BioSystems, 30 (1993) 1–19
Elsevier Scientific Publishers Ireland, Ltd.

Computation of biopolymers: A general approach to different problems

A. V. Finkelstein[a] and M. A. Roytberg[b]
[a]*Institute of Protein Research, Russian Academy of Sciences, Pushchino, Moscow region, 142292, Russia and*
[b]*Institute of Mathematical Problems of Biology, Russian Academy of Sciences, Pushchino, Moscow region, 142292, Russia*

A comparative analysis of some effective algorithms widely used in analysis, computation and comparison of chain molecules is presented. A notion of a stream in an oriented hypergraph is introduced, which generalizes a notion of a path in a graph. All considered algorithms looking over exponential sets of structures in polynomial time can be described as variants of a general algorithm of analysis of paths in graphs and of streams in oriented hypergraphs.

Introduction

Solution of some important problems of molecular biology (e.g. prediction of three-dimensional structures of biopolymers from their primary structures and alignment of nucleotie and amino acid sequences) employs algorithms which can analyze a set of $\sim e^M$ structures by the time $\sim M$ to $\sim M^3$. We have in mind, first, matrix methods of the statistical physics of one-dimensional systems, which are used to determine the properties of the thermodynamically equilibrium state, and, second, the apparatus of dynamic programming used to find out the state of the minimum "energy". These algorithms can be applied to systems in which any arising bond divides the system into two independent parts. It is this independence that allows one to process an exponentially large set of structures in polynomial time (cf. Romanovskiĭ, 1977).

From the formal point of view, many of these algorithms cab be reduced to "summation" of paths in graphs. However, this scheme is not universal, since it does not include, for example, algorithms for RNA secondary structure prediction (Nussinov et al., 1978; Zuker, 1989). Here we propose a new notion: "summation" of streams in directed hypergraphs. This allows us to include all known to us algorithms of this kind into a single general scheme.

Willing to make the paper understandable to a general reader, we present both the necessary mathematical definitions and statements of biological problems together with sketches of algorithms that solve them.

1. Dynamic programming. Search for the optimal alignment of two sequences

The dynamic programming emerged as a general approach to optimization of multistage processes (Dreyfus, 1961; Angel and Bellman, 1972), e.g. in management of economic systems. This method is used also for the numerical solution of variational problems, in particular, for search for stable configurations of physical fields. In molecular biology the method is used for the sequence homology search (Needleman and Wunsch, 1970) and for determination of energetically optimal structures of macromolecules.

In the fundamental Bellman statement (Angel and Bellman, 1972) the method of dynamic programming reduces to the following.

A complete *path* consists of $M-1$ steps (in time, space or another parameter stemming from a particular problem). Let q_i be a *state* before the i-th step ($i = 1, 2, \dots, M-1$) and let q_{i+1} be a state after it (and,

0303-2647/93/$ 06.00 © 1993 Elsevier Scientific Publishers Ireland, Ltd.
Printed and Published in Ireland

respectively, before the next step if $i < M - 1$). The states q_i form a finite set $\{q_i\}$. Possible directions of the i-th step are determined by *control vectors* y_i from a set $\{y_i\}$ so that

$$q_{i+1} = q_{i+1}(q_i, y_i), \quad , i = 1, \dots , M - 1. \tag{1}$$

Each step produces a "profit"

$$\tilde{r}_i(q_i, y_i) = r_{i,i+1}(q_i, q_{i+1}(q_i, y_i)),$$

so that the object function (total profit) on a path $q_1 \to q_2(q_1, y_1) \to q_3(q_2, y_2) \to \dots \to q_M(q_{M-1}, y_{M-1})$ is

$$\tilde{Y}(q_1, y_1, \dots , y_{M-1}) = Y(q_1, \dots , q_M) = \sum_{i=1}^{M-1} r_{i,i+1}(q_i, q_{i+1}). \tag{2}$$

The problem is to find the maximum of this object function (the maximum profit)

$$\Phi = \max_{q_1, \dots , q_M} \left\{ \sum_{i=1}^{M-1} r_{i,i+1}(q_i, q_{i+1}) \right\}. \tag{3}$$

as well as the *optimal path* $Q^{\text{opt}} = (q'_1, q'_2, \dots , q'_M)$ that leads to this maximum, assuming condition (1) to be satisfied.

In order to solve this problem, for each state of the system the following functions are introduced

$$R_i(q_i) = \max_{q_{i+1}, \dots , q_M} \left\{ \sum_{j=i}^{M-1} r_{j,j+1}(q_j, q_{j+1}) \right\},$$

that describe the maximum profit on paths coming from this state and satisfying transition rules (1). The functions R_i can be easily computed in the recurrent manner:

$$R_M(q_M) = 0, \quad \text{for all } q_M,$$
$$R_i(q_i) = \max_{y_i} \{ r_{i,i+1}(q_i, q_{i+1}(y_i)) + R_{i+1}(q_{i+1}(y_i)) \}, \quad (i = M - 1, \dots , 1). \tag{4}$$

In course of these computations one determines also the optimal control vectors $y_i^{\text{opt}}(q_i)$ corresponding to maxima in (4), and retains the optimal transitions $q_{i+1}^{\text{opt}}(q_i) = q_{i+1}(q_i, y_i^{\text{opt}})$ for all $\{q_i\}$, $i = M - 1, \dots , 1$. Now it is possible to find the maximum profit

$$\Phi = \max_{q_i} \{ R_1(q_1) \}, \tag{5}$$

the corresponding state q_1^{opt} that is the beginning of the optimal path, and, finally, the entire optimal path Q^{opt}:

$$q'_1 = q_1^{\text{opt}}, q'_2 = q_2^{\text{opt}}(q'_1), \dots , q'_M = q_M^{\text{opt}}(q'_{M-1}). \tag{6}$$

It should to noted that there can exist several optimal paths of equal value. The Bellman algorithm finds only one of them. It is sufficient for practical (engineering and economics) problems, but often insufficient for analysis of natural objects. Some elaboration of the statement of the problem and the algorithm (Waterman and Byers, 1984) allows one to find all optimal paths (as well as all "suboptimal" paths whose weights differ from the optimal one by a given value or less).

The Bellman problem is often formulated in the graph theory language. Graph vertices (Fig. 1) correspond to states, and arcs correspond to control vectors. Each arc has a weight equal to the profit from

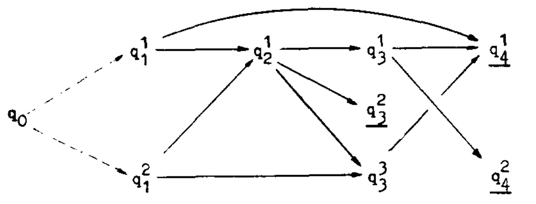


Fig. 1. An acyclic oriented graph having one initial vertex q_0 into which no arcs come. Such a graph always can be stratified into levels with arcs coming from vertices of lower levels to that of higher levels. The level of a vertex is determined by the maximum number of arcs that should be passed before this vertex is reached. The vertex of the level 0 is not entered by any arc. Vertices of the level 1 are entered only by arcs from the vertex of the level 0, vertices of the level 2 are entered by arcs starting in vertices of level 1, and, possibly, 0, etc. Vertices not exited by any arcs are underlined. They form the set $\{q_{\text{end}}\}$. Here this set is formed by three vertices q_4^1, q_4^2 and q_3^2.

```
V:    A T G A C G
      | | | \ |
W:    A A G T A G G
```

Fig. 2. One of possible alignments of sequences V and W. Corresponding letters are connected by lines. If the match weight equals +1, the mismatch weight equals −1, and the gap penalty for any letter equals −3, then the weight of this alignment equals $4 \cdot 1 + 1 \cdot (-1) + 3 \cdot (-3) = -6$.

the corresponding control vector. The *path weight* is defined as a sum of weights of arcs forming it. The Bellman problem is to find the optimal (that is, having the maximum weight) path.

The most direct application of dynamic programing in molecular biology occurs in construction of an optimal alignment of two sequences (Needleman and Wunsch, 1970; Sellers, 1974; Roytberg, 1984; Miller and Myers, 1988).

Various formalizations of this problem differ from each other by the state sets, allowed transitions between the states and their weights. A comprehensive analysis of the dynamic programming method for sequence alignment can be found in the monograph by Waterman (1989). Here we formulate in the graph theory language only the first and the most simple problem by Needleman and Wunsch (1970).

Consider two sequences $V = a_1, a_2, \ldots, a_N$ and $W = b_1, b_2, \ldots, b_M$, where a_i and b_j are symbols from some alphabet. In particular, the alphabet of nucleic acids contains 4 letters (A, T, G, C), while the protein alphabet contains 20 letters (Ala, Gly, ...). Alignment of the sequences V and W means the following: a symbol a_{i_1} is set in correspondence with a symbol b_{j_1}, a symbol a_{i_2} ($i_2 > i_1$) corresponds to b_{j_2} ($j_2 > j_1$), etc. A "bond" between symbols a and b has a weight $S(a, b)$. Letters not participating in the alignment are said to be *deleted*, deletion of a letter a is punished by a penalty $D(a)$. The sum of bond weights and penalties is the weight of the alignment (Fig. 2).

The problem is to find the optimal(that is, having the maximum weight) alignment (if there are several such alignments, then one of those suffices).

When this problem is being solved by the dynamic programming approach, graph vertices correspond to pairs (i, j) where $0 \leq i \leq N$ and $0 \leq j \leq M$. Transitions from a vertex (i, j) to the vertices $(i+1, j)$, $(i, j+1)$ and $(i+1, j+1)$ are possible (Fig. 3). The first transition corresponds to deletion of a symbol $i+1$ in the sequence V, the second one to deletion of a symbol $j+1$ in the sequence W. The transition $(i, j) \to (i+1, j+1)$ corresponds to bonding of the symbols a_{i+1} and b_{j+1}. The initial state is the state $(0, 0)$, the terminal state is the state (N, M) and alignments correspond to paths from $(0, 0)$ to (N, M).

In other algorithms graphs of a more complex structure are considered. In particular, if the bonding weight $S(a, b)$ depends only on matching/mismatching of the symbols (Hirschberg, 1975; Myers, 1989; Roytberg, 1992), then it is sufficient to consider only such vertices (i, j), for which $a_i = b_j$.

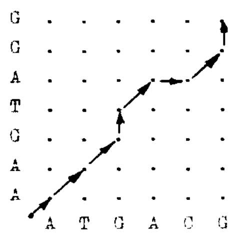
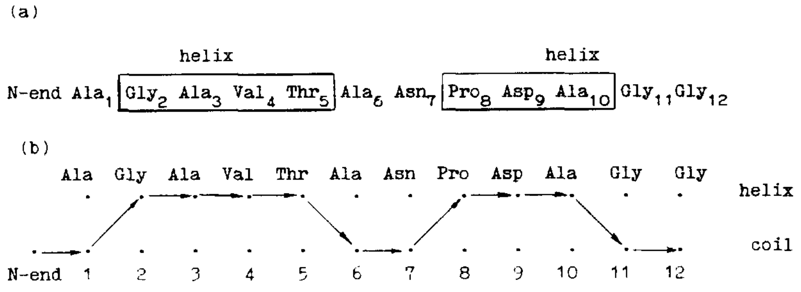


Fig. 3. Vertices of the Needleman-Wunsch graph (dots) and the path corresponding to the alignment presented on Fig. 2.



(a)
helix
helix
N-end Ala1 Gly2 Ala3 Val4 Thr5 Ala6 Asn7 Pro8 Asp9 Ala10 Gly11 Gly12
(b)
Ala Gly Ala Val Thr Ala Asn Pro Asp Ala Gly Gly
helix
coil
N-end 1 2 3 4 5 6 7 8 9 10 11 12

Fig. 4. (A) One of possible arrangements of helices in a polypeptide chain consisting of 12 amino acids. (B) Vertices of the graph describing the secondary structure (helical and coil regions) of a polypeptide chain and the path corresponding to the above arrangement of the secondary structure. In this example two states are possible for each unit: *coil* and *helix*. $\Phi_i(\text{coil}) \sim 0$, $\Phi_i(\text{helix}) = f(a_i)$, where a_i is an amino acid at the i-th position in the chain, $U_i(\text{coil},\text{coil}) = U_i(\text{coil},\text{helix}) = U_i(\text{helix},\text{coil}) = 0$, while $U_i(\text{helix},\text{helix}) = \varepsilon_H$.

2. Generalized matrix apparatus. Statistical physics of polypeptides and DNA

The mathematical apparatus of the statistical mechanics of polymers (Birschtein and Ptitsyn, 1966; Flori, 1969) and, in particular, the theory of helix–coil transitions in polypeptides (Zimm and Bragg, 1959; Levis et al., 1970) and DNA (Vedenov et al., 1967) is largely based on a matrix formalism first introduced by Kramers and Wannier (1941) for computation of the one-dimensional Izing model (Izing, 1925). The latter is the simplest chain of spins each of which can have two possible orientations and interacts only with the nearest neighbors and an external field.

A general algorithm of statistical mechanics of chain molecules can be formulated as follows (Finkelstein,1977): Consider a chain consisting of M units. The i-th unit can assume one of states $\{q_i^1,\dots,q_i^{N_i}\}$ that constitute a set $\{q_i\}$. The energy $E(q_1,\dots,q_M)$ of a chain whose units assume the states $q_1,\dots,q_M$ is determined by the formula

$$E(q_1,\dots,q_M) = \sum_{j=1}^{M} \Phi_j(q_j) + \sum_{j=2}^{M} U_j(q_{j-1},q_j). \tag{7}$$

Here the terms Φ describe the internal energy of units and their interaction with an external field, while the terms U describe the energy of interaction of neighboring units (Fig. 4).

The objective is to find the partition function of the chain

$$Z = \sum_{q_1} \cdots \sum_{q_M} \exp(-E(q_1,\dots,q_M)/kT),$$

where k is the Bolzmann constant and T is the temperature, and, for each unit, the probability that this unit is in a given state.

There exists an algorithm for computation of the partition function Z without explicit combinatorial search over all chain conformations (their number is exponentially large!). Denote $\Phi_1(q_1)$ by $\tilde{U}_1(q_1)$ and denote the sum $U_i(q_{i-1}, q_i) + \Phi_i(q_i)$ by $\tilde{U}_i(q_{i-1}, q_i)$ $(i = 2, \dots, M)$. Let further

$$P_1(q_1) = \exp(-\tilde{U}_1(q_1)/kT);$$
$$r_i(q_{i-1}, q_i) = \exp(-\tilde{U}_i(q_{i-1}, q_i)/kT), \quad i = 2, \dots, M.$$

Clearly, the statistical weight of one conformation is

$$\exp(-E(q_1, \dots, q_M)/kT) = \exp\left(-(\tilde{U}_1(q_1) + \sum_{j=2}^{M} \tilde{U}_j(q_{j-1,j}))\Big/kT\right) \tag{8}$$

while the partition function of the chain is

$$Z = \sum_{q_1} \cdots \sum_{q_M} P_1(q_1) \prod_{j=2}^{M} r_j(q_{j-1}, q_j). \tag{9}$$

The values $r_j(q_{j-1}, q_j)$ form a *transition matrix* R_i of the size $N_{i-1} \times N_i$ such that

$$R_i(k,l) = r_i(q_{i-1}^l, q_i^k).$$

here N_j is the number of possible states of the j-th unit.

In order to find Z for all $i = 2, \dots, M$, one introduces in a recurrent manner vectors $Q_{M-1}, \dots, Q_1$:

Q_M is a vector of the length N_M whose elements equal 1;

$$Q_{i-1} = R_i \cdot Q_i, \quad i = M, \dots, 2 \tag{10}$$

Clearly, (9) can be represented as

$$Z = P_1 \cdot Q_1 \tag{11}$$

where P_1 is a N_1-element vector $(P_1(q_1^1), \dots, P_{N_1}(q_1^{N_1}))$.

This is the essence of the generalized Kramers-Wannier method. The time of computation by formulas (10) and (11) is of the order MN^2, where N is the average number of states of a unit.

Using the vectors $Q_1, \dots, Q_M$, the vector P_1 and additional recurrently computed vectors $P_2, \dots, P_M$, where $P_i = P_{i-1} \cdot R_i$, it is possible to find the probability for a unit i $(i = 1, \dots, M)$ to occupy a state q_j^*:

$$W_j(q_j^*) = 1/Z \sum_{q_1} \cdots \sum_{q_{j-1}} \sum_{q_{j+1}} \cdots \sum_{q_M} \exp(-E(q_1, \dots, q_j^*, \dots, q_N)/kT)$$
$$= P_j(q_j^*) Q_j(q_j^*)/Z. \tag{12}$$

Example: The problem of DNA hybridization

Consider two single-stranded DNA sequences of lengths N and M and the set of base-pairing energies $\{\varepsilon_{ij}\}$, determined by them, $1 \le i \le N$, $M \ge j \ge 1$.

In the simplest case, when interaction between nucleotides within each strand is ignored, the base-pairing energy a fragment $i_1, \dots, i_t$ of the first strand and a fragment $j_1, \dots, j_t$ of the second one (here $1 \le i_1 < \dots < i_t \le N$ and $M \ge j_1 > \dots > j_t \ge 1$ since strands pair in the antiparallel mode, Fig. 5) is

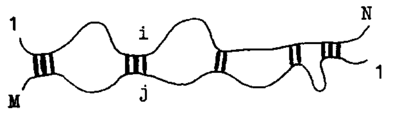


Fig. 5. Hybridization of two DNA strands. Lines correspond to base-pairing (e.g., $i : j$).

$$E(i_1, \dots, i_t; j_1 \dots, j_t) = \sum_{k=1}^{t} \varepsilon_{i_k, j_k}.$$

The search for the most "strong" pairing is similar to the search of the optimal alignment (Section 1). The main difference is that here we consider computation of the *free energy* of strand pairing. This value $F = kT \cdot \ln Z$ (Z is the partition function of pairing) is the one that can be measured experimentally. It takes into account not only the best pairing, but other, not so good but very numerous pairings as well.

In order to solve the problem, we introduce P_{ij} as a partition function of pairing of a fragment $[1, i]$ of the first strand with a fragment $[M, j]$ of the second strand. Assuming that each pairing corresponds to one term of the sum (that is, neglecting logarithmic terms in the loop entropy (Flori, 1971)), the value $Z = P_{N,1}$ can be found from the following recurrent relations:

$$\begin{aligned} P_{0,M+1} &= P_{0,j} = P_{i,M+1} = 1, \\ P_{i,j} &= P_{i-1,j} + \sum_{k=j}^{M} P_{i-1,k+1} \exp(-\varepsilon_{ik}/kT). \end{aligned}$$

The computation time can be reduced at $\sim M$ times if one introduces $P^*_{i,j}$ as partition functions of such pairings of fragments $[1, i]$ with $[j, M]$, that the i-th base of the first strand is necessarily bound to some k-th base of the second strand, where $M \geq k \geq j$, cf. (Roytberg, 1984). Then

$$\begin{aligned} P_{0,M+1} &= P_{0,j} = P_{i,M+1} = 1, \\ P^*_{0,j} &= P^*_{i,M+1} = 0 \\ P^*_{i,j} &= P^*_{i,j+1} + P_{i-1,j+1} \exp(-\varepsilon_{ij}/kT), \\ P_{i,j} &= O_{i-1,j} + P^*_{i,j}. \end{aligned}$$

It is also possible to compute the pairing probability of bases i and j of the first and second strands respectively:

$$W_{ij} = 1/Z \cdot P_{i-1,j+1} \cdot \exp(-\varepsilon_{ij}/kT) \cdot Q_{i+1,j-1}, \tag{13}$$

where the partition function $Q_{i,j}$ of pairing of fragments $[i, N]$ of the first strand and $[1, j]$ of the second strand also is found by recurrent formulas

$$\begin{aligned} Q_{N+1,0} &= Q_{N+1,j} = Q_{i,0} = 1, \\ Q^*_{N+1,j} &= Q^*_{i,0} = 0, \\ Q^*_{i,j} &= Q^*_{i,j-1} + Q_{i+1,j-1} \exp(-\varepsilon_{ij}/kT), \\ Q_{i,j} &= Q_{i+1,j} + Q^*_{i,j}. \end{aligned} \tag{14}$$

with $Q^*_{i,j}$ being a partition function of such a pairing of fragments $[i, N]$ and $[1, j]$ that the i-th base of the first strand is paired with some k-th base of the second one ($j \geq k \geq 1$).

It is clear that expression (13) corresponds to general formula (12), since $P_{i-1,j+1}\exp(-\varepsilon_{ij}/kT)$ is the total statistical weight of all those pairings of the fragment $[1,i]$ of the first strand with the fragment $[j,M]$ of the second one, where the nucleotides i and j are paired, while $Q_{i+1,j-1}$ is the statistical weight of all pairings of the fragments $[i+1,N]$ and $[1,j-1]$ which follow this base-pairing $i:j$.

3. A general algorithm. Acyclic graphs over semirings

Comparison of statements and solutions of the problems arising in the search for an optimal path (3) and the computation of the partition function of a chain molecule (9) demonstrates their close relationship. In both cases

(i) the set of all possible paths from the initial state(s) into the terminal one is considered, each elementary transition between states is ascribed a weight, a path weight is determined by the weights of transitions forming this path, and the object function is determined by the path weights;

(ii) the algorithm is based on the recursion from the terminal states to the initial ones.

The differences are in the determination of the path weight by the weights of the constituent transitions, and in the definition of the object function by the total set of path weights. In the Bellman problem the path weight is the *sum* of transition weights, while the object function is the *maximum* of path weights. In the computation of the partition function, the weight of a path (a conformation of a molecule) is the *product* of transition weights, while the object function is the *sum* of path weights.

These problems can be reduced to the computation of the "sum" of path weights in a graph:

Problem 1 (Aho et al., 1976; Avdoshin et al., 1984). Consider an acyclic directed graph G (Fig. 1) consisting of a non-empty vertex set $\{q\}$ and arcs $\{(q,q')\}$, such that each vertex belongs to at least one arc. One of vertices (q_0) is initial and no arc enters it. A set $\mathcal{P}$ of paths consists of non-empty sequences of arcs (q_0,q_1), $(q_1,q_{,2})$, …, (q_{m-1},q_m) such that each path starts at the initial vertex q_0 and ends at a vertex that is not exited by any arc. Each arc (q,q') is supplied by a weight $r(q,q')$. The *path weight* is defined as a "product" of weights of arcs forming this path:

$$Y(p) = Y(q_0,q_1,\dots,q_{m-1}) = r(q_0,q_1)\otimes\ldots\otimes r(q_{m-1},q_m). \tag{15}$$

The total weight of a set of paths is defined as a "sum" of the path weights:

$$\Phi = \oplus_{\mathcal{P}} Y(q_0,q_1,\dots,q_{m-1},q_m). \tag{16}$$

The objective is to compute this total weight.

Here the arc weights $r(q,q')$ are not necessarily numbers, and operations $\otimes$ and $\oplus$ are not necessarily multiplication and addition. It is required only that the weights belong to a set A that is a semiring with two composition laws $\otimes$ and $\oplus$ and the unit (with respect to $\otimes$) element e (actually, the *left* unit suffices). It means that

(i) the set A is closed with respect to $\otimes$ and $\oplus$:

$$r\oplus r'\in A,\ r\otimes r'\in A;$$

(ii) the operation $\oplus$ is associative and commutative:

$$(r\oplus r')\oplus r'' = r\oplus(r'\oplus r''),$$
$$r\oplus r' = r'\oplus r;$$

(iii) the operation $\otimes$ is associative (not necessarily commutative) and A contains a unit element e with respect to it:

$$(r \otimes r') \otimes r'' = r \otimes (r' \otimes r''),$$
$$r \otimes e = r;$$

(iv) $\otimes$ is distributive relative to $\oplus$:

$$r \otimes (r' \oplus r'') = r \otimes r' \oplus r \otimes r'',$$
$$(r' \oplus r'') \otimes r = r' \otimes r \oplus r'' \otimes r;$$

(the above formulas hold for any r, r' and r'' from A).

If, in particular, $\otimes$ is the arithmetical multiplication, we get formula (8) for computing of the statistical weight of a path, while if $\otimes$ is the arithmetical addition, we compute the total profit on this path and get (2).

If the sign $\oplus$ denotes the arithmetical addition of path weights, we get formula (9) for computation of the partition function Z; if $\oplus$ denotes max (choice of a maximum element), we compute the maximum profit and get formula (3).

All properties (1–4) of the operations $\oplus$ and $\otimes$ are important. Properties (1–3) are used in the statement of the problem itself. If, for example, $\oplus$ is not commutative, formula (16) loses sense. Distributivity allows one to take into account all paths (the number of which is exponentially large!) in polynomial time. More details about graphs over semirings see in (Aho et al., 1976; Avdoshin et al., 1984). Note that the presented statement of the problem is somewhat different from the one considered in (Aho et al., 1976). Since we consider only acyclic graphs G, we can soften restrictions imposed on the weights r. In particular, it is not required that the weights are either all positive or all negative.

Problem 1 is solved in a recurrent manner. For each vertex q not exited by any arc (these vertices form a set $\{q_{\text{end}}\}$) a cost

$$Q(q) = e$$

is ascribed, where e is the unit element, $r \otimes e = r$ for any r (that is, $e = 1$ if $\otimes$ is the arithmetical multiplication, and $e = 0$ if $\otimes$ is addition).

Then, going down by the levels, we compute path costs for each vertex q which is exited by arcs:

$$Q(q) = \oplus_{(q,q')} r(q,q') \otimes Q(q'),$$

where the sum is taken over all arcs (q, q') exiting from q. The process terminates when the value $\Phi = \sum_{\{q_0\}} R(q_0)$ is computed.

We conclude this section by stating in terms of semirings some problems from the physics of chain molecules. In all problems the graph G consists of vertices stratified into levels $0, 1, 2, \ldots, M$, where M is the number of units in a chain. Vertices belonging to a level i correspond to possible states of the i-th unit. Zero level contains a single vertex q_0 (*beginning of the chain*). Arcs connect only the vertices of adjacent levels. An arc $y = (q \to q')$ corresponds to a possible combination of a state q of one unit and a state q' of the adjacent one. The energy of this combination is denoted by ε_y. Energies ε (with or without indices) are real numbers.

1. Statistical mechanics of a chain at a finite temperature T (see Section 2) — *computation of the partition function of all chain conformations*

A is the set of non-negative real numbers r (*statistical weights*, where $r = \exp(-\varepsilon/kT)$);
y has the weight $\exp(-\varepsilon_y/kT)$;
$e = 1$;

$\otimes$ is the arithmetical multiplication, $\times$;
$\oplus$ is the arithmetical addition, $+$.

2. Statistical mechanics of a chain at the zero temperature T (Finkelstein and Reva, 1992) — *computation of the minimum energy and the number of conformations with the minimum energy.*

A is the set of objects of the type $\{n, \varepsilon\}$ where ε is the energy, n is a positive integer (*number of paths*, thus $k \ln n$ equals the chain entropy);
y has the weight $\{1, \varepsilon_y\}$;
$e = \{0, 1\}$;
$\otimes$ is defined by

$$\{n, \varepsilon\} \otimes \{n', \varepsilon'\} = \{nn', \varepsilon + \varepsilon'\};$$

$\oplus$ is defined by

$$\{n, \varepsilon\} \oplus \{n', \varepsilon'\} = \begin{cases} \{n, \varepsilon\}, & \varepsilon < \varepsilon', \\ \{n + n', \varepsilon\}, & \varepsilon = \varepsilon', \\ \{n', \varepsilon'\}, & \varepsilon > \varepsilon', \end{cases}$$

since when two physical systems are combined, their energies are summed, while the numbers of stated are multiplied.

3. Search for one structure with the minimum energy (see Section 1) — *the dynamic programming*

A is the set of objects of the type $\{p, \varepsilon\}$, where p is a path in a graph, ε is the energy;
y has the weight $\{p_y, \varepsilon_y\}$, where p_y is a path consisting of a single arc y;
$e = \{\emptyset, 0\}$, where $\emptyset$ is a null path;
$\otimes$ is defined by

$$\{p_1, \varepsilon_1\} \otimes \{p_2, \varepsilon_2\} = \{p_1 * p_2, \varepsilon_1 + \varepsilon_2\},$$

where $p_1 * p_2$ denotes concatenation of the path p_2 to the path p_1. The operation $\otimes$ is not commutative, since p_2 should extend p_1;
$\oplus$ is defined by

$$\{p_1, \varepsilon_1\} \oplus \{p_2, \varepsilon_2\} = \{p, \min(\varepsilon_1, \varepsilon_2)\},$$

where

$$p = \begin{cases} p_1, & \varepsilon_1 < \varepsilon_2, \\ \min(p_1, p_2), & \varepsilon_1 = \varepsilon_2, \\ p_2, & \varepsilon_1 > \varepsilon_2. \end{cases}$$

The operation $\min(p_1, p_2)$ means that graph vertices are indexed at each level so that each path corresponds to the sequence of indices. For example (Fig. 1), the index $(1, 1, 2)$ of the path $q_2^1 \rightarrow q_3^1 \rightarrow q_4^2$ is smaller than the index $(1, 2)$ of the path $q_2^1 \rightarrow q_3^2$, which in turn is smaller than the index $(1, 3, 2)$ of the path $q_2^1 \rightarrow q_3^3 \rightarrow q_4^2$.

In algorithm implementations it is sufficient to retain only the first arc of a path going out of each vertex. The complete optimal path is reconstituted after an additional processing of the graph in the backwards direction.

4. Search for all structures of the minimum energy (Waterman and Bayers, 1984)

A is the set of objects of the type $\{\mathcal{P}, \varepsilon\}$, where $\mathcal{P}$ is a set of states of a chain fragment (various paths between two vertices of the graph) all of which have the same energy ε;

$e = \{\emptyset, 0\}$, where $\emptyset$ is a zero path;

$\oplus$ is defined by

$$\{\mathcal{P}_1, \varepsilon_1\} \otimes \{\mathcal{P}_2, \varepsilon_2\} = \{\mathcal{P}_1 * \mathcal{P}_2, \varepsilon_1 + \varepsilon_2\},$$

where $\mathcal{P}_1 * \mathcal{P}_2$ denotes the set obtained by all concatenations of paths from $\mathcal{P}_1$ and $\mathcal{P}_2$ (exactly in this order);

$\otimes$ is defined by

$$\{\mathcal{P}_1, \varepsilon_1\} \oplus \{\mathcal{P}_2, \varepsilon_2\} = \{\mathcal{P}, \min(\varepsilon_1, \varepsilon_2)\},$$

where

$$\mathcal{P} = \begin{cases} \mathcal{P}_1, & \varepsilon_1 < \varepsilon_2, \\ \mathcal{P}_1 \cup \mathcal{P}_2, & \varepsilon_1 = \varepsilon_2, \\ \mathcal{P}_2 & \varepsilon_1 > \varepsilon_2. \end{cases}$$

In this problem it is also sufficient to retain for each vertex only a set of outgoing arcs.

Problem 1 is closely related to the following problems:

Problem 2A. In the conditions of Problem 1, find the total weight of all paths passing each vertex of the graph G.

Problem 2B. In the conditions of Problem 1, find the total weight of all paths passing each arc of the graph G.

These problems are solved as follows:

1. Problem 1 is solved and all total weights $Q(q')$ of paths coming out of a vertex q' are retained.

2. The weights $P(q) = r(q_0, q'))$ are ascribed to the vertices of the first level. For higher levels, total weights $P(q)$ of paths entering a vertex q are computed recursively:

$$P(q) = \oplus_{(q'', q)} P(q'') \otimes r(q'', q),$$

where the sum $\oplus$ is taken over all arcs (q'', q) entering q. The total weight of paths passing q vertex q is

$$P(q) \otimes Q(q),$$

while the total weight of paths coming through an arc (q', q) is

$$P(q) \otimes r(q, q') \otimes Q(q').$$

Problem 2A arises in the computation of the unit state probabilities (formulas (12) and (13)), while Problem 2B is related to the search for an optimal path passing a given arc.

4. Prediction of the RNA secondary structure

Prediction of the RNA secondary structure together with prediction of the protein structure, of the non-canonical structures in DNA, and comparison of primary structures of biopolymers is one of the most popular problems of the computational molecular biology.

In this section we consider two algorithms pertaining to this problem: a classical algorithm for prediction of the lowest energy secondary structure of RNA (Nussinov et al., 1978; Zuker, 1989) and an algorithm for prediction of the thermodynamically equilibrium structure (McKaskill, 1990).

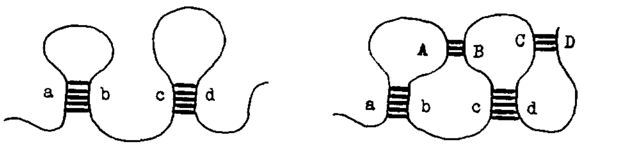


Fig. 6. Secondary structure of a single-stranded RNA. In the left structure (pairings $a-b$ and $c-d$), which is allowed by the considered model, knots are absent. In the right structure the additional pairings $A-B$ and $C-D$ lead to formation of knots. Such structures are forbidden in the considered model.

4.1. Prediction of the optimal RNA secondary structure. One more variant of the dynamic programming

Currently many formal statements of the search for the optimal (that is, having the minimum energy) RNA structure are known. These statements differ in the choice of energy constants, accounting for loops, and the possibility of search for suboptimal structures. We consider only the simplest form of this problem (Zuker, 1989). This statement allows us, however, to illustrate the arising algorithmic problems.

We represent a RNA primary structure as a sequence of symbols $b_1, \dots, b_N$ in the alphabet $\{\mathrm{A, C, G, U}\}$. The secondary structure is formed by hydrogen bonds (pairing) between some bases of a single-stranded RNA. Each base can participate in at most one pairing.

It is assumed that *knots* are forbidden: if the i-th base is paired with the j-th one ($i < j$), then the intermediate bases cannot pair with bases that lie outside the fragment $[i, j]$ (Fig. 6). This assumption is mainly algorithmic. It is not strictly motivated physically or biologically. In applications, prediction is often performed in two steps: first, unknotted structures are found, and then additional knot-forming pairings are considered.

We assume that each pairing is ascribed an energy dependent only on the paired bases. The total structure energy is the sum of the base-pairing energies.

Formally, a secondary structure of a sequence $P = p_1, \dots, p_N$ is a set of pairs $S = \{(i_1, j_1), \dots, (i_t, j_t)\}$ such that

(i) $1 \le i_k < j_k \le N$ $(k = 1, \dots, t)$;

(ii) if $(i, j) \in S$ and $(i', j') \in S$, then either $i' < i < j < j'$, or $i < i' < j' < j$.

Energy of a structure S is

$$E(S) = \sum_k \varepsilon(i_k, j_k), \tag{17}$$

where the energies of base-pairing $\varepsilon(i_k, j_k)$ are assumed to be known. Given a sequence B, the objective is to find the secondary structure having the minimum energy possible for this sequence.

The problem is solved by a recursion. Let $B(i, j)$ be a fragment of B from the i-th to the j-th symbol inclusively (in particular, $B(1, N) = B$), and let $E(i, j)$ be the minimum energy of secondary structures formed by this fragment.

Energies of single-element fragments (i, i) are assumed to be zero, i.e. $E(i, i) \equiv 0$. Energies of fragments of the zero length $E(i + 1, i)$ also equal 0 (they do not participate in the chain energy).

The main recursive relation is

$$E(i, j) = \min\bigl\{E(i + 1, j), \min_{i<k\le j} \{E(i + 1, k - 1) + E(k + 1, j) + \varepsilon(i, k)\}\bigr\}. \tag{18}$$

The first alternative corresponds to the situation when the i-th base (the first base of the fragment $B(i, j)$) is not paired, while in the second case the i-th base is paired with the k-th one, where $i < k \le j$. In the last case the condition of non-knotting allows one to reduce the search for the optimal structure of $B(i, j)$ to independent search for optimal structures of $B(i + 1, k - 1)$ and $B(k + 1, j)$ (Fig. 7).

Search for the optimal structure on B reduces to the sequential search for optimal structures for all fragments of B. Energies of all fragments of length 0 and 1 equal 0. Then, using recursion (18), one finds the optimal structures for fragments of lengths 2, 3, 4, etc. until the entire B is processed.

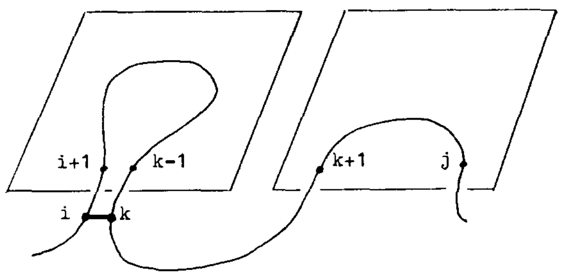


Fig. 7. The structure $B(i,j)$ contains pairings $(i:k)$. In this case its energy equals to the sum of the pairing energy $\varepsilon(i,k)$ and energies of the boxed fragments.

For each segment $B(i,j)$ we retain the value $E(i,j)$ and, retaining k, we remeember the term dominating in $E(i,j)$ (if $E(i,j)$ is dominated by the term $E(i+1,j)$, we assume $k=i$). This allows us to reconstitute by the back search the pairings forming the optimal structure, once the value $E(1,N)$ has been found.

Computation of each energy $E(i,j)$ by recursion (18) requires a time proportional to $|j-i|$, while the entire problem, that is the computation of all $E(i,j)$, requires the $\sim N^3$ time.

4.2. Statistical physics of RNA

The recurrent formulas of the above section are analogous to the formulas of Section 1 used in the construction of the optimal alignment. The partition function for RNA secondary structures is computed by the formulas analogous to (18) (McCaskill, 1990) with the similar substitution of the minimization for the addition of energies of different chain conformations, and of the addition of pairing energies for the multiplication of their exponents.

Let $Q_{i,j}$ be the partition function of a fragment $[i,j]$. Then

$$Q_{i,j} = Q_{i+1,j} + \sum_{i<k\le j} Q_{i+1,k+1} \cdot \exp(-\varepsilon_{ik}/kT) \cdot Q_{k+1,j}$$

(here $Q_{i,i} = Q_{i,i-1} = 1$). Computation of the objective value $Z = Q_{1,N}$ is performed by induction over a length increase of a fragment $[i,j]$.

Back recurrence allows one to determine the probabilities of all base-pairs in the thermodynamically equilibrium state of RNA:

$$W_{i,j} = 1/Z \cdot Q_{i-1,j+1} \cdot \exp(-\varepsilon_{ij}/kT) \cdot Q_{i+1,j-1},$$

where $P_{i,j}$ is the partition function of interaction of a fragment $[1,i]$ with a fragment $[j,N]$ $(1\le i<j\le N)$. $P_{i,j}$ are computed recurrently by decrease of the distance between i and j (cf. (13) and (14) in Section 2):

$$\begin{aligned} P_{0,N} &= P_{1,N+1} = Q_{1,1} = Q_{0,0} = 1, \\ P_{i,j} &= P_{i-1,j} + \sum_{j\le k\le N} P_{i-1,k+1} \cdot \exp(-\varepsilon_{ik}/kT) \cdot Q_{j,k-1} \\ &\quad + \sum_{1\le k<i} Q_{k+1,i-1} \cdot \exp(-\varepsilon_{ik}/kT) \cdot P_{k-1,j}, \end{aligned}$$

(here $1\le i<j\le N$).

5. Dynamic programming on directed hypergraphs: Statement of the problem

The above algorithm for prediction of RNA secondary structure is traditionally considered to be a dynamic programming algorithm. However, this algorithm cannot be reduced to the Bellmann scheme or

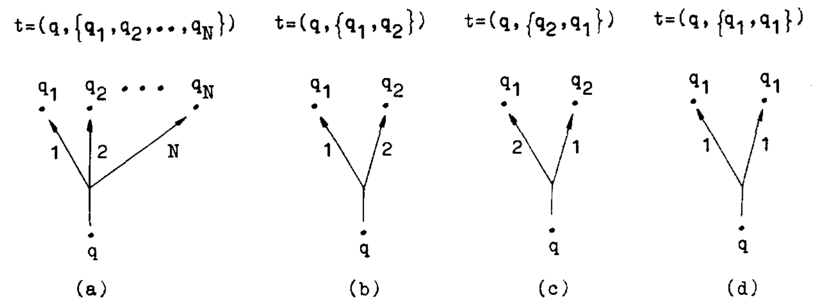


Fig. 8. Graphic representation of hyperarcs with the an initial vertex q and a set of terminal vertices $q_1, \ldots, q_N$. The order of terminal vertices is important: hyperarcs (B) and (C) are different (since the order of terminal vertices q_1 and q_2 is different). Among the terminal vertices of a hyperarc there can be coinciding ones (D).

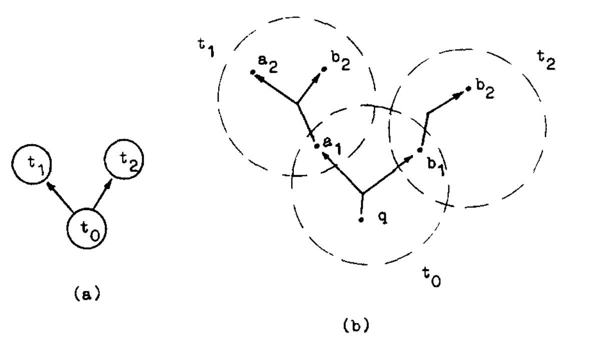


Fig. 9. A stream on the set of vertices $\{q, a_1, b_1, a_2, b_2\}$. It is formed by hyperarcs $t_0 = \big(q, \{a_1, b_1\}\big)$, $t_1 = \big(a_1, \{a_2, b_2\}\big)$ and $t_2 = \big(b_1, \{b_2\}\big)$. Two possible representations of this stream as a tree are presented. (A) Tree nodes correspond to hyperarcs, while vertices of the given DH-graph are not shown explicitly. (B) A more extended representation which would be predominantly used below: tree nodes correspond to vertices of the given DH-graph, hyperarcs (tree nodes in (A)) are encircled. Note that a vertex of the DH-graph can appear in several different tree nodes (here it is b_2, cf. Fig. 8B).

the summation of paths in a graph described in Section 3: successor of a state (here states are fragments of the given RNA molecule) is not a single new state, but two states, namely, a pair of fragments obtained after splitting of the initial fragment by a new pairing.

In order to describe such situations, we introduce the notion of a *directed hypergraph (DH-graph)*. It is a finite set of *vertices* (corresponding, as above, to states of subsystems), and a set of *hyperarcs* connecting these vertices. A hyperarc $t = (q, V)$ sets in correspondence an initial vertex q and a non-empty set of terminal vertices $V = \{q_1, \ldots\}$ (some of the states q_i can coincide, see Fig. 8). The vertices which are not initial for any arc are called the *dead-end vertices*.

A *stream* in a DH-graph is an analog of a path in an ordinary directed graph. These arcs form a path in a directed graph when the terminal vertex of an arc (q_0, q_1) coincides with the initial vertex of an arc (q_1, q_2). Analogously, if the terminal vertices of a hyperarc $t_0 = \big(q, \{q_1, \ldots, q_n\}\big)$ coincide with the initial vertices of hyperarcs $t_1 = \big(q_1, \{q_1^1, \ldots, q_{m_1}^1\}\big)$, ..., $t_n = \big(q_n, \{q_1^n, \ldots, q_{m_n}^n\}\big)$, all these hyperarcs form a stream, t_0 being the *root arc* of this stream (Fig. 9).

Like the paths in the models described in Sections 1–3, a stream describes one of the possible states of a system consisting of subsystems. When the RNA secondary structure is considered (Fig. 10), such subsystems are the separate chain fragments. They correspond to DH-graph vertices, while hyperarcs of this graph correspond to the different pairings of the first base of each fragment, or to the absence of any pairing.

In Section 3 path weight was introduced as a "product" of arc weights and the computation of the "sum"

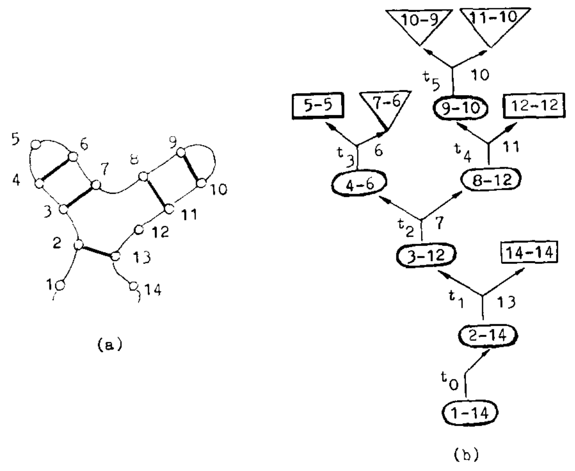


Fig. 10. One of possible configurations of a 14–unit RNA (A). Circles denote nucleotides. Configuration (A) contains 5 base-pairings: 2 : 13, 3 : 7, 4 : 6, 8 : 11, 9 : 10. A stream (B), corresponding to this configuration contains 6 hyperarcs $t_0, t_1, \ldots, t_5$. Vertices of the DH-graph are fragments of the chain. Fragments of a length 2 or more are circled, fragments of length 1 are boxed, fragments of length 0 are set in triangles. Fragments of lengths 0 and 1, and only such fragments, are the dead-end vertices of the DH-graph. The initial vertex of the arc t_0 corresponds to the entire chain 1–14. The first nucleotide is not paired, and thus t_0 has only one terminal vertex: the fragment 2–14. Other hyperarcs correspond to base-pairings and have two terminal vertices each. The number at the "fork" of a hyperarc is the number of the nucleotides paired with the first nucleotide of the corresponding fragment. Thus, the arc t_1 corresponds to the base-pairing 2 : 13 in the fragment 2–14. After that two fragments are left, namely, 3–12 and 14–14. The vertex 14–14 is dead-end, since further pairing in a fragment of length 1 (as well as of length 0, e.g. 10–9) is impossible.

of path weights was considered. Similarly, we define *stream weights* as products of the hyperarc weights and compute the sum of weights of all streams.

We turn now to formal definitions.

Definition 1. An oriented hypergraph (DH-graph) G is a triple (V, q_0, E) where V is a non-empty finite set of vertices, q_0 is the *initial vertex*, E is a finite set of hyperarcs on G.

In order to define formally a stream in a DH-graph, we need the notion of *tree* (Aho et al., 1976). Recall, that a tree is a directed acyclic graph in which (a) there is a single vertex (*root*) with no entering arcs, and (b) for any vertex there exists exactly one path to it from the root.

In order to avoid misunderstanding, we use below a term *node* for tree vertices. Nodes with no exiting arcs are called *leaves*.

Definition 2. A stream in a DH-graph $G = (V, q_0, E)$ is a tree such that

(i) each node of the tree corresponds to a vertex $q \in V$, where one vertex can correspond to several different nodes;

(ii) a set of arcs coming from node α which is not a leave corresponds to a hyperarc $t(\alpha) \in E$, so that the initial vertex of t is the vertex $q(\alpha)$ corresponding to the node α, while the terminal vertices of t are the vertices corresponding to the nodes into which the arcs from α go.

Examples of streams are presented on Fig. 9, 10 and 11.

Now we need only to define the weights of the streams. Let $G = (V, q_0, E)$ be a DH-graph and let each hyperarc $t \in E$ be ascribed its weight $r(t)$, which is an element of a semiring A. The weight $R(T)$ of a stream T is defined in a recurrent manner.

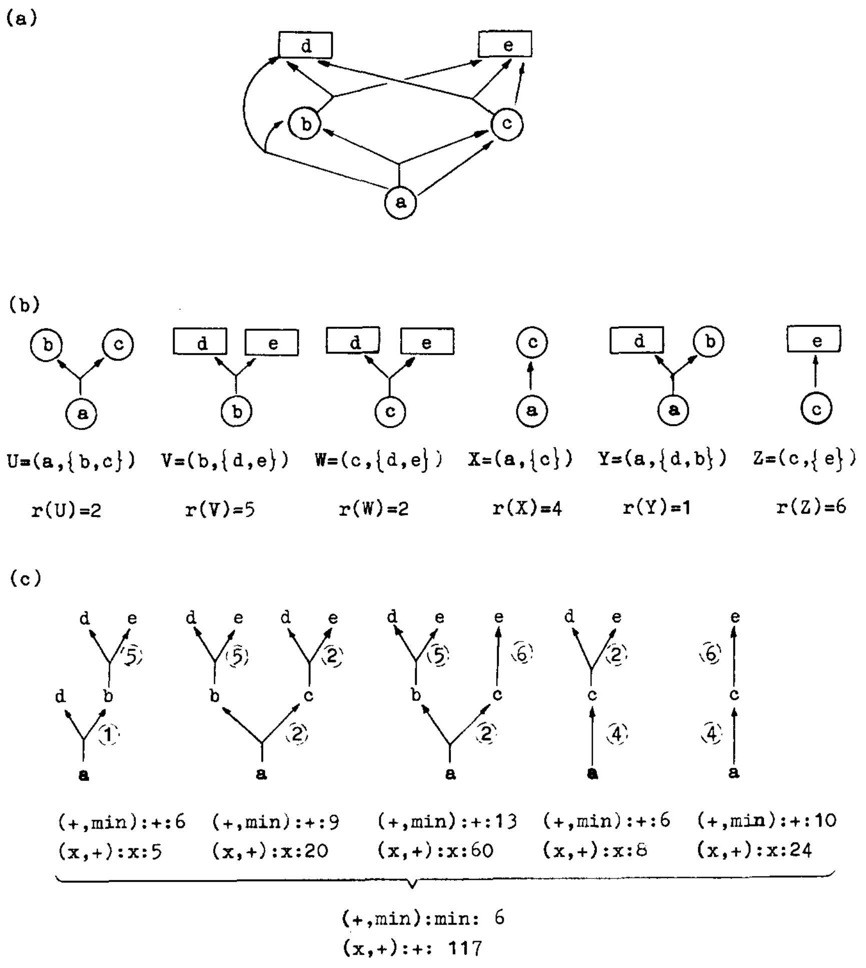


Fig. 11. (A) A DH-graph with 5 vertices and 6 hyperarcs. The vertex a is initial, the boxed vertices d and e are terminal. (B) Hyperarcs and their weights (real numbers). (C) All full streams in the DH-graph. Weights of hyperarcs are circled. Below each stream its weights relative to two semirings are presented: $(\times, +)$: $\otimes$ and $\oplus$ are multiplication and addition of real numbers respectively, $(+, \min)$: $\otimes$ and $\oplus$ are addition and minimum, respectively. Below the brace the weight of the entire DH-graph relative to these semirings is given.

Definition 3.

(i) If a stream T consists of a single node α (i.e. if no arcs go from the root of T), then the weight $R(T)$ is assumed to be equal to the unit e of the semiring A.

(ii) If $N \geq 1$ arcs (corresponding to the hyperarc $t(\alpha)$) go from the root α of a stream T, then

$$R(T) = r(t(\alpha)) \otimes R(T_1) \otimes \cdots \otimes R(T_N).$$

Here $T_1, \ldots, T_N$ are substreams of the stream T coming out of the terminal vertices of the hyperarc $t(\alpha)$; they are "multiplied" in the order of the terminal vertices of $t(\alpha)$.

Definition 4. A stream is called *terminal* if all its leaves correspond to the dead-line vertices of a DH-graph. A terminal stream is called *full* if its root corresponds to the initial vertex q_0 of the DH-graph (V, q_0, E).

Definition 5. *Weight* of a DH-graph $G = (V, q_0, E)$ whose hyperarcs have the weights from a semiring A is the sum (in the sense of A) of the weights of all its full streams.

Definition 6. A DH-graph is called acyclic if it contains no streams in which the vertex corresponding to the stream root corresponds also to some other node of it.

Clearly a finite acyclic DH-graph contains only a finite number of streams.

Now we can formulate *the problem of "dynamic programming" on DH-graphs.*

Problem 3. Consider a finite acyclic DH-graph $G = (V, q_0, E)$, each hyperarc $t \in E$ of which has a weight $t(e)$, which is an element of a semiring A. The objective is to find the weight of the DH-graph G.

In particular, the algorithms for computation of RNA secondary structure considered in Section 4 follow this scheme. In this case (Fig. 10):

(i) vertices of the DH-graph are fragments $[i, j]$;

(ii) hyperarcs are transitions from a fragment $[i, j]$ to the fragment $[i + 1, j]$ and to a pair of fragments $[i, k - 1]$ and $[k + 1, j]$;

(iii) arc weights are: the energies of pairings ε_{ik} (with $\varepsilon_{ii} \equiv 0$) when the minimum energy is computed, the statistical weights $r_{ik} = \exp(-\varepsilon_{ik}/kT)$ when the partition function is computed, and the pairs (pairing $i : k, \varepsilon_{ik}$) when the structure of the minimum energy is searched for (see Problem 3A in Section 4).

(iv) semirings are: ($\otimes$ is arithmetical addition, $\oplus$ is the minimum) when the minimum energy is computed, and ($\otimes$ is arithmetical multiplication, $\oplus$ is arithmetical addition) when the partition function is computed. Operations employed in the search for the minimum energy structure(s) are described in Problems 3 and Section 3.

Solution of Problem 3

Define the weight of a vertex $q \in V$ as a sum $Q(q)$ of all terminal streams with the initial vertex q. Similarly to the algorithm of Section 3, we compute the values $Q(q)$ in the recurrent manner, going from terminal vertices to the initial ones.

If q is a dead-end vertex, that is, if it has no exiting hyperarcs, then $Q(q) = e$ (that is the unit of the semiring A).

Let now q be a non-terminal vertex, and let $t_1 = \left(q, \{q_1^1, \ldots, q_{N_1}^1\}\right), \ldots, t_s = \left(q, \{q_1^s, \ldots, q_{N_s}^s\}\right)$ be all hyperarcs coming out of q. Then

$$Q(q) = \oplus_{t_i} r(t_i) \otimes Q(q_1^i) \otimes \ldots \otimes Q(q_{N_i}^i), \tag{19}$$

where the sum is taken over all hyperarcs $t_1, \ldots, t_s$, $R(q_j^i)$ is the weight of the vertex entered by the j-th end of the i-th hyperarc.

In order to prove formula (19), consider an arbitrary hyperarc $t = \left(q, \{q_1, \ldots, q_N\}\right)$. Denote by ρ_t the sum (in the sense of the semiring A) of weights of all trees whose roots correspond to the hyperarc t. Let $\{T_j^1, T_j^2, \ldots\}$ be all terminal streams with the initial state q_j. Then

$$Q(q) = \oplus_t \rho_t. \tag{20}$$

Clearly,

$$Q(q_j) = \oplus_{i_j} R\left(T_j^{i_j}\right),$$

where i_j spans all terminal streams with the initial state q_j. On the other hand,

$$\begin{aligned}\rho_t &= \oplus_{i_1,\dots,i_N} r(t) \otimes R\left(T_1^{i_1}\right) \otimes \cdots \otimes R\left(T_N^{i_N}\right)\\ &= r(t) \otimes \left(\oplus_{i_1} R\left(T_1^{i_1}\right)\right) \otimes \cdots \otimes \left(\oplus_{i_N} R\left(T_N^{i_N}\right)\right)\\ &= r(t) \otimes Q(q_1) \otimes \cdots \otimes Q(q_N). \end{aligned} \tag{21}$$

Equations (20) and (21) prove equation (19), which provides the desired recurrent algorithm. Clearly its working time is proportional to the total number of terminal vertices in all hyperarcs of the DH-graph G.

Similarly to Problems 2A and 2B, one can state the following problems:

Problem 4A. In the conditions of Problem 3 find the total weight of all streams passing each vertex of the DH-graph G.

Problem 4B. In the conditions of Problem 3 find the total weight of all streams passing each hyperarc of the DH-graph G.

Here a sentence "a stream passes a vertex q" means that one of its nodes corresponds to the vertex q, while a sentence "a stream passes a hyperarc t" means that t corresponds to the arcs exiting from some node of the stream.

Unlike Problems 2A and 2B, these problems can be solved effectively only with additional restrictions.

Definition 7. A DH-graph is called *strongly acyclic* if it does not contain any stream T such that two of its nodes correspond to one and the same vertex q.

Note that the acyclicity condition forbids only the second use of the vertex corresponding to the root of the stream, while two branches can include the same vertex.

Algorithms analogous to algorithms of Section 3 solve Problems 4A and 4B if

(i) the DH-graph is strongly acyclic;

(ii) the multiplication $\otimes$ is commutative.

In order to describe these algorithms, we introduce some new notions. Let q be a vertex of G. A stream T in G is called *q-stream* if one of its leaves corresponds to the vertex q, while all other leaves correspond to dead-end vertices. The sum of weights of all q-streams is denoted by $P(q)$.

From conditions (1) and (2) it follows that the sum of weights of all paths passing a vertex q equals

$$P(q) \cdot R(q),$$

while the sum of weights of all paths passing an arc $t = (q, \{q_1, \dots, q_n\})$ equals

$$P(q) \cdot r(t) \cdot \otimes_{j=1}^{n} R(q_j).$$

The values $P(q)$ are computed in the recurrent manner, moving from the vertex q_0 to vertices of higher levels (here the level of a vertex is the maximum path length from the root of a full stream to the node marked by the vertex q). The recursion looks as follows. Let $t_1, \dots, t_s$ be all hyperarcs for which q is a terminal vertex, and for an arbitrary $j = 1, \dots, s$ let q_j be the initial vertex of t_j, and $\{q_j^k\}$ be the set of terminal vertices (one of them is the vertex q). Then

$$P(q) = \oplus_j r(t_j) \cdot P(q_j) \cdot \otimes_{q_j^k \neq q} R(q_j^k).$$

Conclusion

The model of streams in DH-graphs allows us to describe all known algorithms that are usually considered to be "based on the dynamic programming method". This model is applicable in computation of all structures that can be described as a set of bonds each of which divides a structure into several independent parts. If the condition of "independence of parts" is not satisfied (e.g. when knotted structures occur in RNA), it is useful to solve a simplified problem with a forcefully introduced independence condition, and then to solve the general problem using search algorithms.

The formulation of the problems in terms of semirings allowed us, given an algorithm searching for the energy minimum, to construct a dual algorithm of the computation of the partition function of the same system (Sections 4.1, 4.2 and 6), and *vice versa*. This analogy seems to be rather promising.

On the other hand, there is an important difference between the analysis of oriented hypergraphs and ordinary oriented graphs: DH-graphs are in general irreversible, i.e. hyperarcs, unlike ordinary arcs, cannot be reversed. Backtracking in Problems 4A and 4B, unlike Problems 2A and 2B, is possible only in specific cases. Problem 3, that is solved by algorithms not requiring backtracking, can be solved in the general case, similarly to Problem 1.

Finally, we note the existence of a large class of problems not satisfying the described scheme, but closely related to it. These are the problems where the operations $\otimes$ (computation of path (stream) weights by the (hyper)arc weights), and $\oplus$ (computation of the object function by the path (stream) weights) do not satisfy the semiring axioms (Section 3), in particular, the distributivity condition (Lengauer and Theune, 1991). Such a "non-distributive" problem arises, for instance, in the problem of prediction of the exon–intron structure in higher eukaryote genomes (Gelfand and Roytberg, this volume). In the cited papers some approaches to the non-distributive problems on graphs are suggested. It would be interesting to extend these approaches to the analysis of streams in oriented hypergraphs.

Acknowledgements

We are grateful to the anonymous referee for critique and suggestions (in particular, the term *oriented hypergraph*). M.R. was partially supported by a grant from the Human Genome Council of the Russian Academy of Sciences.

References

Aho, A., Hopcroft, J. and Ullman, J., 1976, The Design and Analysis of Computer Algorithms (Addison–Wesley, Reading, MA).

Angel, E. and Bellman, R., 1972, Dynamic programing of a partial differential equation (Academic Press, New York, London). (Mir, Moscow) (Russian translation)

Avdoshin, S.M., Belov, V.V. and Maslov, V.P., 1984, Mathematical Aspects of Software Synthesis (VINITI, Moscow) (in Russian).

Birschtein, T.M. and Ptitsyn, O.B., 1966, Conformation of Macromolecules (Interscience, New York).

Dreyfus, S., 1961, Dynamic programming, in: Progress in Operations Reserch, vol. 1 (New York, London).

Finkelstein, A.V., 1977, Theory of protein molecule self-organization. III. A calculating method for the probabilities of the secondary structure formation in an unfolded protein chain. Biopolymers 16, 525–529.

Finkelstein, A.V. and Reva, B.A., 1992, Search for stable state of a short chain in molecular field. Protein Engineering 5 (no. 6).

Flori, P., 1969, Statistical Mechanics of Chain Molecules (Interscience, New York, London).

Gelfand, M.S. and Roytberg, M.A., A dynamic programming algorithm for prediction of the exon–intron structure. This volume.

Hirshberg, D.S., 1975, A linear space algorithm for computing maximal common subsequences. Commun. ACM 18, 341–343.

Izing, E., 1925, Beitrag zut Theorie des Ferromagnetizmus. Zeitschr. Phys. 31, 253–258.

Kramers, H.A. and Wannier, G.H., 1941, Statistics of the one-dimensional ferromagnet. Phys. Rev. 60, 252–276.

Lengauer, T. and Theune, D., 1991, Unstructured path problems and the making of semirings. Proc. WADS'91.

Lewis, P.N., Go, N., Go, M., Kotelchuk, D. and Sheraga, H.A., 1970, Helix probability profiles of denatured proteins and their correlation with the native structures. Proc. Natl. Acad. Sci. 65, 810–815.

McCaskill, J.S., 1990, The equilibrium partition function and base pair binding probabilities for secondary structure. Biopolymers 26, 1105–1119.

Miller, W. and Myers, E., 1988, Sequence comparison with concave weighting functions. Bull. Math. Biol. 50, 97–120.

Myers, E.W., 1989, An $O(ND)$ difference algorithm and its variations. Algorithmica 1, 251–266.

Needleman, S.B. and Wunsch, C.D., 1970, A general method applicable to the search for similarities in amino acid sequence of two proteins. J. Mol. Biol. 148, 443–453.

Nussinov, R., Pieczenik, G., Griggs, J.R. and Kleitman, D.J., 1978, Algorithms for loop matchings. SIAM J. Appl. Math. 35, 68.

RomanovskiĭV.I., 1972, Algorithms of Solution of Extremal Problems, Ch. 6. Dynamic Programming Models (Nauka, Moscow) (in Russian).

Roytberg, M.A. 1984, An algorithm of finding homologies among primary structures (Pushchino) (in Russian).

Roytberg, M.A., 1992, A search for common patterns in many sequences. Comput. Appl. Biosci. 8, 57–64.

Sellers, P.H., 1974, On the theory and computation of evolutionary distance. SIAM J. Appl. Math. 26, 787–793.

Vedenov, A.A., Dykhne, A.M., Frank-Kamenetsky, A.D., Frank-Kamenetsky, M.D., 1967, To the theory of the transition helix–coil in DNA. Mol. Biol. (USSR) 1, 313–318.

Waterman, M.S. and Byers, W.A., 1984, Determining all optimal and near-optimal solutions when solving shortest path problems by dynamic programming. Oper. Res. 32, 1381.

Waterman, M.S. (ed.), 1989, Mathematical Methods for DNA Sequences (CRC Press, Boca Raton, FL).

Zimm B.H. and Bragg, J.R., 1959, Theory of the phase transition between helix and random coil in polypeptide chains. J. Chem. Phys. 31, 526–535.

Zuker, M., 1989, The use of dynamic programming algorithms in RNA secondary structure prediction, in: Mathematical Methods for DNA Sequences, M.S. Waterman (ed.) (CRC Press, Boca Raton, FL) pp. 159–184.