P. W. K., Adleman, L. M.: Solution of a 20-variable 3-SAT problem on a DNA computer. Science **296** (2002) 499–502
3. Braich, R. S., Johnson, C., Rothemund, P. W. K., Hwang, D., Chelyapov, N., Adleman, L. M: Solution of a Satisfiability Problem on a Gel-Based DNA Computers. A. Condon, G. Rozenberg eds, Proceedings of 6th International Workshop On DNA Based Computers, Leiden Netherlands, (2000) 31–42
4. Franco, G., Giagulli, C., Laudanna, C., Manca, V.: DNA Extraction by XPCR. C. Ferretti G. Mauri C. Zandron et al. eds, DNA 10, LNCS 3384, Springer-Verlag Berlin Eidelberg, (2005) 106–114
5. G. Franco, V. Manca, *Combinatorial Aspects of DNA Solution Spaces generated by XPCR Recombination*, in preparation.
6. Faulhammer, D., Cukras, A. R., Lipton, R. J., Landweber, L. F.: Molecular computation: RNA solution to chess problems. Proc. Natl. Acad. Sci. USA **98** (2000) 1385–1389
7. Head, T.: Formal language theory and DNA: An analysis of the generative capacity of specific recombinant behaviors. Bulletin of Mathematical Biology **49** (1987) 737–759
8. Lee, J. Y., Lim, H-W., Yoo, S-I., Zhang, B-T., Park, T. H.: Efficient Initial Pool Generation for Weighted Graph Problems Using Parallel Overlap Assembly. G. Mauri G. Rozenberg C. Zandron eds, Preliminary Proceedings of the 10th International Meeting on DNA Based Computers, Milan, Italy, (2004) 357–364
9. Lipton, R. J.: DNA solutions of hard computational problems. Science **268** (1995) 542–544
10. Jonoska, N., Sa-Ardyen, P., Seeman, N.C.: Computation by self-assembly of DNA graphs. Journal of Genetic Programming and Evolvable Machines **4** (2003) 123–137
11. Kaplan, P. D., Ouyang, Q. O., Thaler, D. S., Libchaber, A.: Parallel overlap assembly for the construction of computational DNA libraries. J. Theor. Biol. **188** (1997) 333–341

12. Manca, V., Zandron, C.: A Clause String DNA Algorithm for SAT. N. Jonoska N.C. Seeman eds, Proceedings of the 7th International Workshop on DNA-Based Computers: DNA 7, LNCS 2340, Springer, (2002) 172–181
13. Rose, J. A., Hagiya, M., Deaton, R. J., Suyama, A.: A DNA-based in vitro Genetic program. Journal of Biological Physics **28** 3 (2002) 493–498
14. Stemmer, W.: DNA shuffling by random fragmentation and reassembly: in vitro recombination for molecular evolution. Proc. Natl. Acad. Sci. USA **91** (1994) 10747–10751