Vehicular Network Analysis Using Probabilistic Model Checking

Tommaso Peletta, Jonathan Lo
{tommaso.peletta@etu.unige.ch,
jonathan.lo@etu.unige.ch}

May 26, 2021

Contents

1 Introduction

The goal of this project is to use a probabilistic model checker to verify some properties of a model. More precisely, we use the model checker to analyze road traffic properties in order and to observe if these tools (model checkers) are well suited in this context. We mainly use two probabilistic model checkers: "PRISM" and "GreatSPN".

In the first part of this project, we analyze properties of situations of "Roundabout insertion" using PRISM. We try to observe the collision probability of such a situation in order to avoid accidents if possible. In the second part of the project, we analyze the topology of a small portion of roads (small region of a town) using GreatSPN.

2 Model checker

A Probabilistic Model Checker (PMC) is a tool used to automatically analyze complex systems. It allow to verify properties by enumerating all the reachable states of the system. A PMC has the capability to answers complex questions like "What is the probability that we encounter a specific event ?". For example, a PMC we can detect deadlocks or states that are never reachable in any circumstances. The three main steps of a PMC are :

1. Specify the properties we want to analyze
2. Construct the formal model of the system
3. Launch the verifier to validate the properties

2.1 PRISM

PRISM is a probabilistic model checker that provides formal modelling and analysis capabilities for systems with probabilistic, non-deterministic and real-time characteristics, through support for verification of probabilistic times automata [4].

PRISM provides a programming language and model checking tool for describing and analysing discrete, continuous-time Markov chains and Markov decision

processes. The software has been widely used for different applications such as Bluetooth network performance analysis [2] and behavioural predictions for biological signalling pathways [3].

The tool is provided with complete and up-to-date documentation, making it easy to install and use.

2.2 GreatSPN

GRaphical Editor and Analyzer for Timed and Stochastic Petri Nets (GreatSPN [1]) is a software package for modeling, validation and performance evaluation of distributed systems using Generalized Stochastic Petri Nets and their extensions. The tool provides a friendly user interface to experiment with timed-based Petri nets modelling techniques and it implements efficient analysis algorithms to allow its use on rather complex application.

GreatSPN is composed of multiple separate programs that cooperate in order to provide several modelling and analysis tools. More over, it allows to run different analysis modules on different machines in a distributed computing environment. All modules are written in C programming language to guarantee portability and efficiency.

The software main features includes:

- A graphical interface allowing to edit Petri Nets and to run analysis without the need of learning a specific programming language.
- Structural properties for nets with priorities and inhibitor arcs.
- Linear programming performance bounds for timed P/T nets.
- Reachability graph generation and analysis program.
- Markovian solvers for steady-state.
- An interactive event-driven simulation framework.
- Colored nets extension.
- Nets composition through algebra

GreatSPN provides a lot of functionalities allowing complete analysis of various models. On the other hand, the given documentation of the software is not

up-to-date making it hard to use for specific and complex analysis even if its user interface looks friendly and simple at first sight.

3 Autonomous Vehicle Insertion Into A Roundabout

3.1 Problem

In this part of the project, we try to analyze the insertion of vehicles in a single-lane roundabout. There are vehicles circulating inside and outside of the roundabout. The model start with a random configuration of the vehicules' positions to ensure non-determinism. Our idea is to model the vehicles and the road using "Stochastic Cellular Automata" and "Nagel-Schreckenberg model".

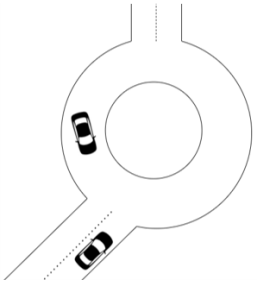

Figure 1: Roundabout

3.2 Properties

The properties that we want to verify are :

- Determine the probability of a collision (with random starting configuration)
- Determine the quantity of braking (mean) for a successful insertion
- Determine the time (mean) needed for a successful insertion

3.3 Cellular Automata

A "cellular automata" consists of a grid of cells. Each cell has a state that will change with time. The state of a cell at time t+1 depends on the state of this cell and all the cells around at time t. At each iteration of time, some rules are applied to each cell at the same time. The new generation of cells depends entirely on the preceding generation [5].

3.4 Our roundabout model

For simplicity, we unfold our roundabout to a linear road. Each square represents a portion of the road and the vehicles are represented by colored dots. More precisely each cell represents a portion of road of length 1.0 [m].

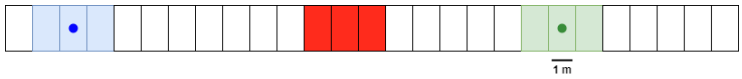

Figure 2: Unfolded roundabout

- Red square : intersection zone (where the insertion happens)
- Blue dot : a vehicle inside the roundabout
- Green dot : a vehicle wanting to insert in the roundabout
- Blue squares : represent the size of the blue vehicle
- Green squares : represent the size of the green vehicle

3.5 Nagel-Schreckenberg model

The Nagel-Schreckenberg model is based on cellular automata and is used to simulate road traffic [6]. In this model, the road are represented by cells. Each cell can contain one or zero vehicle and each vehicle has a speed. The specificity of this model is that time and space are discrete. At each timestep, 4 steps are applied to each car at the same time :

1. Acceleration : speed is increased by one unit if not already at maximum speed
   $v_{i+1} \rightarrow min(v_i + 1, v_{max})$
2. Braking : we verify the distance available to avoid collision
   $v_{i+1} \rightarrow min(v - i, headway)$
3. Randomization : we reduce the speed by one unit with probability p
   $v_{i+1} \rightarrow (v_i - 1)$
4. Motion : we move the vehicle at their new location (number of cells to move is equivalent to speed)
   $x_{i+1} = x_i + v_i$

3.6 Nagel-Schreckenberg modified

For our our specific model, we have adapted the Nagel-Schreckenberg model. We have the following steps for the vehicle inside the roundabout :

1. Acceleration : $v_{i+1} \rightarrow min(v_i + 1, v_{max})$ with probability p_1
2. Braking : $v_{i+1} \rightarrow min(v - i - 1, headway)$ with probability p_2
3. Motion : $x_{i+1} = x_i + v_i$

In this case we have a probabilistic behaviour of the vehicle driven by a human defined by the probabilities p_1 and p_2.

We have the following steps for the vehicle inserting in the roundabout :

1. Acceleration : $v_{i+1} \rightarrow min(v_i + 1, v_{max})$
2. Braking : $v_{i+1} \rightarrow$ if $|d_A/v_A - d_H/v_H| < \epsilon$ then brake
3. Motion : $x_{i+1} = x_i + v_i$

In this case, since the vehicle is automatic, the behaviour is deterministic and depends only on the behaviour of the vehicle inside the roundabout.

3.7 Problem encountered

We have observed that the problem of "Roundabout insertion" seems to be not very well suited for a probabilistic model check using "PRISM". It happens that results could be obtained analytically without using "PRISM". To give our main tool more usefulness we had to add some probabilities in our system. However these probabilities lead us to an avoidable explosion of cases. Adding these probabilities complexifies our system without providing significant results. We also encountered problems as we modeled our system. There was a question of granularity. Indeed we had problems to determine if the model has to be low level (physic-based model) or high level (behaviour-based model). The level abstraction was unclear and seems a bit hard to decide.

For all these reasons, we tried to find another problem more suited for a probabilistic model checker. The new problem found is described in the second part of our report.

4 Vehicular Traffic and Road Topology Analysis

Nowadays, one of the main challenges faced in large metropolitan ares is traffic congestion. To address this problem, many studies have been done on the analysis of vehicular flow and traffic control. In this section we propose an analysis method for evaluating road typology efficiency based on a probabilistic model checker.

4.1 Model

Our model used for the analysis of the vehicular flow consist in:

- A predetermined section of a town road-map.
- Every direction of travel of each road is represented by a given state.
- Vehicles have the same probability of choosing one direction rather than the other when arriving at an intersection.
- Roads may have different travel times depending on the length of the road itself and its characteristics.

- Vehicles exiting the model are re-inserted from a given model entry roads. This means that the number of vehicles is constant during the execution of the simulation. Later in the paper we will discuss different techniques of re-insertion.

More precisely, we thought about two different formal model to represent our problem, the first is a Markov-chain based model and the second a Stochastic Petri Nets based model.

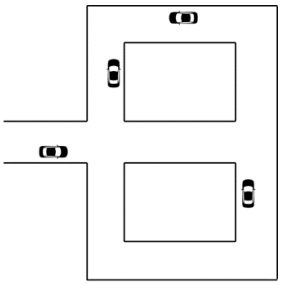

Figure 3: Model

4.1.1 Markovian Model

Markov chain can be used to describe our model. More precisely each direction of travel of each road will be represented by a given state in the Markov chain. Intersections in the road are represented with edges from one state to an other. Since a vehicle has the same probability of choosing one direction rather than the other, the weight of each edge will be equal to one divided by the number of edges exiting by the same node. Each vehicle represent an instance randomly walking in the Markov chain. At each step of time the vehicle will pass from one state to another representing the movement of the vehicle in the road map (Fig. 4).

In order to represent the different time travel of each road we introduce two different possible method. The first one is based on a cost-synchronisation of all instances running on the model. More precisely, each edge will be characterized by a cost representing the average travel time of this road. In order to analyse the vehicle traffic we will study the average number of vehicle on a given state for each cost value (Fig. 5). The second approach is based on a

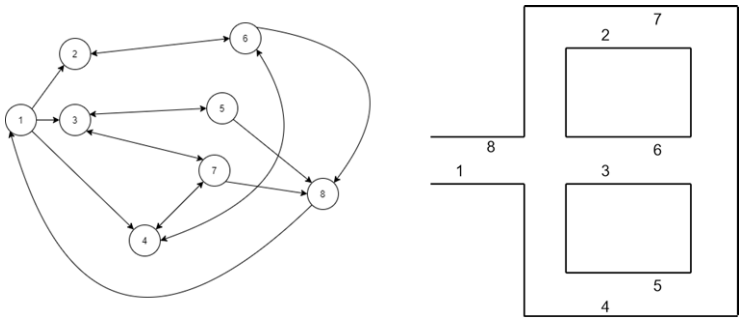

Figure 4: Markov model

step-synchronisation; the idea is to add multiple states for each state depending on the average travel time of the corresponding road. For instance if a road takes on average 3 time step to be travelled, then int he Markov chain this road will be represented by 3 different states. The first representing the entry point of the road, the second (reachable from the first with probability 1) the middle part of the road, and the third (reachable from the first with probability 2) the exit point of the road. In this case we will compute the average number of vehicle on a road during the same time step by summing the average number of vehicle on each sub-section of a road (Fig. 6).

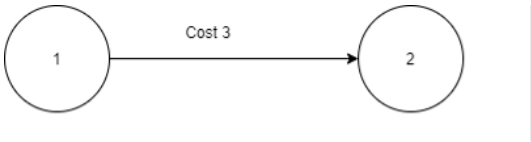

Figure 5: Reward synchronization

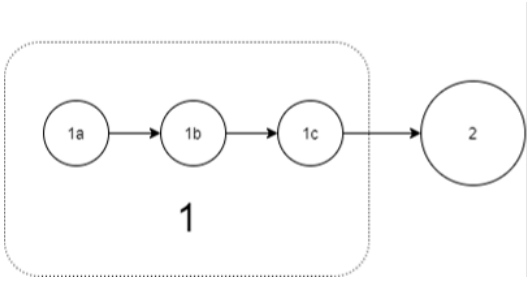

Figure 6: Step synchronization

4.1.2 Stochastic Petri Net Model

The second formal model that can be used to efficiently describe our problem is Stochastic Petri Nets model. In this case, each direction of travel is represented by a place of the Petri net, intersection are represented by transition from one place to another, and vehicle are represented by tokens in the network. The weight to determine the probability of choosing a direction rather than another is attributed to each transition. More precisely, a transition weight is equal to the least common multiple among the number of choice from each place divided by the number of choice of the place preceding the transaction. In this case, for our analysis, we focus on the average number of tokens in each place.

In order to represent the different time travel we can add multiple place for each place similarly to the method described in the Markovian model section. In addition the transition preceding the entering of a road will produce two tokens, the first entering the normal sequence of place representing the travel time of the road, and the second entering a new place representing the global road. The exiting transition will then consume two tokens, the first form the global place and the second from the last place of the sequence representing the travel time (Fig. 7). Using this method, we will focus only on the average number of tokens of each global place.

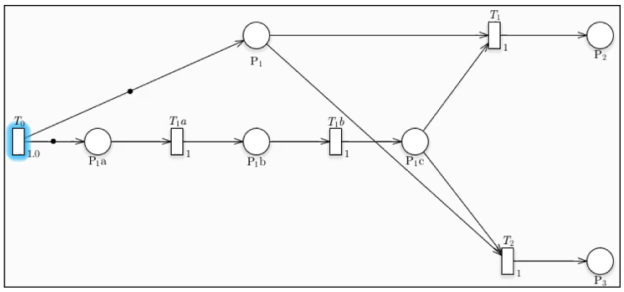

Figure 7: Distance representation in a Petri Nets

Two different method of re-insertion of vehicle exiting the model can be used. The first is a deterministic method where exiting vehicle re-enter the model from the same road on the opposite travelling direction. The second method is a probabilistic approach that re-insert the vehicle into a random entry of the model. In order to represent this second approach with Petri Nets, we create a new place that every exiting token enters. Then, a transition with very high

weight is created for each entry place, going form the global outside-place to the entry place. This ensure that tokens are re-inserted in the model after one step with a very high probability.

4.2 Analysis and Results

For realism purpose and to analyze the topology of a real location, we had to choose a specific region of a real town. We chose to model the region named "Les Acacias" which is a part of the town of Geneva located in Switzerland. Before modelling into a Petri Net, we had to put a label on each road. A road that is only one-way has a unique label like "42" and a road that is two-way has two label "42f" and "42b" (one for each way). We then modelled the area into a Petri Net and analysed using *GreatSPN*. (See fig. 8)

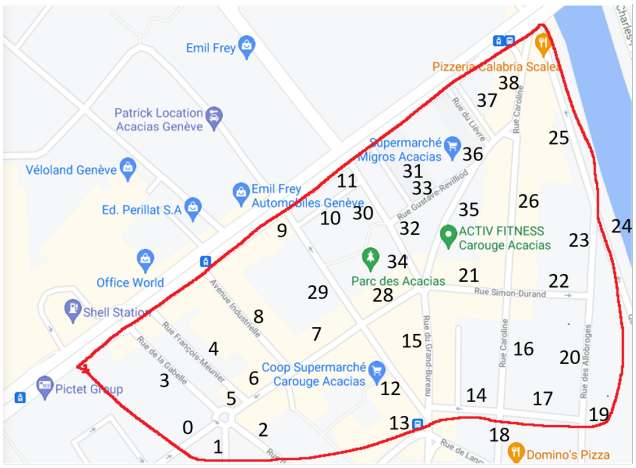

Figure 8: Les Acacias

4.2.1 Deterministic Entry Points

At first, we decided that each vehicle exiting the model is reinserted directly in the road of opposite travel direction he has exited from. We used *GreatSPN* to compute the average number of tokens in each place. To do this, *GreatSPN* simulate the model until a convergence of the average number of tokens is reached. The results are shown in Fig. 13.

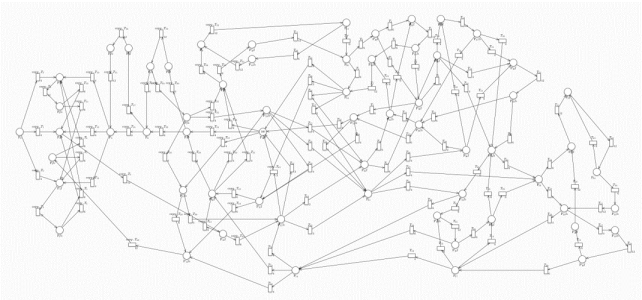

Figure 9: Petri Net of Acacias

As the table show, there are roads that are overloaded (for instance P_{20b}) and other that are under-loaded (for instance P_{7b}). This means that the road topology is not optimal in order to have a balanced vehicular flow.

Using the deterministic entry points method we notice that vehicular flow in some areas is inaccurately described. For instance P_0 can not be reached by any token since it represents a one-way entry road on the model (no vehicle can exit and re-inserted from and in P_0).

4.2.2 Randomly Chosen Entry Points

To resolve the problems mentioned right before, we modified our model to randomly reinsert vehicles that exited into an entry point of the circuit (Fig. 10).

Using such a random reinsertion pattern we obtain different results that are closer to reality. As shown in Fig. 14 there are no more places with zero average token. Moreover the overloaded and under-loaded roads have changed in some cases.

4.2.3 Local Analysis

To simplify the analysis of our model, we tried to split the global analysis into several local analysis. This split would lower the number of places/transitions of each part. This let us work on big model part by part, making the modelling and analyze simpler and faster.

We compared the global analysis and the local analysis in order to discover if these two approaches' results are equivalent. In our case, we split the "Acacias"

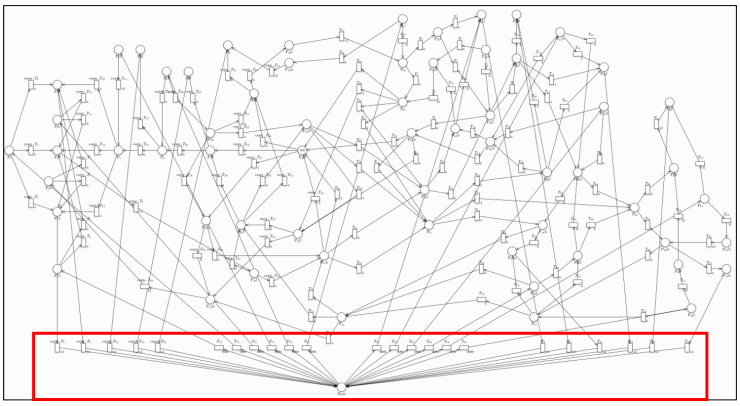

Figure 10: Random entry in the Petri Net

in two part (left-side and right-side). Then we execute the same analysis on the local parts and the global part. For what it concern the left local part (Fig. 15), we obtain and total token error of 4.373 (on a total of 47.185 tokens in the model), and an average error of 0.15. In particular, only exit and entry places count a total error of 2.51 with an average of 0.358. This shows that entry and exit places are the least accurate part of our analysis. The right part total error is 3.088 (on a total of 54.017 tokens in the model), and an average error of 0.088 (Fig. 16). In this part, entry-exit places counts an average error of 0.083 that is similar to the average error of all places. This study show that the local part analysis is possible with a good precision. In addition, it shows that, depending on the model, entry and exit places are the least accurate. In order to mitigate this border error, we could model an area slightly bigger than what we need to analyze. This would concentrate the error in the areas that are not meaningful to us, making our model more accurate in the part we are interested in (Fig. 11).

4.2.4 Complexity Analysis

Our idea is to determine experimentally the complexity of the GreatSPN statistical analysis. We start by creating complete Petri nets with n places and $n = 3,..,7$. A complete Petri Net is a Petri Net where each place have a transaction of weight 1 going to each place. To make the complexity analysis we executed GreatSPN statistical tool for each n five times with different random seeds. For each execution we are interested on the number of batches (each one charac-

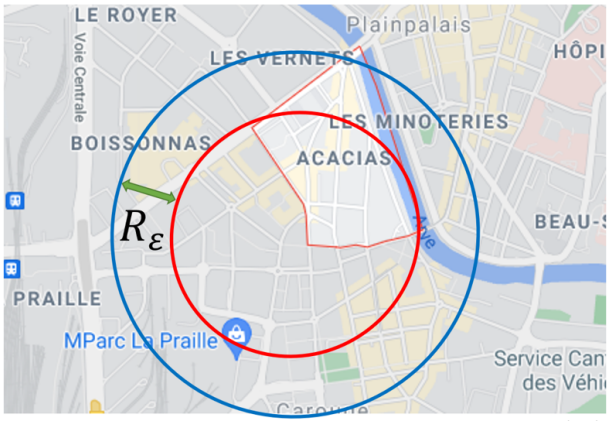

Figure 11: Mitigate border error

terized by 5000 events) need to reach a convergence with error lower than *1%*. As the Results in Fig. 12 shows, the average number of batches needed grow exponentially with the number of places. More over, by executing the same analysis on the same model with more tokens in the initial marking we notice that this parameter does not have an impact on the number of batches needed to reach a convergence.

From this results we can conclude that making multiple local analysis instead of a single global one with more places would reduce the execution time drastically.

Execution\Places	3	4	5	6	7
seed1	74	141	253	385	588
seed2	87	178	256	402	529
seed3	80	140	252	390	535
seed4	69	154	264	373	557
seed5	71	144	270	380	556
Avarage	76,2	151,4	259	386	553

Figure 12: Complexity analysis

4.3 Further works

Multiples studies can be done on this topic, in this section we give some ideas of further researches that we didn't cover during this project but they would be

interesting to do. More precisely, it would be interesting to:

- Compare the results obtained with our probabilistic analysis with real data on the traffic flow. This would give a method to evaluate our model.
- Try to find solutions for overloaded roads of the Acacias region. For example, by modifying the map topology (adding or deleting streets).
- Have traffic flow from South to North or North to South depending on the moment (rush hour).
- Further study the local analysis by finding some method to better estimate the transition weights in order to better approximate the global analysis.
- Compute mathematically the real complexity of the analysis tool.
- Add the notion of road length and travel time in the analysed model.
- Better estimate transition weight based on proximity areas of interest.

5 Conclusion

In this project, we tried to analyze properties of the insertion of vehicles in a single-lane roundabout using PRISM. We modeled this traffic using Nagel-Schreckenberg model's idea. We discovered that our problem was not well suited for a probabilistic model check using PRISM. We then switched to another problem that we thought was more suited for a probabilistic model checker. The new idea was to analyze the topology of a road circuit and find overloaded streets. We chose to model "Les Acacias" (a region of the city of Geneva) into a Petri Net. We used GSPN to do statistical analysis of the Petri Net model. We discovered where were the roads that are susceptible to be overloaded/underloaded. We also tried to divide the map into smaller parts in order to analyze them separately and independently. After several global/local analysis we discovered that smaller parts were faster to analyze and simplier to model. We then tried to prove that it was faster by doing an experimental complexity analysis of GSPN statistical analysis.

To conclude, this project allowed us to discover that PRISM and GSPN are powerful tools for analyzing properties of different models.

After having seen model checking theoretically during our studies, this project was a wonderful opportunity for us to discover in practice how model checking are done.

References

[1] http://www.di.unito.it/ greatspn/index.html.

[2] M. Duflot, M. Kwiatkowska, G. Norman, and D. Parker. A formal analysis of bluetooth device discovery. *International Journal on Software Tools for Technology Transfer*, 8:pages621–632, 2006.

[3] J. Heath, M. Kwiatkowska, G. Norman, D. Parker, and O. Tymchyshyn. Probabilistic model checking of complex biological pathways, October 2006.

[4] M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In G. Gopalakrishnan and S. Qadeer, editors, *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.

[5] Sukumar Nandi, Bibhas Kar, and Pushkar Pal. Theory and applications of cellular automata. *Computers, IEEE Transactions on*, 43:1346 – 1357, 01 1995.

[6] A. Schadschneider. The nagel-schreckenberg model revisited. *The European Physical Journal B - Condensed Matter and Complex Systems*, 10:573–582, 1999.

Appendices

Place	Average token count
P0	0
P7b	0.415806319
P26f	0.617217787
P37b	0.624062627
P37f	0.625613341
P4b	0.62745158
P4f	0.628265045
P20f	0.760140114
P36b	0.826727773
P38f	0.833662841
P38b	0.834959411
P8b	0.835404995
P8f	0.835997245
P12b	0.856074586
P35b	0.89883942
P10f	0.900016906
P32f	0.901921853
P10b	0.902316097
P33f	0.972824551
P31f	1.045710495
P31b	1.047023469
P28b	1.087979534
P32b	1.117192031
P15f	1.170102291
P33b	1.188161938
P6	1.252565507
P5	1.252820906
P36f	1.258830907
P19b	1.516901032

Place	Average token count
P18f	1.517724795
P18b	1.517999323
P16b	1.518616927
P19f	1.519609427
P16f	1.519643532
P35f	1.544149338
P7f	1.670154628
P2b	1.769340048
P13b	1.773003798
P30	1.801764639
P9	1.802426524
P11	1.803588023
P29	1.804539659
P26b	1.805970849
P21	1.845245388
P23b	1.898150309
P23f	1.899959591
P34	2.02034984
P28f	2.154884028
P14	2.267865634
P22	2.275180293
P17	2.276766598
P1f	2.605000435
P1b	2.605724864
P3f	2.60872585
P3b	2.610047859
P15b	2.698125474
P12f	2.851968793
P13f	3.021606776
P2f	3.025479718
P25	3.787817581
P24	3.793466631
P20b	4.550512225

Figure 13: Acacias results with deterministic entrees

Place	Average token count
Pout	0.001
P7b	0.717
P10b	0.720
P20f	0.739
P23b	0.742
P4f	0.742
P37f	0.751
P26f	0.758
P8f	0.768
P32f	0.829
P12b	0.854
P10f	0.887
P19b	1.025
P33f	1.148
P28b	1.167
P31f	1.242
P38f	1.244
P35b	1.316
P16f	1.319
P15f	1.397
P8b	1.435
P4b	1.438
P1b	1.440
P3b	1.440
P0	1.442
P18f	1.467
P38b	1.469
P37b	1.470
P11	1.472
P24	1.474

Place	Average token count
P31b	1.475
P19f	1.476
P36b	1.481
P6	1.485
P36f	1.493
P7f	1.539
P18b	1.569
P32b	1.573
P9	1.608
P30	1.650
P33b	1.663
P17	1.765
P29	1.773
P28f	1.853
P35f	1.854
P13b	1.875
P2b	1.879
P16b	1.955
P23f	2.047
P5	2.185
P14	2.248
P21	2.286
P26b	2.395
P34	2.398
P15b	2.500
P22	2.617
P13f	2.644
P2f	2.646
P12f	2.769
P25	2.779
P3f	2.863
P1f	2.872
P20b	3.076

Figure 14: Acacias results with random entrees

Place	Global	Local	Error
P0	1.442	1.383	0.059
P10b	0.72	1.384	0.664
P10f	0.887	0.908	0.021
P12b	0.854	0.938	0.084
P12f	2.769	2.624	0.145
P13b	1.875	1.807	0.068
P13f	2.644	2.570	0.074
P14	2.248	1.395	0.853
P15b	2.5	2.220	0.280
P15f	1.397	1.755	0.358
P1b	1.44	1.385	0.055
P1f	2.872	2.769	0.103
P28b	1.167	1.107	0.060
P28f	1.853	2.128	0.275
P29	1.773	1.806	0.033
P2b	1.879	1.811	0.068
P2f	2.646	2.560	0.086
P3b	1.44	1.384	0.056
P3f	2.863	2.773	0.090
P4b	1.438	1.385	0.053
P4f	0.742	0.740	0.002
P5	2.185	2.134	0.051
P6	1.485	1.487	0.002
P7b	0.717	0.693	0.024
P7f	1.539	1.580	0.041
P8b	1.435	1.375	0.060
P8f	0.768	0.788	0.020
P9	1.608	2.297	0.689
Pout	0.001	0.001	0.000

Figure 15: Local left analysis results (highlighted in yellow are the places that connect leftpart and right part)

Place	Global	Local	Error
Pout	0,001	0,001	0,000
P20f	0,739	0,682	0,057
P23b	0,742	0,691	0,051
P37f	0,751	0,804	0,053
P26f	0,758	0,811	0,053
P32f	0,829	0,985	0,156
P19b	1,025	1,025	0,000
P33f	1,148	1,198	0,050
P31f	1,242	1,244	0,002
P38f	1,244	1,262	0,018
P35b	1,316	1,306	0,010
P16f	1,319	1,364	0,045
P18f	1,467	1,369	0,098
P38b	1,469	1,369	0,100
P37b	1,470	1,371	0,099
P11	1,472	1,371	0,101
P24	1,474	1,376	0,098
P31b	1,475	1,378	0,097
P19f	1,476	1,38	0,096
P36b	1,481	1,408	0,073
P36f	1,493	1,534	0,041
P18b	1,569	1,54	0,029
P32b	1,573	1,605	0,032
P30	1,650	1,675	0,025
P33b	1,663	1,709	0,046
P17	1,765	1,966	0,201
P35f	1,854	1,984	0,130
P16b	1,955	2,007	0,052
P23f	2,047	2,053	0,006
P21	2,286	2,3	0,014
P26b	2,395	2,442	0,047
P34	2,398	2,557	0,159
P22	2,617	2,588	0,029
P25	2,779	2,673	0,106
P20b	3,076	2,989	0,087

Figure 16: Local right analysis results (highlighted in yellow are the places that connect leftpart and right part)