Chapter 5

Conclusions and Future Work

Set covering and strategies to solve NP-complete problems have two complementary sides of view. They can be analyzed from the perspective of time and from the one of exactness. But when different strategies are mixed, exactness and approximation, interesting results can be obtained.

Differences between the modalities of the exact algorithm were analyzed. The prunes proved to be very useful; however, using a greedy algorithm in order to get an initial guess for the upper bound for the pruning was not successful with the data sets tested.

The algorithms were tested in random inputs as well as real inputs. Both parameters were registered for every program, time and solution. Exact algorithm worked in a very good time according to expected results. It grows exponentially; then, it can not be used for too big inputs. In the input of fifty subsets and fifty elements this algorithm has an excellent performance, time is almost zero seconds. This is a very good performance is observed that the algorithm is $\Theta(2^n)$ where n is the number of subsets. However, it has a reasonable time until around 100 elements and 100 subsets in a random input. Some times it has also a very good performance when the best solution is low, as was observed in the HPV data set. The number of elements does not affect too much in time but in memory. The number of subsets is part of exponential function which bounds

the Θ of the algorithm. It is also observed a good performance on DLV specification, even if it is not an algorithm ad-hoc.

An aspect which could be improved in exact algorithm is the selection of subsets. It takes any subset and proves it. Being more specific, it takes the first set between those that have not been taken. An approach could be selecting a big one.

On the other hand, polynomial algorithms have interesting results when they use a different strategy to avoid their own weakness. For example, greedy algorithm has the problem of finding the best solution for only part of the problem. Therefore, it is adopted a strategy of finding the best solution for a bigger portion of the problem. This strategy works for this specific problem because each subset considered considerably increases the size of the considered portion of the solution. At the end, there are solutions at a higher level. The inexactness of the strategy proves then what has been suspected by people who studies NP, this is, the lack of a pattern on finding best solutions. In this case the lack is presented when greedy approach is used.

The same case is found in dynamic programming approach. It considers partial solutions to find the best solution, and shares these lower-level solutions. Therefore, it is expected to have a good performance, but sometimes their answers are far of the exact algorithm. It would be important to find a function of error bound. Although the algorithm is less important than

If properties of instances of a special analyzed problem are found, this fact will help on the search of the best algorithm for some kind of problems. Sometimes the NP-problems of some kind are solved satisfactorily with an approximation algorithm. This occurs if they, for example, fulfill some property, like the number of elements that belong to the sets, the number of ties between cardinalities and others.

Now that software for this problem is working, it will be interesting to analyze other natural data sets to find which algorithm works better in this case, or to find

cases where a specific algorithm has a better performance. Simple application software can be programmed to use them all. It would define, for example, if a data set is small enough to be treated with exact algorithm, or if after getting an approximate solution with the greedy algorithm, can be asked for a better result using dynamic programming algorithm, its recursive version which uses memory.

An interesting aspect to analyze is the instances where ties are presented. The classic greedy takes the set with the lowest i. It would be interesting to find how strong is the relation between the number of ties found and the final error in the answer obtained by the algorithm.

Another parameter is the density of the sets. This could be important to select the kind of algorithm to use. There are many algorithms designed to solve set covering, as it was explained in chapter 2. For example, greedy algorithm takes the biggest set. It can be easily seen that some degree of error is introduced when a selected set intersects with many other sets. Therefore, in this case the greedy algorithm does not seem a very good option. But these observations are merely intuitive so it would be interesting to get statistical information about these aspect, or even better, mathematical proofs of some properties of instances. The idea of properties can be generalized to number of intersections, ties, average size of sets and so on.

As NP complete problems are supposed to be unpredictable, to find a set of polynomial algorithms which assures to solve them all, is a very ambitious idea. However, for some polynomial algorithms like greedy, a proven error bound has been found. Then, if an algorithm has an error bound, it could also be found some properties when could be assured that some algorithm will find the best solution, or an error bound for specific instances according to the parameters found before.

In the context of DNA sequences, there are more algorithms to solve set covering. It is expected that different approaches, like linear programming or neural networks

could be tested. Different conversions from DNA typing problem to other NP complete problems would be a point to analyze. For example, traveler agent has been used also to solve other problems related to DNA. The more diversity of algorithms and approaches, the better solutions will be obtained, even if it is not the best.

In the case of HPV, RFLP technique to distinguish DNA sequences introduces some other factors. For example, sometimes it is not that important to find an algorithm to classify a sequence into a defined type, but only to know if a virus is dangerous or not. As types are classified in high risk, low risk and unknown risk types, the problem introduces a restriction of only three different sequences. Set covering could be adapted to solve the specific problem. Even more, it could be found a polynomial algorithm which solves a restricted problem like this one.

There is other problem related to RFLP. It is about priority of enzymes, for example, if they are more difficult to find, or more expensive. An element of complexity is introduced. Some matrices from instances of DNA would have more cost than others. This problem can be seen as weighted set covering because matrices are finally represented as sets. Weighted set covering is also an NP complete problem, therefore, some algorithms like greedy can be applied if they are adapted.

NP problems are present in real life problems. Even if no exact algorithm is found, they are important and can not be ignored. There is no proof about if it is possible or not to construct an algorithm. Therefore, polynomial approximations have to be still used to solve these problems satisfactorily. Data about the properties, correlations and any kind of information that helps to optimize the algorithms must be produced and selected. In this way, even if NP completeness is not understood yet, practical solutions work for the real problems, expecting to contribute to this comprehension.