Chapter 4

Results

Now, results obtained from the proofs of the algorithms will be presented. Three sets of different randomly created instances have been tested, using all the algorithms explained in the Chapter 3. Results from natural data sets are presented also.

The first set contains 20 inputs with number of elements going from fifty to 800 elements and subsets in the range of 4 to 150. The second set contains 60 inputs of 50 elements and the number of subsets greater or equal to 50 and less than 360 elements. The third set contains 80 inputs of 50 subsets and the cardinality of base set is from 50 to 2025 elements.

4.1 Exact algorithm

The greedy algorithm used to find a maximum solution worked as expected; this is, with a very good approach. However, an interesting result is that even with the inclusion of the greedy approximation for the definition of a starting bound of the exact branch and bound algorithm there is no significant improvement in the execution time, according to the proofs no more than 8%. Table 4.1 presents a time comparison between these algorithms using some instances with 50 elements and number of subsets from 50 to

205 subsets, from the second set of proofs.

As an explanation, it is supposed that the algorithm finds a reasonable bound very fast. Of course, this does not imply that the greedy bound does not result at all in advantages over the execution time. However, at least with the instances tested they were not significant.

The implementation of the algorithm was experimentally compared against the DLV specification, in order to correct any implementation errors that could have been generated. Most of the times (95%) the answer has been gotten faster than the DLV answer, but there are some instances where this has not happened. The smaller the number of concurrent best solutions, the faster DLV gets an answer. In Table 4.1 some comparison times obtained in SUN environment are shown. The first set of proofs was used, and it is also included a data set from HPV-RFLP.

Three natural data sets have been also tested: RFLP data from the HPV-typing problem and RFLP and hybridization data from the HLA-DQ typing problem. The sets were provided by Dr. Javier Garcés, who got them in the public electronic database from the National Center of Biotechnology Information [13].

In the last two cases HLA typing were analyzed without the complications that arise due to diploidy. In the first two, the exact algorithm found an answer in reasonable time: In the case of HPV it was 2 seconds and in HLA-RFLP it was 0.01 seconds. This is because the number of enzymes was small (2 enzymes), so there were not too many recursive calls.

In Figure 4.1 total time of the algorithm for different instances is shown, incrementing the number of subsets, and in Figure 4.2 total time function is appreciated when the number of elements is incremented.

Table 4.1: Comparison in time (seconds) between exact algorithm using (EG) and not using (ENG) a bound based on greedy algorithm. It's also presented the number of prunes done using maximum size limit, with and without this first bound. It can be observed that they are almost the same.

Sets	Elements	T. EG	T. ENG	Prunes EG	Prunes ENG
50	50	3.3985	3.36434	12858	12858
60	50	0.1144	0.102285	227	226
70	50	1.2390	1.1997	2777	2774
80	50	2.7783	2.57918	9219	9217
85	50	5.0843	4.81146	10925	10909
90	50	5.5408	5.56254	9946	9946
95	50	17.2640	20.6799	32434	44677
100	50	4.8546	6.4686	6245	6245
105	50	1.8361	2.64004	2673	2673
110	50	4.4751	6.69684	6072	6064
115	50	7.7159	7.71706	11621	11619
120	50	0.8905	0.820072	321	321
125	50	46.3130	46.1713	74214	74241
130	50	24.7477	23.9185	41802	41850
135	50	52.2379	51.8721	58778	58775
140	50	44.7466	49.327	76505	76505
145	50	10.9418	12.1232	15946	15968
150	50	19.7741	20.4814	25228	25227
155	50	134.5595	181.679	115951	215115
160	50	1.4955	2.56748	359	359
160	50	78.947	87.245	97441	109433
165	50	12.9901	12.9136	7037	7036
170	50	6.4636	6.18707	3570	3568
175	50	174.355	171.262	152884	153012
180	50	18.5014	19.5978	14317	14316
185	50	82.6539	80.8053	67636	67635
190	50	211.909	212.581	162979	162979
195	50	18.7409	22.2631	9133	21719
200	50	40.2083	40.7717	33035	33035
205	50	154.422	151.799	195264	195271

Table 4.2: DLV compared against exact algorithm in C++ using SUN operative system and Solaris environment using 14 random inputs and HPV data set. Time presented in seconds

input #	m	n	Time Exact algorithm	Time DLV	Solution	Number of optimal solutions
1	16	4	0	0	2	2
2	16	8	0	0	3	2
3	32	8	0	0	3	3
4	32	16	0	0	4	5
5	50	50	4	17.8	4	1
6	60	50	1	58	6	15
7	70	50	2	36	6	4
8	80	50	1	56	8	42
9	100	50	192	231	6	1
10	200	50	49	15	9	2
11	800	50	4898	418	17	10
12	60	60	14	723	6	61
13	70	70	215	1362	6	27
14	80	80	199	1526	5	1
hpv	1081	205	20	562	2	8

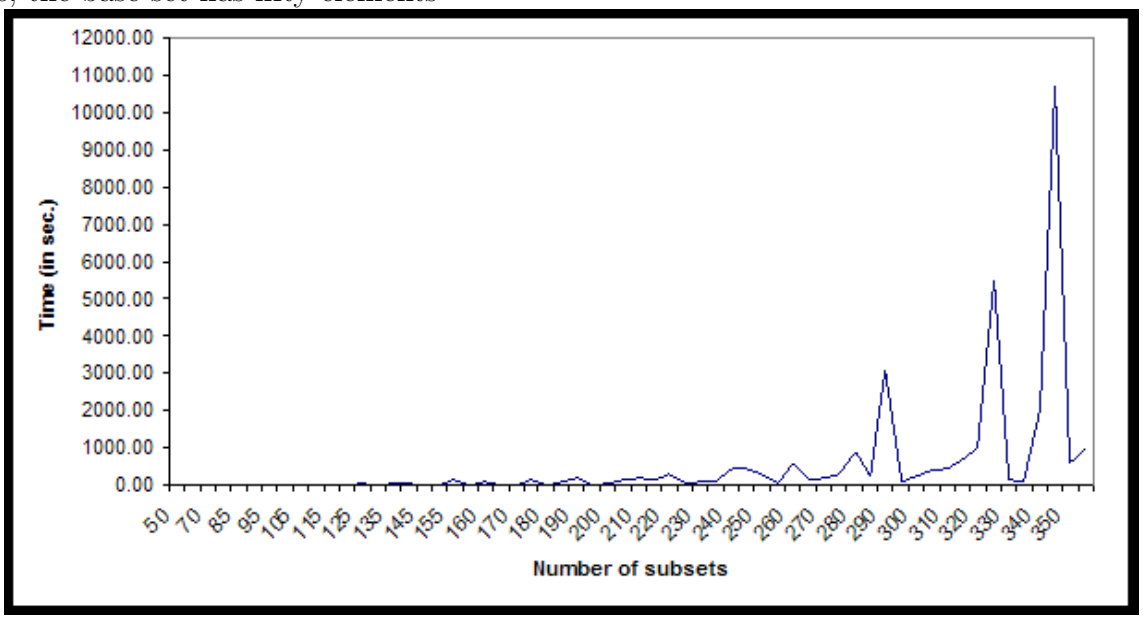
4.2 Approximation algorithms

Essentially, two parameters were registered in polynomial algorithms. First of all, it was important to find the error bound of the algorithms, as well as which one of them got the best solution, the nearest to the exact solution. Time was also an important element because although it does not increment exponentially, the algorithms are bounded by different time complexity functions, as it was explained in the previous chapter. For example, simple greedy algorithm is Θn^2 and greedy-by-pairs algorithm is Θn^4.

4.2.1 Greedy algorithm and variants

Different algorithms with greedy approach were tested. Greedy algorithm is one of best approaches found for set covering because it has a very good bound [14]. Bound function was explained in Section 2.2.

Figure 4.1: Time of solution in seconds, Exact algorithm. Proofs of the second set, this is, the base set has fifty elements



It yielded the same optimal solution than exact algorithm for HPV and HLA sequences. For the third natural data set, hybridization in HLA, the greedy algorithm got an answer of 14 elements, as well as the other polynomial algorithms, although it was impossible to compare it with the exact algorithm. Table 4.3 presents the solutions.

Greedy by pairs (GP) is same than greedy (SG) but it takes pairs of subsets instead of single subsets. This approach reduces problems of "local minimum", as it can be seen in this example.

Example 4.1. Let $F = \{S_1, S_2, S_3, S_4, S_5\}$ and $X = \{a, b, c, d, e, f, g, h\}$

$S_1 = \{a, b, c, d\}$

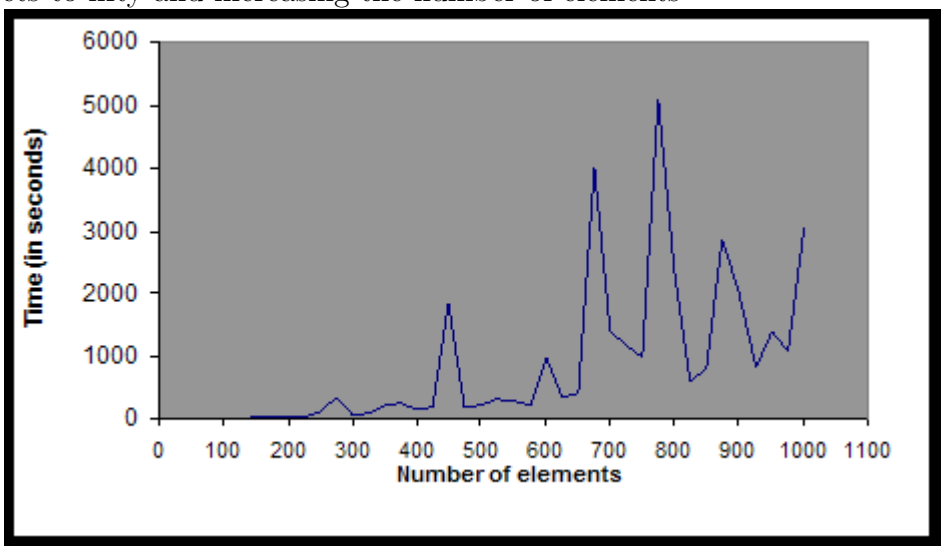
$S_2 = \{a, c, h\}$

$S_3 = \{b, e\}$

$S_4 = \{c, g\}$

$S_5 = \{d, f\}$

Figure 4.2: Time of solution in seconds, Exact algorithm. Proofs of the third set, fixing the subsets to fifty and increasing the number of elements



In this simple example can be seen how greedy algorithm will select all the subsets as the best solution, because it takes S_1 as the subset with the highest cardinality. However, the best solution is obtained from subsets $\{S_2, S_3, S_4, S_5\}$. This is because S_1 can be easily obtained from the other sets. If a pair strategy is used, the first selection will be the pair $< S_2, S_3 >$, and then it will get the optimal solution.

However, another problem appears with this algorithm. Observe this example:

Example 4.2. Let $F = \{S_1, S_2, S_3, S_4\}$ and $X = \{a, b, c, d, e, f\}$

$S_1 = \{a, b, c\}$

$S_2 = \{a, d\}$

$S_3 = \{b, e\}$

$S_4 = \{c, f\}$

As the previous one, there is a problem selecting the set with the biggest cardinality, because it does not belong to the best solution. Greedy by pairs (GP) selects the correct pair, in this case $< S_2, S_3 >$, in the first iteration. However, the second iteration has

no choice than selecting the pair $< S_1, S_4 >$. In this case, there is a problem when the algorithm is finishing because it takes always a pair.

Even with problems of selecting some extra sets, GP algorithm got an answer better or equal than greedy algorithm in 70 % of times in the proofs. It got solutions with a difference of 1 subset in 30% of proofs. However, some times (3%) GP algorithms got worst solutions with a difference of 2 or more subsets.

The last variation used was a Greedy algorithm that calls the Exact algorithm at a lower level (GE). As it was shown in Section 3.3, it is calculated a bound to call this algorithm at an appropriate level. This algorithm has the guarantee to find always a solution $C \subseteq F$ which is better than the solution of greedy algorithm. Moreover, the problem with greedy-by-pairs algorithm is easily solved if the same strategy of calling an exact algorithm at the end of the problem is used. It was experimentally observed that exact algorithm got an immediate answer with an input around fifty subsets and fifty elements.

GE algorithm returns always better solutions than SG, but it got better solutions than GP in 10% of proofs, and worst solutions through 40% of times, in the second set of 60 proofs.

Combining both strategies (GE and GP), an Hybrid Greedy Algorithm (HG) was implemented. It works better than GP all the times. It also worked better than SG in all the proofs; however, nothing could be mathematically assured. The GE algorithm got better answers than HG in 5% of proofs and worst solutions in 20% of times.

4.2.2 Dynamic programming algorithm

Dynamic Programming algorithm (DP) had the objective of experimenting with other approach to set covering. In general, both polynomial algorithms did not always find

the best solution but they yielded a good approach, as it can be seen in the Figures 4.5 and 4.8.

In general, the greedy algorithm had a better performance than dynamic programming approach, approximately 41% of inputs. However, in 10% of the instances, dynamic approach yielded a better result. GP algorithm had better answers 36% of times and worst in 25% of them. GE algorithm got better answers in 56% of proofs and worst answers in 3%. Even HG algorithm, with the best of all error bounds in the examples, got better answers 65% times and worst in 1%. The input where it got the best solution of both had 50 elements and 200 subsets (file `in29.in`). All the inputs can be found at [19]. It was observed that dynamic programming had more error bound when the number of elements increased; however, this is not definitive. More experiments are needed.

Therefore, in Figure 4.3 time comparison among algorithms is presented. Differences among polynomial algorithms can not be appreciated because time is too small. In Figure 4.4 is presented a time comparison among the three polynomial algorithms which do not use exact algorithm in the lower level. These algorithms got the best times in the proofs. In Figures 4.5 and 4.6, solutions of these algorithms are presented with the second set of instances of 50 elements. It can be observed that none of the algorithms got an especially better or worst performance in the set of examples.

Simple polynomial algorithms had a very interesting performance. Even if the five algorithms were polynomial, HG and GE, this is, the algorithms that call the exact algorithm have a great difference of time in this set of proofs. However, they do not get the best answers.

In Figures 4.7 and 4.8 are presented the time and solution by number of elements; this is, the third set of proofs. This set has instances of 50 subsets and the number of elements goes from 50 to 1000. As it can be observed, the best time is for simple

Figure 4.3: Time comparison among the six algorithms. Exact algorithm is not completely presented because its times are too high and does not let to be compared against the others. Instances of 50 elements in the base set.

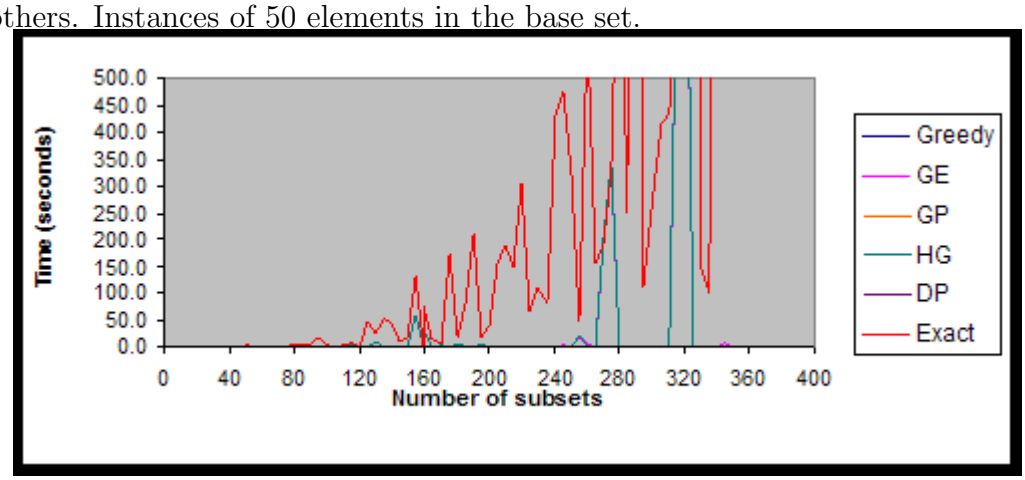


Figure 4.4: Time comparison among SG, GP and DP They got the lowest times. Instances of 50 elements in the base set (second set of proofs).

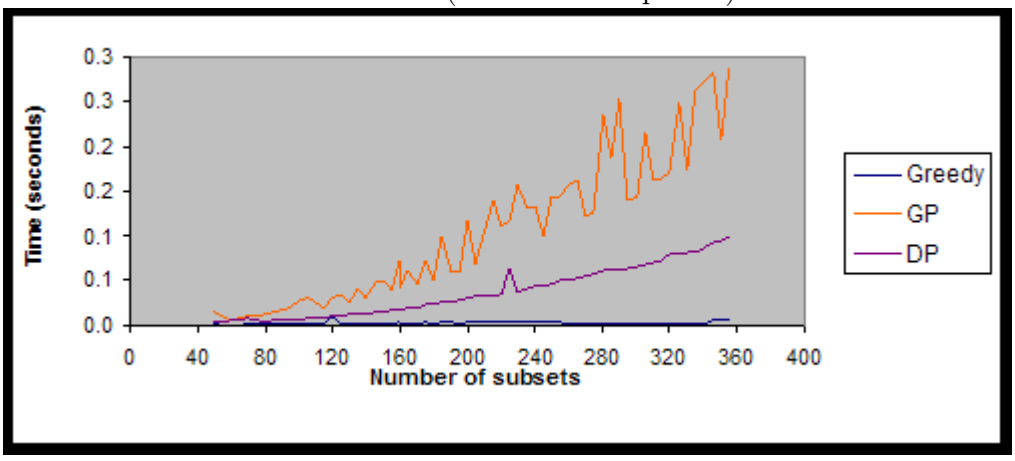


Figure 4.5: Solutions comparison basic polynomial algorithms and the Exact Algorithm in the second set of proofs, base set with 50 elements

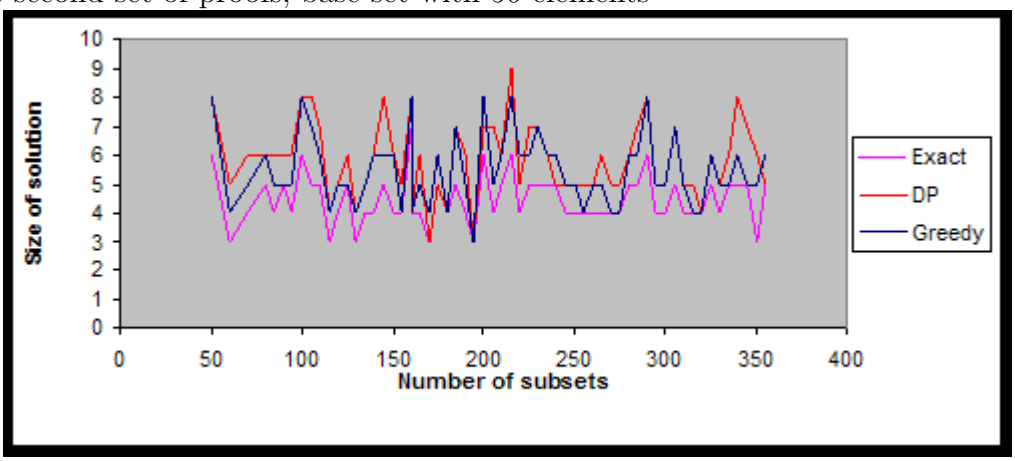


Figure 4.6: Solutions comparison among Greedy variants. Second set of proofs, 50 elements

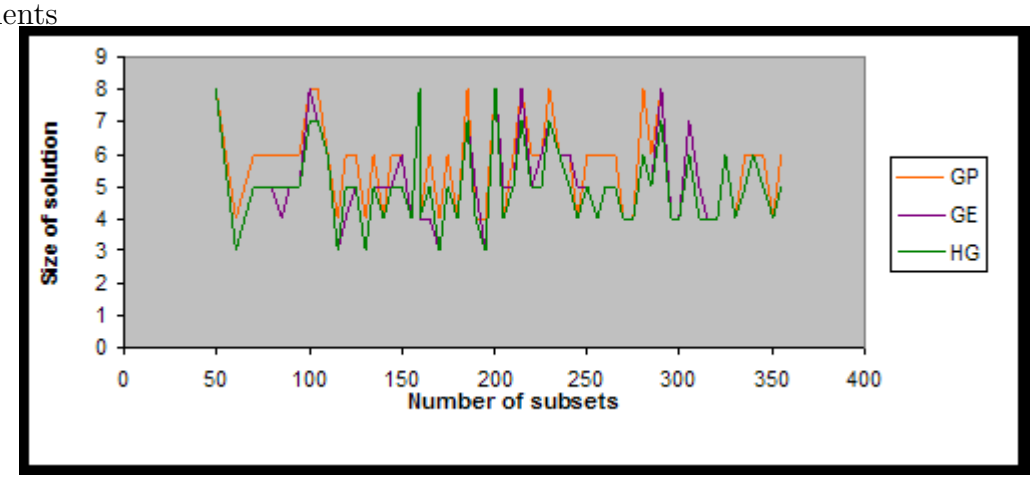


Figure 4.7: Time comparison among the six algorithms on instances of 50 subsets

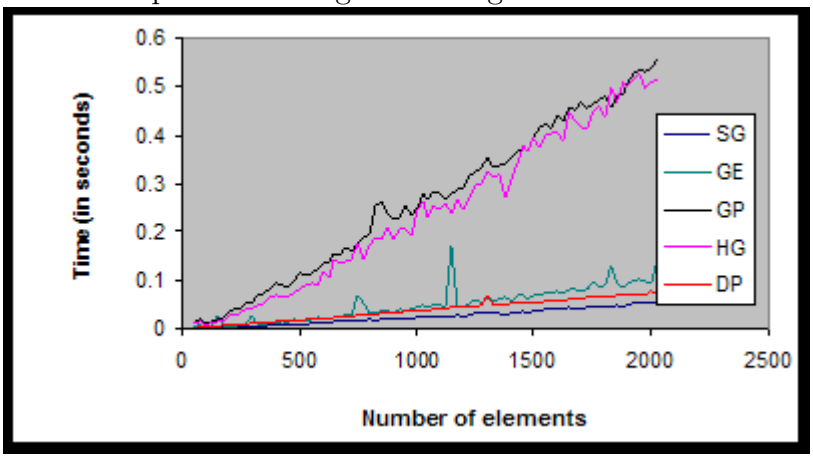
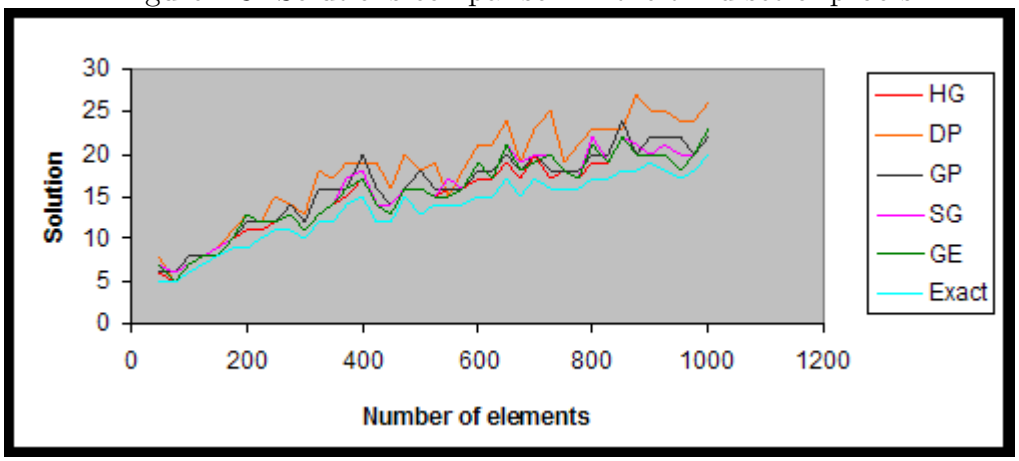


Figure 4.8: Solutions comparison in the third set of proofs



greedy algorithm and the worst is for GP algorithm. Moreover, dynamic programming solutions presents higher error ratio than the other algorithms. Some other proofs for these algorithms are presented in [16].

Dynamic programming didn't perform as good as greedy algorithm with the natural data, but it shows the same pattern described before. However, conclusions about natural data set are not definitive; more proofs are needed. In Table 4.3 results for the real data are presented.

Table 4.3: Real Data solutions given by the algorithms

Sequences	Elements	Sets	Exact	SG	GE	GP	HG	DP
HPV	1081	205	2	2	2	2	2	2
HLA-DQ	561	34	13	13	13	14	13	17
HLA-DQ (Hybridization)	1081	111	-	13	13	14	13	5