Chapter 1

Algorithmic Evaluation and Improvement

This chapter examines the algorithmic performance of TAR2. Even though we have successfully applied TAR2 on various domains, as will be shown below, its exponential runtime with respect to the treatment size is not a desirable algorithmic feature. We then present TAR3 - an improved learner that achieves the same result as TAR2 in liner time by adopting a series of strategies including random sampling.

1.1 Algorithm Performance of TAR2

Table 1.1 reports TAR2 runtimes(sec) on data sets of different sizes. It shows TAR2 is suitable for handling small to medium sized data set. For example, the algorithm learnt treatments in 23 seconds from a dataset containing 250,000 examples: see the *reachness2* domain in table 1.1.

domain	#example	#continous	#discrete	#class	size(T)	time(sec)
iris	150	4	0	3	1	< 1
wine	178	13	0	3	2	< 1
car	1,728	0	6	4	2	< 1
autompg	398	6	1	4	2	1
housing	506	13	0	4	2	1
pageblocks	5,473	10	0	5	2	2
circuit	35,228	0	18	10	4	4
cocomo	30,000	0	23	4	1	2
pilot	30,000	0	99	9	5	86
reacheness	25,000	4	9	4	2	3
reacheness2	250,000	4	9	4	1	23

Table 1.1: Runtimes for TAR2 on different domains (on a 333 MHz Windows machine with 200MB of ram). First 6 data sets come from the UC Irvine machine learning data repository; “circuit” is described in this paper; “cocomo” comes from the COCOMO software cost estimation model [MH01]; “pilot” comes from the NASA Jet Propulsion Laboratory [FM02]; “reachness” and “reachness2” come from other source [MH02].

1.1.1 Runtime vs. Data Size

To examine the runtime with respect to data size, we generated data set of different sizes from the COCOMO risk model [ACDC$^+$98]. By simulating the model, we could generate data set of any size.

Figure 1.1 shows three studies where the size of the treatments(size(T)) was held constant, and the size for the data set was increased. TAR2’s runtimes are linear on dataset size. However, larger treatment size decreases the efficiency.

1.1.2 Runtime vs. Treatment Size

Figure 1.2 shows one study where the size of the data set was held constant(3MB), and the size of treatment(size(T)) was increased. TAR2’s run-

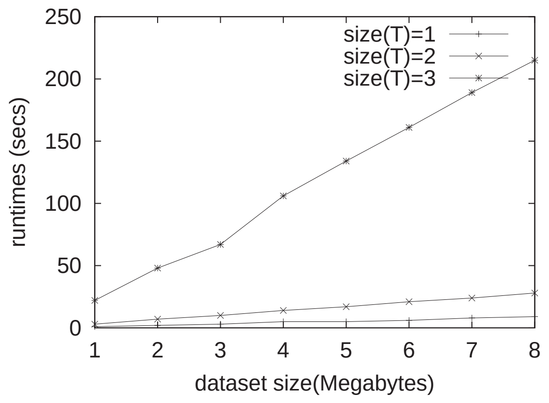

Figure 1.1: Runtime vs dataset size. Datasets are generated from COCOMO model.

times are exponential on treatment size.

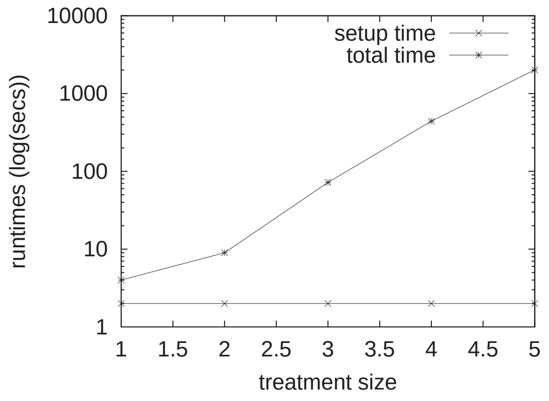

Figure 1.2: Runtime vs treatment size. Data set size is fixed to 3MB. Datasets are generated from COCOMO model. Note the Y-axis is the logarithm of the runtime.

Recall that treatments are generated by exploring subsets of pairs whose $confidence1$ value are greater than a threshold. For treatment size=r, if N pairs occurring above the threshold, TAR2 will explore $\binom{N}{r}$ such pairs. To find treatments of different sizes, there are total

$$\binom{N}{1} + \binom{N}{2} + \binom{N}{3} \ldots + \binom{N}{N-1} + \binom{N}{N} \approx 2^N$$

candidates to explore. Although the *value exclusion* property(items of the same attribute e.g. A1=a and A1=b can never be contained by the same instance) of the data set can preliminarily shrink the search space by not producing candidates with more than one value for the same attribute, the search is still intractable when N is large and will incur exponential runtime.

1.1.3 Runtime in Practice

The exponential runtime with respect to treatment size seems to rule out TAR2's feasibility for real world application. Yet surprisingly, it is not the usual case in practice. We have applied TAR2 to more than 20 domains so far. In all of those domains have we observed the same narrow funnel feature. Figure 1.3 shows the confidence1 distributions seen in eight example sets. In all cases, the $confidence1$ distribution has a small right tail; i.e. a small number of variables exert a disproportionately large influence on the overall behavior of the system. When this is true, effective treatments can usually be found with $treatmentSize < 6$

In chapter **??**, we have made an average case mathematical analysis showing that the odds of narrower funnels are millions of times more likely than wider funnels. Although such a statistical analysis may not apply to a particular domain, it does support our observation in practice. When narrow funnel exists, the exponential impact will not be fatal. However, if very large treatments are required, an incremental treatment learning approach could

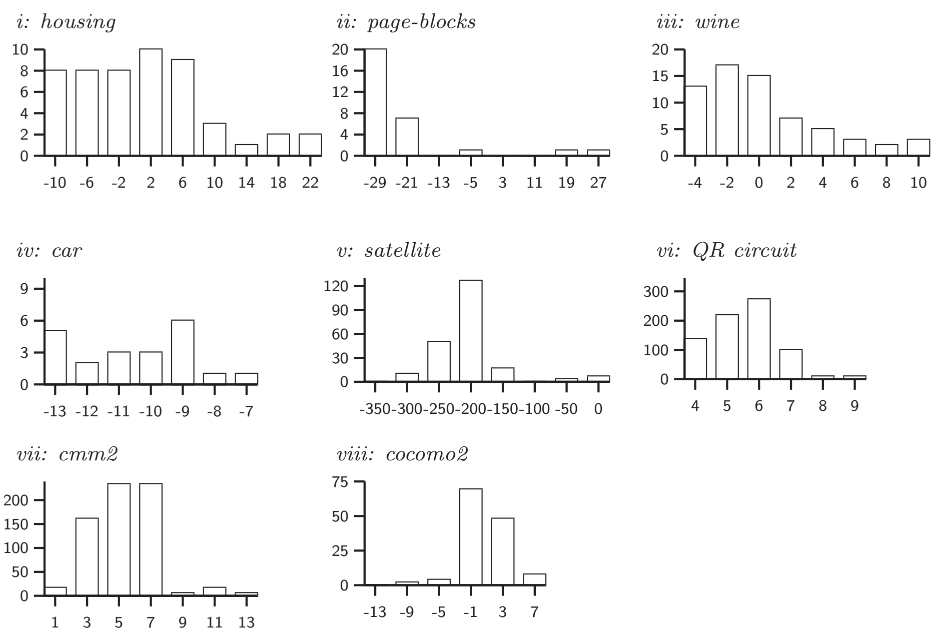

Figure 1.3: Confidence1 distributions seen in eight domains. Y-axis is the number of times a particular confidence1 was seen. (i)-(iv) come from datasets taken from the UC Irvine machine learning repository. (i)-(iv) were generated from other domains discussed in this thesis.

be considered as an alternative, such as used in the `pilot` case study.

1.2 TAR3: The Improvement

Exponential runtime is certainly not a desirable feature for a data miner. TAR3 is our solution to the problem. We have integrated various strategies into the basic algorithm, which are discussed below:

1.2.1 Random Sampling

In TAR2, confidence1 measure ($\triangle$) is a heuristic assessing the contribution each attribute-value pair makes toward changing the class distribution. If we think of $\triangle$ values as weights associated with attribute-value pairs, those with larger weights have higher probability to be selected into treatments than those with smaller weights. Let:

$$\begin{aligned} a_i &= \text{attribute-value pair in the data set} \\ w_i &= \triangle \text{ value associated with } a_i \end{aligned}$$

What we need is to sample a_i from a multinomial distribution with discrete weights w_i. This strategy is employed in TAR3 and is done as follows:

1. Place a_i in increasing order according to w_i.
2. Compute the CDF(Cumulative Distribution Function) value of w_i:
$$CDF(i) = \frac{\sum_{x=1}^{i} w_x}{\sum_{x'=1}^{N} w_{x'}} \quad \text{(N = the total number of } w_i\text{)}$$
3. Sample a uniform value u in the interval [0,1].
4. The sample a_j is the least a_i such that $u < CDF(i)$

The above process is repeated until we get a treatment of $size(T)$. As a side effect, random sampling also eliminates the necessity to specify the confidence1 threshold.

1.2.2 Treatment size

TAR2 requires $size(T)$ be specified by the user. Each run of TAR2 returns treatments of that fixed size. In TAR3, $size(T)$ is a uniform value sampled from the interval $[1..maxTreatmentSize]$, where $maxTreatmentSize$

could be the total number of attributes in the data set. This allows us to obtain treatments of different sizes in one run. In practice, we found that $maxTreatmentSize$ seldom exceeds half of attribute number.

1.2.3 $lift(T)$ pruning

A candidate treatment T is evaluated by calculating $lift(T)$. Treatment T is qualified only if $lift(T) >$ certain threshold. We eventually abandoned this approach. Instead, treatment T is reported if and only if it is among the top N treatments found whose $lift(T) > 1$ ($lift(T) > 1$ ensures the treatment indeed makes improvement on class distribution), where N is the maximum number of treatments user wants.

1.2.4 $bestClassSupport$ penalization

For treatment T, $size(T)$ has noticeable impact on *worth* of the treated data set (and hence $lift(T)$). Usually treatments of larger sizes tend to achieve higher *worth* than those of smaller sizes. In the association rule mining community, Ke Wang et.al,[KW01] found that the $confidence$ of a rule holds a monotonous property known as the *universal-existential upward closure*. Our $confidence1$ is a function of $confidence$, and exhibits a similar yet not monotonous behavior. Consider one extreme case where a dataset has total 5 attributes. A treatment of size 5 might select only one example and this example belongs to the best class. This treatment however, has the highest *worth*: on the class distribution graph, 100% of examples(in this case, only 1 example) in the treated subset belong to the best class. Similarly, treatments of very large size may achieve outstanding *worth*, however, they usually select so few examples that lack statistical significance.

This problem is solved by introducing in the *BestClassSupport* parameter. *BestClassSupport* is defined as **ratio of examples belonging to the best class in the treated set to those in the original set**. i.e.,

$$BestClassSupport = \frac{|C_{best}, T|}{|C_{best}|}$$

minBestClassSupport specifies the minimum *BestClass* ratio a treatment T must achieve. For instance, assuming a data set contains 500 examples, among which only 50 belong to the best class. With $minBestClassSupport =$ 80%, if applying a treatment T results in a treated set containing 100 examples, among which 45 belong to the best class. Then the *BestClassSupport* of this treatment T is $\frac{45}{50} = 90\% > minBestClassSupport$, thus T is considered qualified. In fact, T is a very good treatment, as it raises the best class percentage from $\frac{50}{500} = 10\%$ to $\frac{45}{100} = 45\%$.

In TAR3 implementation, we didn't simply reject treatments that do not meet the *minBestClassSupport*. Instead, we use the *minBestClassSupport* as a regularizer that penalizes their *worth*. This ensures some potentially highly predictive treatments still be reported although their *BestClassSupport* are slightly below the threshold.

1.2.5 Stopping point

A theoretical drawback with any random search is that such random exploration can miss significant parts of the option space. TAR3 addresses this issue by taking the following strategy: To generate N treatments, TAR3 makes multiple iterations. Each iteration, X new treatments are generated and checked. Only qualified top N are remained in the treatment set. If the current iteration doesn't contribute any new treatments to the top N set, it is called a failure iteration. The next iteration, more(X+N) treatments are generated. The procedure stops after M failure runs in a row. In practice we found that M=[5..10] was often sufficient to return stable results.

1.2.6 Interface

The changes in algorithm also result in a more friendly interface of TAR3 . Table 1.2 lists parameters required by TAR2 and TAR3 excluding those common to both. The replacement of threshold parameters with upper-bound ones makes parameter setting easier and more intuitive. While thresholds are domain specific, upper-bounds are less sensitive. We normally use default values or set them to some larger values, the controlling strategy employed in TAR3's random process(e.g., the stopping point) was able to accommodate domains accordingly. Further, we no longer have to run it several times to get treatments of different sizes.

TAR2	TAR3
1. $\triangle$ threshold	1. maxTreatmentNumber
2. worth threshold	2. maxRandomIterations
3. treatmentSize	3. maxTreatmentSize

Table 1.2: Different parameters required by TAR2 and TAR3.

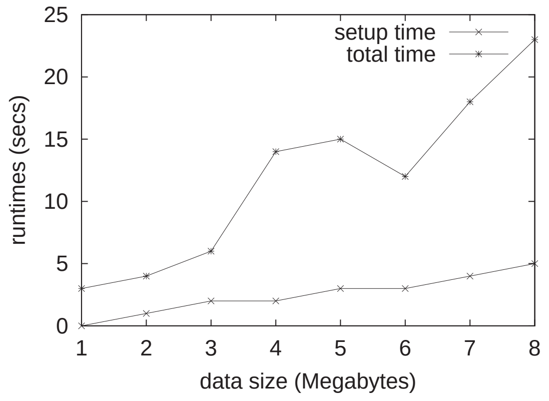

Figure 1.4: Runtime vs dataset size. Datasets are the same as in the performance study for TAR2 (figure 1.1)

1.3 Performance Improvement

Figure 1.4 reports TAR3 runtimes on datasets seen in figure 1.1. The curve fits a linear trend line with $R^2 = 0.8292$, suggesting TAR3's runtime is linear with respect to data size. Given the fact that all effective treatments are returned in one run, TAR3 has made a significant improvement on algorithmic performance.

Figure 1.3 compares TAR2 and TAR3 runtimes on the same datasets seen in table 1.1. Column 6 lists the maximum treatment size returned by TAR3 on each domain. Column 7 lists TAR2's runtimes with respect to maximum treatment size. In domains where $size(T)$ is small (e.g., the first 6 datasets), TAR2's runtimes are comparable to TAR3's. However, when $size(T)$ is large(e.g. the pilot domain), TAR3 is significantly faster than TAR2 (195sec vs 2842sec). This indicates that TAR3's random sampling is able to scale when TAR2's exponential search becomes intractable.

domain	#example	#continous	#discrete	#class	maxSize(T)	TAR2(sec)	TAR3(sec)
iris	150	4	0	3	2	1	< 1
wine	178	13	0	3	2	1	< 1
car	1,728	0	6	4	4	3	< 1
autompg	398	6	1	4	4	3	< 1
housing	506	13	0	4	4	4	1
pageblocks	5,473	10	0	4	2	2	1
circuit	35,228	0	18	10	6	18	6
cocomo	30,000	0	23	4	3	104	28
pilot	30,000	0	99	9	7	2842	195
reacheness	25,000	4	9	3	4	28	4
reacheness2	250,000	4	9	4	4	293	42

Table 1.3: Runtimes for TAR3 on different domains (on a 333 MHz Windows machine with 200MB of ram).

1.4 Experiments Results Comparison

This study aims at examining TAR3's stability in terms of the returned treatments. We wanted to know whether and to what extent would the randomness introduced in TAR3 effect its results. Firstly, we ran TAR3 on a domain and recorded the size of the best treatment returned. We then configured TAR2 so that it would return treatments of that size on the same domain. We compare both the $Worth(T)$ and the individual attribute-value pairs appeared in the best treatments returned by TAR2 and TAR3 respectively. Table 1.4 lists the results from 10 domains, in which TAR3 found the same best treatments as did TAR2. In the "pilot" domain (table 1.5), TAR3 found a best treatment of size 10, which had a much higher $Worth$ than those TAR2 returned (the max treatment size TAR2 could handle on this domain is 5). Baselined on TAR2, this comparison shows that TAR3 is able to find the same results, which we believe, is contributed to the control strategy of stopping point.

domain	attribute	range	TAR3	TAR2
iris	petal width	[1..1.6)	x	x
	petal length	[3.5..4.6)	x	x
		worth:	1.71	1.71
wine	attr11	[0.4874)	x	x
	attr12	[1.27..1.78)	x	x
		worth:	1.81	1.81
car	buying	low	x	x
	safety	high	x	x
		worth:	2.21	2.21
auto-mpg	cylinders	[4..5)	x	x
	horsepower	[46..75)	x	x
	weight	[1613..2223)	x	x
		worth:	1.97	1.97
housing	rm	[6.65..9.78)	x	x
	ptration	[12.6..15.9)	x	x
		worth:	2.35	2.35
page-block	blackand	[493..46113]	x	x
	height	[9..804]	x	x
	p_black	[0.052287)	x	x
	eccen	[0.007..2.889)	x	x
	area	[660..143993]	x	x
		worth:	9.28	9.28
circ (30k examples)	B3c	ok	x	x
	Sw2c	off	x	x
	Sw1c	on	x	x
		worth:	9.07	9.07
cocomo (30k examples)	pcap	[3..4]	x	x
	ruse	[1..2)	x	x
	acap	[3..4]	x	x
	sced	[3..4]	x	x
		worth:	1.53	1.53
reachness (25k examples)	orpMean	[6..8)	x	x
	andfMean	0.1	x	x
	noMean	[0..8)	x	x
		worth:	1.65	1.65
reachness2 (250k examples)	orpMean	[9..10]	x	x
	noMean	[0..8)	x	x
	andfMean	0.1	x	x
		worth:	1.65	1.65

Table 1.4: Best treatment returned by TAR3 and TAR2 on various domains.

domain	attribute	range	TAR3	TAR2	TAR3(best)
pilot	P44	Y	x		
(50k examples)	P317	N	x		
	P755	N	x	x	x
	P1066	N	x	x	x
	P706	Y	x	x	x
	P688	Y		x	
	P2160			x	
	P761	N			x
	P1065	N			x
	P690	Y			x
	P1069	N			x
	P2111	N			x
	P1971	Y			x
	P704	N			x
		worth:	5.93	5.84	8.16

Table 1.5: Best treatment returned by TAR3 and TAR2 on the pilot domain.

1.5 Case Study: The Pilot Domain Again

We have discussed the TAR2 application on the "pilot" case in requirement optimization domain (see chapter **??** section **??**). To compare both the performance and experimental results, we ran TAR2 and TAR3 on the same "pilot" domain again.

The model used in this case study is a revised version of the original one, which contains fewer details. The data set obtained after simulation has 58 attributes instead of 99. Each example is also evaluated by a pair of cost and benefit figures. According to the domain experts, we combined cost and benefit into a single attribute using the same balanced scheme but with different dividing thresholds. The combination resulted in 16 classes representing 16 levels of "goodness".

Figure 1.5 shows the initial cost-benefit distribution from the model simulation. The data points are widely spread across the possible cost and benefit ranges. Further, most low cost points correspond to low benefit level and high benefit points have high cost values. The desired low-cost high-benefit points are very few: less than 3% of the entire data.

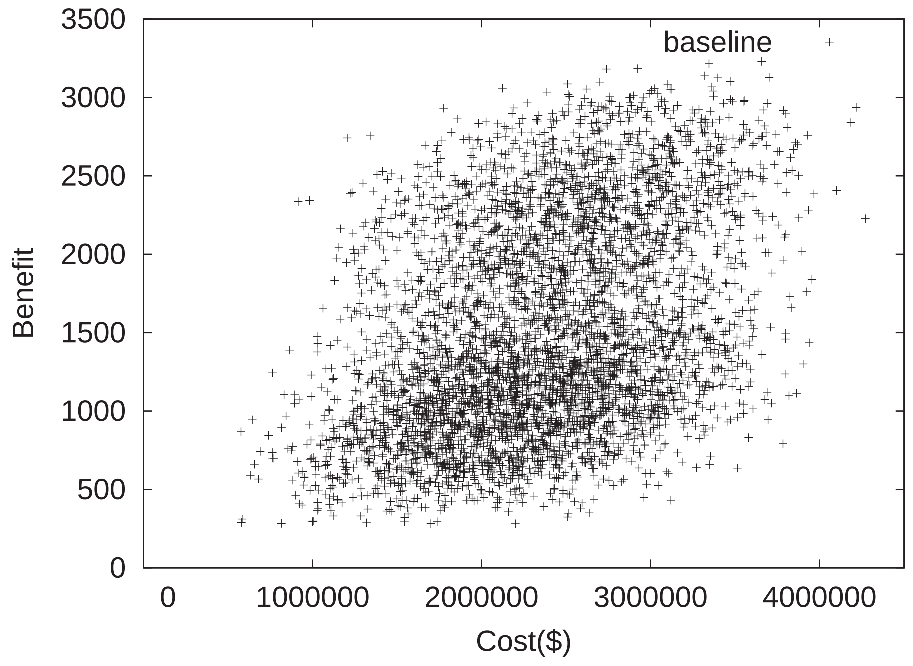

Figure 1.5: The cost-benefit distribution of the initial simulation from the pilot domain.

We followed the same incremental learning approach as discussed in the last chapter, namely the following steps:

1. Ran TAR2 and TAR3 on the baseline (initial simulation) data, and generated two set of treatments.
2. The top ranked treatment was chosen from each treatment set. For the

purpose of comparison, we didn't ask domain experts to examine the individual treatments, we simply chose the top one instead.

3. We then imposed the 2 chosen treatments (1 from TAR2, 1 from TAR3) on the model respectively; simulated it again and got another 2 sets of data examples.
4. Step 1-3 were repeated until the stopping point was reached.

1.5.1 Comparison of the Cost-Benefit Distribution

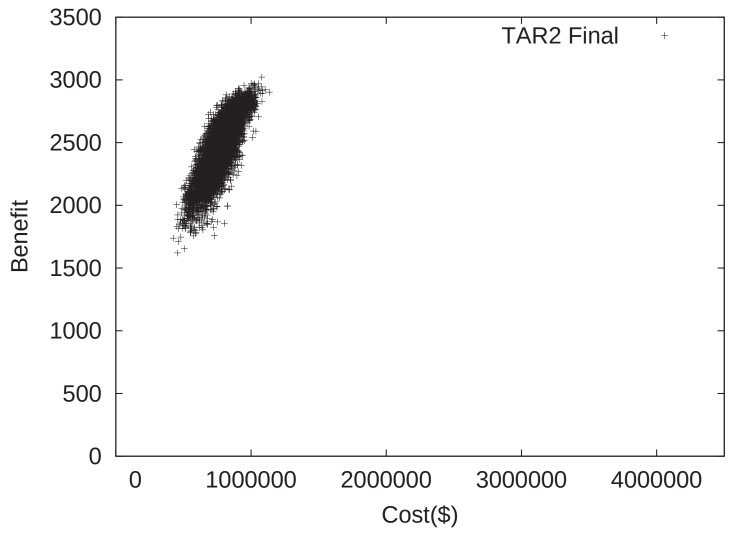

Figure 1.6: The cost-benefit distribution from executing the model of pilot domain when it was constrained after the 5th iteration of TAR2.

Figure 1.6 shows cost-benefit distribution after the 5th iteration of TAR2. Compared to figure 1.5, the variation is relatively small. Most of the data points are grouped at the upper-left corner of the graph, indicating a tight cluster of low-cost high-benefit results. Figure 1.7 is the result from TAR3

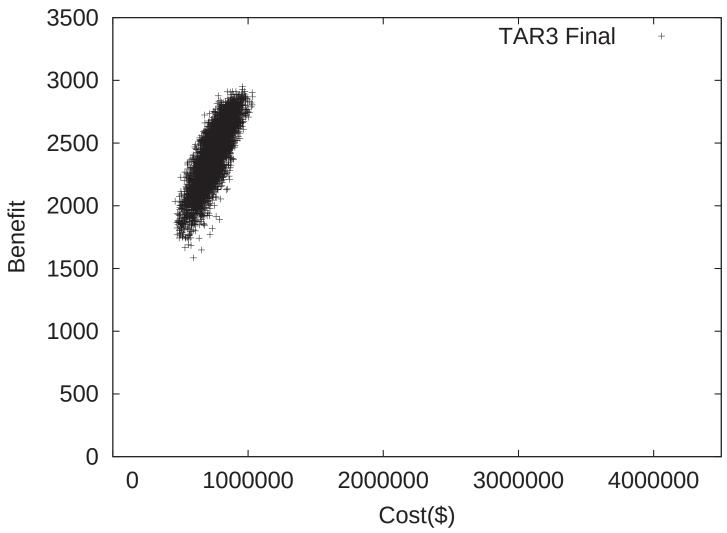

Figure 1.7: The cost-benefit distribution from executing the model of pilot domain when it was constrained after the 4th iteration of TAR3.

experiments following the 4th iteration. The two graphs are visually the same, indicating very similar results.

1.5.2 Comparison of the Best 3 Class Distribution

For a closer comparison, table 1.6 records the best 3 class distribution of each round. The best 3 out of total 16 classes correspond to a region of desired zone that the the domain experts interested. TAR2 reaches the stopping point after 5 rounds, fixing total 19 attributes; TAR3 reaches the stopping point after 4 rounds, fixing total 20 attributes. At the stopping point, both TAR2 and TAR3 achieved a similar class distribution. Further learning didn't offer significant improvement (i.e., the distribution improvement is less than 10%).

TAR2	baseline	run1	run2	run3	run4	run5
size(T)	0	4	4	4	4	3
Class14	3%	33%	68%	22%	7%	2%
Class15	0%	1%	7%	38%	19%	5%
Class16	0%	0%	4%	28%	74%	93%
Total	**3%**	**34%**	**79%**	**88%**	**100%**	**100%**

TAR3	baseline	run1	run2	run3	run4	run5
size(T)	0	6	6	5	4	—
Class14	3%	47%	50%	11%	0%	—
Class15	0%	2%	19%	27%	7%	—
Class16	0%	1%	13%	60%	93%	—
Total	**3%**	**50%**	**82%**	**98%**	**100%**	—

Table 1.6: Comparison of the best 3 class distributions for TAR2 and TAR3 experiments.

1.5.3 Comparison of Each Round

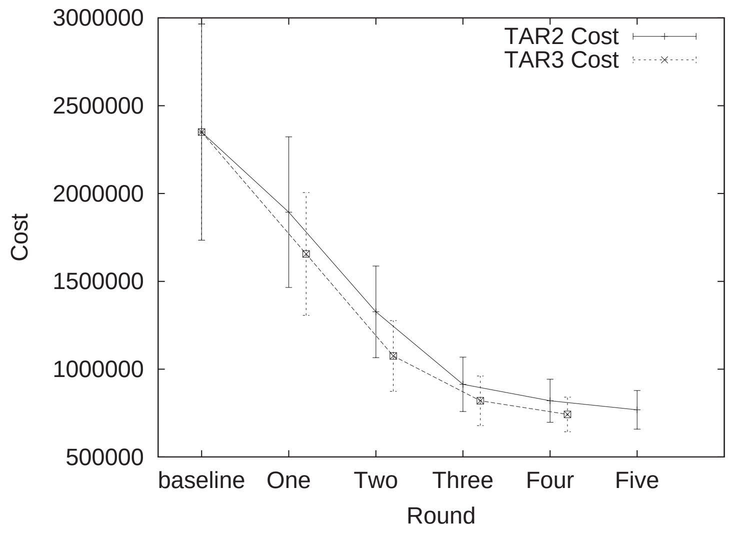

Figure 1.8: The mean and standard deviation of cost at each round.

At the beginning of this experiment, TAR2 and TAR3 started from the

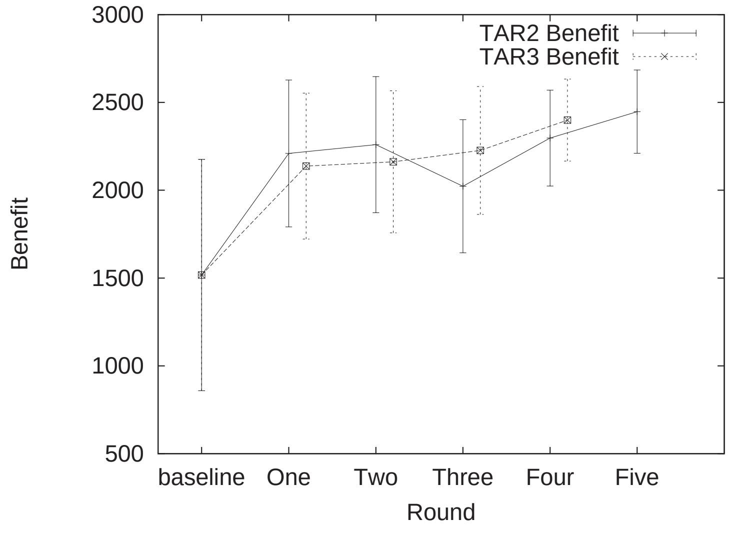

Figure 1.9: The mean and standard deviation of benefit at each round.

same point (i.e. the first baseline data) and came up with different treatments. They later followed their own path toward the final destination. Figure 1.8 compares their performance on each round in terms of the mean and standard deviation of the cost figure. Each round, TAR3 achieved lower cost and smaller deviation making it get to the stopping point one iteration earlier. Figure 1.9 compares the benefit figure. Again, TAR3's deviation is smaller at each round. It is interesting to notice the dip in the TAR2 curve, which indicates a slowing down of the progress. But it eventually catches up in round 4 and round 5.

1.5.4 Comparison of the Final Treatments

TAR2 gave a final treatment of size 19 after 5 iterations, TAR3's final treatment is of size 20 after 4 iterations. Although in each run, TAR2 and TAR3 generated quite different treatments, the combined final treatments are almost the same. Table 1.7 compares the two treatments attribute by attribute, showing that they have 18 attributes in common.

No.	Attribute	TAR2	TAR3	No.	Attribute	TAR2	TAR3
1	[P63=N]	✓	✓	12	[P1310=Y]	✓	✓
2	[P70=N]	✓	✓	13	[P529=N]	✓	✓
3	[P72=N]	✓	✓	14	[P544=N]	✓	✓
4	[P73=Y]	✓	✓	15	[P551=N]	✓	✓
5	[P74=N]	✓	✓	16	[P555=Y]	✓	✓
6	[P126=Y]	✓	✓	17	[P575=N]	✓	
7	[P135=N]	✓	✓	18	[P960=N]		✓
8	[P137=N]	✓	✓	19	[P1047=N]	✓	✓
9	[P145=Y]		✓	20	[P1260=Y]	✓	✓
10	[P154=Y]	✓	✓	21	[P1287=N]	✓	✓
11	[P166=N]	✓	✓	Total		19	20

Table 1.7: Comparison of the final treatments found by TAR2 and TAR3, respectively.

1.5.5 Comparison of Runtimes

The data size we used is $20,000$ examples $\times$ 58 attributes at each round. Table 1.8 compares their runtimes. For reference reasons, column 3 and 5 list the size of best treatment found at that round. The average runtimes of TAR3 is only $\frac{1}{5}$ to $\frac{1}{3}$ TAR2's runtime. TAR3 runs much faster even with larger treatment size.

Round	TAR2(sec)	size(T)	TAR3(sec)	size(T)	TAR3/TAR2
1	1243	4	320	6	25.8%
2	1170	4	348	6	29.7%
3	927	4	235	5	25.3%
4	650	4	126	4	19.4%
5	103	3	—	—	—

Table 1.8: Comparison of the runtimes of each round.

1.6 Conclusion

From the above case study, we have the following observations:

- TAR3's runtime is much shorter than TAR2, average $\frac{1}{5}$ to $\frac{1}{3}$ TAR2's runtime.
- TAR3 usually achieves a better class distribution than TAR2 each run, with a slightly larger treatment.
- TAR2 eventually reaches the same distribution as TAR3 after more runs, possibly with total less attributes fixed (i.e., the size of the final treatment is smaller in TAR2's case).

In summary, by adopting random sampling as well as other strategies, TAR3 has made major improvement in terms of algorithmic efficiency. Comparison experiments on the case study demonstrates TAR3's capability of obtaining stable results more efficiently without degrading its performance.

Specifically, the key idea of random sampling algorithm is that the attribute ranges with higher $confidence1$ value should be picked up with higher probability. This leaves space for us to control the random process: For example, probability can be changed/quantitized by using some functions when computing the CDF value. Also, even though TAR3 achieves near linear runtime, it still requires the entire data set be fitted into the ram. One possible approach could be running TAR3 separately on subsets of the data and doing an ensemble based learning. Another alternative would be to design a online version where examples are read in one at a time, and the $confidence1$ distribution is built dynamically. In this case, we must change the discretization process or even abandon it completely. In the future we are interested in attempting those ideas.

Bibliography

[ACDC$^+$98] C. Abts, B. Clark, S. Devnani-Chulani, E. Horowitz, R. Madachy, D. Reifer, R. Selby, and B. Steece. COCOMO II model definition manual. Technical report, Center for Software Engineering, USC,, 1998. `http://sunset.usc.edu/COCOMOII/cocomox.html#downloads`.

[FM02] M. Feather and T. Menzies. Converging on the optimal attainment of requirements. In *In IEEE Joint Conference On Requirements Engineering ICRE'02 and RE'02, 9-13th September, University of Essen, Germany, 2002.*, 2002. Available from `http://tim.menzies.com/pdf/02re02.pdf`.

[KW01] David Cheung Francis Chin Ke Wang, Yu He. Mining confident rules without support requirement. In *the 10th ACM International Conference on Information and Knowledge Management (CIKM 2001), Atlanta*, 2001.

[MH01] T. Menzies and Y. Hu. Reusing models for requirements engineering. In *Submitted to the first International Workshop on*

Model-based Requirements Engineering, 2001. Available from `http://tim.menzies.com/pdf/01reusere.pdf`.

[MH02] T. Menzies and Y. Hu. Agents in a wild world. In *Book chapter, submitted to Formal Approaches to Agent-Based Systems*, 2002.