A Multiple Graph Layers Model with Application to RNA Secondary Structures Comparison

Julien Allali[1] and Marie-France Sagot[2]

[1] Institut Gaspard-Monge, Université de Marne-la-Vallée, Cité Descartes, Champs-sur-Marne, 77454 Marne-la-Vallée Cedex 2, France, allali@univ-mlv.fr
[2] Inria Rhône-Alpes et LBBE, Université Claude Bernard, Lyon I, 43 Bd du 11 Novembre 1918, 69622 Villeurbanne cedex, France, Marie-France.Sagot@inria.fr, and King's College, London, UK

Abstract. We introduce a new data structure, called MiGaL for "Multiple Graph Layers", that is composed of various graphs linked together by relations of abstraction/refinement. The new structure is useful for representing information that can be described at different levels of abstraction, each level corresponding to a graph. We then propose an algorithm for comparing two MiGaLs. The algorithm performs a step-by-step comparison starting with the most "abstract" level. The result of the comparison at a given step is communicated to the next step using a special colouring scheme. MiGaLs represent a very natural model for comparing RNA secondary structures that may be seen at different levels of detail, going from the sequence of nucleotides, single or paired with another to participate in a helix, to the network of multiple loops that is believed to represent the most conserved part of RNAs having similar function. We therefore show how to use MiGaLs to very efficiently compare two RNAs of any size at different levels of detail simultaneously.

keywords: graph layers, graph comparison, edit distance, RNA, secondary structure

1 Introduction

We introduce in this paper a new data structure, called *MiGaL* for "*Multiple Graph Layers*", that is composed of various graphs linked together by relations of abstraction/refinement. The new structure is useful for representing information that can be described at different levels of abstraction, each level corresponding to a graph. Similar structures have already been used for modelling a spatial environment [4, 5] or plant architectures [8, 6]. It could also be used for analysing web pages or source codes. MiGaL is a more general structure in that it can model graphs besides trees, and edges can be added between the different layers.

After giving a formal presentation of the MiGaL structure, we propose an algorithm for comparing two MiGaLs. The algorithm performs a step-by-step comparison starting with the most "abstract" level. The result of the comparison at a given step is communicated to the next step using a special colouring scheme.

We then present an application of MiGaL to the analysis of RNA secondary structures. RNAs have different functions in a cell, these being strongly related to the spatial fold adopted by the molecule. It is therefore meaningful to compare such folds in order to infer or understand function. For RNAs, the comparison is usually done by considering the secondary structure which, in the case of RNAs, provides a planar view of the fold. A secondary structure may be seen at different levels of detail, going from the sequence of nucleotides, single or paired with another to participate in a helix, to the network of multiple loops that is believed to correspond the most conserved part of RNAs having similar function.

MiGaLs represent therefore a natural way of modelling such secondary structures. To perform the comparison at each different level, an edit distance algorithm is applied to trees that are rooted and ordered (reflecting the orientation of the RNA molecule). We use for this the algorithm particularly adapted to RNAs that was introduced in a recent work by Allali *et al.* [1]. In this paper, we show how to optimise it using the colouring scheme of the algorithm for comparing two MiGaLs introduced earlier in the paper in order to compare RNAs at different levels of detail simultaneously. We show that our method leads to quite satisfying results when applied to the analysis of RNA secondary structures. Furthermore, we show that this new algorithm for comparing RNAs allows to address the *scattering effect* problem mentioned in the literature [1]. Finally, the divide strategy used as the different levels are considered in turn allows also to very efficiently compare RNAs of any size.

2 Multiple Graph Layers

We present in this section the MiGaL data structure. After a formal definition, we provide an algorithm to compare such structures.

2.1 Definition

From now on, we adopt the following notations. Given a set S, we denote by $|S|$ its cardinality. Let $G(V,E)$ be a graph with vertex set $V(G) = \{v_1, \ldots, v_{|V(G)|}\}$ and edge set $E(G) = \{e_1, \ldots, e_{|E(G)|}\}$.

We start by defining the new data structure. It can be described as a layered sequence of graphs, with each graph linked to its neighbours by two applications, one of which corresponds to an abstraction from the previous graph, the other a refinement leading to the next graph. The formal definition of the MultIGrAphLayers data structure is as follows.

Definition 1 (Definition of a $MiGaL$**).** *A* $MiGaL$ *structure* $M(G,R)$ *of size* $|M|$ *is defined as a layered sequence* $G = G_1, \ldots, G_{|M|}$ *of* $|M|$ *graphs and a sequence* $R = \alpha_1, \ldots, \alpha_{|M|-1}$ *of* $|M|-1$ *applications called* refinements*. Each refinement* α_i *is an application from* $V(G_i)$ *to* $\mathcal{P}(V(G_{i+1}))$*, that is, each vertex in*

$V(G_i)$ has a subset of $V(G_{i+1})$ as image. The inverse application of α_i, denoted by β_i, is a surjection called an abstraction *that maps every vertex in $V(G_{i+1})$ to a vertex of $V(G_i)$. In addition, $M(G, R)$ satisfies the three conditions:*

1. *For each vertex $v \in V(G_i)$, the subgraph induced by $\alpha_i(v)$ is connected and not empty.*
2. *For each edge $(u, v) \in E(G_i)$, there exists at least one vertex of $\alpha_i(u)$ connected to a vertex of $\alpha_i(v)$.*
3. *For each pair of unconnected vertices $u, v \in V(G_i)$, there is no edge between a vertex of $\alpha_i(u)$ and a vertex of $\alpha_i(v)$.*

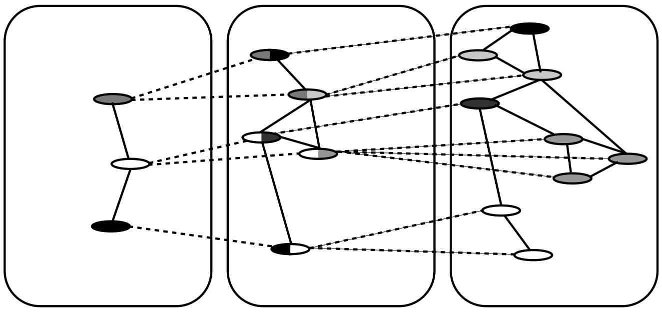

Fig. 1. Example of a *MiGaL* structure of size three defined by three graphs and two refinements.

Figure 1 illustrates this definition. From now on, we refer to the first graph of a MiGaL as the *top* graph and to the last one as the *bottom* graph.

This structure is clearly useful to encode data at different levels of detail. A comparable data structure called *Multihierarchical Graph* has been used in [4, 5] to represent the environment (towns, buildings, rooms, ...) and drive robots in space. Godin *et al.* in [8, 6] use a similar structure, that they call *quotiented trees*, to model and study plants. As compared to the latter, MiGaLs can represent graphs and not just trees, and contrary to Multihierarchical Graphs, MiGaLs can have edges between nodes belonging to different layers.

2.2 Top-Down comparison algorithm

In this section, we present an algorithm to compare two MiGaLs. The algorithm performs a top-down traversal of the structure. We assume both MiGaLs have the same number of layers, and compare them layer by layer starting with the graph at the top. The result of the comparison at a given layer is transmitted to the next layer by colouring vertices and edges of the graph and using the refinement application. We then compare the two graphs of the next layer taking into account such colouring. The process continues until the last layer is reached (bottom graph).

This approach assumes the existence of an algorithm allowing to compare two graphs of a same layer. This algorithm clearly depends on the data that is modelled using a MiGaL. In what follows, we consider a black box that is able to produce an *extended mapping* between two graphs where by an extended mapping is meant the following:

Definition 2 (Extended mapping). *Given two graphs G_1 and G_2, we define an extended mapping $\mathcal{M}$ between them as a set of couples of vertex sets of the two graphs: $\mathcal{M} = \{(S_1, S_2)/S_1 \subset V(G_1) \text{ and } S_2 \subset V(G_2)\}$ such that for all $(S_1, S_2) \in \mathcal{M}$:*

- *S_1 is a connected component of G_1*
- *S_2 is a connected component of G_2*
- *$\forall (S'_1, S'_2) \in \mathcal{M}$, $S_1 \cap S'_1 = \emptyset$ or $S_2 \cap S'_2 = \emptyset$*

We define a colour-partitioned graph G with vertex set $V(G)$ as a graph fitted with an application $C_G : S \subset V(G) \to \mathbb{N}^+$ which gives a colour to each vertex of $S \subset V(G)$ such that subgraphs defined by vertices of the same colour are connected. For convenience, uncoloured vertices are given the colour 0. The vertices with colour 0 are therefore elements of $V(G) \setminus S$.

We now extend the definition of an *extended mapping* to colour-partitioned graphs as follows.

Definition 3 (Colour-constrained extended mapping). *Given two colour-partitioned graphs G_1 and G_2, a* colour constrained extended mapping *$\mathcal{M}$ between them is defined as an extended mapping which further satisfies the following conditions:*

- *$\forall (S_1, S_2) \in \mathcal{M}$, every vertex of S_1 has the same colour c and every vertex of S_2 also has this colour c.*
- *There is no vertex of G_1 or G_2 coloured with 0 that is involved in $\mathcal{M}$.*

Now that we have defined a mapping on colour-partitioned graphs, we give a general algorithm for comparing two MiGaLs, $M(G, R)$ and $M'(G', R')$, having the same number of layers. This number is denoted by $|M| = |M'|$. The algorithm assumes again that we have a black box B that computes a colour-constrained mapping between two colour-partitioned graphs.

The initialisation step colours the vertices of G_1 and G'_1 with 1.

The algorithm is then divided into $|M|$ steps. For each layer i from 1 to $|M|$, we compute the mapping $\mathcal{M}_i$ between G_i and G'_i using B and (except for the last step) colour the vertices of G_{i+1} and G'_{i+1} such that vertices associated by the mapping have the same colour and vertices absent from $\mathcal{M}_i$ are coloured with 0. The result of the algorithm is the set of all mappings between each pair of layers.

The pseudo-code of this algorithm is shown in Figure 2. Let b be the time complexity of algorithm B; the time required to compare two MiGaLs of size n is $O(n * b)$.

Compare$(M(G,R),\ M'(G',R'))$
$colour = 2$
Set all vertices of G_1 and G'_1 to 1.
for each layer i from 1 to $|M|$
    $\mathcal{M} = B(G_i, G'_i)$
    set the color of $V(G_{i+1})$ and $V'(G'_{i+1})$ to 0
    for each couple of sets (u,v) in $\mathcal{M}$
        for all nodes n in u
            set the colour of nodes in $\alpha(n)$ to *colour*
        for all nodes m in v
            set the colour of nodes in $\alpha'(m)$ to *colour*
        set *colour* to *colour* $+ 1$
return $\mathcal{M}_{1\ldots|M|}$

Fig. 2. Algorithm for the comparison of two MiGaL structures using an external algorithm B that computes a colour-constrained mapping between two colour-partitioned graphs.

It is important to notice that the main idea of this algorithm is to compute mappings from the top to the bottom layer. Each layer is compared using the mapping of the previous one without reconsidering the choices implied by this mapping.

We now present a practical application of the MiGaL structure and of the comparison algorithm to the study of RNA secondary structures. In particular, we show, in this case, how to significantly improve the complexity of algorithm B by taking advantage of the node-colouring.

3 RNA-MiGaL

RNAs are one of the most important molecules in the cell. They are composed by a succession of nucleotides named A,C,G and U (also referred to as bases). Inside a cell, RNAs do not keep a linear form but instead fold in space. The fold is given by the set of interactions between nucleotides. An RNA can be described at three different levels, respectively called its primary, secondary and tertiary structures. The primary structure refers to the sequence of nucleotides. The secondary structure is composed of the list of base pairs that participate in an helix (see below). The tertiary structure corresponds to all interactions (base pairings) in the RNA that is, to its 3D structure.

The function of an RNA (be it the well known ribosomal and transfert RNAs or the more recently discovered snoRNAs, microRNAs, etc.) is strongly linked to the shape adopted by the RNA in space. It is accepted that two RNAs that have the same function will have closely similar secondary structures but not necessarily similar primary structures. Considering this, it is fundamental to have efficient algorithms to compare RNAs from the point of view of their secondary structures.

Various structural elements can be distinguished in the secondary structure of an RNA: *helices* which correspond to consecutive base pairs, *hairpin-loops* which are unpaired bases at the end of an helix, *internal-loops* defined by the unpaired bases between two helices and called *bulges* when one side is empty, *multi-loops* which are the meeting point of at least three helices and, finally, *stems* that are series of helices, internal-loops and bulges.

Many approaches have been used for modelling secondary structures. Mainly, three codings has been proposed to represent RNA secondary structures: rooted ordered trees [16]; arc annotated sequences [3]; 2-intervals [14].

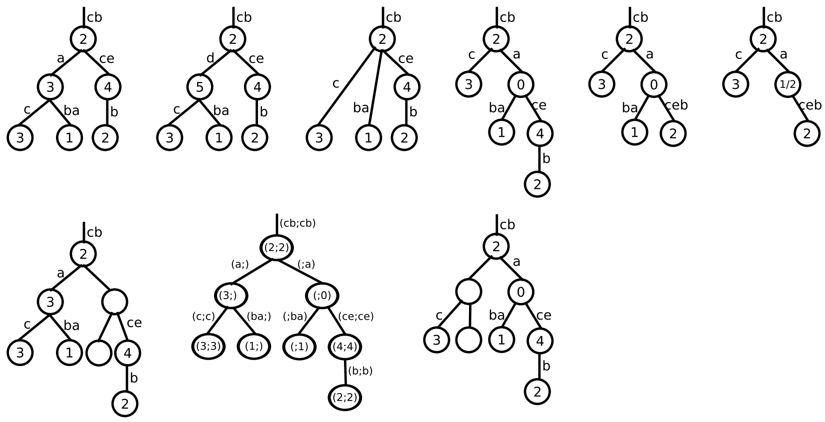

Fig. 3. Edit operations, edit distance and alignment on rooted ordered labelled (on nodes and edges) trees. On the first line, the first tree is edited by a substitution of $a-3$ to $d-5$. Then the node 5 is deleted and after that the node 0 is inserted. From this point, if operations have unit cost (1 for mismatch, deletion, insertion), the edit cost is 3 (the edit distance is 2 because the substitution is useless). The last two trees on the first line give an example of edge fusion (edges ce and b, the new label is constructed by concatenation) and node fusion (nodes 0 and 1, the new label is computed using mean). The second line shows how to align the first tree of the first line with the fourth. We see here that the alignment cost is four (two insertions and two deletions).

Here we focus on the tree model. We represent secondary structures by rooted ordered trees because the root correspond to the beginning and the end of the molecule and the order between the children of a node corresponds to the orientation of the sequence. In fact, we can use different tree models depending on the information we want to encode. In [16], Zuker and Sankoff use trees where internal nodes code for base pairs and leaves for free nucleotides. This tree can be compacted [7] into a homeomorphically irreductible tree where internal nodes correspond to helices and leaves correspond to fragments of unpaired bases.

Shapiro in [12] uses a tree where edges code for helices and nodes are labelled on $\{M, B, I, H, R\}$ (multi-loop, budge, internal-loop, hairpin-loop and the root). We can also use a more abstract tree where edges code for stems and nodes for multi-loops and hairpin-loops, or just for multi-loops.

To compare rooted ordered trees, there exist essentially two methods. The first is the edit distance [11, 13, 15] based on three edit operations. A *substitution* changes the label of a node. A *deletion* removes a node of the tree, its children are then re-attached to the node's father preserving the relative order between nodes. An *insertion* is the opposite operation of a deletion. If we assign a score to each of these operations, we can define the edit distance between two trees as the minimum of the score of a series of edit operations (sum of the score of each operation) that transforms the first tree into the second one. Recently, Allali *et al.* [1] extended this distance by introducing four new operations: *node fusion*, *edge fusion* and the opposite operations of *node split* and *edge split*. These operations allow to address some of the limitations of the classical edit distance when comparing RNA secondary structures using high level trees (where nodes and edges code for secondary structure elements only, not for nucleotides). The complexity of the classical edit distance is $O(n^4)$ where n is the size of the trees and $O((2d)^l n^4)$ for the algorithm with node and edge fusions where d is the degree of the trees and l the maximum number of consecutive fusions per node.

The second method for comparing rooted ordered trees is by performing a tree alignment [10, 9]. In this case, the goal is to insert blank labelled nodes in the two trees such that they become isomorphic. The score of the alignment corresponds to the sum of the scores of the association of node labels (the blank character assumes the role of insertions and deletions in sequence alignment). The optimal score, maximal or minimal depending on the scoring scheme adopted, is then sought. A tree alignment can be expressed as an edit distance computation where insertions must precede all deletions. We do not give further details here on alignments as we only use edit distance later in the paper.

We now suggest a new modelling of RNA secondary structures based on the MiGaL data structure introduced earlier in the paper. We thus call RNA-MiGaL a MiGaL structure composed by four layers, each modelled by a rooted ordered tree. The next section is dedicated to a description of RNA-MiGaL.

3.1 The four layers definition

The four layers contained in an RNA-MiGaL correspond to the secondary structure of an RNA observed at different levels of detail. Thus, each layer is modelled by a rooted ordered labelled tree. In the bottom layer, nodes and leaves code for nucleotides while the top layer encodes the network of multi-loops of an RNA. This choice has been dictated by two assumptions. The first, already mentioned, is that structure is more important than sequence. Thus we introduce information on nucleotides at the bottom layer only which will be treated last by the comparison algorithm. The second is that the network of multi-loops can be considered as the skeleton of the secondary structure of an RNA. RNAs of the same family should thus have strongly conserved multi-loop networks. For this reason, the top layer of an RNA-MiGaL is a tree that codes for the multi-loop network. Intermediate layers correspond to the structure encoded using stems or helices.

We provide below a summary of the layers and of the meaning of a node and a leaf in each layer:

– The tree of layer 1 corresponds to the multi-loop network. The nodes encode multi-loops and the edges encode stems. In the nodes, we store the number of helices connected by the multi-loop. In the edges, we store the number of nucleotides contained in the stem.
– Layer 2 consists in the structure defined by the stems. The internal nodes represent multi-loops, leaves represent hairpin-loops and edges encode stems. In the edges, we store the number of base pairs and the number of unpaired bases contained in the stem. In the internal nodes and leaves, we store the number of unpaired bases contained in the multi-loops and hairpin-loops.
– The tree of layer 3 encodes secondary structure elements: nodes encode hairpin-loops, multi-loops, internal-loops and bulges and store the number of unpaired bases of the corresponding element. The edges represent helices and store the number of base pairs of the helices.
– The last tree models the RNA primary structure. The internal nodes thus represent base pairs and leaves unpaired bases. Both store the names of the corresponding bases.

3.2 RNA-MiGaL comparison

The problem now is to compare two secondary structures using RNA-MiGaLs. To do so, we use edit distance with fusions and the algorithm described in [1] to compare the pairs of trees of layers 1, 2 and 3. To take into account the colour of the nodes and edges, we just have to test if two nodes or edges have the same colour to allow them to be fusioned (in a tree) or substituted (between the trees). The trees of layer 4 are compared using the classical edit distance as fusions make no sense for nucleotides.

Figures 4 and 5 show the result of the comparison using RNA-MiGaLs between two Group I Intron RNAs retrieved from [2]. The left RNA is found in *Acanthamoeba griffini* and the right one in *Chlorella sorokiniana*.

The time complexity required to compare two trees of a same layer is $O((2d)^l n^4)$ for the first 3 layers and $O(n^4)$ for the last layer with n the size of the trees. Since we colour nodes while we progress in the comparison, and, once trees are coloured, substitutions and fusions can only be performed on nodes of same colour, we may wonder if it is not possible to compare separately the subtrees defined by each colour.

The response is positive but it requires some attention as is shown in the following example. Figure 6 presents on the top left a colour-partitioned tree, with white corresponding to colour 0. In the box, we have split each tree into subtrees according to their colour and computed the edit distance between these trees separately. The nodes associated by the mapping are linked with dashed

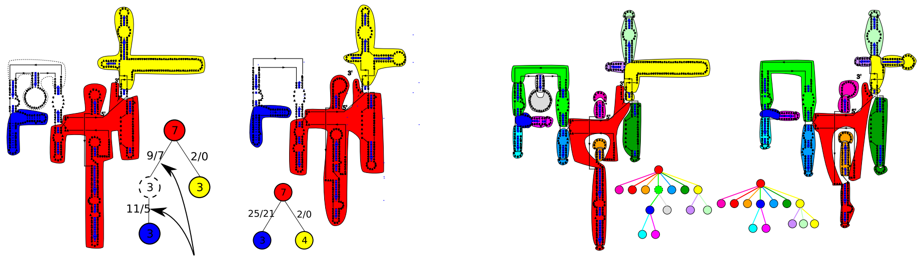

Fig. 4. Result of the comparison of two Group I Introns. On the left, the result of the comparison of the trees of layer 1; on the right, the result of the comparison of the trees of layer 2.

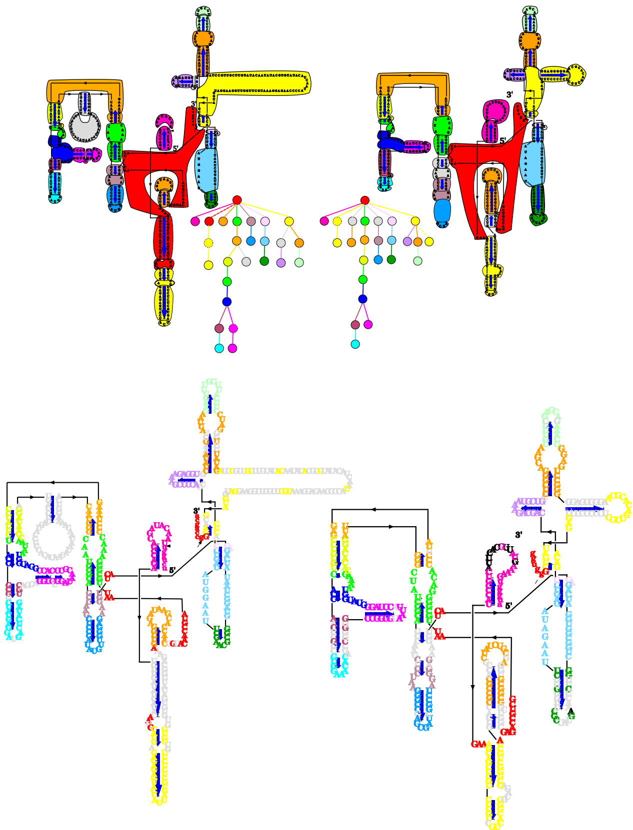

Fig. 5. Result of the comparison of two Group I Introns. On the top, the comparison of the trees of layer 3; on the bottom, the result of the comparison of the trees of layer 4.

lines. We have then reported the result of the computations on the original trees. The problem is pointed to by the bold dashed lines in the tree at the bottom. We have two couples of nodes associated via two different computations. Inside each computation, the relative order of the nodes is respected by the edition (we work on ordered trees). However, on the final tree, the associations do not respect the order between the nodes.

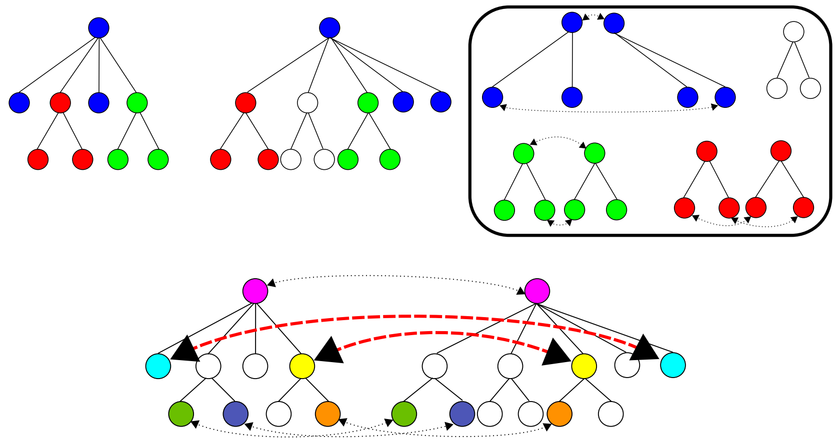

Fig. 6. Problem about splitting tree according to colour to optimize edition computation.

We therefore have to modify the algorithm such as to take into account the presence of a subtree of a colour x hanging from an edge of a subtree of colour $y \neq x$. To do so, we introduce anchors during the splitting step. When we split trees into subtrees according to their colour, we add anchors to the subtrees. These anchors represent the subtrees of another colour that hang from an edge of the subtree being considered as shown in Figure 7. The anchors have the same colour as the subtrees they stand for. We then have to modify the edit distance scoring scheme such that:

- deleting the anchor of colour c costs the deletion of the subtree corresponding to the anchor;
- an anchor of colour c can only match with an anchor of the same colour.

Finally, we order the edit computations according to the dependence implied by the colouring scheme, that is we consider the subtree from the leaves to the root.

The consequence on the complexity is important as the time required to compare two trees of a given layer has a time (and space) complexity which depends on the size of the biggest subtree and not on the size of the whole trees anymore.

In the worst-case, the time complexity is $0((2d)^l \times n^4)$ for each level but in the case of RNAs, on average, the tree at each layer contains a number of subtree that is proportional to the size of the tree, and the observed time complexity is close to linear except for the first layer.

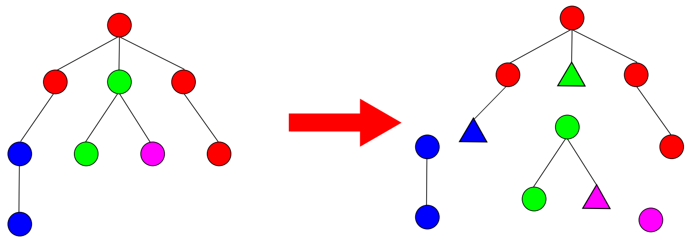

Fig. 7. Splitting the left tree into subtrees according to the colours of the nodes and addition of anchors (represented by triangles).

4 Conclusion

We introduced in this paper a new data structure and an algorithm for comparing such structures. Both are generic and may be applied to very different types of problems. They are particularly interesting for comparing objects that may be described at different levels of abstraction.

RNA secondary structures are one example of such objects and we therefore showed how to optimise the generic algorithm, in particular its special colouring scheme for going from one level of abstraction to the next, to compare two RNA secondary structures. The results obtained are very satisfying. In particular, the algorithm addresses the so-called scattering effect described in the literature [1]. The new algorithm allows to compare large sized structures in a fast way while the multiple layer approach represents a biologically very natural way of modelling and analysing with RNAs.

References

1. Julien Allali and Marie-France Sagot. A new distance for high level rna secondary structure comparison. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 2(1):3–14, 2005.
2. J. J Cannone, S. Subramanian, M. N. Schnare, J. R. Collett, L. M. D'Souza, Y. Du, B . Feng, N. Lin, L. V. Madabusi, K. M. Muller, N. Pande, Z. Shang, N. Yu, and R. R. Gutell. The comparative RNA web (CRW) site: an online database of comparative sequence and structure infor mation for ribosomal, intron, and other RNAs. *BMC Bioinformatics*, 3(1), 2002.

3. P. A. Evans. Finding common subsequence with arcs and pseudoknots. In *Proceedings of the 10th Annual Symposium on Combinatorial Pattern Matching (CPM'99)*, number 1645 in LNCS, pages 270–280, 1999.
4. Juan A. Fernandez and Javier Gonzalez. Hierarchical graph search for mobile robot path planning. In *ICRA*, pages 656–661, 1998.
5. Juan-Antonio Fernández-Madrigal and Javier González. Multihierarchical graph search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(1):103–113, 2002.
6. P. Ferraro and C. Godin. An edit distance between quotiented trees. *Algorithmica*, 36:1–39, 2003.
7. W. Fontana, D. A. M. Konings, P. F. Stadler, and P. Schuster. Statistics of RNA secondary structures. *Biopolymers*, 33:1389–1404, 1993.
8. C. Godin and Y. Caraglio. A multiscale model of plant topological structures. *Journal of theoretical biology*, 191:1–46, 1998.
9. Matthias Höchsmann, Thomas Töller, Robert Giegerich, and Stefan Kurtz. Local similarity in RNA secondary structures. In *Proceedings of the IEEE Computer Society Conference on Bioinformatics*, page 159. IEEE Computer Society, 2003.
10. Tao Jiang, Lusheng Wang, and Kaizhong Zhang. Alignment of trees - an alternative to tree edit. In *Proceedings of the 5th Annual Symposium on Combinatorial Pattern Matching*, pages 75–86. Springer-Verlag, 1994.
11. S. M. Selkow. The tree-to-tree editing problem. *Inform. Process. Lett.*, 6(6):184–186, 1977.
12. B. Shapiro. An algorithm for multiple RNA secondary structures. *Comput. Appl. Biosci.*, 4(3):387–393, 1988.
13. Kuo-Chung Tai. The tree-to-tree correction problem. *J. ACM*, 26(3):422–433, 1979.
14. S. Vialette. Pattern matching problems over 2-interval sets. In *Proceedings of the 13th Annual Symposium on Combinatorial Pattern Matching*, pages 53–63. Springer-Verlag, 2002.
15. K. Zhang and D. Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM J. Comput.*, 18(6):1245–1262, 1989.
16. M. Zuker and D. Sankoff. RNA secondary structures and their prediction. *Bull. Math. Biol.*, 46:591–621, 1984.