El Farol Bar problem using learning automata

by

Harald Unander, Torbjørn Vaaje

Supervisor: Morten Goodwin Olsen

Project report for Distributed Systems in Spring 2007

based on report template version 3.0 (2006)

Agder University College

Faculty of Engineering and Science

Grimstad, 14 May 2007

Status: Final

Keywords: learning automata, El Farol Bar problem

Abstract:

This project wanted to look into if Learning Automata is suitable for solving the El Farol Bar problem. We chose to try out the Tsetlin algorithm for the problem. We also made one algorithm on our own, but we were not able to fully neither develop it nor evaluate it within the scope of this project. As a reference we implemented the El Farol Bar problem as described by Arthur.

The Tsetlin implementation has proved to be accurate, efficient, scaleable and not least, easy to implement and it really outperforms the competition.

It is believed that the Jeannequin (Artur) agent would gain if more effort was put into making good prediction methods. However, this also the main weakness with this agent. A lot of tuning and effort is needed to make it perform for a certain data set.

Version Control

Version[1]	***Status***[2]	***Date***[3]	***Change***[4]	***Author***[5]
1.0	Final	2007-05-14	Created	Torbjørn Vaaje Harald Unander

1 **Version** indicates the version number starting at 0.1 for the first draft and 1.0 for the first review version.

2 **Status** is DRAFT, REVIEW or FINAL

3 **Date** is given in ISO format: yyyy-mm-dd

4 **Change** describes the changes carried out since the previous version

5 **Author** is the one who did the change

1 Table of Contents

2 Introduction

2.1 Problem Statement

El Farol Bar problem

Ref: http://en.wikipedia.org/wiki/El_Farol_Bar_problem

There is a particular, finite population of people. On Thursday night, all of these people want to go to the El Farol Bar. However, the El Farol is quite small, and it's no fun to go there if it's too crowded. So much so, in fact, that the following rules are in place:

If less than 60% of the population goes to the bar, they'll all have a better time than if they stayed at home.

If more than 60% of the population goes to the bar, they'll all have a worse time than if they stayed at home.

Unfortunately, it is necessary for everyone to decide at the same time whether they will go to the bar or not. They cannot wait and see how many others go before deciding to go themselves.

The assignment includes using machine learning techniques to solve the el Farol bar problem.

2.2 Report outline

Chapter 3 gives a detailed description on the problem and the requirement for this project.

Chapter 4 gives an overview over what have been done in this field previously and what literature that we base our solution on.

Chapter 5 describes how we solved the task.

Chapter 6 describes the experiments that we did

Chapter 7 discusses the pros and cons of different solutions.

Chapter 8 sums up and give some conclusions.

3 Problem description

We will look at possible solutions to the El Farol Bar problem by using machine learning techniques.

- Implement and repeat the experiment as described in [3].
- Solve the El Farol Bar problem by using a learning automata of Tsetlin type [4].
- Compare the two methods

We will collect information on:

- How well the different scenarios meets the overall goal (i.e. close to 60 people at the bar each night.)
- How well the distribution is between all the agents (are the distribution equal or meeting the individual needs)
- Which (set of) algorithm works best?

Challenges:

- For learning automata find a way to satisfy each agent desire to be present at the bar. The Tsetlin automat will without any modification soon adjust so that some will always go to the bar and others will always stay home.
  This problem is also present for the Jeannnequin solution (it does neither support individual wishes for how frequent they want to go to the bar.)

Implementation:

- We will use Java as implementation language
- Java interfaces will be used to define agents and prediction methods
- The same framework will be used for both implementations
- An experiment may include agents of both types (Tsetlin & Jeannequin)
- Test results will be written to data files and presented in MS Excel where applicable.

Further work:

- Intruduce external events or noise

4 Background

4.1 Literature Review

The El Farol Bar problem was first described by Brian Arthur in the paper "Inductive Reasoning and Bounded Rationality (The El Farol Problem)", Published in *Amer. Econ. Review (Papers and Proceedings)*, **84**, 406, 1994. (Reference [2])

N people decide independently each week whether to go to a bar that offers entertainment on a certain night. Space is limited, and the evening is enjoyable if things are not too crowded—specifically, if fewer than 60 are present. There is no way to tell the numbers coming for sure in advance, therefore a person or agent: *goes*—deems it worth going—if he expects fewer than 60 to show up, or *stays home* if he expects more than 60 to go. The only information available is the numbers who came in past weeks.

"El Farol Bar Problem" Nicolas Jeannequin "MSc in Mathematical Modelling and Scientific Computing Mathematical Modelling II" (Reference [3])

This paper is an experiment on the El Farol Bar Problem as described by Brian Arthur. It is more in detail on the implementation and also more extensive on the result. That is more statistic is provided.

"Reinforcement Learning: A Survey" Leslie Pack Kaelbling & Michael L. Littman (Reference [4]) This web page gives a short introduction to the Tsetlin automaton.

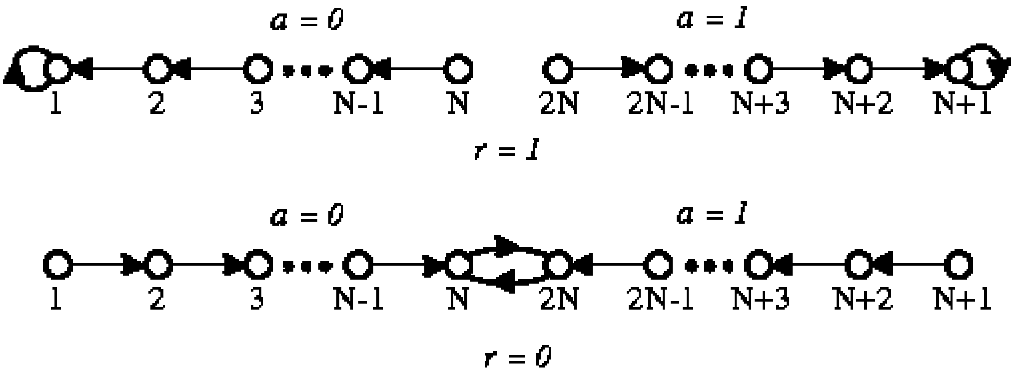

Figure 1 A Tsetlin Automaton

A Tsetlin automaton with 2*N* states. The top row shows the state transitions that are made when the previous action resulted in a reward of 1; the bottom row shows transitions after a reward of 0. In states in the left half of the figure, action 0 is taken; in those on the right, action 1 is taken.

5 Solution

5.1 Design Specification and Implementation

The learning agents are implemented in Java and we used Eclipse as development tool.

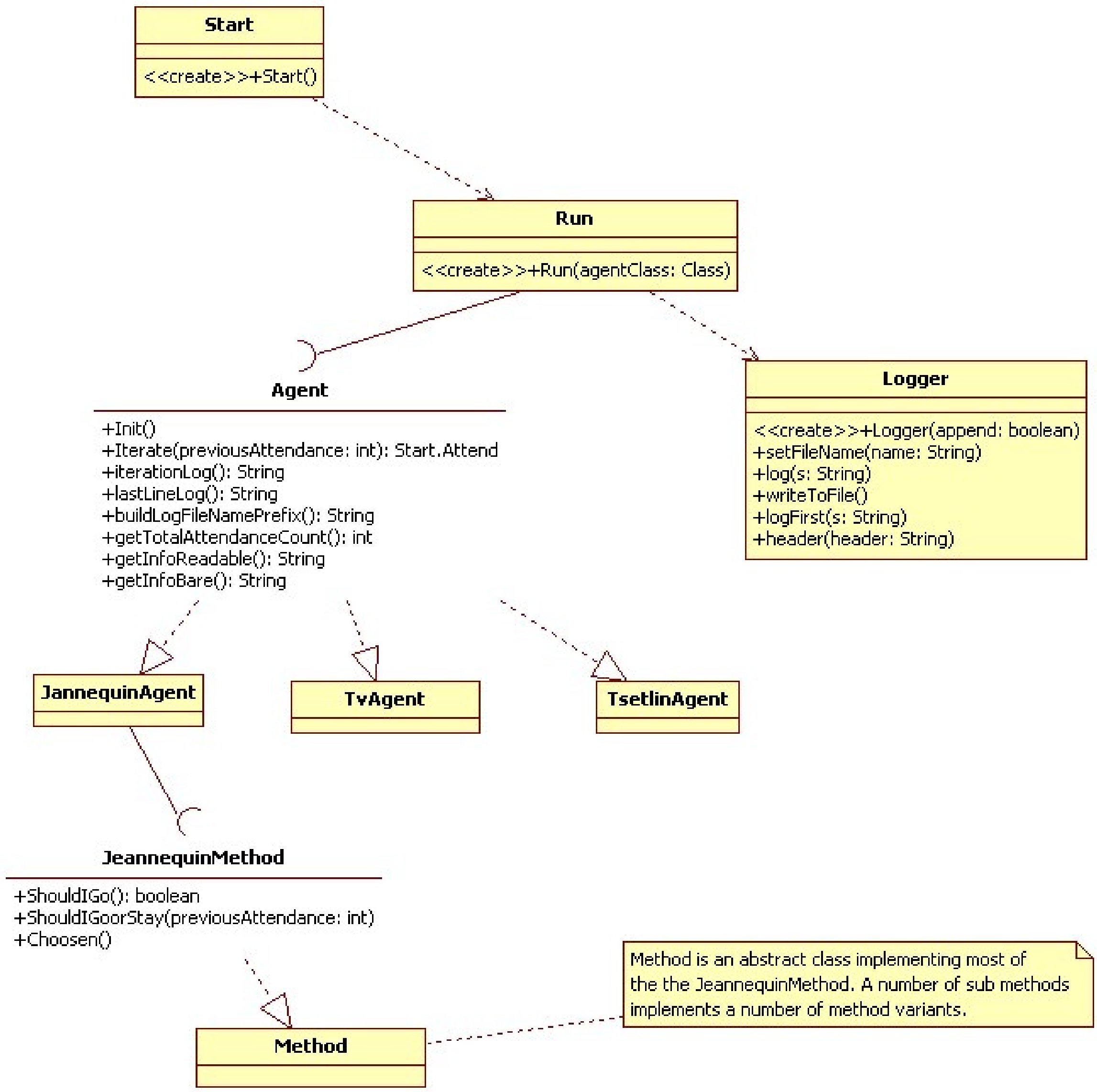

Figure 2 El Farol Bar - Java Class Diagram

The Start class is used to initiate a run for a certain agents. Parameters in the Start.java method define log location, agent count, number of iterations and goal (bar comfort level).

```
String LOG_DIR ="C:\\ElFarolBar\\logs\\";
int AGENT_COUNT = 1000; // The potential number of guests
int ROUNDS = 20000; // The total number of bar nights
int GOAL = 60; // The bar's comfort level in number of visitors
```

Agent to run is selected by the start method:

```
public Start() {
        new Run(TsetlinAgent.class);
        new Run(JeannequinAgent.class);
        new Run(TvAgent.class); }
```

The agent implementations define tunable parameters for the specific agent. The result of a run is stored in a file with a file name constructed from the run's parameters. A summary log per agent keeps one line of information per run. New agents can easily be added, and they will be easily included in the runs, and logging and data collection will be for free.

5.2 The Tsetlin agent

This is a standard implementation of the Tsetlin automaton as described in ref[4].

The Tsetlin agent can be tuned on the following parameters:

```
int N = 8; // Number of Tsetlin automata states/2
int MARGIN = 1; // Don't penalize if distance to GOAL is less the MARGIN
float PENALIZE_CHANCE = 0.7f;
float REWARD_CHANCE = 0.7f;
```

The main logic is as follows:

- If the previous attendance (or no attendance) is contributing to getting closer to the goal, there is a reward with a REWARD_CHANCE. Otherwise, there is penalization with a PENALIZE_CHANCE. This is really the "brain" of the tsetlin automaton.
- Depending on reward or penalization, the agents state is updated as described in chapter 4.1. If the state is less than N, attend, otherwise don't attend.

5.3 The TV agent

TV agent got its name from the well known computer scientist Torbjørn Vaaje of Austre Moland.

This agent keeps his/hers strategy on go or stay home for some iterations. After a number of iterations the Agent reconsider his/hers strategy depending on how well it contribute to the overall goal. The closer the group is to the overall goal, the less likely is it that the Agent changes strategy.

We have implemented a first version of this agent, but there are a lot of space for refinement of the method and implementation of this agent.

5.4 The Jeannequin (Artur) agent

An agent inspired by the papers of Artur and Jeannnequin.

This is implemented as described in reference [2] and [3]. In [2] Arthur writes that he had implemented "*several dozen focal predictors replicated many times".* Due to lack of time we were not able to implement that many predictors. This is a weakness with our implementation.

6 Experiments

First a little guidance to reading the results:

AgentCount	Rounds	Goal	Mean	SD	95%L	95%U	Zero	Full	Lower	Higher	N	Margin	Penalize	Reward

- Agent count: the potential customer number
- Rounds: the number of bar nights
- Goal: the bar's comfort level
- Mean: the average attendance count over number of rounds
- SD: Standard deviations for attendance over number of rounds
- 95%L: mean – 2*SD, the lower limit of the 95% confidence interval
- 95%H: mean + 2*SD, the upper limit of the 95% confidence interval
- Zero: the number of agents never showing up
- Full: the number agents always present
- Lower: the lowest participation over the rounds
- Upper: the higest participation over the rounds
- N: for Tsetlin agent, N
- Penalize, reward: for Tsetlin agent, peanlize and reward chances

6.1 Tsetlin Agent

For the Tsetlin agent, the following data sets have been tried:

- GOAL = 60
- AGENT_COUNT = 100/1000/10000
- ROUNDS = 1000/10000/100000
- N = 4/8/12/16/20
- MARGIN = 1
- PENALIZE_CHANCE = REWARD_CHANCE = 0.4/0.5/0.6/0.7/0.8/0.9/1.0

A summary of the results from all the Tsetlin runs follows:

TsetlinResult.xls

The result from a selected Tsetlin run is given here. It lists every agents state and attendance (X) or not (-) for every round. (Such a log is generated for every run):

TsetlinAgent N4 0.7 0.7 R1000 A100 G60 run.log

For a number of runs we seem to have perfect results with SD equal to zero:

AgentCount	Rounds	Goal	Mean	SD	95%L	95%U	Zero	Full	Lower	Higher	N	Margin	Penalize	Reward
10000	10000	60	61	0	61	61	4660	5	61	61	20	1	0.8	0.8

Table 1 The Tsetlin agent result at first sight

The problem with this result is that around half of the customers never attend the pub.

Limiting the results to those giving fair attendance give the following results:

AgentCount	Rounds	Goal	Mean	SD	95%L	95%U	Zero	Full	Lower	Higher	N	Margin	Penalize	Reward
100	1000	60	59	1,5	56	62	0	3	47	66	4	1	0.8	0.8
100	1000	60	59	1,6	56	62	1	0	55	64	8	1	0.6	0.6
100	1000	60	59	2,2	55	63	0	0	52	66	4	1	0.7	0.7
100	1000	60	59	2,5	54	64	0	0	50	68	4	1	0.7	0.7
100	1000	60	59	2,8	53	65	0	0	50	68	4	1	0.6	0.6

Table 2 Best results with agent count = 100, zero&full < 10%

AgentCount	Rounds	Goal	Mean	SD	95%L	95%U	Zero	Full	Lower	Higher	N	Margin	Penalize	Reward
1000	10000	60	60	1,7	57	63	4	0	51	74	8	1	0.7	0.7
1000	10000	60	61	2,4	56	66	0	0	53	74	12	1	0.6	0.6
1000	10000	60	61	8,1	45	77	0	0	22	147	4	1	0.9	0.9

Table 3 Best results with agent count = 1000, zero&full < 10%

AgentCount	Rounds	Goal	Mean	SD	95%L	95%U	Zero	Full	Lower	Higher	N	Margin	Penalize	Reward
10000	10000	60	63	3,6	56	70	613	0	50	79	16	1	0.6	0.6
10000	100000	60	61	1,6	58	64	248	0	54	73	20	1	0.6	0.6
10000	100000	60	63	3,6	56	70	0	0	50	82	16	1	0.6	0.6

Table 4 Best results with agent count = 10000, zero&full < 10%

Let's take a closer look at runs with penalize and reward chances of 0.7:

No	AgentCount	Rounds	Goal	Mean	SD	95%L	95%U	Zero	Full	Lower	Higher	N
1	100	1000	60	59	2,4	54	64	0	0	51	69	4
2	1000	1000	60	72	11,5	49	95	0	0	44	112	4
3	1000	10000	60	72	11,5	49	95	0	0	41	113	4
4	10000	10000	60	326	17,9	290	362	0	0	266	396	4
5	10000	100000	60	326	17,8	290	362	0	0	254	409	4
6	100	1000	60	59	1,2	57	61	15	6	58	62	8
7	1000	1000	60	60	1,5	57	63	223	0	56	65	8
8	1000	10000	60	60	1,6	57	63	2	0	52	76	8
9	10000	10000	60	68	9,1	50	86	4	0	44	110	8
10	10000	100000	60	68	9,1	50	86	0	0	40	114	8
11	100	1000	60	60	1,0	58	62	31	3	60	61	10
12	1000	1000	60	60	1,0	58	62	373	4	58	64	10
13	1000	10000	60	60	1,0	58	62	229	1	57	64	10
14	10000	10000	60	62	4,5	53	71	338	0	46	92	10
15	10000	100000	60	62	4,2	54	70	0	0	41	99	10
16	100	1000	60	59	0,4	58	60	38	0	58	60	12
17	1000	1000	60	60	1,0	58	62	429	4	60	62	12
18	1000	10000	60	60	0,9	58	62	381	7	57	63	12
19	10000	10000	60	60	1,3	57	63	3850	0	55	71	12
20	10000	100000	60	60	1,2	58	62	1597	0	56	67	12
21	100	1000	60	60	0,0	60	60	37	2	60	60	16
22	1000	1000	60	60	1,0	58	62	432	8	60	61	16
23	1000	10000	60	60	0,8	58	62	477	6	60	62	16
24	10000	10000	60	60	1,0	58	62	4426	2	60	62	16
25	10000	100000	60	60	1,0	58	62	4380	1	58	63	16
26	100	1000	60	61	0,0	61	61	33	1	61	61	20
27	1000	1000	60	61	0,0	61	61	449	2	61	61	20
28	1000	10000	60	59	0,0	59	59	438	5	59	59	20
29	10000	10000	60	59	0,0	59	59	4550	6	59	59	20
30	10000	100000	60	60	1,0	58	62	4526	7	59	62	20

Table 5 A closer look at Tsetlin change 0.7 results

Observations:

1. For all runs we reach an average level close to the goal, except for runs 4 and 5.
2. Low N values (4/8) in combination with large agent counts give negative results.
3. The runs marked green have more than 95% of the samples within +/-5% of mean so they can be considered positive runs.
4. For some of the positive runs there are a huge number of agents (~40%) with no participation.
5. Run 19-20 is interesting, the zero attendance count going down. A further decrease is expected if the number of rounds is increased.
6. Run pairs 12-13, 17-18, 19-20, 22-23, 23-24, 27-28 and 29-30 shows that increasing number of rounds make no difference.
7. Looking closer into the results of run 30 shows that the situation of what agents are active is not changing at all.

6.2 The Jeannequin (Artur) agent

For the Jeannequin agent, the following data sets have been tried:

- GOAL = 60
- AGENT_COUNT = 100/1000/10000
- ROUNDS = 1000/10000
- Methods = 2/4/8

No	AgentCount	Rounds	Goal	Mean	SD	95%L	95%U	Zero	Full	Lower	Upper	Number of methods
1	**100**	**1000**	**60**	**59**	**7**	**45**	**73**	**0**	**0**	**43**	**75**	**2**
2	**100**	**1000**	**60**	**59**	**7,2**	**45**	**73**	**0**	**0**	**38**	**82**	**4**
3	**100**	**1000**	**60**	**60**	**8,8**	**42**	**78**	**0**	**0**	**11**	**80**	**8**
4	1000	1000	60	78	43,8	-10	166	37	0	4	140	2
5	**1000**	**1000**	**60**	**60**	**6,9**	**46**	**74**	**22**	**0**	**42**	**82**	**4**
6	1000	1000	60	85	59,3	-34	204	0	0	24	377	8
7	1000	10000	60	85	29,9	25	145	0	0	3	272	2
8	**1000**	**10000**	**60**	**60**	**9,1**	**42**	**78**	**0**	**0**	**25**	**284**	**4**
9	1000	10000	60	94	63,9	-34	222	0	0	25	322	8
10	10000	10000	60	236	320,9	-406	878	0	0	38	2577	2
11	10000	10000	60	86	117,5	-149	321	0	0	21	1477	4
12	**10000**	**10000**	**60**	**60**	**7,9**	**44**	**76**	**0**	**0**	**31**	**94**	**8**

Table 6 Results of Jeannequin agent

Observations:

- The best result is with few agents.
- This method does not scale as good as the Tsetlin
- The number of methods to choose from is probably one important reason for the unstable result.

6.3 The TV Agent

For the TV agent, the following data sets have been tried:

- GOAL = 60
- AGENT_COUNT = 100/1000/10000
- ROUNDS = 1000/10000

No	AgentCount	Rounds	Goal	Mean	SD	95%L	95%U	Zero	Full	Lower	Upper
1	**100**	**1000**	**60**	**60**	**0**	**60**	**60**	**0**	**60**	**60**	**60**
2	**100**	**1000**	**60**	**60**	**0**	**60**	**60**	**0**	**59**	**60**	**60**
3	**100**	**1000**	**60**	**60**	**0**	**60**	**60**	**0**	**60**	**60**	**60**
4	**1000**	**1000**	**60**	**60**	**0**	**60**	**60**	**0**	**0**	**60**	**60**
5	**1000**	**1000**	**60**	**60**	**0**	**60**	**60**	**0**	**0**	**60**	**60**
6	**1000**	**1000**	**60**	**60**	**0**	**60**	**60**	**0**	**0**	**60**	**60**
7	**1000**	**10000**	**60**	**60**	**0**	**60**	**60**	**0**	**0**	**60**	**60**
8	**1000**	**10000**	**60**	**60**	**0**	**60**	**60**	**0**	**0**	**60**	**60**
9	**1000**	**10000**	**60**	**60**	**0**	**60**	**60**	**0**	**0**	**60**	**60**
10	**10000**	**10000**	**60**	**60**	**0**	**60**	**60**	**0**	**0**	**60**	**60**
11	**10000**	**10000**	**60**	**60**	**0**	**60**	**60**	**0**	**0**	**60**	**60**
12	**10000**	**10000**	**60**	**60**	**0**	**60**	**60**	**0**	**0**	**60**	**60**

Table 7 Results of TV agent

Seems like a good method, but there is no random, so the agent either stay or go as soon as this method has stabilize.

7 Discussion

7.1 Tsetlin

The Tsetlin agent turned out to be easy to implement and it needs very little tuning when changing data sets. Penalize chances and reward chances bewteen 0.6 and 0.7 were found to give the best results. Its efficiency is also quite good; it uses around 30 seconds for a run with the 10000/10000 data set.

When it comes to the values for N the conclusion is that N must grow with the number of agents. This confirms earlier studies. Good values for the different agent counts seem to be:

- Agent count 100, N=4 (ref Table 2)
- Agent count 1000, N=8 or N=12 (ref Table 3)
- Agent count 10000, N=16 (ref Table 4)

In a number of runs we see that the standard deviation quickly settles to 0. This means the distribution becomes static. This situation is more frequent with higher N values, and when the N value is high in relation to the agent count. In this case, increasing the number of runs does not improve the attendance distribution (page 11, item 6).

We also have the case with a great agent count and a high N value (page 11, item 5), where a higher number of rounds reduces the the zero attendance count.

7.2 Jeannequin (Artur)

The Jeannnequin solution requires a selection method and a number of methods to choose between. We believe that we have made an efficient and easy to implement selection method. Still we had a few fault before we got it to work properly. Then we had to implement a number of methods for the agents to choose between. An advantage here is that it is not a problem if a method turns out to be wrong or useless. The only consequence is that it is hardly ever chosen and then only during the first iterations. But we need a number of useful methods. It seems that in our experiment that the numbers of available methods are too small and thus the number of useful methods. It is much more demanding when it comes to CPU resouces when compared to the Tsetlin agent; it uses 7 minutes with the 10000/10000 data set, so the biggest data set was excluded from the tests.

7.3 Tv

The TV Agent is a very simple agent, serving as an example for anyone who wants to implement their own. Not much effort has been put into this agent.

7.4 Comparison

The behaviour of the Tv agent is very similar to the Tsetlin with chances set to 1.0, giving a static agent attendance distribution. There is not so much more to say about this agent. We are not so interested in the static distribution case. This can easily be achieved in numerous ways.

We will concentrate on comparing Tsetlin with Jeannequin.

Efficiency

For the 10000/10000 the computation time varies from Tsetlin 30 seconds to Jeannequin of 7 minutes.

Reaching the attendance goal

When it comes to reaching the attendance goal Tsetlin is better than Jeannequin with a significantly lower standard deviation and a smaller confidence interval.

Attendance distribution

Figure 3 compares the agent types' distribution for selected data sets. If we look at the participation for some agent for some time with the configuration 100/1000 we got this (red is go and blue is stay; X-axis is agent and Y-axis is iterations with the first iteration on top.

The figure clearly shows the different characteristics, with -

- the Tsetlin distribution slowly changing attendance state reflecting the state behaviour and randomness
- the Jannequin reflecting the selected methods with repeating patterns
- the Tv agent rapidly reaching the static distribution

Implementation

The Tsetlin is a standard implemenation. Only one method needs implementation, and only the N value must be tuned with changing data sets. The Jeannequin agent needs a number of methods to work well, and a wrong selection of and methods types give bad results.

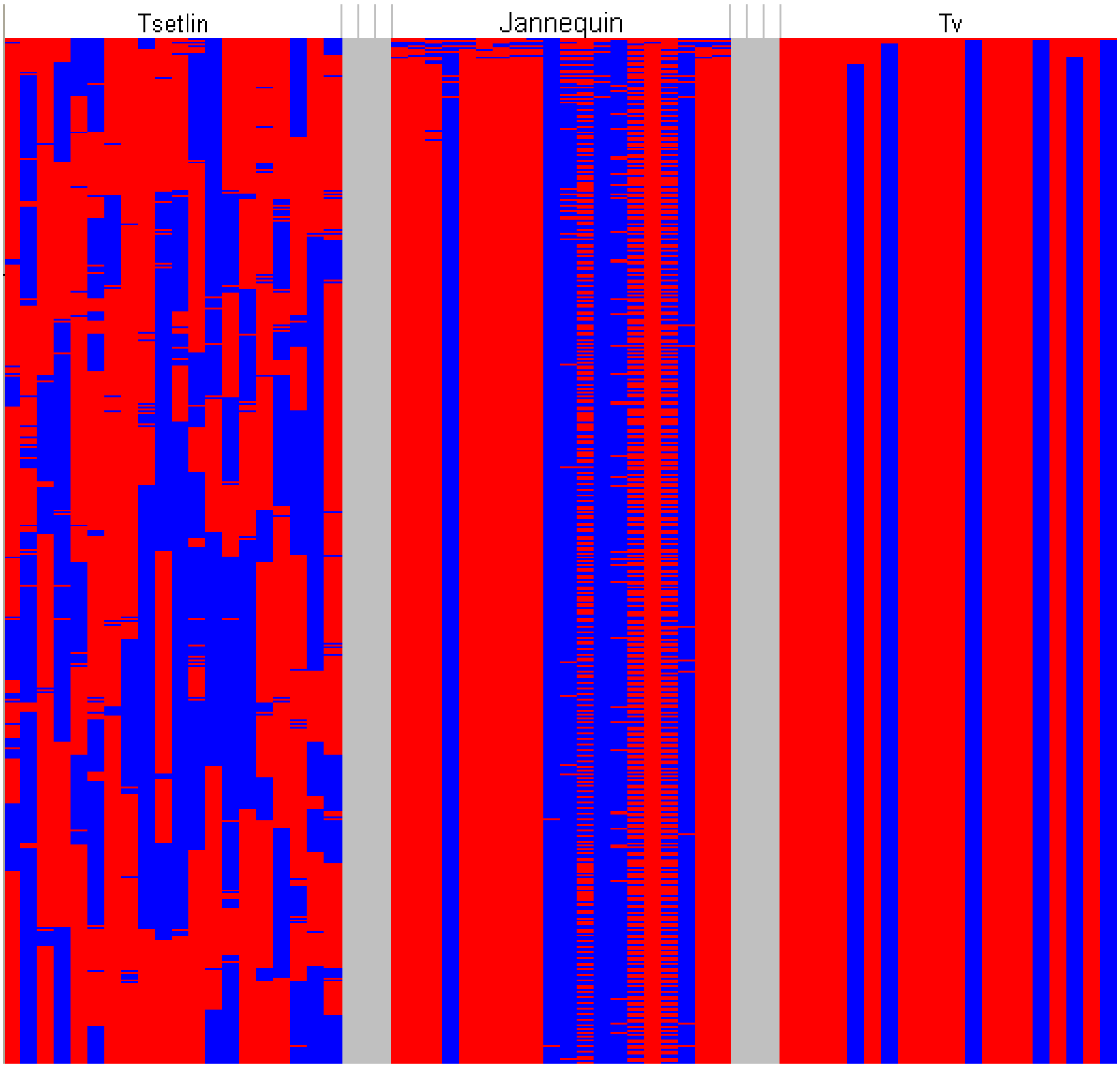

Figure 3 Attendance distribution overview

8 Conclusion

We have implemented the Tsetlin and Jeannequin agents, and even an additional agent, and they all works, reaching the desired attendance goal for the El Farol bar. Experiments show that the Tsetlin agent in all respect behaves best. It also requires less tuning of method implementations.

The Tsetlin implementation has proved to be accurate, efficient, scaleable and not least, easy to implement and it really outperforms the competition.

It is believed that the Jeannequin agent would gain if more effort was put into making good prediction methods. This is however also the main weakness with this agent. A lot of tuning and effort is needed to make it perform for a certain data set.

There has not been time to implement mechanisms to take into account individual attendance goals of the bar customers, and we have not reached to add noise introduction, so this is left for further study.

The implementation in Java and Eclipse has given a good framework for agent implementation, and quite some effort has been put into the logging part so that results can easily be extracted.

By using the agent interface, new agents can easily be added, taking advantage of the provided execution and logging framework.

Appendices

Appendix 1 Glossary & Abbreviations

Agent count	The potential customer number
Rounds	The number of bar nights
Goal	The bar's comfort level
Run	A season in the bar

Appendix 2 List of Tables

Appendix 3 List of Figures

Appendix 4 References

HiA, Bruk av kilder i skriftlige arbeider ved Høgskolen i Agder, August 2006, http://www.hia.no/stud/eksam/kilder.htm

[1] Peter J. Denning, Douglas E. Comer, David Gries, Michael C. Mulder, Allen Tucker, A. Joe Turner, and Paul R. Young, *Computing as a Discipline, Communications of the ACM, Volume 32, Number 1, 1989*

[2] W.B. Arthur, *Inductive reasoning and bounded rationality, America Economics Association Review, 1994*
http://www.santafe.edu/~wbarthur/Papers/El_Farol.html

[3] Nicolas Jeannequin, *El Farol Bar Problem*
http://web.comlab.ox.ac.uk/oucl/work/nicolas.jeannequin/Site/Research_files/ELFAROL-1.pdf

[4] Leslie Pack Kaelbling, Michael L. Littman**,** Reinforcement Learning: A Survey,
http://www.cs.cmu.edu/afs/cs/project/jair/pub/volume4/kaelbling96a-html/node10.html