Nat Comput
DOI 10.1007/s11047-010-9199-8

Algorithmic applications of XPCR

Giuditta Franco · Vincenzo Manca

Abstract An emerging trend in DNA computing consists of the algorithmic analysis of new molecular biology technologies, and in general of more effective tools to tackle computational biology problems. An algorithmic understanding of the interaction between DNA molecules becomes the focus of some research which was initially addressed to solve mathematical problems by processing data within biomolecules. In this paper a novel mechanism of DNA recombination is discussed, that turned out to be a good implementation key to develop new procedures for DNA manipulation (Franco et al., DNA extraction by cross pairing PCR, 2005; Franco et al., DNA recombination by XPCR, 2006; Manca and Franco, Math Biosci 211:282–298, 2008). It is called XPCR as it is a variant of the polymerase chain reaction (PCR), which was a revolution in molecular biology as a technique for cyclic amplification of DNA segments. A few DNA algorithms are proposed, that were experimentally proven in different contexts, such as, mutagenesis (Franco, Biomolecular computing—combinatorial algorithms and laboratory experiments, 2006), multiple concatenation, gene driven DNA extraction (Franco et al., DNA extraction by cross pairing PCR, 2005), and generation of DNA libraries (Franco et al., DNA recombination by XPCR, 2006), and some related ongoing work is outlined.

Keywords Algorithmic analysis · DNA extraction · DNA mutagenesis · DNA recombination · PCR · Splicing

1 Introduction

Information processing requires a strategy of systematic data transformation by basic operations. The definition of such operations depends on the type of data to which they apply, as well as on their organization within physical supports. According to the seminal idea of DNA computing (Adleman 1994; Lipton 1995), information can be stored in biopolymers, and enzymes (besides molecular biology and genetic engineering techniques)

G. Franco (✉) · V. Manca
Department of Computer Science, University of Verona, Strada Le Grazie 15, 37134 Verona, Italy
e-mail: giuditta.franco@univr.it

Published online: 28 April 2010

manipulate them in a massively parallel way, according to strategies producing universal computations (Franco and Manca 2005; Franco and Margenstern 2008; Păun et al. 1998). An algorithm is such a strategy, whereas a computational process is a procedure composed of the computations running onto a specific system.

In the last two decades, a more physically and biologically grounded theory has stimulated substantial changes in our understanding of computing and in design of "unconventional" computers. Universal computations of a Turing machine can be simulated at a molecular level (Rothemund 1996), where selective cutting, pasting (concatenating), reading, pairing and elongating strings are natural operations, most of the times performed by enzymes. Moreover, nature computes not only by sequence rewriting, but also by crossing-over, annealing, insertion-deletion and so on. Nonetheless, although biological sequence reading and writing are not so obvious as they are for formal strings, massive parallelism of computation and compactness of information storage are given for free by nature, and the limits to miniaturization of silicon technology do not exist anymore. Interestingly enough, novel algorithms are thus designed in order to make molecular computations robust, that are based on different, appropriate kinds of operations.

The formulation of theoretical molecular computational models (Conrad 1985; Head 1987; Păun et al. 1998) has spurred a production of actual implementations, favored by continuous improvements in underlying technologies. On the other hand, laboratory possibilities to manipulate molecules have offered a fresh paradigm for operations, and several models for DNA-based computations have been accordingly developed, including insertion–deletion systems (Kari and Thierrin 1996), splicing (finite, linear, circular, distributed) systems (Head 1987), sticker systems and molecular finite automata (Păun et al. 1998), forbidding-enforcing systems (Ehrenfeucht et al. 2000) and their variant for graphs to model DNA self-assembly (Franco and Jonoska 2005). A recent idea in this context is to use DNA as "fuel" other than as a structural material (Yurke et al. 2000), for construction of DNA machines, walkers, switchers, and DNA transducers (Goel et al. 2008).

The highly and genuinely interdisciplinary field of molecular computing is an evident example where theory and practice grow up together, just because of their interchange and complementarity, in a tight duality. A couple of good examples are represented by the development of self-assembly and DNA encoding theories. Self-assembly inspired models (Winfree et al. 2001) grew up in parallel with the actual production, by self-assembly experiments, of DNA cubes (Chen and Seeman 1991), octahedrons (Zhang and Seeman 1994), Borromean rings (Mao et al. 1999), complete graphs (Jonoska et al. 2003), and, most recently, cylinders and Möbius strips (Goel et al. 2008). Issues of DNA encoding, longly studied and mainly raised by experimental mismatch problems (Hussini et al. 2002), have been resolved by a couple of notable recent results. One has been the criterion according to which a longer double stranded DNA region is preferentially formed over one with a shorter double stranded region (Yurke et al. 2000), and the other one is the origami method, proposed to fold long single stranded DNA molecules into arbitrary two-dimensional shapes, by oligonucleotides "staple strands" (Rothemund 2006). Yet a free software may be found, on http://www.cdna.dk/origami, which imports any shape as design object, automatically finds a folding path and generates the 3D atomic model.

The potential of using DNA molecules for solving computationally hard problems has been traditionally addressed by DNA computing community, and extensively investigated from both experimental and theoretical points of view. Satisfiability problem (SAT) was perhaps the most widely studied NP-complete problem, of which theoretical solving methods may be found in (Manca and Zandron 2002; Rozenberg and Spaink 2003), while experimental solutions of small instances of SAT were proposed (for example) in (Braich

et al. 2002; Sakamoto et al. 1999). There have been several attempts to reduce the space and/or the time complexity of DNA algorithms solving NP-complete problems, until very recently (Goel et al. 2008). However, a scaling up of the proposed models from toy instances to realistic sizes remains an open problem, mainly because of the increasing quantity of DNA strands necessary to encode initial data. According to the results on this issue from literature, DNA computers cannot be competitive with computers in silicon for solving hard problems precisely.

In the streamline of some recent trends (which attempt to understand natural phenomena in terms of information processing (Kari and Rozenberg 2008)), we here investigated in a somewhat reverse direction, which could be defined as "Computing DNA" versus "DNA Computing". Nature is employed as a medium not only for computation but also to test concepts and techniques imported from computer science to understand and describe biomolecular processes.

DNA algorithms reported in subsequent sections, rather than focused on the solution of hard combinatorial problems (of computer science interest), are proposed as biotechnological procedures, aimed at suggesting new methods for DNA manipulations, deduced by combinatorial and computational analyses. Indeed, an attractive task nowadays is to characterize models within the scope of given experimental limitations, and to improve or obtain protocols that are sufficiently reliable, controllable and predictable. As a matter of fact, several standard biomolecular protocols do not ensure the needed precision for computation. Usually such techniques have to go through several adjustments, and sometimes completely new protocols are necessary in order to improve their yield. However, the goal of Computing DNA is rather to optimize the efficiency of biomolecular methodologies, not only to improve the reliability of computing, but also for medical and biological applications. Other research along this direction regards the study of whiplash PCR, a technique which allows a single strand state machine to perform a state transition by polymerase extension (Sakamoto et al. 1999). It is known to suffer from a serious efficiency problem called back-hybridization, and researchers are working to overcome it both by introduction of primers as external signals (Goel et al. 2008) and by exploiting the polymerase $\phi 29$, having an excellent displacement capability (Goel et al. 2008).

Analyzing the DNA molecule by non-biochemical principles but from a computational viewpoint, appeared helpful yet to understand its associative capacities (Reif et al. 2002), its recombinant behaviour (Ehrenfeucht et al. 2001; Manca and Franco 2008) and its own internal structure (in (Franco and Manca 2005; Manca 2005; Manca 2002) the double helix structure, together with the bilinearity, complementarity and antiparallelism of DNA molecules were proved to be a logical consequence of informational and computational features of efficient string duplication algorithms). A nice example of such an approach is the work (Manca et al. 2008), where a phenomenon related to the founding effect of evolution known as *genetic drift* was proved by a simple combinatoric argument related to the Asymptotic Equipartition Property (AEP) theorem of Shannon. Beside, the whole evolutive process was reconstructed in biomolecular terms, by exploiting PCR as a mechanism to grow biomolecular populations, and a drift process toward homogeneity of molecular types has been experimentally achieved. Laboratory experiments in progress are aimed to test that this evolution process is visible even at a higher scale, with microbial colonies, chosen as examples of biological systems which grow and can be sampled. Therefore, the design of molecular models of evolutive phenomena, to analyze the evolution of DNA molecules corresponding to the evolution of populations with given properties, is pointed out as a promising trend of Computing DNA.

2 DNA algorithms

As it has been briefly discussed in (Manca and Franco 2008), a feature common to molecular and quantum computations is the possibility to be observed and captured only at certain moments. In both cases indeed, one cannot control step by step what is the exact effect of the operations, and one alters the system when observing the result. A sound way to account for what happens between the moment of preparing an initial system and the moment of measuring it is to think of the complex components of the state as amplitudes, or proportions (rather than probabilities). As an example, a quantum coin can be in a state of both head and tail, in some proportions, simultaneously, until one measures it. This feature is just a mean of exploring several possibilities simultaneously (Arrighi 2003). The case of DNA operations is somehow similar: they are performed on *pools*, formally identified with multisets (sets with a positive multiplicity associated to their elements) of both strings and double strings over a given alphabet (of nucleotides). However, in a real pool we do not have the complete control of how the operation precisely affects the multiset. In terms of solving a mathematical problem, the output pool is only a mixed population of "solutions" and "other outcomes", and no theory of molecular computation may presume a single unambiguous, exact result of computation. Despite this kind of approximation, bio-algorithms driving system behaviours can be written such that the final pool of molecules contains the desired solution.

A key point for overcoming this apparent contradiction relies in the different levels on which computation and observation are performed. In fact, the microstate of a DNA pool P is given by the multiset of strings contained in the test tube, while the macrostate is the language $L(P)$ of strings which are observable. When observing a system, we can distinguish the types of molecules present in a sufficient quantity, but not the exact number of strands which instantiate them, and we can measure their length. Presences, relative concentrations, and lengths are observable macroscopic parameters which we modify by operations (e.g., amplification varies the quantitative level of specific molecules, and sometimes it is necessary just to detect the presence of such molecules), while massive and recombinant capabilities of computation depend on microscopic effects which are basilar in the complex internal dynamics of the system. Despite this unaffected microscopic non-determinism, main DNA operations work on populations, and they are deterministic in the effect they produce on macrostates. This factor ensures that encoding and decoding information into macrostates allow us to design reliable information elaborations. In other words, by leaving unknown the individual behaviour of molecules, DNA computation manipulates matter at a population level, by transforming macro-states of DNA pools, where behaviours become almost deterministic.

This approach seems to be a best way to really manipulate computations in nature, rather than trying to fit (mysterious) natural processes into pre-conceived, exact computational models. Another important feature of molecular computation is their validation by comparison with experimental results: laboratory experiments are crucial to test any DNA algorithm designed according to above considerations. Experimental and computational efficiency of DNA algorithms are then a further issue to be taken into account.

In this paper we would like to emphasize a foundational, algorithmic understanding of the interaction between DNA molecules in the PCR-based amplification process, along with the study proposed in Manca and Franco (2008), where a theorem classifying all the possible PCR outcomes was proved, and along with some novel experiments which were carried on for this purpose. In next sections we describe the XPCR procedure and recall the main algorithms designed to (i) extract from a pool all the strings containing as a substring

a given string γ (i.e., all γ-superstrings) (Franco et al. 2005) and (ii) produce an n-dimensional binary library, that is, all the possible sequences of type $\alpha_1\alpha_2\ldots\alpha_n$, where α_i can be one of two different strings X_i and Y_i, for each i from 1 to n (Franco et al. 2006). Moreover, we propose an XPCR-based algorithm to perform mutagenesis (Franco 2006), which replaces any occurrence of a given substring γ with another given string δ, within all the strings of a pool, and finally we show that even parallel XPCRs can be implemented, in order to concatenate several strings by overlapping.

Let us recall some notations we will use in the following, while the reader can refer to Păun et al. (1998) for basic notions of DNA computing and to Rozenberg and Salomaa (1997) for basic notions of formal language theory. The symbol of a string α that occurs at position i, with $1 \leq i \leq |\alpha|$ (where $|\cdot|$ gives the length of a string), is denoted by $\alpha(i)$, while the sequence of symbols of α occurring (in the given order) from position i to position j, with $1 \leq i \leq j \leq |\alpha|$, is denoted by $\alpha[i, j]$ (and it is a substring of α). In particular, it is called *prefix* if $i = 1$ and *suffix* if $j = |\alpha|$. A string having a given α as a prefix and a given β as a suffix is also indicated by $\alpha\ldots\beta$, where central dots represent an unknown non-empty string. $PCR_{(\xi,\eta)}$ (P) represents the amplification product of all the ξ-prefixed and η-suffixed (sub)strings present in the pool P, by means of PCR with primers ξ and $\bar{\eta}$ (i.e., the reverse Watson–Crick complementary strand of η). El_k (P) is a pool containing all the strings of length k of the pool P selected by gel electrophoresis. We denote by $mix(P_1, P_2)$ the union $P_1 \cup P_2$ of pools P_1 and P_2, and with $(P_1, P_2) = split$ (P) the production of two pools P_1 and P_2 such that $mix(P_1, P_2) = P$ and $L(P_1) = L(P_2) = L(P)$.

3 XPCR as a new technology

The idea of using PCR as a "very elegant and effective detection method" to check the existence of solutions of mathematical problems, solved in the DNA Computing context, was first considered in Rozenberg and Spaink (2003), where a theoretic method "to block" wrong sequences with PNA strands was proposed. A special type of PCR, called Cross Pairing PCR (shortly XPCR), was later introduced in Franco et al. (2005), as an efficient extraction method, experimentally tested in several situations. The simple technology of this approach turned out to be interesting both in itself and as an implementation tool for many DNA algorithms (Manca and Franco 2008), having useful applications in biological contexts, such as the efficient generation of DNA libraries (Franco 2006; Franco et al. 2006).

In formal language theory, a string rewriting rule of this form:

$$\alpha x\gamma y\beta, \alpha v\gamma w\beta \rightarrow \alpha x\gamma y\beta, \alpha v\gamma w\beta, \alpha x\gamma w\beta, \alpha v\gamma y\beta \tag{1}$$

is called *null context splicing rule* (Head 1987). Let us briefly call it γ-recombination.

$XPCR_\gamma$ procedure implements massively parallel γ-recombinations on couples of DNA molecules, along with a common substring γ, out of a heterogeneous pool of double stranded molecules having a same length n, a common prefix α and a common suffix β. Such properties of the pool are not restrictive, since enzymatic infix-operation, concatenating a common prefix or suffix to molecules of a pool, is standard in molecular biology, as well as discrimination by lengths (that allows the partition of molecules in sub-pools having a same length). Intuitively, $XPCR_\gamma$ works only on molecules of the form $\alpha x\gamma y\beta$ (that is, having γ as a substring), by copying and storing both the left-hand strand $\alpha x\gamma$ and the

Table 1 XPCR procedure

XPCR − Extraction (P, α, γ, β)
1. Let $(\mathbf{P_1}, \mathbf{P_2}) := \mathbf{split}(\mathbf{P})$;
2. copy in parallel the left-hand and right-hand parts of strings including γ:
$\mathbf{P_1} := \mathbf{PCR}_{(\alpha,\ \gamma)}\ (\mathbf{P_1})$ and $\mathbf{P_2} := \mathbf{PCR}_{(\gamma,\beta)}(\mathbf{P_2})$;
3. collect the strings produced in the previous step: $\mathbf{P} := \cup_{\mathbf{k}\ <\ \mathbf{n}}\mathbf{El_k}\ (\mathbf{P_1} \cup \mathbf{P_2})$;
4. restore by γ-recombination (1) the strings of the initial pool including γ:
$\mathbf{P} := \mathbf{PCR}_{(\alpha,\beta)}(\mathbf{P})$;
5. output is given by $\mathbf{P} := \mathbf{El_n}\ (\mathbf{P})$.

right-hand strand $\gamma y \beta$, and by finally recovering the initial string by concatenating these two strands by γ-overlapping.

For clarity of explanation, we assume that in the initial pool there do not exist two different molecules having the substring γ occurring at the same position. In other words, the string γ is located at different positions when it occurs as a substring of different molecules. As an example, by looking at the rule (1), if we start from a couple of different molecules of length n, since they have γ at a different position, after the application of a γ-recombination we have the same initial two molecules together with chimeras having a length different than n (one type will be shorter and the other one will be longer). There exists a way to free XPCR from this assumption without hindering its function, given by a more sophisticated extraction algorithm avoiding chimeras that may be found in (Manca and Franco 2008, p. 294).

If P is an input pool such that $L(P) = \{\alpha \ldots \beta || \alpha \ldots \beta| = n\}$, then the $XPCR_\gamma$ algorithm can be described as in Table 1. If we look at P after the first three steps of the algorithm, it contains all the strings $\alpha \ldots \gamma$ and $\gamma \ldots \beta$ which were present as substrings in the initial pool. In Fig. 1 the effect of step 2 is depicted, while, more interestingly, in Fig. 2 the combinatorial effect of step 4 is shown. As one can see in Fig. 2, differently from standard PCR the primers hybridize with single strands of two different dsDNA molecules, so releasing the respective partners in each molecule. At this point, these single strands can hybridize with each other by means of their (reversed) complementary parts γ and $\overline{\gamma}$, and polymerase uses the single strand components of this structure as templates in order to complete the double strings. In other words, the sequences $\alpha \ldots \gamma \ldots \beta$ are firstly generated by a sort of

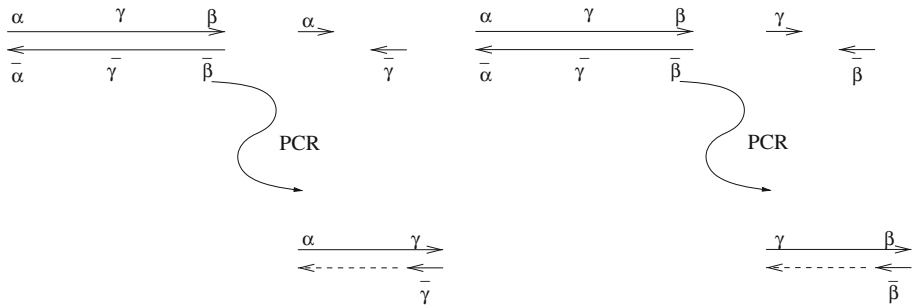


Fig. 1 Second step of XPCR_γ procedure (Franco et al. 2006)

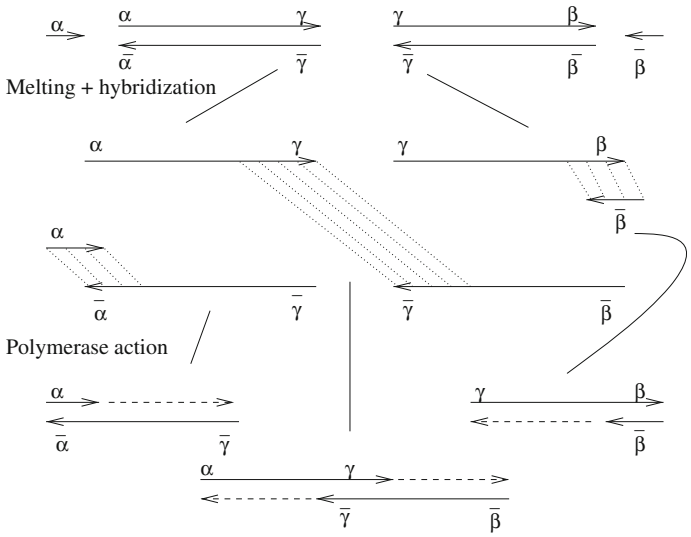


Fig. 2 Recombination basic step of XPCR$_\gamma$ (Franco et al. 2006)

overlapping juxtaposition of the two initial strands, and then amplified. This step has been verified in laboratory with three sizes of γ (229, 95, 15 bp) (Franco et al. 2005).

It is important to perform step 3 before the XPCR recombination step, otherwise all the strings which were not γ-superstrings would be amplified in the fourth step. An interesting feature is that in the recombination step there is no production of redundant material, because only the recombined (long) strings are amplified and the quantity of short molecules remains constant (Franco 2006). In fact, strands out of the initial short pieces in every step work as a template for generation of other short pieces, while the newly generated short pieces glue to form a long string including γ (see Fig. 2).

After the fourth step of the algorithm, P contains γ-superstrings of several lengths (and the splicing rule (1) at this point has been yet realized), and only those of length n (selected by step 5, which is essential for extraction) are those which were present in the initial pool. Indeed, all the strings with a different length obviously were not present in the pool, and those of length n were present because of the assumption above for clarity of explanation.

This algorithm was tested in vitro, where, from a very heterogeneous pool, all and only the types of strands having substrands of a given type were extracted (Franco et al. 2005; Manca and Franco 2008). Namely, a pool was considered containing both γ-superstrands and γ'-superstrands ($\gamma \neq \gamma'$), where all γ-superstrands were either γ_1-superstrands, or γ_2-superstrands, or γ_3-superstrands (γ_1, γ_2, γ_3 all different and 15 bp long). The experiment resulted in the correct and complete extraction (under several experimental conditions) of only γ-superstrands, and of all the three kinds of γ-superstrands (Franco et al. 2005).

Nevertheless, the XPCR technique should be furthermore validated and optimized in terms of laboratory conditions which guarantee its experimental reproducibility. Indeed, even if the steps of XPCR procedure are performed by means of standard and quite reliable techniques such as PCR and gel electrophoresis, experimental conditions to optimize the performance of the fourth step and to understand the limits of applicability have to be investigated.

More in details, we could wonder how much of primers α and $\bar{\beta}$ is required (relatively to the amount of either the γ-superstrings or all the molecules in the initial pool), and around which annealing temperature the PCR of the fourth step has better performance. Indeed, once the overlapped molecules have been formed, their amplification increases exponentially as for usual PCR, but to optimize the process we would like to create appropriate conditions to favor the overlapping formations rather than the linear amplification of short strings (including α or β). In order to get good enough amplifications (meaning to make the amplification product visible enough), in the experiments reported in (Franco et al. 2005, 2006) we decreased the annealing temperature (to 40–45°C), and this worked out with specific lengths of the strings. Then, the questions arise about what the length ranges for XPCR functioning are: how long can be the γ strings we are looking for, and how long the initial strings? Longer is γ, and longer should be taken the infixes α and β (because to perform step 2 it is required they have a similar annealing temperature): how PCR amplification and the overlapped molecules production behave with respect to these parameters? Some limitations of the efficiency of overlapped forms may be found in (Franco 2006; Lee et al. 2005). A Cooperint Project has been funded by Verona University for some ongoing work in collaboration with the Department of Biological Sciences, University of Binghamton, NY, on validation and optimization of XPCR, and efficient generation of DNA libraries.

Concatenation of two strings can be easily performed by XPCR (Franco 2006) as well as other operations which can be described in terms of splicing rules. In the next section we will discuss a few XPCR-based algorithms.

4 XPCR-based algorithms and experiments

We report here an XPCR-based procedure to generate large DNA libraries in linear time, that outperforms other methods in literature (Franco 2006; Franco et al. 2006). A theoretical XPCR generation algorithm working in logarithmic-time was designed as well (Franco 2005), but it was not experimentally implemented yet. Special strings called recombination witnesses were discovered, by argumentations of combinatorics, such that detecting their presence in the output pool guarantees that the whole library has been actually produced (Franco 2006; Manca and Franco 2008).

The goal here is to efficiently generate a DNA n-dimensional binary library, that is the pool of all the possible sequences of type $\alpha_1\alpha_2\ldots\alpha_n$, where α_i can be one of two different strings X_i and Y_i, for each i from 1 to n, by starting from as less sequences as possible (since sequence synthesis is significantly costly).

Our algorithm, introduced in Franco et al. (2006), is called quaternary algorithm as it starts from four sequences I_1 , I_2 , I_3 , I_4, each of which is of the form $\alpha_1\alpha_2\ldots\alpha_n$, where α_i can assume a binary value. For the sake of simplicity, we describe this method for a solution space of dimension $n = 6$. It may be easily generalized to any dimension n, and even to the non-binary case, where α_i can assume k different values with $k > 2$ (while the initial number of sequences increasing as k^2) (Franco 2006).

Let us then consider the sequences: $I_1 = X_1\ X_2\ X_3\ X_4\ X_5\ X_6$, $I_2 = Y_1\ Y_2\ Y_3\ Y_4\ Y_5\ Y_6$, $I_3 = X_1\ Y_2\ X_3\ Y_4\ X_5\ Y_6$, $I_4 = Y_1\ X_2\ Y_3\ X_4\ Y_5\ X_6$. Intuitively, it is clear that any string $\alpha_1\alpha_2\alpha_3\alpha_4\alpha_5\alpha_6$ can be obtained as the concatenation of substrings of I_1 , I_2 , I_3 , I_4, and then produced by copying and recombining along suitable common substrings. As an instance, the sequence $X_1\ X_2\ Y_3\ X_4\ Y_5\ Y_6$, can be namely obtained by applying an X_2-recombination to the couple I_1, I_4 (according to the rule (1)), and by a subsequent Y_5-recombination

Table 2 An XPCR recombination algorithm to generate libraries

XPCR − Recombination (P, α, X_2, $Y_2, \ldots, X_5$, Y_5, β)

Let $L(P) = \{\alpha I_1\beta, \alpha I_2\beta, \alpha I_3\beta, \alpha I_4\beta\}$;
for $i = 2, 3, 4, 5$ **do**
1. $(\mathbf{P_1}, \mathbf{P_2})$:= **split**(**P**);
2. $\mathbf{P_1}$:= $\mathbf{XPCR_{X}}_i$ ($\mathbf{P_1}$) and $\mathbf{P_2}$:= $\mathbf{XPCR_{Y}}_i$($\mathbf{P_2}$);
3. **P** := **mix** ($\mathbf{P_1}$, $\mathbf{P_2}$);

applied to the molecules I_2 and $X_1\ X_2\ Y_3\ X_4\ Y_5\ X_6$, the last one obtained as a product of the X_2-recombination.

Therefore, the output pool may be identified with the null context splicing language (Head 1987) deduced by applying the X_i/Y_i-recombinations, for every $i = 2, \ldots, 5$ (in general up to $n - 1$), to the axioms $\{I_1, I_2, I_3, I_4\}$ (Franco 2006). The generation algorithm is reported in Table 2, and a laboratory experiment validating the method was reported in (Franco et al. 2006). Currently, there is some work in progress at Binghamton University, NY, to built up a DNA library of 4^7 elements by a generalized version of this method. In experimental terms, the largest generated (20-dimensional) DNA library dates back to 2002 (Braich et al. 2002) and it was a product of the sophisticated mix and split method. A 12-dimensional library was constructed on the basis of thermodynamical data (Penchovsky and Ackermann 2003), whereas, more recently, cheap overlapping methods to produce libraries are under development (e.g., a variant of POA in (Gal et al. 2008) and the XPCR quaternary method in (Franco et al. 2006)).

According to our algorithm, variables X and Y are implicitly assumed to have the same length. In algorithmic terms, this is not a must of the generation method, because binary libraries encoded by variables of different length would be as well generated by redefining the XPCR procedure (in Table 1) by eliminating both steps 3 and 5. In this case, indeed, we would not have any filter by length, and the generation algorithm could work equivalently on strings of different length, with the only disadvantage to produce a relatively reduced amplification of newly formed molecules (at step 2). In terms of laboratory implementation instead, there are significant constraints for the admissible variable lengths. In fact, variables are employed as primers by XPCR, then they all need to have an annealing temperature comparable with those of α and of β in order PCR amplification to work, and this definitely limits the range of acceptable lengths for variables X and Y. The exact limitations in this context need further investigation, which can be addressed along the questions posed in the final part of Sect. 3.

Differently than for the extraction problem, in the case of generation algorithm we are more interested in the formation of chimeras (new molecules), when applying the rule (1), than in the retrieval of the initial ones. It would be interesting in this context to evaluate the relative concentrations of the two chimeras as products of XPCR procedure: are they formed with a same probability? In biochemical terms, another open question could be: do the "5′–5′" overlappings have the same probability to be formed as the "3′–3′" ones?

In principle, XPCR recombination methods could be applied to gene shuffling. In fact, some experimental results significantly imply that the gene order within an operon of biosynthetic pathway is a key element to precisely tune the final amount of the protein product. These studies revealed also the importance of gene disposition in a field where

even the functional role and the reciprocal influences of single genetic network components are still unknown.

In the following we propose novel XPCR-based mutagenesis and concatenation methods as combinatorial algorithms validated by a few experiments.

4.1 XPCR mutagenesis

Once strings including γ are extracted, one could be interested in performing a mutagenesis process, that is, to obtain from a pool of type $\{\alpha\gamma\beta\}$ an output pool of type $\{\alpha\delta\beta\}$ where the substrands of type γ are replaced by substrands of type δ. In the special case where γ is the empty word, we have 'insertional mutagenesis', that is used to block the expression of a gene because such mutagenesis may cause the loss of functionality. In general, mutagenesis in vitro of cloned genes is a standard tool for functional analysis of polymers like DNA and proteins. Namely, randomly distributed mutations are first induced by mutagenesis to identify a functional region in a given fragment, and then the functionality of a specific region is analyzed by altering its sequence by 'situ-specific' (that is, site directed) mutagenesis. Traditionally, techniques used to perform mutagenesis have been restriction enzymes or other kinds of enzymes, oligo synthesis, hybridization and sequencing. Recently, a rapid and efficient polymerase chain reaction-based strategy for one/two-site mutagenesis has been proposed in Gong et al. (2004).

An algorithm based on XPCR that performs mutagenesis operation was tested in vitro with a sequence extracted from the human gene RhoA (a smallGTPase),where a substring of 123 bp was replaced by a substring of 150 bp. A pool of type $\{\alpha\gamma\beta\}$ was transformed, by the algorithm proposed in Table 3 (negative indexes of a substring mean that we count the positions in the backward order, from the right side to the left side of the string), into a pool of type $\{\alpha\delta\beta\}$; where $\alpha\gamma\beta$ is the sequence we briefly call RhoA (actually it is RhoA[152,733] that was extracted from the human gene RhoA), γ is the 123 bp long string starting at position 231 of the RhoA, and δ is a string 150 bp long. If we call α_1 and α_2, respectively, the prefix and the suffix 18 bp long of α, and β_1, β_2, respectively, the prefix and the suffix of β with length 20 bp, a flowchart of the XPCR mutagenesis algorithm can be described as in Table 4. PCR and XPCR results of XPCR-Mutagenesis algorithm are indicated in Fig. 3.

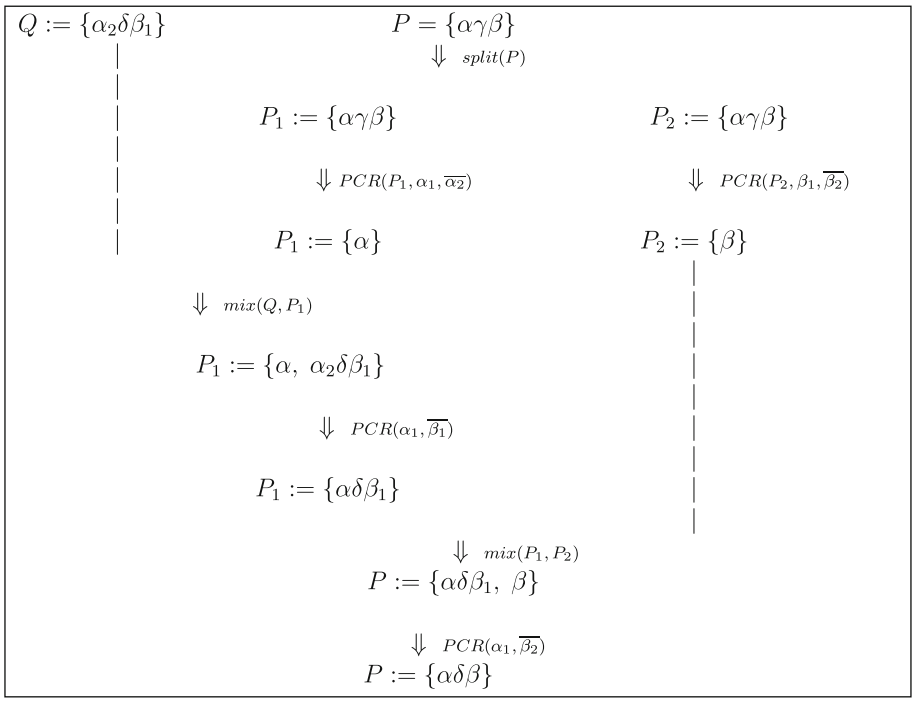
Table 3 An XPCR mutagenesis algorithm

XPCR − Mutagenesis(P , γ, δ)

Let $L(P) = \{\alpha\gamma\beta\}$ and $L(Q) = \{\alpha[-18, -1]\ \delta\ \beta[1, 20]\}$;

1. $(\mathbf{P_1}, \mathbf{P_1}) := \text{split}(\mathbf{P})$;
2. $\mathbf{P_1} := \mathbf{PCR}_{(\alpha[\mathbf{1,18}],\alpha[\mathbf{-18,-1}])}(\mathbf{P_1})$ and $\mathbf{P_2} := \mathbf{PCR}_{(\beta[\mathbf{1,20}],\beta[\mathbf{-20,-1}])}(\mathbf{P_2})$;
3. $\mathbf{P_1} := \mathbf{El}_{|\alpha|}(\mathbf{P_1})$ and $\mathbf{P_2} := \mathbf{El}_{|\beta|}(\mathbf{P_2})$;
4. merge the first of the obtained pools with Q, that is, $\mathbf{P_1} := \mathbf{mix}(\mathbf{P_1}, \mathbf{Q})$;
5. $\mathbf{P_1} := \mathbf{PCR}_{(\alpha[\mathbf{1,18}],\ \beta[\mathbf{1,20}])}(\mathbf{P_1})$;
6. $\mathbf{P_1} := \mathbf{El}_{|\alpha|+|\delta|+\mathbf{20}}(\mathbf{P_1})$;
7. $\mathbf{P} := \mathbf{mix}(\mathbf{P_1}, \mathbf{P_2})$;
8. $\mathbf{P} := \mathbf{PCR}_{(\alpha[\mathbf{1,18}],\ \beta\ [\mathbf{-20,-1}])}(\mathbf{P})$;
9. output is given by $\mathbf{P} := \mathbf{El}_{|\alpha|+|\beta|+|\delta|}(\mathbf{P})$.

Table 4 Flowchart of an XPCR mutagenesis algorithm transforming a pool P = $\{\alpha\gamma\beta\}$ into a pool $\{\alpha\delta\beta\}$, where $\alpha = \alpha_1 \ldots \alpha_2$ and $\beta = \beta_1 \ldots \beta_2$



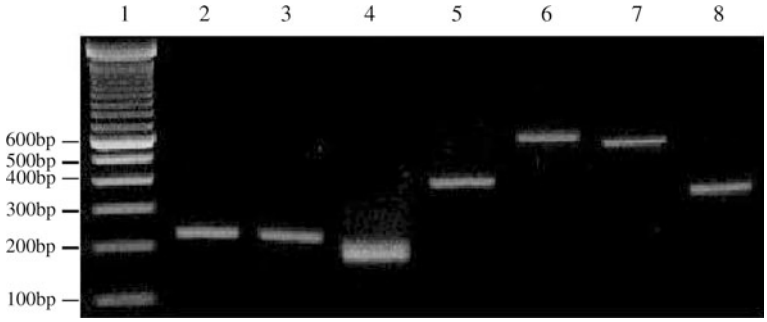


Fig. 3 DNA mutagenesis. Intermediate and final products of the XPCR metagenesis algorithm described in Table 4 are here reported. *Lane 1* molecular size marker ladder (100 bp). *Lane 2* amplification of strand α (230 bp). *Lane 3* amplification of strand β (229 bp). *Lane 4* amplification of strand $\alpha[-18, -1]\delta\beta[1, 20]$ (188 bp). *Lane 5* cross pairing amplification of α and $\alpha[-18, -1]\delta\beta[1, 20]$ (400 bp). *Lane 6* cross pairing amplification of β and $\alpha\delta\beta[1, 20]$ (609 bp). *Lane 7* RhoA (582 bp). *Lane 8* amplification by PCR ($\alpha[26, 46]$, $\beta[-20, -1]$) 354 bp. All PCRs are performed at temperature 58°C

4.2 Multiple XPCR

Assembly of multiple DNA fragments in few steps is relevant for reconstruction of complex genomes, and most of successful techniques use plasmids (Pachuk et al. 2000).

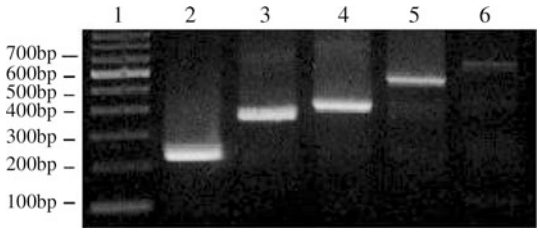


Fig. 4 Multiple XPCR results. Multiple concatenation by XPCR has been experimentally performed as a first attempt to assemble four, five, six, seven, and eight strings. *Lane 1* molecular size marker ladder (100 bp). *Lane 2* successful 4-XPCR of strands α_5, α_6, α_7, α_8 (217 bp); *lane 3* successful 5-XPCR of strands α_4, α_5, α_6, α_7, α_8 (356 bp); *lane 4* successful 6-XPCR of strands α_3, α_4, α_5, α_6, α_7, α_8 (407 bp); *lane 5* successful 7-XPCR of strands α_2, α_3, α_4, α_5, α_6, α_7, α_8 (538 bp); *lane 6* not good enough amplification of an 8-XPCR on strands α_1, α_2, α_3, α_4, α_5, α_6, α_7, α_8 (626 bp), all at annealing temperature of 56°C. Strands α are all reported in the final section

In our contexts, multiple forms of XPCR also work, referred to as n-XPCR when n different types of DNA double strings are concatenated in one step. For example, when $n = 4$, given a pool P such that $L(P) = \{\alpha\delta_1\gamma_1, \gamma_1\delta_2\gamma_2, \gamma_2\delta_3\gamma_3, \gamma_3\delta_4\beta\}$, $PCR_{(\alpha,\beta)}(P)$ provides an exponential amplification of the string $\alpha\delta_1\gamma_1\delta_2\gamma_2\delta_3\gamma_3\delta_4\beta$. We achieved good experimental results by applying multiple XPCR method to concatenate from $n = 4$ up to $n = 7$ strings (see lanes 2, 3, 4, 5 of Fig. 4). The experiment to test an n-XPCR with $n = 8$ was not working as expected, and the scarcely visible amplification is reported in lane 6 of Fig. 4. Hence, further experiments have to be carried out in order to check the feasibility and the performance of n-XPCR for $n \geq 8$. Nevertheless, the result obtained for seven strings means that, in principle, from initial types

$$\{\alpha\delta_1\gamma_1, \gamma_{48}\delta_{49}\beta\} \cup \{\gamma_i\delta_{i+1}\gamma_{i+1} \mid i = 1, \ldots, 47\}$$

where any δ_i may have a binary value, we could achieve a recombination of 2^{49} different types of molecules, by seven parallel 7-XPCRs, and by a further 7-XPCR on a pool where the seven previous results are mixed up together (that is, in the time of two usual PCRs).

An application of our extraction methods based on XPCR is of interest for Department of Cellular Therapy and Hematology, San Bortolo Hospital, Vicenza, Italy, where pathologies related to multi-loci mutations of patients' genes can be revealed by applying multiple XPCR methods. Indeed, while PCR retrieves strings containing a couple of ordered substrings, XPCR retrieves strings containing three given substrings, and multiple XPCR retrieves strings having n ordered substrings. Some work in progress in Vicenza is testing the performance of real time XPCRs (XPCR methods where PCRs are executed as RT-PCRs).

5 Perspectives of computing DNA

In this work we have presented the standpoint of a new methodology, which is based on discrete mathematical models (and computer simulations) of DNA recombination. We discussed the application of XPCR-based methodologies in different contexts, such as DNA extraction, recombination, and mutagenesis. Open problems and current projects focussing on future developments of the applications of XPCR have been discussed along with the presented research. The efficiency of XPCR in manipulating DNA under critical

conditions may open completely new perspectives in different research fields such as DNA computing and molecular biology, as well as in medicine, by facilitating the identification of genetic mutations, recombinations and polymorphisms in human diseases. We think that our work could suggest new investigations of "Computing DNA" where discrete and combinatorial aspects could be the key for a new comprehension of bio-molecular phenomena. More in general, we think that investigating the potential and the limitations of algorithmic methods in the study of bioprocesses can discover new and interesting features of natural computation.

Experimental protocols and DNA sequences

Experimental protocols and all the DNA sequences employed in the experiments described above are reported in the following.

Reagents. 25 and 100 bp marker DNA ladder and agarose (Promega); PCR buffer, $MgCl_2$ and dNTPs (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 1.25 U 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), thermocycled 30 times: 95°C for 30 s (denaturing), different temperatures (see figures) for 30 s (annealing), 72°C for 15 s (elongation); final extensions were performed at 72°C for 5/10 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 V/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).

DNA sequences employed in XPCR mutagenesis experiment (Fig. 3)

δ = GCA GTC GAA GCT GTT GAT GCC AAG AGA TGG TCG TCT GCT AGC ATG TCA CGC CAC GGA ACG GTG AGC GCG AGT GTG ATA TGC AAT GAT CTG ATC CGT CCC GAT AAG TAT TGT CAC GCA TCG GTA CGT AAC TAG ACG CTG CCG TAG TCG ACG, $\alpha[1, 18]$ = ATGGCTGCCATCCGGAAG, $\overline{(\alpha[-18,-1])}$ = GTA TCT GGG TAG GAG AGG, $\alpha[-18,-1]$ = CCT CTC CTA CCC AGA TAC, $\beta[1, 20]$ = GAA GGA TCT TCG GAATGATG, $\overline{(\beta[1,20])}$ = CAT CAT TCC GAA GAT CCT TC, $\overline{(\beta[-20,-1])}$ = TCA CAA GAC AAG GCA ACC AG, $\alpha[-18, 1]$ $\delta[1, 19]$ = CCT CTC CTA CCC AGA TAC GCA GTC GAA GCT GTT GATG, $\delta[-17,-1]\beta[1, 20]$ = CGC TGC CGT AGT CGA CGG AAG GAT CTT CGG AAT GATG, $\alpha[26, 46]$ = GAT GGT CGT CTG CTA GCA TG.

DNA sequences employed in multiple XPCR experiment (Fig. 4)

Extremal parts γ are separated by a hyphen: α_1 = GCA GAT ATC GAG GTG GAT G-GA AAG CAG GTA GAG TTG GCT TTG TGG GAC ACA GCT GGG CAG GAA GAT TAT GAT CGC CTG AGG CC-C CTC TCC TAC CCA GAT AC, α_2 = CCT CTC CTA CCC AGA TAC - CGA TGT TAT ACT GAT GTG TTT TTC CAT CGA CAG CCC TGA

TAG TTT AGA AAA CAT CCC AGA AAA GTG GAC CCC AGA AGT CAA GCA TTT CTG TCC CAA CGT GCC CAT CAT CCT GGT TGG GAA TAA - GAA GGA TCT TCG GAA TGA TG, $\alpha_3 =$ GAA GGA TCT TCG GAA TGA TGA GCA CAC AAG GCG GGA GC-T AGC CAA GAT GAA GCA GGA G, $\alpha_4 =$ TAG CCA AGA TGA AGC AGG AG-C CGG TGA AAC CTG AAG AAG GCA GAG ATA TGG CAA ACA GGA TTG GCG CTT TTG GGT ACA TGG AGT GTT CAG CAA AGA CCA AAG ATG GAG TGA GAG AGG TTT TTG AAA TGG CTA CGA GAG C-TG CTC TGC AAG CTA GAC GTG G, $\alpha_5 =$ TGC TCT GCA AGC TAG ACG TGG - GAA GAA AAA ATC TGG TTG CCT TGT CTT GTG A, $\alpha_6 =$ CTG GTT GCC TTG TCT TGT GA-T TC-C TCT GAG ATA AGT TTC TGT TC, $\alpha_7 =$ CTC TGA GAT AAG TTT CTG TTC - TCA T-GC AGT CGA AGC TGT TGA TGC, $\alpha_8 =$ GCA GTC GAA GCT GTT GAT GC-C AAG AGA TGG TCG TCT GCT AGC ATG TCA CGC CAC GGA ACG GTG AGC GCG AGT GTG ATA TGC AAT GAT CTG ATC CGT CCC GAT AAG CAA GTC AGA TTG ACC GCA CGT AAC TA-G ACG CTG CCG TAG TCG ACG.

Acknowledgements The experimental research reported in this paper was funded by the Italian National Research Project FIRB 2003, RBA01PHZS. The experiments were performed at the laboratories of the Department of General Pathology at University of Verona, and the authors wish to thank Cinzia Giagulli and Carlo Laudanna for their technical support. They are also grateful for very helpful comments and suggestions of anonymous referees. The first author is thankful for the interesting discussions with Susannah Gal and Tony Macula (from SUNY Binghamton University, USA) about experimental and combinatorial analyses of XPCR protocol.

References

Adleman LM (1994) Molecular computation of solutions to combinatorial problems. Science 266: 1021–1024

Arrighi P (2003) Quantum computation explained to my mother. Bull EATCS 80:134–142

Braich RS, Chelyapov N, Johnson C, Rothemund PWK, Adleman L (2002) Solution of a 20-variable 3-SAT problem on a DNA computer. Science 296:499–502

Chen JH, Seeman NC (1991) Synthesis from DNA of a molecule with the connectivity of a cube. Nature 350:631–633

Conrad M (1985) On design principles for a molecular computer. Commun ACM 28:464–480

Ehrenfeucht A, Hoogeboom HJ, Rozenberg G, van Vugt N (2000) Forbidding and enforcing. In: Winfree E et al (eds) DNA based computers V, DIMACS Series AMS, vol 54. pp 195–206

Ehrenfeucht A, Prescott DM, Rozenberg G (2001) Computational aspects of gene (un)scrambling in ciliates. In: Landweber LF et al. (eds) Evolution as computation. Springer, Berlin, pp 216–256

Franco G (2005) A polymerase based algorithm for SAT. In: Coppo M et al (eds) Proceedings ICTCS 2005, LNCS 3701. Springer-Verlag, pp 237–250

Franco G (2006) Biomolecular computing—combinatorial algorithms and laboratory experiments. PhD Thesis, University of Verona, Italy

Franco G, Giagulli C, Laudanna C, Manca V (2005) DNA extraction by cross pairing PCR. In: Ferretti C et al. (eds) Revised selected papers from DNA 10, LNCS 3384. Springer-Verlag, pp 106–114

Franco G, Jonoska N (2005) Forbidding—enforcing conditions in DNA self-assembly. In: Chen J, Jonoska N et al. (eds) Nanotechnology, science and computation, pp 105–118

Franco G, Manca V (2005) An algorithmic analysis of DNA structure. Soft Comput 9(10):761–768

Franco G, Manca V, Giagulli C, Laudanna C (2006) DNA recombination by XPCR. In: Carbone A et al (eds) Revised Selected Papers from DNA 11, LNCS 3892. Springer-Verlag, pp 55–66

Franco G, Margenstern M (2008) A DNA computing inspired computational model. TCS 404:88–96

Gal S, Monteith N, Macula AJ (2008) Successful preparation and analysis of a 5-site 2-variable DNA library. Nat Comput 8(2):333–347

Goel A, Simmel FC, Sosik P (eds) (2008) In: Preliminary proceedings of the 14th international meeting on DNA computing, Prague, Czech Republic

Gong Z, Zhang H, Gabos S, Li XF (2004) Rapid and efficient polymerase chain reaction-based strategies for one-site and two-site mutagenesis. Anal Biochem 331:404–406

Head T (1987) Formal language theory and DNA: an analysis of the generative capacity of specific recombinant behaviors. Bull Math Biol 49:737–759
Hussini S, Kari L, Konstantinidis S (2002) Coding properties of DNA languages. In: Jonoska N et al (eds) Revised selected papers from DNA 7, LNCS 2340, Springer-Verlag. pp 57–69
Jonoska N, Sa-Ardyen P, Seeman NC (2003) Computation by self-assembly of DNA graphs. J Genet Progr Evolvable Mach 4(2):123–137
Kari L, Rozenberg G (2008) The many facets of natural computing. Commun ACM 51(10): 72–83
Lee JY, Lim HW, Yoo S-I, Zhang BT, Park TH (2005) Efficient initial pool generation for weighted graph problems using parallel overlap assembly. In: Ferretti C et al (eds) Revised selected papers from DNA 10, LNCS 3384. Springer-Verlag, pp 215–223
Lipton RJ (1995) DNA solutions of hard computational problems. Science 268:542–544
Manca V (2005) On the logic and geometry of bilinear forms. Fundamenta Informaticae 64:261–273
Manca V (2002) On the logic of DNA bilinearity. In: Hagiya M et al (eds) Preliminary proceedings of DNA 8, pp 330
Manca V, Franco G (2008) Computing by polymerase chain reaction. Math Biosci 211:282–298
Manca V, Franco G, Lampis S, Vallini G (2008) The phenomenon of sampling and growing in bio-populations. In: Extended abstract in proceedings of the 14th international meeting on DNA computing, Prague, Czech Republic
Manca V, Zandron C (2002) A clause string DNA algorithm for SAT. In: Jonoska N et al (eds) Revised selected papers from DNA 7, LNCS 2340. Springer-Verlag, pp 172–181
Mao C, Sun W, Seeman NC (1999) Designed two-dimensional holliday junction arrays visualized by atomic force microscopy. J Am Chem Soc 121:5437–5443
Kari L, Thierrin G (1996) Contextual insertions/deletions and computability. Inf Comput 131(1):47–61
Pachuk CJ, Samuel M, Zurawski JA, Snyder L, Phillips P, Satishchandran C (2000) Chain reaction cloning: a one-step method for directional ligation of multiple DNA fragments. Gene 243:19–25
Penchovsky R, Ackermann J (2003) DNA library design for molecular computation. J Comput Biol 10(2):215–230
Păun Gh, Rozenberg G, Salomaa A (1998) DNA computing. New computing paradigms. Springer, Berlin
Reif JH, LaBean TH, Pirrung M, Rana VS, Guo B, Kingsford C, Wickham GS (2002) Experimental construction of very large scale DNA databases with associative search capability. In: Jonoska N et al (eds) Revised selected papers from DNA 7, LNCS 2340, Springer-Verlag, pp 231–247
Rothemund PWK (1996) A DNA and restriction enzyme implementation of turing machines. In: Lipton RJ et al (eds) DNA based computers. Proceedings of a DIMACS 27. Princeton University, American Mathematical Society, pp 75–119
Rothemund P (2006) Folding DNA to create nanoscale shapes and patterns. Nature 440:297–302
Rozenberg G, Salomaa A (1997) Handbook of formal languages, vol 3. Springer-Verlag, Berlin
Rozenberg G, Spaink H (2003) DNA computing by blocking. Theoret Comput Sci 292:653–665
Sakamoto K, Kiga D, Komiya K, Gouzu H, Yokoyama S, Ikeda S, Sugiyama H, Hagiya M (1999) State transitions by molecules. In: Kari L et al (eds) Bio systems—special issue. Proceedings of The fourth international meeting on DNA based computers, vol 52, pp 81–91
Yurke B, Turberfield A, Mills A, Simmel F, Neumann J (2000) A DNA-fuelled molecular machine made of DNA. Nature 406:605–608
Winfree E, Eng T, Rozenberg G (2001) String tile models for DNA computing by self-assembly. In: Condon A et al (eds) Revised selected papers from DNA 6, LNCS 2054, Springer, pp 63–88
Zhang Y, Seeman NC (1994) The construction of a DNA truncated octahedron. J Am Chem Soc 116: 1661–1669