Application of Maximal Constraint Satisfaction Problems to RNA Folding

Christine Gaspin
INRA, UBIA
Chemin de Borde-Rouge,
BP27 Auzeville,
31326 Castanet-Tolosan cedex,
FRANCE

Jean-Charles Régin
ILOG S.A.
9 rue de Verdun BP 85
F-94253 Gentilly Cedex,
FRANCE

Abstract

The formalism of constraint satisfaction problems (CSP) applies very well in a number of various applications. However, when it is not possible to compute a complete assignment of variables, Partial CSP or dynamic CSP solving is the alternative. Generally, *Branch & Bound* based algorithms allow to compute solutions maximizing requirements when a cost function g is available. In this paper, we propose an efficient algorithm allowing to compute all the partial solutions of a CSP in which all the constraints are satisfied when a cost function g is not available and a given subset of variables must appear in the final assignment. The algorithm is described and experimental results are given for RNA structure determination.

1 MOTIVATION

The formalism of constraint satisfaction problems (CSP) is a powerful framework for representing and solving a large number of various applications. Searching for a complete assignment of the variables is the usual query in CSP and a variety of general techniques have been developed for finding in an efficient way a consistent assignment of values to such a predefined set of variables ([1], [2], [3], [4], [5], [6]).

However, it is not always possible to find complete assignments either because the problem is over-constrained ([7]) or we do not know what are the variables to assign ([8]). In over-constrained problems, one can relax the constraints or try to satisfy as many requirements as possible. This class of problems is known as Partial CSP (PCSP) where a PCSP is defined from a CSP and a function g which maps a value to each partial instantiation. Solving a PCSP amounts to finding a partial instantiation that gives an optimal value with regard to g. Recently more attention has been given to PCSP where the main objective is to maximize the satisfaction of constraints ([7]). If good estimations with regard to g are calculable, then the best technique for solving a PCSP is *Branch & Bound*. When the constraints on a subset of variables must all be satisfied, the problem is called the maximal utility problem (MUP) in [9]. When no function g is associated to any partial assignment or if g assigns the same cost to any assignment the task in a MUP is to find a maximum assignment which contains the given subset of variables. One extension of these ideas is to compute all the partial solutions in which all the constraints are satisfied for a given subset of variables. We identify this class as $Max_V CSP$.

Applications are found in resource allocation in job-shop scheduling when it is necessary that a given subset of tasks are realized, in configuration tasks where the presence of some variables often depends on the presence of other variables or the values of other variables... Mittal and Falkenheimer ([8]) proposed a language and an ATMS based algorithm to solve this type of problem which they call dynamic constraint satisfaction problem. In this paper, we propose a new algorithm to solve problems in $Max_V CSP$. The algorithm works on the graph of consistency which represents the CSP. The main difficulties with this algorithm are to cut off branches that will not contain the given set of variables and to compute maximal (or saturated) solutions *i.e.*, solutions that are not contained in any other. We consider finite (the domains are finite sets of values) and binary CSP. This algorithm was used successfully to compute all the saturated RNA secondary structures [10] that satisfy experimental and thermodynamic folding requirements. One of its main interest in solving that problem is its capacity to consider pseudoknot constraints while computing RNA secondary structures.

After giving some preliminary definitions in section 2, we show that $Max_V CSP$ is related to the problem of finding a subset of all the maximal complete subgraphs (cliques) of the consistency graph.The algorithm is described in section 3. Finally, experimental results are given for the problem of RNA secondary structure determination.

2 PRELIMINARY DEFINITIONS

A finite *CSP* $P = (X, D, C)$ is defined as:

- a set X of N variables $x_1, ..., x_N$,
- a set D of finite domains $d_1, ..., d_N$,
  each domain $d_i = \{v_{i1}, ..., v_{ik}\}$ gives the finite set of possible values for the variable x_i,
- a set (C_R) of constraints $c_1, ..., c_p$ where each constraint c_i is defined on a set of variables $(x_{i_1}, ..., x_{i_{j_i}})$ by a subset $r_i \in R$ of the cartesian product $d_{i_1} \times ... \times d_{i_{j_i}}$ (r_i gives the values of variables which are compatible with each other).

A solution is an assignment of values to all variables which satisfies all the constraints. In binary CSP, all the constraints are unary or binary. The consistency graph $\mathcal{C}(\mathcal{P})$ which represents a binary CSP P is a graph $G = (Y, E)$ where each vertex $y_i \in Y$ represents a pair *(variable,value)* and an edge $e_j \in E$ connects two vertices if they represent compatible pairs (cf. Figure 1).

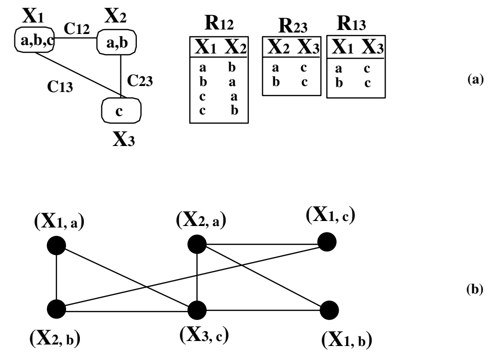

Figure 1: (a) The constraint graph. (b) The consistency graph of the constraint graph

A *maximal* clique in a graph G=(X,V)[1] is a clique (a complete subgraph) that is not contained in any other one. A clique is *maximum* if there is no clique of G of larger cardinality. The *size* of a clique is the number of vertices it contains. The clique problem is the optimization problem of finding a clique of maximum size in a graph. The associated decision problem is to ask whether a clique of a given size k exists in the graph. The clique problem is NP-complete [11].

If the number of variables of a CSP is N, the problem of finding a solution to the CSP is equivalent to the problem of finding an N-clique in the consistency graph $\mathcal{C}(\mathcal{P})$. In $\mathcal{C}(\mathcal{P})$, an N-clique is both a maximal and maximum clique.

[1]X is the set of vertices and V the set of edges

Proposition 1 *Let P be a CSP and $\mathcal{C}(\mathcal{P})$ the associated consistency graph. A locally consistent k-partial instantiation I_k of G is a k-clique in $\mathcal{C}(\mathcal{P})$.*

Indeed, if all the constraints are satisfied in the consistency graph associated with the subgraph induced by the variables of I_k then all the pairs of compatible values *i.e.* all the edges in the consistency graph are adjacent and form a k-clique.

A locally consistent k-partial instantiation of a CSP P is *maximal* if it cannot be extended to a locally consistent k'-partial instantiation (k'>k).

Proposition 2 *A maximal locally consistent k-partial instantiation of G is a maximal k-clique in $\mathcal{C}(\mathcal{P})$.*

A locally consistent k-partial instantiation of a CSP P is *maximum* if no other locally consistent k'-partial instantiation is of greater cardinality, i.e., k $\geq$ k' for each k'.

Now, we can define the problem $Max_V CSP$ of interest.

Max$_V$ CSP

- **Given** a CSP $P = (X = X_{\overline{V}} \cup X_V, D, C)$ (X_V is the given subset of variables which must appear in final assignments) and the associated consistency graph $\mathcal{C}(\mathcal{P})$,
- **Find** in $\mathcal{C}(\mathcal{P})$ all the maximal locally consistent k-partial instantiations, where each consistent partial instantiation
  $I_k = \{(x_1, v_1), ..., (x_k, v_k)\}$:
  1. satisfies the constraints in the subgraph induced by
     $X_k = \{x_1, ..., x_k\}$,
  2. $X_k \supseteq X_V$ and
  3. no superset of I_k is a complete maximal subgraph in $\mathcal{C}(\mathcal{P})$.

3 SEARCHING FOR $Max_V CSP$ SOLUTIONS

In $\mathcal{C}(\mathcal{P})$, each vertex is associated with a pair $(variable, value)$. An edge connects two vertices if they are compatible pairs. The algorithm we propose works on the graph $\mathcal{C}(\mathcal{P})$ and relies on the main principles used in [12] where the problem is to enumerate all the maximal cliques of a graph. The problem consists now in computing only those solutions that both are maximal cliques and contain a given subset $IMPOS$ of assigned variables.

3.1 Data structures

At each node of the tree, the following four arrays are maintained:

- SOL contains at each level of the search tree the vertices of $\mathcal{C}(\mathcal{P})$ that are currently part of the potential clique solution, *i.e.* the vertices of a completely connected subgraph.
- At the beginning, $IMPOS$ gives the subset X_V of variables for which all the constraints must be satisfied in computed assignments. At each level of the search tree $IMPOS$ contains all the variables which both must appear in computed solutions and that have not yet appeared in SOL but which belong to a pair $(variable, value)$ in $CAND$.
- $CAND$ contains, at each level, vertices which are possible candidates for selection and addition to SOL ($x_i \in CAND$ is connected to every $x_i \in SOL$). $CAND$ is shared into two sub-arrays. The first sub-array contains all the vertices in which variables from $IMPOS$ appear. This facilitates the selection of such pairs at the beginning and makes the algorithm more efficient in terms of openning new branches. The second sub-array contains other pairs.

- NOT contains, at each level, vertices which are connected to every member of SOL, which have appeared in previously discovered cliques, and which (to avoid repetition) must therefore not appear in a current clique. This avoids finding cliques or subgraphs of cliques that have been found previously. Because NOT and $CAND$ have no common element, the sum of the sizes of these two arrays is never greater than the total number of pairs.

3.2 Algorithm

The algorithm is a depth first, branch and bound procedure with search ordering.

SearchForMaxSol(in $CAND, NOT, IMPOS$)
ChooseCand($CAND, NOT, Cand, NbDisc$)
while $NbDisc > 0$ **do**
  AddToSol($SOL, Cand$)
  $NewCAND \leftarrow$ ComputeNewCand($CAND, Cand$)
  $NewNOT \leftarrow$ ComputeNewNot($NOT, Cand$)
  $NewIMPOS \leftarrow$ComputeNewImpos($IMPOS, Cand$)
  if AllImposedInCand($NewIMPOS$) **then**
    if $NewCAND = \emptyset$ **and** $NewNOT = \emptyset$ **then** printMaxClique(SOL)
    else SearchForMaxSol($NewCAND, NewNOT, NewIMPOS$)
  endif
  $nb \leftarrow nb - 1$
  RemoveFromSOL($SOL, Cand$)
  AddToNot($NOT, Cand$)
  $Cand \leftarrow$ NextDisconnected($CAND, Cand$)
endw

The tree search begins at the root node, corresponding to the initial level of the tree search. The formation of new nodes of the search tree at subsequent levels involves selectively choosing graph vertices (pairs $(variable, value)$ in $\mathcal{C}(\mathcal{P})$) to form potential cliques using a search ordering heuristic that consists in choosing first a well-chosen vertex ($Cand$) in $CAND$ by using the function (ChooseCand).

Selection of a pair $(variable, value)$
The selected element ($Cand$) is always the least disconnected vertex. When $IMPOS$ is not empty, pairs in $CAND$ containing variables of $IMPOS$ are first selected. The number of disconnections ($NbDisc$) of $Cand$ with other vertices is kept and used later in the algorithm as a counter which is decremented each time a new branch is developed (that will give raise to a new series of cliques) from one of the disconnected vertices of $Cand$. The choice of the less disconnected vertex has the effect that the largest cliques tends to be found early in the tree search and the cutting off of branches early in the search tree. This is possible when there exists a point in NOT connected to all points in $CAND$ ($Cand \in NOT$ and $NbDisc = 0$) or when a variable from $IMPOS$ does not appear in pairs $(variable, value)$ of $CAND$. Search ordering causes the appearance of such points as soon as possible while searching.

Updating NOT, $CAND$ **and** $IMPOS$

- Updating $CAND$. When a vertex $Cand$ is selected, the sets $CAND$, $IMPOS$ and NOT are updated. A first consequence of the choice of $Cand$ is to propagate constraints by the function ComputeNewCand. The result is a set $NewCAND$ computed by eliminating from $CAND$ those pairs which are not compatible with $Cand$.

- Updating NOT. $NewNOT$ is computed by using ComputeNewNOT which selects in NOT those vertices which are compatible with $Cand$. This is very important because if at some tree node, some member of NOT is connected to every member of $CAND$ there is no chance of finding a new clique and the search backtracks.

- Updating $IMPOS$. When a vertex $Cand$ is selected that is a pair containing a variable of $IMPOS$, then $IMPOS$ is updated by removing this variable (ComputeNewImpos). In this way, at the next level, this variable, that is already in SOL, is not verified for its membership in SOL or $CAND$. With the test realized by the function AllImposedInCand we look at the membership of all the variables which are still in $IMPOS$ in the set $CAND$. If there is only one variable that is not in a pair $(variable, value)$ of $CANDIDATE$, then the search backtracks.

If, at some tree node, $CAND$ and NOT are empty, then SOL is a solution and is reported.

3.3 Example

Figure 2 illustrates the application of this algorithm to a small example. There are four maximal cliques in the consistency graph. Now, let us consider that x_3 is imposed in the computed solutions: only two assignments give desirable solutions.

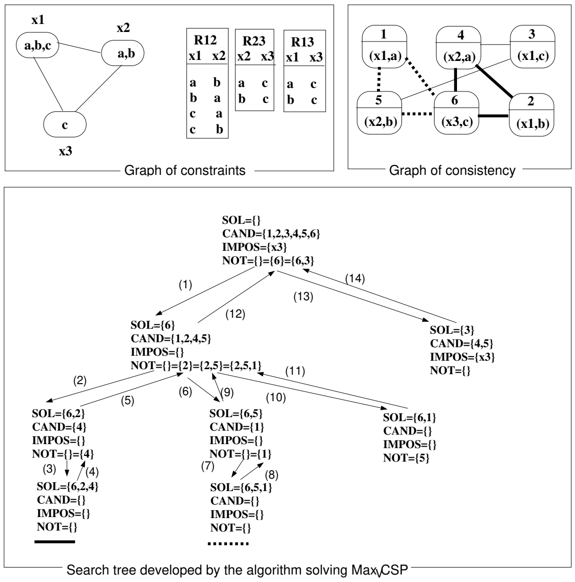

Figure 2: Search tree associated to the CSP above. Each vertex shows the sets SOL, CAND, IMPOS and NOT, respectively.

At the beginning, the array $CAND$ contains all the pairs: $(x_1, a), (x_1, b), (x_1, c), (x_2, a), (x_2, b), (x_3, c)$. The least disconnected vertex in $IMPOS$ is the pair (x_3, c) which is also the only pair containing the imposed variable. $IMPOS$ contains x_3, NOT and SOL are empty. Once (x_3, c) is selected (after (1)), only those pairs connected in $\mathcal{C}(\mathcal{P})$ are kept as future candidates in $NewCAND$. $NewIMPOS$ is updated by removing x_3 and SOL contains the selected pair. In the next step, the least disconnected selected vertex of the graph with vertex in $CAND$ is the pair (x_1, b) (2). This vertex is added in SOL. $NewCAND$ is computed by removing all the vertices not connected with (x_1, b). NOT is empty. The next selected vertex is (x_2, a) (3). At this step, $CAND$ becomes empty, NOT is always empty and $IMPOS$ too. A solution is reported which contains x_3: $S1 = \{(x_3, c), (x_1, b), (x_2, a)\}$. The algorithm backtracks with NOT containing the last pair selected: (x_2, a) (4). (x_2, a) has no disconnected vertices in $CAND$: the algorithm bactracks with NOT containing the pair (x_1, b) (5). At this step $CAND$ contains vertices disconnected from (x_1, b) : (x_2, b) and (x_1, a). (x_2, b) is selected and the same process as before is applied ((6), (7)). Another solution is found : $S2 = \{(x_3, c), (x_2, b), (x_1, a)\}$. The algorithm backtracks again ((8) and (9)). When in (9), the next disconnected vertex from (x_1, b) is (x_1, a) which is selected to extend a new branch. But when $NewNOT$ is computed (10), it contains the pair (x_2, b) which is connected to the only vertex in $CAND$. This is a condition to bactrack (11) because a solution containing these pairs has already been computed. At this step, there is no other vertex disconnected from (x_1, b). The algorithm backtracks (12). The next selected vertex is (x_1, c) but when selected, it is no longer possible to compute solutions containing the x_3 variable (13): (x_1, c) is not connected to (x_3, c). Then the algorithm backtracks: no other vertex can be selected. The algorithm stops (14).

4 EXPERIMENTATIONS

4.1 The RNA problem

In molecular biology, a molecule of ribonucleic acids (RNA) is composed of ribonucleotide units linked together by covalent bonds. Each unit consists of a phosphate group, a ribose sugar and a base. There are four types of bases commonly identified by a single letter: A(denine); G(uanine); C(ytosine) and U(racile). Such a molecule is commonly represented in databanks as a word built from these four letters. The secondary structure of a RNA molecule is an intermediate level of folding of the molecule where individual units interact with each other through hydrogen bonds to form complementary base pairs (C-G, A-U and G-U) according to a set of constraints. With basic constraints, the number of possible secondary structures is about 1.8^N with N the number of ribonucleotide units. The problem to solve is to determine a good model of a secondary structure.

Two computationnal techniques are used currently which have in common the essential knowledge of the primary structure. The comparative analysis consists in a systematic search for compensatory mutations in an alignment of homologous RNA sequences from several organisms whereas the thermodynamic optimization proposes optimal and suboptimal foldings when only one sequence is available. Each of these approaches brings different structural informations and has its own limitations. This paper adresses the case of only one available sequence.

Algorithms used to find the secondary structure with the lowest free energy associate all the regions that can form helices and loops with a numerical value calculated from empirical base pairing, stacking and loop energies [13]. The combinatorial approach works by generating all the possible helices that can be formed from the sequence and then combines them into optimal and suboptimal valid structures [14]. But the most commonly used approach is based on dynamic programming. First applied by Nussinov [15] to maximize base pairing, it was extended to energy minimization by both the same author [16] and by Zuker [17]. The main advantage of dynamic programming is speed and thus the ability to fold large molecules (up to 2000 bases). Both approaches are, however, ultimately limited in that they rely on our partial understanding of the parameters necessary for the calculation of the free energy [18, 19]. They allow to compute optimally folded structures that may not represent the actual base pairing relationships found in the RNA molecule because many folded structures are possible with very similar free energies and because other cellular elements may stabilize active RNA structures that are otherwise thermodynamically unstable. A partial solution to this problem was to extend folding programs to allow for the calculation of a series of possible structures that take into account a given set of biochemical data. The most popular of these is that of Zuker [20], but there are others [21, 22]. There are three main drawbacks to this approach. Firstly, knowledge cannot be incorporated dynamically during the process of specification of the secondary structure. This makes difficult to test new ideas about RNA folding and to incorporate new data without restarting the program. In Zuker's program, it is possible, before the process of folding, to incorporate knowledge but this introduces a bias in the energy function. Secondly, the approach relies entirely on the thermodynamic model, which has been shown recently to contain bad parameters for loops [23]. However, Zuker's program takes into account some tetraloops when detected [19]. And finally, some of the computed secondary structures are unsaturated (non maximal solutions), that is some of the structures listed are included into others.

In comparison to those latter approaches, we propose an interactive approach working with single bases or pairs of bases. It relies on the formulation of the problem in the framework of binary constraint satisfaction problems (CSP) both to specify a secondary structure through a set of folding constraints and to compute all the suboptimal saturated RNA secondary structures which satisfy all the folding constraints. At the start, it relies on the computation of the probabilities of pairing of each base with all others [24]. The constraint satisfaction formulation of the problem deals dynamically with a chosen set of folding constraints and, finally, a search algorithm computes all the suboptimal saturated secondary structures which satisfy those folding constraints. Within such a framework, it is possible to test new ideas about RNA folding and secondary structures including pseudoknots can be computed. Constraints are either the classic folding restrictions (e.g. three unpaired bases at least in a hairpin loop between two paired bases, absence of pseudoknot, canonical pairs, forced paired bases, forced unpaired bases) or could be any other unary or binary constraints. In order to compute a series of suboptimal secondary structures, we use the probabilities of pairing of each base with each other as unary constraints in order that only those pairs which are the most probable can be selected. These probabilities of pairing are computed with McCaskill's program [24] which is based on the calculation of the partition function within the thermodynamic model. In the end, a matrix of probabilities of pairing of each base with all others is available and offers a view of the search space. Thus, pairs of bases or unpaired bases can be selected in the matrix of probabilities as a part of the structure and a series of suboptimal saturated secondary structures are computed that agree with all the constraints. The main advantages of this approach consist in computing structures that may contain pseudoknots and the interactive management of constraints.

4.2 Representation

In specification step, a variable x_i is associated with a base b_i in the sequence and represents the partner of b_i in the secondary structure. Each variable x_i has a domain d_i of candidate values v_{ij} which are the bases that can pair with it. The assignment $x_i = v_{ik}$ represents the pair (b_i, b_k). Because all the bases are not paired in a secondary structure, an additional value is necessary in domains meaning the base is unpaired. Thus, each pair $(variable, value)$ represents a possible pair of bases or a base unpaired in the secondary structure.

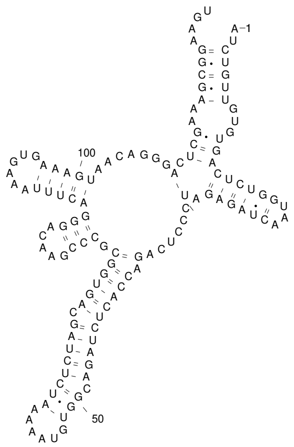

Figure 3: A secondary structure in accordance with usual constraints. Pairs A-U, G-C and G-U are represented by different symbols: a dot for a G-U pair; a line for a A-U pair and a double line for a G-C pair.

This representation makes it possible to apply arc-consistency algorithms to maintain arccoherence. Unary constraints come from usual restrictions among them pairing (A pairs only with U, G pairs with C or U) and stereochemical (three unpaired bases between two paired bases, size of helices). Other unary constraints come from available experimental data (a base is paired with another, a base is unpaired, a base is paired but its partner is unknown)or encode the thermodynamic model through a probabilistic model. Binary constraints encode the absence of pseudoknot and the fact that a base is paired to only one another (see figure 3). A detailed description of constraints is given in [10].

In second step of searching, each vertex of the consistency graph represents a possible pair in a secondary structure. Pairs meaning that bases are unpaired are removed in order to compute only saturated structures *i.e.* structures that are not contained in others. Because all the bases are not paired in a secondary structure, complete assignments cannot be found. With this representation, searching for all the maximal assignments amounts to computing all maximal cliques of the consistency graph.

When data assert that some nucleotides are paired but the partner is unknown, variables associated with these nucleotides must appear in computed assignments. The algorithm proposed in section 3 to solve $Max_V CSP$ problems was used in that second step to compute RNA secondary structures according to given pairing experimental data.

4.3 Results

First experiments show the interest of using **SearchForMaxSol** to solve $Max_V CSP$. Other results illustrate the folding of RNA sequences based on the proposed representation when experimental data are available, pseudoknot exists and interactive folding is desirable.

4.3.1 Comparison between search procedures

Tests were realized with the same RNA sequence. Variables that must appear in computed assignments (X_V) were randomly chosen according to compatible possible base pairs. 10 files were generated with $\mid X_V \mid = 5$. Two approaches were compared in our experiments (Figure 4).

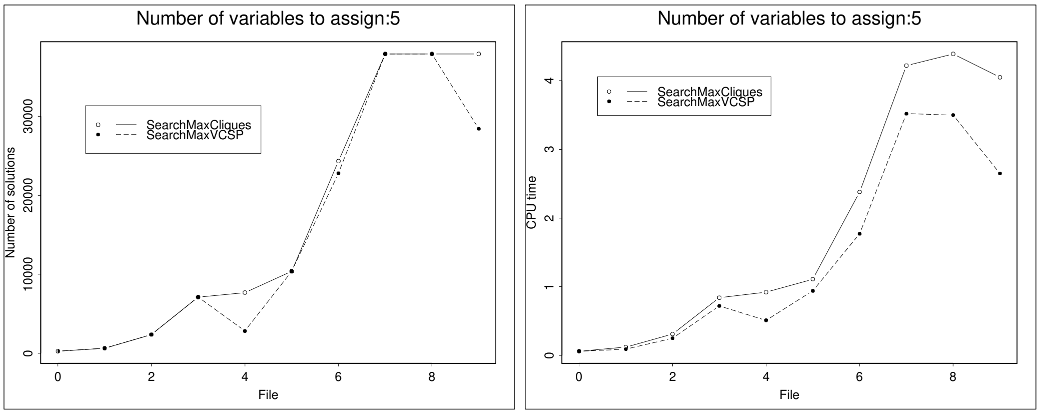

Figure 4: (*Left*). SearchMaxCliques gives the number of maximal cliques when no variable is imposed in computed assignments. SearchMaxVCSP gives the number of maximal cliques computed by *SearchForMaxSol*. (*Right*). SearchMaxCliques gives the CPU time when computing all the maximal cliques then filtering them to give only those which contain a given subset of 5 variables. SearchMaxVCSP gives the CPU time spent by using the algorithm *SearchForMaxSol*.

With the first one, all the maximal cliques were computed then filtered in order to give as solutions only those which contain variables in X_V. The algorithm used is practically the same as the algorithm described in section 3. The only difference is that the set $IMPOS$ is not maintained and a filtering procedure evaluates computed assignments. The algorithm described in section 3 was used as a second approach. Figure 4 (left) shows the effect of considering variables in X_V on the number of maximal cliques computed. One can note that the number of solutions differs a lot according to the variables in X_V. In figure 4 (right), CPU times are reported in order to estimate the cost differences. In all cases, $SearchMax_V CSP$ gives better CPU times. This argues that the cost of maintaining a set $IMPOS$ and testing each partial solution is cheaper than computing all the maximal cliques then filtering them.

4.3.2 Real data

In the case of structured RNA determination, the knowledge of the state (paired or unpaired) of bases in a molecule strongly relies on results of chemical and enzymatic probes or comparative analysis. These data allow to constrain the folding of the molecule and only those structures that agree the data are explored as the secondary structure candidates.

$SearchMax_V CSP$ is implemented in the SAPSSARN program [10]. The potential of how SAPSSARN can help in 2D-folding was illustrated with three molecules the length of which is respectively around 400 nucleotides for the group I intron *Td* [25] and the RNaseP of *Escherichia coli* [26] and around 200 nucleotides for the mRNA of *Escherichia coli* ribosomal protein S15. A protocol which mimics the hierarchical folding of RNA was tested in the case of longest RNA sequences. Results revealed that hierarchical view applies very well in the case of the group I intron *Td*. The case of RNaseP of *Escherichia coli* is more complicated because a lot of choice remain possible after a first selection of pairs. The case of mRNA S15 was chosen because it can fold in two mutually exclusive conformations that are in dynamic equilibrium and experimental data were available for the wild-type and a set of mutants [27]. It is a good example to test the local removal of pseudoknot constraints based on experimental data. We used the CFP5517 mutant in our tests (G45 instead of C45). For both wild type and mutant, only residues with strong reactivity in chemical probing were used to force unpaired bases. In the case of the wild type, bases A13, U16, A17, G18, A19, G20, U61 and G62 are forced unpaired. Bases G9, C10, U11, G12, G27, U28, C29, U65, C66, U67 and A69 are forced paired. The only structure found contains helices H1a, H1b and H2 (Figure 5, on the left).

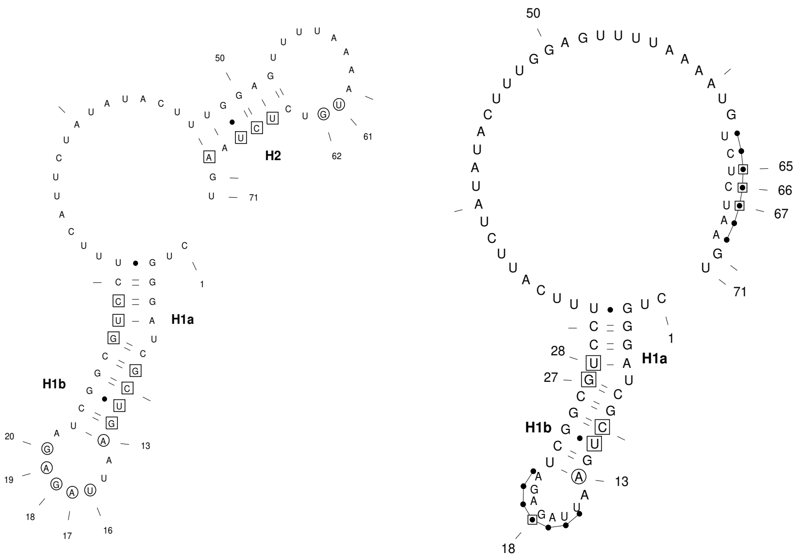

Figure 5: mRNA S15 tagged according to experimental data. Circles represent unpaired nucleotides and squares represent paired nucleotides.

In the case of the CFP5517 mutant, A13 only was forced unpaired and C10, U11, G18, G27, U28, U65, C66 and U67 were forced paired. In that case, the matrix of probability reveals possible pairs in agreement with experimental data (G18 paired with C66) allowing for the formation of a new helix containing 8 base pairs but leading to a pseudoknot. The pseudoknot constraint was then removed for the concerned bases (from 63 to 69 and from 15 to 21). Finally, all the saturated structures were computed leading to the pseudoknotted structure on the right side of Figure 5. Evidence for this structure as an alternative conformation for S15 is given by Philippe and coworkers [27].

5 CONCLUSION

Standard constraint satisfaction problems and partial constraint satisfaction problems solution techniques do not answer the problem of computing solutions when no numeric measure of the quality of the solution is available or when the measure is not a numeric one. The algorithm we proposed shows a real advantage with respect to the calculation and filtering of all the partial solutions when it is not possible to assign a value to each variable and somes variables have to be assigned. We experimented successfully this algorithm on 2D-folding of RNA molecules in order to highlight its behaviour but other applications of this idea are found in design, configuration and resource allocation problems.

References

[1] R. J. Dechter and J. Pearl, "Network-based heuristics for constraint-satisfaction problems," *Artificial Intelligence*, vol. 34, no. 1, pp. 1–38, 1987.

[2] E. C. Freuder, "A sufficient condition for backtrack-free search," *Journal of the ACM*, vol. 29, no. 1, pp. 24–32, 1982.

[3] E. C. Freuder, "A sufficient condition for backtrack-bounded search," *Journal of the ACM*, vol. 32, no. 4, pp. 755–766, 1985.

[4] A. K. Mackworth, "Consistency in networks of relations," *Artificial Intelligence*, vol. 8, pp. 99–118, 1977.

[5] R. M. Haralick and G. L. Elliot, "Increasing tree search efficiently for constraint satisfaction problems," *Artificial Intelligence*, vol. 14, pp. 263–313, 1980.

[6] C. Bessiere, E. C. Freuder, and J. Régin, "Using inference to reduce arc-consistency computation," in *Proceedings of the 14th IJCAI*, (Montréal, Canada), August 1995.

[7] E. C. Freuder and R. J. Wallace, "Partial constraint satisfaction," *Artificial Intelligence*, vol. 58, pp. 21–70, 1992.

[8] S. Mittal and B. Falkenhainer, "Dynamic constraint satisfaction problems," in *AAAI90, Proceedings of the Eighth National Conference on Artificial Intelligence*, 1990.

[9] E. Tsang, *Foundations of constraint satisfaction.* Academic Press Inc., 1993.

[10] C. Gaspin and E. Westhof, "An interactive framework for RNA secondary structure prediction with a dynamical treatment of constraints," *Journal of Molecular Biology*, vol. 254, pp. 163–174, 1995.

[11] M. R. Garey and D. Johnson, *Computers and Intractability, A Guide to the Theory of NP-completeness.* W.H. Freeman ans company, 1979.

[12] C. Bron and J. Kerbosch, "Finding all cliques of an undirected graph," *Communications of the ACM*, vol. 16, September 1973.

[13] D. Turner, N. Sugimoto, and S. M. Freier, "RNA structure prediction," *Ann. Rev. Biophys. Chem.*, vol. 17, pp. 167–192, 1988.

[14] M. Gouy, "Secondary structure prediction of RNA," *Nucleic Acids and Protein Sequence Analysis, A Practical Approach*, pp. 259–284, 1987.

[15] R. Nussinov, G. Pieczenick, J. Griggs, and D. Kleitman, "Algorithms for loop matching," *SIAM J. Appl. Math.*, vol. 35, pp. 68–82, 1978.

[16] R. Nussinov and A. B. Jacobson, "Fast algorithm for predicting the secondary structure of single stranded RNA," *Proc. Natl. Acad. Sci. USA*, vol. 77, pp. 6309–6313, 1980.

[17] M. Zuker and P. Stiegler, "Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information," *Nucleic Acids Research*, vol. 9, pp. 133–148, 1981.

[18] J. A. Jaeger, D. H. Turner, and M. Zuker, "Improved predictions of secondary structures for RNA," *Proc. Natl. Acad. Sci. USA*, vol. 86, pp. 7706–7710, 1989.

[19] A. E. Walter, D. H. Turner, J. Kim, M. H. Lytlle, P. Muller, D. H. Mathews, and M. Zuker, "Coaxial stacking of helixes enhances binding of oligoribonucleotides and improves predictions of RNA folding," *Proc. Natl. Acad. Sci. USA*, vol. 91, pp. 9218–9222, 1994.

[20] M. Zuker, "On finding all suboptimal foldings of a RNA molecule," *Science*, vol. 244, pp. 48–52, 1989.

[21] K. Yamamoto and H. Yoshikura, "An improved algorithm for the prediction of optimum and suboptimum folding structures of long single-stranded RNA," *Computer Applications in BioSiences*, vol. 3, pp. 31–35, 1987.

[22] A. Williams and I. Tinoco, "A dynamic programming algorithm for finding alternative RNA secondary structures," *NAR*, vol. 14, pp. 299–315, 1986.

[23] V. Antao and I. Tinoco, "Thermodynamic parameters for loop formation in RNA and DNA hairpin tetraloops," *Nucleic Acids Research*, vol. 20, pp. 819–824, 1992.

[24] J. McCaskill, "The equilibrium partition function and base pair binding probabilities for RNA secondary structure," *Biopolymers*, vol. 29, pp. 1105–1119, 1990.

[25] F. Michel and E. Westhof, "Modelling of the three-dimensional architecture of group I introns based on comparative analysis," *J. Mol. Biol.*, vol. 216, pp. 585–610, 1990.

[26] B. James, G. Olsen, and N. Pace, "Phylogenetic comparative analysis of RNA secondary structures," *Methods Enzymol.*, vol. 180, pp. 227–239, 1989.

[27] C. Philippe, L. Bénard, C. Portier, E. Westhof, B. Ereshmann, and C. Ehresmann, "Molecular dissection of the pseudoknot governing the translational regulation of *Escherichia coli* ribosomal protein S15," *Nucleic Acids Research*, vol. 23, pp. 18–28, 1995.