Traffic lights management using belief propagation

CÉLESTIN QUANG PHAM NGOC THINH

MSc by Research in Pattern Analysis and Neural Networks

ASTON UNIVERSITY

August 2006

ASTON UNIVERSITY

Traffic lights management using belief propagation

CÉLESTIN QUANG PHAM NGOC THINH

MSc by Research in Pattern Analysis and Neural Networks, 2006

Thesis Summary

There is a considerable interest in the investigation of traffic control in road networks. Road congestion is now part of daily life in urban environment. As cities develop, traffic congestion only worsens, and constructing more roads does not always resolve the problem. One of the approaches is to improve traffic lights management and make them adaptive to the traffic conditions.

Using the traffic information, one aims to estimate the best traffic lights configuration for each junction. Using probabilistic methods, message passing techniques [1] can be applied to a road network, represented by a sparse graph, such that each junction shares traffic and traffic light control information with neighbouring junctions.

It is the aim of the project to devise a belief propagation algorithm for efficient traffic lights management of a given road network according to the current traffic information. Two inherent cost/success measures will be employed based on equating traffic flow in all roads and on limiting traffic density per road below a fraction of its capacity. Simulations were carried out for both cases, and the algorithm gives good results. Depending on the cost function used, the algorithm manages to balance the loads or reduce the number of congested roads.

Keywords: Belief propagation, Traffic management, Roads networks

Acknowledgements

I would like to thank all the NCRG staff. Especially my supervisor David Saad, without who none of this work would have been possible. I would like to thank Pierre, Jack, Thomas, Boremi, Michel, and Radji for our talks and for playing badminton.

My last attention goes to my family, in particulary my parents, for letting come and study at Aston for greatest benefit.

Finally, thanks to Laetis and Titine.

Contents

List of Figures

Chapter 1

Introduction

Several approaches exist to tackle the problem of traffic congestion. It is a class of problem we encounter in many areas: road networks, the Internet or any information network. Some of the approaches propose offline and centralized solutions, where traffic is routed using fixed rules from historical data on traffic flow. The main drawback is that these methods are not flexible enough to cope with dynamic changes in the traffic volume. In addition, centralized methods typically require higher computional costs. In the case of road networks, adaptive management of the traffic lights offer a much better solution. Cooperative methods [2] have been introduced to change the traffic rules in correspondance to the local current traffic demands. Here we are interested in a locally computed global solution, and use a probabilistic method called belief propagation [1, 3, 4, 5] to infer the optimal traffic lights configuration from the traffic information.

1.1 Road traffic management

Traffic management in roads is actually achieved by traffic lights and sign posts. In the case of recurrent congestion in an area, constructing new roads can reduce loads and decrease traffic jams. However they typically offer a costly and long-term solution to this complex problem. That is one of the reason to focus on traffic lights management. A classical approach is to coordinate them so that traffic flows. In rural areas, where traffic patterns are more easily identified we can define fixed rules from historical data, to synchronize traffic lights. However, in an urban environment, the dynamics are complex and require a more flexible approach as the traffic changes quickly. To coordinate the traffic lights, decentralized solutions are also considered but usually involve communication between the traffic lights and therefore a costly implementation. Intermediate solutions between centralized and decentralized approaches also exist, using multiagent systems [6].

1.2 Traffic modelling

Real road networks are difficult to describe exactly mathematically. A large variety of models are being used to simulate traffic for traffic forecasting and control. Some of them simulate the behavior of each car on a given network [7, 8]. K. Nagel [8] proposed a microscopic model based on cellular automata. Each car is described as a particle moving on a cell network, with its own speed, acceleration, destination etc. Other models consider macroscopic properties, like the Lighthill, Whitham and Richards model which uses the analogy between traffic and the flow of fluids[8].

We formulate the traffic routing problem as a flow optimisation problem. We take into account the current traffic loads, roads capacities and the network topology to determine the appropriate routing rules which minimize a cost function. We first describe a macroscopic model used to represent road networks. It is based on bipartite factor graphs composed of two sets of nodes representing respectively junctions and roads. We also model the interactions between junctions. The traffic flows are simulated according to a model similar to fluid dynamics. We will then consider two different cost functions depending on our objectives. The first one aims to balance the traffic over the roads, ignoring the roads capacities. Afterwards, we will focus on congestion in the network and use an associated cost function with capacity constraints.

In both cases, a Belief Propagation algorithm is devised to minimize the cost function. Experiments were carried out on random graphs on which we create traffic and simulate it according to our model.

Chapter 2

Belief propagation algorithm

2.1 Bayesian networks

2.1.1 Definition

A Bayesian network is a directed acyclic graph, in which nodes represent random variables and directed links represent probabilistic dependencies among the variables. They are used for modelling decision support systems and many other systems which involve probabilistic inference of dependent variables.

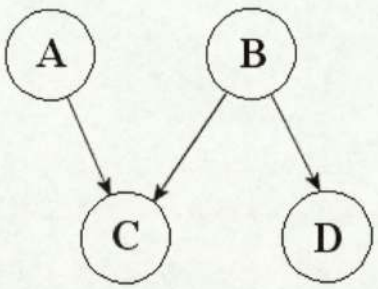

Figure 2.1: Bayesian network

The figure 2.1 is an example of Bayesian network. There is a directed link from node B to node D, it means variable D depends directly on variable B. The variable B is called a parent of D. For each node, there is a conditional probability distribution $P(A \mid pa(A))$, where $pa(A)$ are the parents of A. In the case of a node A without any parents, its probability is represented by an unconditional probability distribution $P(A)$. For some nodes, the variables have known values and are called evidence nodes. With this information, we can infer the conditional marginals of the variables on the non-evidence nodes.

2.1.2 Inference

The goal of inference in a Bayesian network is to find the conditional probability distribution of certain variables from the information on other variables with known values, by summing or integrating over any remaining undetermined variables. For example, in figure 2.1, one can determine the conditional probability $P(C \mid A, B)$, from the known distributions $P(A)$ and $P(B)$.

Several methods exists to infer these probabilities, exact and approximative ones. We are only interested in the approximative algorithm called belief propagation (BP), as it is the only algorithm used in this project.

2.2 Belief propagation in Bayesian networks

Given a Bayesian network, Pearl [3] proposed an iterative algorithm based on local propagation rules to infer the most likely values of its variables. The idea is to factor the marginal probability at a given node into a product of contributions coming from its neighbours with the following rules (using his notation):

Consider a node X, having children $Y_1, \ldots, Y_m$ and parents $U_1, \ldots, U_n$. e^+ and e^- denote the evidences coming respectively from its parent and child nodes.

Its marginal probability is given by:

$$P(X = x) = \alpha\lambda(x)\pi(x), \qquad \text{with } \alpha \text{ a normalizing factor} \tag{2.1}$$

where

$$\lambda(x) = P(e^- \mid X = x) = \prod_{i \in 1 \ldots m} \lambda_{Y_j}(x) \tag{2.2}$$

$$\pi(x) = P(X = x \mid e^+) = \sum_{u_1, \ldots, u_n} P(x \mid u_1, \ldots, u_n) \prod_i \pi_X(u_i) \tag{2.3}$$

All those quantities are calculated with the messages sent by both its parents $\pi_X(u_i)$ and children $\lambda_{Y_j}(x)$. In the same way X sends the following messages which are probabilities:

$$\text{to } U_i, \qquad \lambda_X(u_i) = P(U_i = u_i \mid X, U_{j \neq i})$$

$$\lambda_X(u_i) = \beta \sum_x \lambda(x) \sum_{u_k,\ k \neq i} P(x \mid u_1, \ldots, u_n) \prod_{k \neq i} \pi_X(u_k) \tag{2.4}$$

where β is a normalizing term

(2.5)

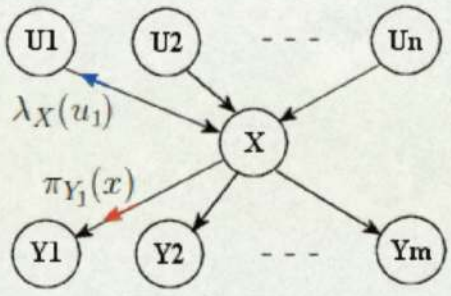

Figure 2.2: BP in Bayesian networks

$$\begin{aligned} \text{to } Y_i, \qquad \pi_{Y_i}(x) &= P(X = x \mid Y_{j\neq i}, U_1, ..., U_n) \\ \pi_{Y_i}(x) &= \frac{P(X = x)}{\lambda_{Y_i}(x)} \end{aligned} \tag{2.6}$$

This algorithm is executed for all the nodes of the network, and is guaranteed to converge in Bayesian networks to the correct posterior probabilities. It offers the other advantage to have a computational cost growing linearly with the system size, but exponentially with the connectivity.

2.3 Belief Propagation in bipartite factor graphs

A bipartite factor graph is composed of variable nodes $\underline{\mathbf{X}} = \{X_1, \ldots, X_n\}$ and factor nodes $\underline{\mathbf{Y}} = \{Y_1, \ldots, Y_m\}$. $\mathcal{L}(A)$ represents the neighbouring nodes of $\underline{\mathbf{X}}$ or $\underline{\mathbf{Y}}$ connected to node A. The factor nodes represents probabilistic dependencies between the variables nodes and are functions whose arguments are subsets of $\underline{\mathbf{X}}$. This representation expresses how the joint probability $P(\underline{\mathbf{X}} = \mathbf{x})$ can be approximately factorised into the product of the local functions $Y_i(\mathbf{x}_{\mathcal{L}(Y_i)})$. Those functions have for arguments the values of the neighbouring nodes.

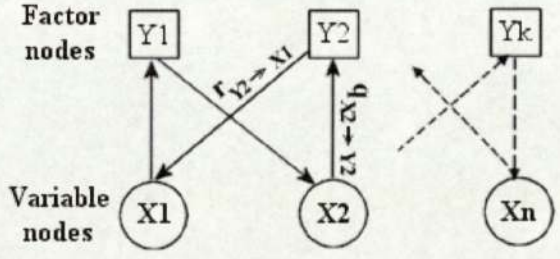

Figure 2.3: Factor graph

Joint probability mass function

$$\begin{aligned} P(\mathbf{X}=\mathbf{x}) &= \prod_i P(X_i \mid \{X_t\}_{t\neq i}) \approx \prod_i P(X_i \mid \mathcal{L}(X_i)) \\ &\approx \frac{1}{\mathcal{Z}} \prod_i Y_i(\mathcal{L}(Y_i) = \mathbf{x}_{\mathcal{L}(Y_i)}) \\ \mathbf{x} &= \{x_1, \ldots, x_n\} \text{ and } \mathcal{Z} \text{ is a normalizing term} \\ & \quad \mathbf{x}_{\mathcal{L}(Y_i)} \text{ are the values taken by the neighbouring nodes of } Y_i \end{aligned}$$

It has been demonstrated [4, 5] that BP is mathematically equivalent in factor graphs and Bayesian networks. However in this case, the message are sent between the two different types of nodes.

Message sent from Y_i to X_j:

$$\begin{aligned} r_{Y_i \to X_j}(x_j) &= P(Y_i \mid \{Y_t\}_{t\neq i}, X_j = x_j, \{X_k\}_{k\neq j}) \\ &= \sum_{\mathbf{x}_{\mathcal{L}(Y_i)} \setminus x_j} Y_i(\mathbf{x}_{N(Y_i)}, X_j = x_j) \prod_{X_k \in N(Y_i) \setminus X_j} q_{X_k \to Y_i}(x_k) \end{aligned}$$

Message sent from X_j to Y_i:

$$\begin{aligned} q_{X_j \to Y_i}(x_j) &= P(X_j = x_j \mid \{X_k\}_{k\neq j}, \{Y_t\}_{t\neq i}) \\ &\propto \prod_{Y_k \in \mathcal{L}(X_j) \setminus Y_i} r_{Y_k \to X_j}(x_j) \end{aligned} \tag{2.7}$$

This algorithm have been employed in graphs with loops and have given excellent experimental results for decoding in turbo-codes and Low Density Parity Codes [9]. A reason to explain convergence of the BP algorithm in loopy graphs has been still to be found and has been studied by Y. Weiss [10].

Chapter 3

Simple model

To manage traffic lights, we first need a mathematical model of road networks to apply the algorithm. Our approach is similar [11], we devise an algorithm to minimize a cost function.

As it was explained in the previous chapter, the BP algorithm has a computational complexity that grows exponentially with the connectivity. Road networks by nature have a graph structure with a connectivity which rarely exceeds 5, that is why the BP algorithm is suitable to the problem. It should give us a solution in a reasonable time scales whatever the size of the system is, as it grows linearly with the system size. By sending messages about the traffic lights configuration between junctions, we expect the algorithm to give a solution that minimizes traffic loads in the road network. In this project, we assume the traffic loads on roads to be known and usable as evidence.

In this chapter, we first model a road network as a factor graph. The second task will be to use the BP algorithm to optimize the traffic lights management given the information we have on the loads. In this first cost measure considered here, we do not consider any capacity constraints on the roads to simplify the problem.

3.1 Representation

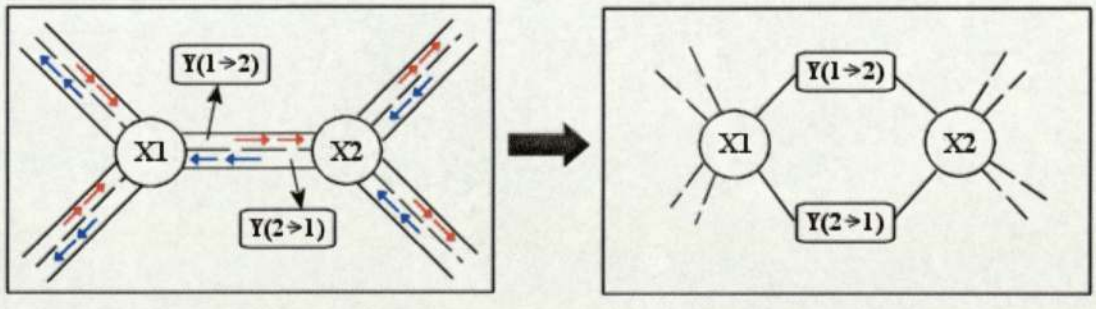

Figure 3.1: Conversion of the network to a factor graph

We consider a road network of N junctions, with a fixed number C of bidirectional separated roads connected to each of them. Each lane of these roads has a limited capacity denoted as κ_{ij} (for the lane $(i \rightarrow j)$). To apply the algorithm, the road network is converted to the associated factor graph as in figure 3.1. The factor nodes are the functions $f_{(ij)}(x_i, x_j)$ which give probabilistic information about the load given traffic informations Y_{ij}^t at time step t we have on each lanes of the roads. The variable nodes are the traffic light states X_i^t at time step t.

We do not model any driver behavior, so the car drivers follow exactly the traffic lights orders during their journey. Our model is pretty similar to information networks. The cars behave as information packets travelling along the edges of the graph, and being redirected accordingly to their destination at the nodes. However in our case, the drivers do not have any destination and drive in the network without stopping at any time. We assume the the total traffic within the network to remain constant.

Definition of X_i^t: The variable X_i^t represents the state of the traffic lights at junction i. In this model, no realistic and complex traffic lights configuration are used. Each state is just an indication of how to redirect the incoming flow, as routing rules. The traffic lights redirect the traffic flow of incoming roads to outgoing roads; however U-turns are not allowed.

$$x_i \in \Omega(i) \quad = \quad \{j \rightarrow k \mid j \in \mathcal{L}(i),\ k \in \mathcal{L}(i) \setminus j\}$$

with $\mathcal{L}(i)$ denoting all the junctions connected to i

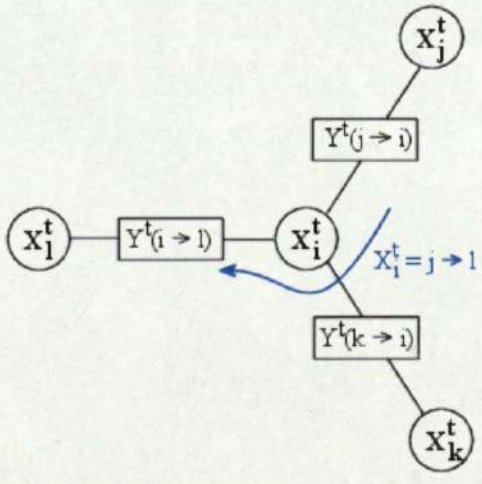

Figure 3.2: Example of X_i^t

We assume that the traffic lights cycle through a finite set of states for a duration

δ_i^t in a given order. Each of the states has a duration $\delta^t(x_i)$ so that:

$$\sum_{x_i \in \Omega(i)} \delta^t(x_i) = \delta^t$$

The sum of the fractions of cycle $\frac{\delta^t(x_i)}{\delta^t}$ allocated to each state is equal to 1.

$$\sum_{x_i \in \Omega_i} \frac{\delta^t(x_i)}{\delta^t} = 1$$

So we can define the fraction $\frac{\delta^t(x_i)}{\delta^t}$ as the posterior probabilities of having this state x_i given the global traffic information $\underline{\mathbf{Y}}^t = \{Y_i^t\}$ and others traffic lights configuration $\underline{\mathbf{X}}^t \setminus X_i^t = \{X_k^t\}_{k \neq i}$. The more probable that state is, the more time it will be allocated.

$$\frac{\delta^t(x_i)}{\delta^t} = P(X_i^t = x_i \mid \underline{\mathbf{Y}}^t, \underline{\mathbf{X}}^t \setminus X_i^t)$$

Definition of $Y_{i\to j}^t$: The variable $Y_{i\to j}^t$ represents the traffic information on the lane connecting junction i to junction j. It can have any value between 0 and κ_{ij}. It is not a discrete model, so that traffic information represents a real flow similar to the density of cars rather than their number on a given road. It is the evidence we are going to use.

The variables X_i^t are the values we are trying to find in this project, by using them as variables to optimise a cost measure. The BP algorithm will be used to determine the conditional propability distributions $P(X_i^t \mid \underline{\mathbf{Y}}^t, \underline{\mathbf{X}}^t \setminus X_i^t)$. Then the cost function will measure the quality of the expected traffic distribution this set of variables generates. It will work in a similar way as a cooperative system in which all the traffic lights share their information with their neighbouring junctions.

3.2 Next time step generation

The system has real dynamics, at each time step the traffic flows through the roads and junctions depending on the loads and traffic lights. We consider dynamics which are similar to fluids flowing through pipes. The outgoing traffic at a junction is equal to the incoming traffic and we assume that there is conservation of the traffic over time so there is no creation or loss of traffic between each time step. The main difference is that the state of each traffic light affects the traffic flow coming in and out of a given junction. That's why the junctions interact with one another as the state of traffic lights of a certain junction is function of the neighbouring ones. It has repercussions on the flow coming in and out of this junction.

The traffic on a road is defined as the incoming traffic through its origin minus the outgoing flow. The incoming flows are weighted by the fraction of traffic light cycle times allocated to the various traffic light states, so that only part of this traffic is contributing to the flow in the considered road.

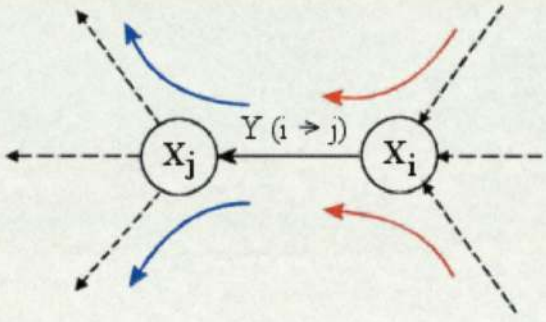

Figure 3.3: Interations between variables nodes

From that, we can generate the incoming traffic in a road at the next time step:

$$\Delta^{+}Y_{ij}^{t+1} = \sum_{k \in \mathcal{L}(i)/j} P(X_i^t = k \to j) Y_{ki}^t$$

The outgoing traffic is function of the time allocated to the road $(i \to j)$ which is the total duration of all traffic lights states $(i \to *)$ redirecting traffic from this road . It is defined as:

$$\Delta^{-}Y_{ij}^{t+1} = Y_{ij}^t \sum_{k \in \mathcal{L}(j)/i} P(X_j^t = i \to k)$$

The traffic increment is then the difference between the two previous values:

$$\Delta Y_{ij}^{t+1} = \Delta^{+}Y_{ij}^{t+1} - \Delta^{-}Y_{ij}^{t+1} \tag{3.1}$$

3.3 Cost function

3.3.1 Definition

Our main criteria to define the quality of a traffic lights configuration is to study the distribution of the traffic we expect it to generate. It would be interesting to observe how the loads on the roads are divided given this configuration. A good traffic light management would then lead to a balanced load, where there is no congestion. This is one method, and it is commonly used to qualify traffic networks. In information theory, the resource utilisation to route packets is also often used as a criteria. As we do not have any information of that kind, and will focus on traffic distribution.

The quadratic energy-cost function we are going to use is:

$$E_B(\underline{\mathbf{Y}^t}) = \frac{1}{NC} \sum_{i,j} (Y_{ij}^t)^2 \tag{3.2}$$

To normalize the cost function, it is divided by NC the total number of lanes.

The choice of this function can be explained by the fact that it is closely related to the variance of the traffic distribution. The variance of a traffic distribution is by definition:

$$\begin{aligned} V(\underline{\mathbf{Y}^t}) &= \frac{1}{NC} \sum_{i,j} \left(Y_{ij}^t - \frac{\overline{Y}}{NC} \right)^2 \\ &= \frac{1}{NC} \left(\sum_{i,j} (Y_{i,j}^t)^2 - \frac{2\overline{Y}}{NC} \underbrace{\sum_{i,j} Y_{i,j}^t}_{=\overline{Y}} + \sum_{i,j} \frac{\overline{Y}^2}{(NC)^2} \right) \\ &= \frac{1}{NC} \sum_{i,j} (Y_{i,j}^t)^2 - \left(\frac{\overline{Y}}{NC} \right)^2 \\ \text{where } \overline{Y} &= \sum_{i,j} Y_{ij}^t \quad \text{the total traffic} \end{aligned} \tag{3.3}$$

If we neglect the constant term $\left(\frac{\overline{Y}}{NC}\right)^2$, the cost function is equal to the variance. This cost function is relevant as it measures the variance of the traffic distribution, and therefore how balanced are the traffic loads over the network. The more distributed the traffic on the roads is, the closer the loads on the roads are and the lowest the cost function is.

Our goal is to optimise this cost function so that loads are spread equally on each road. We will also use it as a termination criteria for the algorithm and stop its iterations once variations are below a defined threshold.

3.3.2 Optimal distribution

In this toy problem, the optimal solution which minimizes the cost function is easy to find. The only constraint we have is that the overall traffic $\overline{Y}$ is constant over time.

$$\forall t, \ \overline{Y} = \sum_{i,j} Y_{ij}^t$$

Due to the way the cost function is related to the variance of the system, the minimum of the variance is the minimum of the cost function. Actually the variance goes to 0 when each Y_{ij}^t is the mean traffic load $\frac{\overline{Y}}{NC}$. It obviously still satisfies the traffic conservation constraint and represents a state of minimum difference for the variables.

The minimum cost is then:

$$E_B(\underline{\mathbf{Y}}^*) \quad = \quad \frac{1}{NC} \sum_{i,j} \left(\frac{\overline{Y}}{NC}\right)^2$$

3.4 Factor nodes

3.4.1 Expected loads calculation

What we need in the case of the factor nodes is not the generated loads, but the expected value of loads we would get if we set the two traffic lights X_i, X_j at certain states x_i, x_j. We consider uniquely the contribution those traffic lights states make to the current road. It means that the incoming and outgoing flows still need to be weighted by the posterior probabilities $P(X_i^t = x_i \mid \underline{\mathbf{Y}}^t, \underline{\mathbf{X}}^t \setminus X_i^t)$.

If both states involve the considered road then they both contribute to change the traffic load.

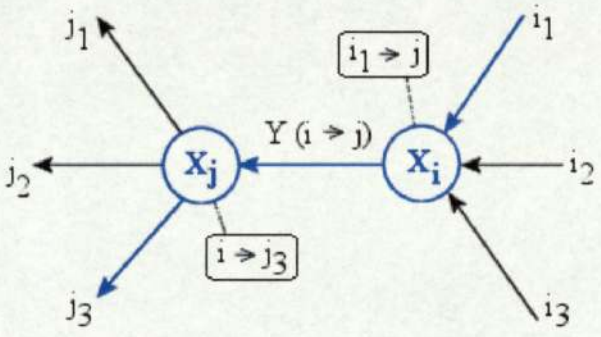

Figure 3.4: The two traffic lights contribute to update the flow of Y_{ij}^t

$$\begin{aligned} Y_{ij}^{t+1}(x_i, x_j) \quad = \quad Y_{ij}^t \quad &+ \quad P(X_i^t = (k \to j) \mid \underline{\mathbf{Y}}^t, \underline{\mathbf{X}}^t \setminus X_i^t)\, Y_{ki}^t \\ &- \quad P(X_j^t = (i \to k') \mid \underline{\mathbf{Y}}^t, \underline{\mathbf{X}}^t \setminus X_j^t)\, Y_{ij}^t \end{aligned}$$

Otherwise that means one of the state does not redirect traffic in or from this road. Thus the lane might have only an incoming or outgoing flow.

$$\begin{aligned} \text{if } x_i \neq (* \to j),\ Y_{ij}^{t+1}(x_i, x_j) \quad &= \quad Y_{ij}^t \quad - \quad P(X_j^t = (i \to k') \mid \underline{\mathbf{Y}}^t, \underline{\mathbf{X}}^t \setminus X_j^t)\, Y_{ij}^t \\ \text{if } x_j \neq (i \to *),\ Y_{ij}^{t+1}(x_i, x_j) \quad &= \quad Y_{ij}^t \quad + \quad P(X_i^t = (k \to j) \mid \underline{\mathbf{Y}}^t, \underline{\mathbf{X}}^t \setminus X_i^t)\, Y_{ki}^t \end{aligned}$$

The last case is if none of the states concerns the current road, then:

$$\text{if } x_i \neq (* \to j) \text{ and } x_j \neq (i \to *),\ Y_{ij}^{t+1}(x_i, x_j) \quad = \quad Y_{ij}^t$$

3.4.2 Definition of factor node functions

By definition the factor nodes represent the interactions between variables. Previously, we saw that the traffic lights states have obviously repercussions on the system's evolution. Two neighbouring traffic lights will interact and modify the loads on the lanes connecting them. From the load on a lane, we can compute an expected load $\hat{Y}_{ij}^{t+1}(x_i, x_j)$ for the next time step if traffic lights X_i^t and X_j^t are set to the states x_i and x_j. This expected value concerns only the contribution of those two states x_i and x_j, neglecting any others incoming or outgoing flows. We define the factor nodes to represent the posterior probabilities of having these expected contributions $\hat{Y}_{ij}^{t+1}(x_i, x_j)$. The global problem is to minimize the cost function, by distributing all the traffic over the graph, but locally it corresponds to having each lane to minimize its traffic load.

We will model the posterior probabilities to favour low expected values of flow, by giving them a higher probability. This will affect the BP algorithm. Thus each lanes will try to reduce its own load, but the message-passing algorithm will compare their belief and expectation to get a global solution.

We will use a gaussian probability distribution, as they are easy to deal with and multiply it with the Θ function to forbid any negative value:

$$f_{(ij)}(x_i, x_j) = P(\hat{Y}_{ij}^{t+1} \mid X_i^t = x_i,\ X_j^t = x_j,\ \underline{\mathbf{Y}^t}) \tag{3.4}$$

$$\propto \exp\big(-\beta\,(\hat{Y}_{ij}^{t+1}(x_i, x_j))^2\big)\ \Theta(\hat{Y}_{ij}^{t+1}(x_i, x_j)) \tag{3.5}$$

β : temperature parameter

$\Theta(x)$: the Θ function returns 1 for a non-negative x, and 0 otherwise

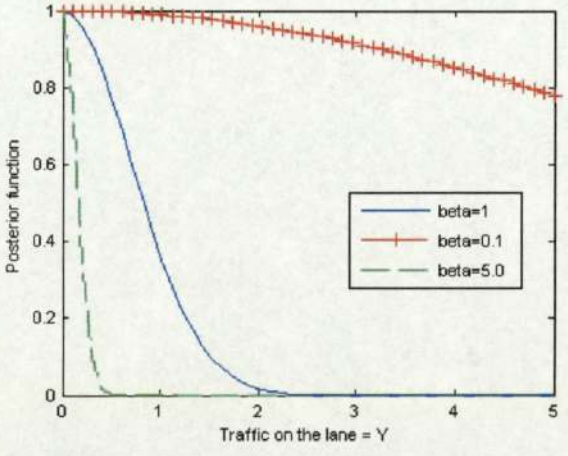

Figure 3.5: Gauss function

The β parameter controls the stiffness of the $f_{(ij)}(x_i, x_j)$ function, as we can see in figure 3.5. For too high values of β, the function favours strongly low values of flow, and can result in giving only very small values which give rise to numerical problems. On the other hand, too small values of β will give messages that won't differentiate between solutions. In this case, it would behave similarly as an uniform distribution and the low loads would not be favoured.

As $\hat{Y}_{ij}^{t+1}$ can only reach value between 0 and $+\infty$, $f_{(ij)}(x_i, x_j)$ has to be normalised by $\frac{1}{\mathcal{Z}}$ to ensure it to be a probability distribution.

$$\mathcal{Z} = \int_0^{+\infty} e^{-\beta u^2} \, \mathrm{d}u = \frac{1}{2}\sqrt{\frac{\pi}{\beta}}$$

Chapter 4

Algorithm

The traffic lights control algorithm consists in sending message between the junctions X_i^t and the lanes Y_{ij}^t connecting them. Gradually these messages should converge and give an approximative solution to the posterior distribution given the evidence. In this chapter, we will focus on describing how to compute these messages and how to apply the algorithm.

4.1 Message passing equations

There are two types of messages to send, depending if it is sent from a variable node X_i^t or a factor node Y_{ij}^t. They are respectly the $q_{i\to(ij)}(x)$ and $r_{(ij)\to i}(x)$ messages. We update each type of message at the same time for all nodes.

We can distinguish two different cases depending on whether the message are sent from the origin of the roads or the destination of the road.

If the messages are sent from a lane to its origin junction (i.e. $r_{(ij)\to i}(x)$ messages), the marginalization is carried out over all the possible states of the destination traffic lights. The associated update-rules are:

$$r_{(ij)\to i}(x_i) \quad = \quad \sum_{x_j \in \Omega(j)} f_{(ij)}(x_i, x_j) \quad q_{j\to(ij)}(x_j) \tag{4.1}$$

Otherwise, if they are sent between a lane and its destination junction, the equations are:

$$r_{(ij)\to j}(x_j) \quad = \quad \sum_{x_i \in \Omega(i)} f_{(ij)}(x_i, x_j) \quad q_{i\to(ij)}(x_i) \tag{4.2}$$

Here the marginalization is done over the states of the traffic lights at the incoming junction.

The message $q_{i\to(ij)}(x_i)$ is sent from the junction X_i to the lane Y_{ij}. It is the product of the messages $r_{(*i)\to i}(x_i), r_{(i*)\to i}(x_i)$ coming from the other incoming lanes

Y_{*i} and outgoing lanes Y_{i*} connected to junction X_i. It has to be normalized over all the possible values of x_i.

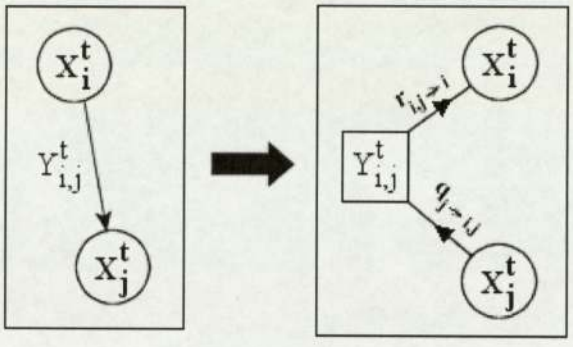

Figure 4.1: Gauss function

$$q_{i\to(ij)}(x_i) \quad \propto \quad \prod_{k\in\mathcal{L}(i)\setminus j} r_{(ki)\to i}(x_i) \quad \prod_{k\in\mathcal{L}(i)} r_{(ik)\to i}(x_i)$$

We chose to initialise the messages at random normalized values as it does not change the efficiency of the algorithm.

4.2 Description

The input to the optimisation is the network topology, the link capacities and an estimate of the demand between each pair of edge nodes in the network. The output of the optimisation is a routing that gives the optimal flow on each link, according to a cost function.

The parameter ϵ, which define the tolerance for the variations of the cost function, is set at 0.0001.

At step time step $t = 0$
Compute the system variance σ^2 (Eq. 3.3)
Compute $\beta = \frac{1}{2\sigma^2}$
Initialise the messages to random values
Compute the cost function and save the value in *oldCost*

for time step $t = 1$ to $t = t_{max}$ **do**
  for $it = 1$ to $it = it_{max}$ **do**
    for every road $(i \rightarrow j)$ of the graph **do**
      Compute the r-messages (Eq. 4.1 and 4.2)
    end for
    for every junction i of the graph **do**
      Compute the q-messages (Eq. 4.3)
    end for
    Compute the posterior probabilities from the current messages (See 2.3)
    Estimate the expected traffic loads from the current configuration (See 3.4.1)
    Compute the cost function and save it into *cost*
    if $abs(cost - oldCost) \leq \epsilon$ **then**
      Go out of the it loop
    end if
  end for
  Generate next time step
end for

Chapter 5

Experiments and results

Before starting the experiments, we implemented a random road network generator. To avoid any finite-size effect, we used it to generate large networks of $N = 1000$ junctions. We only consider graphs of fixed connectivity to simplify the problem, but the algorithm can be extended to any connectivity profile. As in most traffic models, we assume the traffic to be Poisson distributed; we set the initial traffic values accordingly.

The experiments are carried out by applying the BP algorithm at each time step to find a configuration for the traffic lights. The solution is then used to generate the next time step, giving us new traffic loads on the roads. Finally, the system evolution is achieved by iterating those steps.

In order to study the performance of the algorithm, for each generated network, we also let the system evolve from the initial state without applying any algorithm. We assign the traffic lights at a precise configuration, for which the probability distribution $P(X_i^t \mid \underline{\mathbf{Y}}^t, \underline{\mathbf{X}}^t \setminus X_i^t)$ is uniform.

$$\forall x_i \in \Omega(i),\ P(X_i^t \mid \underline{\mathbf{Y}}^t, \underline{\mathbf{X}}^t \setminus X_i^t) = \frac{1}{\mid \Omega(i) \mid}$$

where $\mid \Omega(i) \mid$ is the cardinality of $\Omega(i)$

That is effectively giving the same duration to all traffic lights state. Thus the outgoing traffic of any roads is shared equally with the roads connected to its outgoing end. We will also compare the algorithm's efficiency to the optimal distribution $E_B(\underline{Y}^*)$.

We will study how the algorithm converges to the optimal value. A second experiment is done to analyse its behavior given the network connectivity. Finally, we will observe the number of iterations the algorithm requires to converge to a solution at each time step.

5.1 Definition of the β parameter

The only parameter we can manipulate in the model is the β parameter. It drastically changes the performance of the algorithm, sometimes giving no convergence at all if chosen wrongly . The best performance was achieved by using a value related to the system variance $V(\underline{\mathbf{Y}}^t)$. We thought about updating the β parameter during the system evolution, and changing it to the new variance at each time step. We will see further that, as the time progresses, the algorithm manages to equilibrate the loads. The variance tends then to become very small and we encountered numerical problem in dealing with these small values. We therefore decided to use a constant value for β which is the variance of the system at the first time step.

During all the experiments, the β parameter has the value:

$$\beta = \frac{1}{2\,V(\underline{\mathbf{Y}}^0)} \tag{5.1}$$

$$\text{where } V(\underline{\mathbf{Y}}^0) = \frac{1}{NC}\sum_{i,j}\left(Y_{ij}^t - \frac{\overline{Y}}{NC}\right)^2$$

5.2 Convergence to the optimal solution

In figure 5.1, we show one example with a $N = 1000$, $C = 3$ network. The β parameter is set as describe previously. The initial state is generated from a Poisson probability distribution of mean= 2.5.

We notice that as the time goes by, the algorithm manages to get close to the optimal distribution. Moreover it converges to a steady state and keeps the loads balanced on all roads. The optimal distribution is not reached in one step due to the time it takes traffic to migrate on the graph; it may take several time steps for the algorithm to be able to distribute the traffic on all the roads. The traffic reduction of a congested road takes time, especially if it is located in a congested area.

We notice that the algorithm behaves better that an uniform allocation of the traffic lights. It favours the highly loaded roads to distribute their traffic. Using flat posteriors, we still converge to the optimal distribution as it stillscatters the traffic. But that would not be necessarily achieved with a random allocation, because of the possible existence of congestion clusters. They are areas where the traffic lights tend to promote the incoming flows. It results in an increasing overall traffic in the area over time.

We can see in figure 5.2 that the traffic loads on each roads tend to converge to the mean of the traffic over the network. It is logical as the cost is getting close to its minimum, the variance of system is becoming close to zero. It represents a state of nearly perfect distribution of the traffic so that the load is shared by all.

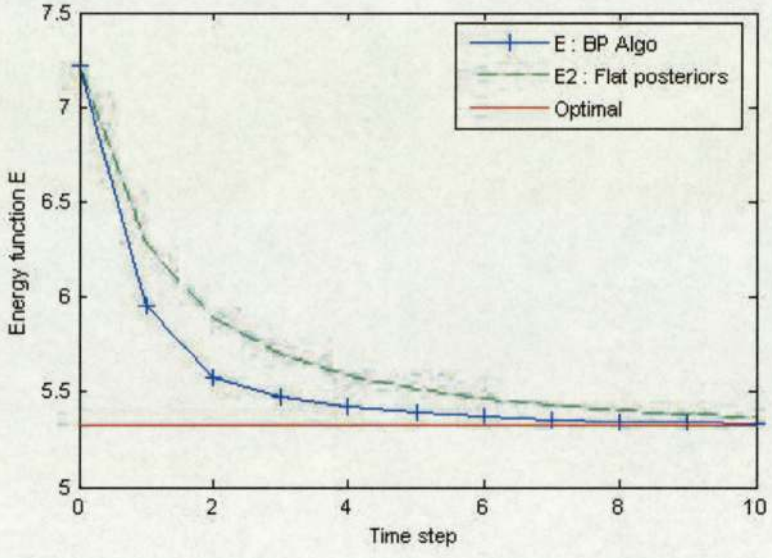

Figure 5.1: Cost function given the time

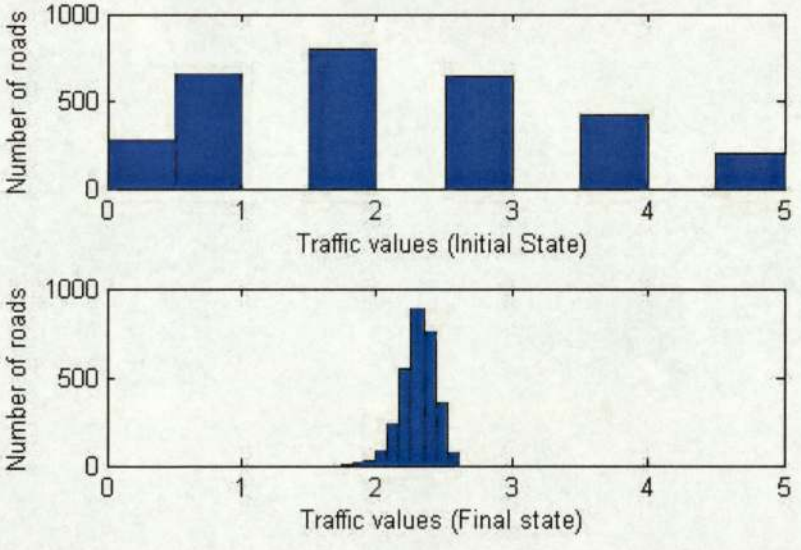

Figure 5.2: Histograms of the traffic values at the initial and final states

5.3 Performance given the connectivity

To study the comparative performance between our algorithm and an equal-time assignement of the traffic lights, we measure the percentage of improvement by using the BP algorithm at each time step.

Improvement measure definition:

$$\text{Perf}(t) = 100 \times \frac{E_B(\underline{\mathbf{Y}}_F^t) - E_B(\underline{\mathbf{Y}}_{BP}^t)}{E_B(\underline{\mathbf{Y}}_F^t) - E_B(\underline{\mathbf{Y}}^*)} \tag{5.2}$$

$$\text{where } E_B(\underline{\mathbf{Y}}^*) = \frac{1}{NC} \sum_{i,j} \left(\frac{\overline{Y}}{NC}\right)^2$$

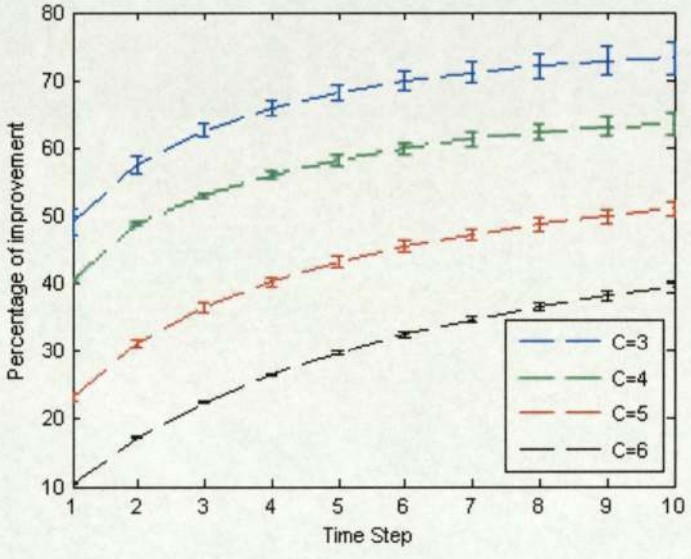

Figure 5.3: Performance measurement given the connectivity (run on 500 randomly generated networks)

The statistics on figure 5.3 show the performance of the BP algorithm compared to the flat posteriors case. For any connectivity, the performance goes up as the system evolves, however at the same time we observe a slow down of the increase. As we reach the optimal distribution, the traffic is more distributed than at the initial state, so the traffic loads tend not to change very much. The functions $f_{ij}(x_i, x_j)$ give very close probabilities whatever are the states x_i and x_j. The more loaded roads are not favoured anymore, and it results in an nearly equal allocation of time for all traffic lights. That is why the improvement rate is decreasing with time.

The choice of using a constant β is also part of the reason of the slowing down. By updating it at each time step, the smaller value of β we use in the lastest time step would make the little traffic difference sensible.

Another important point to notice is that we get a smaller improvement as the connectivity increases. The more roads are connected to a junction, the easier it becomes to distribute the traffic. Congested roads can reduce their traffic much faster, and have more solutions to do so. In this case, an equal-time assignement is numerically

'closer' to the solution found by the algorithm. That explains why the difference in performance diminishs when the connectivity is growing.

5.4 Iterations at each time step

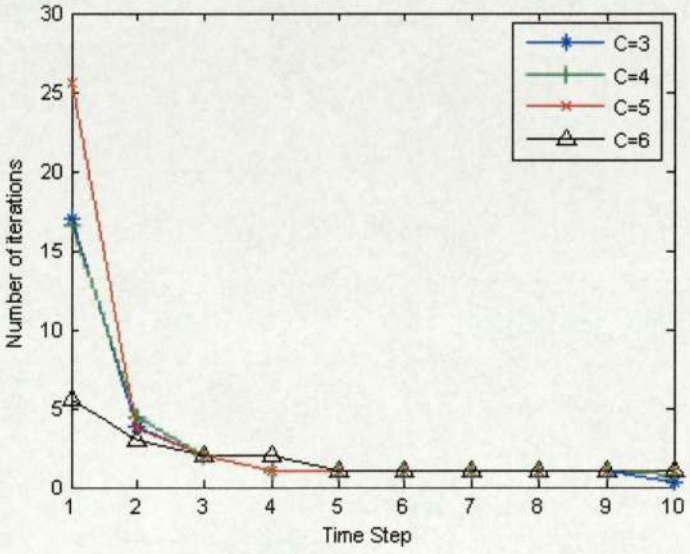

Figure 5.4: Average number of iterations of the BP algorithm at each time step given the connectivity (variance is smaller than the symbol)

I also measured, at each time step and for different connectivities, the number of iterations needed for the algorithm to find a solution. In figure 5.4 we show the results averaged over 500 networks. We notice that for the connectivity $C = 6$ the algorithm definitely converges faster. As previously, the more choice we have to distribute the traffic, it becomes easier to find a solution. Thus the algorithm requires less iterations to reach a stable state of its messages. The biggest difference is at the first time step, that is when the the network is most unbalanced and traffic light control is the most difficult to manage.

However there is a behavior we cannot explain in the results we got. The $C = 5$ connectivity case is troubling, because it takes statistically more iterations for the algorithm to converge to a solution at the first time step. The structure of the graph with this connectivity does not differ from the other cases and cannot be the cause of this. After a long time looking for errors in the implementation of the algorithm, I could not find the reason for this behaviour. Understanding this behaviour would require further study.

Chapter 6

Limited capacity model

We saw that the algorithm manages to balance the loads on the roads and minimizes the cost function. It was a simple problem that we make more complicated by considering limited capacity constraints on the roads. From now on each road has its own capacity κ_{ij} that we have to take into account in the optimisation problem. This will give us a much more realistic model, in a sense that real traffic networks cannot handle an unlimited number of cars. In this chapter, instead of studying the traffic distribution, we will focus on the network congestion.

Our task is to introduce a new cost function as our criteria have changed, and modify the factor nodes definition to fit the new problem.

6.1 Cost function

6.1.1 Definition

The new cost function is a global congestion measure of the network. For that it gives the percentage of congested roads. We consider that a road is congested, if its traffic is above a certain ratio α of its capacity κ_{ij}.

$$E_C(\underline{\mathbf{Y}}^t) = \frac{1}{NC} \sum_{i,j} \Theta\left(Y_{ij}^t - \alpha\, \kappa_{ij}\right) \tag{6.1}$$

$\Theta(x)$: the Θ function returns 1 for a non-negative x, and 0 otherwise

The optimal value is obviously zero, if the traffic is distributed so that none of the roads is congested.

6.1.2 Satisfiable case

The satisfiable case is when the cost function can be nullified, from the knowledge we have about the network. It exists then a solution to distribute the traffic so that all

the loads are below the threshold. The following inequality should be obeyed if such a minimum could be found.

$$\sum_{i,,j} Y_{ij}^t \quad \leq \quad \alpha \sum_{i,j} \kappa_{ij}$$

If the inequality is satisfied, then it means that we have enough ressources to tackle all the traffic demands, the global traffic demand being smaller than a considered fraction of the network total capacity.

6.2 Factor nodes

From now we will model the factor nodes, so that they only accept not-congested roads. We define them as:

$$\begin{aligned} f_{(ij)}(x_i, x_j) &= P(\hat{Y}_{ij}^{t+1} \mid X_i^t = x_i,\ X_j^t = x_j,\ \underline{\mathbf{Y}^t}) \\ &= \frac{1}{\alpha \kappa_{ij}}\, \Theta(Y_{ij}^t)\, \Theta\!\left(\alpha - \frac{Y_{ij}^t}{\kappa_{ij}}\right) \end{aligned} \tag{6.2}$$

By using the step function Θ, we only allow positive traffic values below a fraction α of the road capacity.

Practically, if we use a function of this form, it might generate a chain reaction which nullify all the messages. Let consider the case where whatever are the ingoing and outgoing flows of a road, the expected contribution $\hat{Y}_{ij}^{t+1}$ is above $\alpha\ \kappa_{ij}$. So for all values of (x_i, x_j), the function $f_{(ij)}(x_i, x_j)$ returns 0. From the equations (4.4) and (4.6), we notice that the $r_{(ij)\to j}(x)$ messages are nullified. Therefore the $q_{i\to(ij)}(x)$ are also equal to 0. We end up with no messages sent between nodes.

In order to solve that issue, we need to approximate the step function and give some tolerance for the limited capacity constraints. For that we use the complementary error function erfc(x). It is defined by:

$$\operatorname{erfc}(x) \quad = \quad \frac{2}{\sqrt{\pi}} \int_x^{+\infty} e^{-t^2}\, \mathrm{d}t$$

We approximate the factor function by multiplying the erfc(x) with the step function $\Theta(x)$ to forbid any negative traffic values:

$$f_{(ij)}(x_i, x_j) \quad \approx \quad \frac{1}{\mathcal{Z}_{ij}} \operatorname{erfc}\!\left(\beta \left(\frac{Y_{ij}^t}{\kappa_{ij}} - \alpha\right)\right) \Theta(Y_{ij}^t)$$

Once again, the β parameter controls the stiffness of the curve. The higher it is, the better is the approximation. On the other hand the constraints are stricter, as the values above the threshold are less tolerated. We can observe that on figure 6.1.

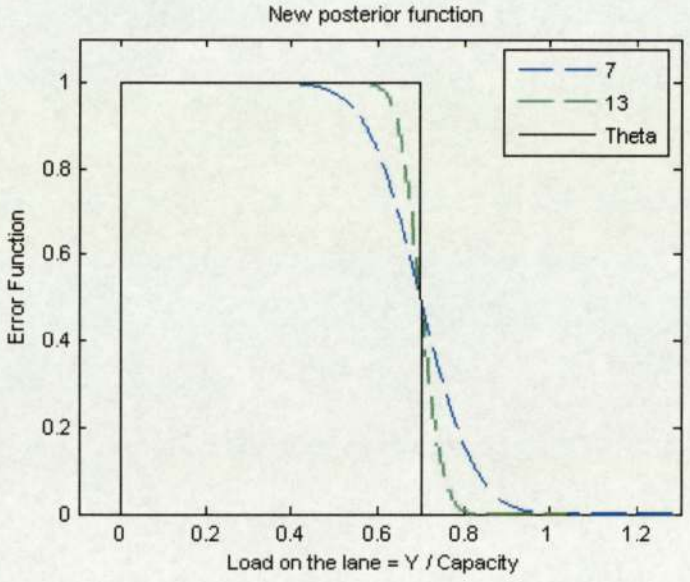

Figure 6.1: Error function

To ensure that $f_{(ij)}(x_i, x_j)$ describes a probability distribution, it has to be normalized by:

$$\begin{aligned} \mathcal{Z}_{ij} &= \int_0^{\kappa_{ij}} \operatorname{erfc}\Big(\beta\left(\frac{u}{\kappa_{ij}} - \alpha\right)\Big)\,\mathrm{d}u \\ &= \frac{\kappa_{ij}}{\beta}\int_{-\alpha\beta}^{\beta(1-\alpha)} \operatorname{erfc}(v)\,\mathrm{d}v \\ &\quad \text{by substituting with } v = \beta\,(\frac{u}{\kappa_{ij}} - \alpha) \end{aligned}$$

From [12], we know that the indefinite integral of the complementary error function is:

$$g(u) = \int \operatorname{erfc}(u)\,\mathrm{d}u = u\operatorname{erfc}(u) - \frac{e^{-u^2}}{\sqrt{\pi}} + C$$

We conclude that:

$$\mathcal{Z}_{ij} = \frac{\kappa_{ij}}{\beta}\Big(g(\beta(1-\alpha)) - g(-\alpha\beta)\Big)$$

Chapter 7

Experiments and results

Simulations were carried out with networks of size N=1000. However, difference in the graph generation from the previous chapter 5 is that, from now on we assume that the roads have symmetrical capacities. It means that $\kappa_{ij} = \kappa_{ji}$. It is usually a property of road networks and it makes the problem much simpler. I have noticed that with small connectivity ($C = 3$) graphs, having asymmetrical edges prevents the algorithm from finding a solution. Locally, we can imagine an overloaded junction where outgoing capacities are too small to let the traffic disperse. That would make the computation impossible to carry out, especially in small connectivity graphs when there are less choices to distribute the traffic. Our simulation implementation of the traffic, allows loads to go above the capacity.

We will use a congestion threshold set at $\alpha = 0.7$. The termination criteria is defined so the algorithm stops when the variations in the cost function are smaller than 1%.

As the algorithm did not manage to give any consistent results for a connectivity C= 3, we do not consider this case. I encountered many numerical errors for this connectivity.

7.1 Influence of β

7.1.1 Relationship between β and connectivity

The performance of the algorithm are highly dependent on the connectivity and the stiffness parameter β, as we can observe in figure 7.1 and 7.3.

For $C = 4$, the problem is more difficult than for graphs with higher connectivities, as we have less choice to distribute the traffic. Among the values examinded the lowest value $\beta = 7$ offers the best performance, as we relax the capacity constraints and allow more traffic over the threshold. Using this constant value of β is enough to make the algorithm converge.

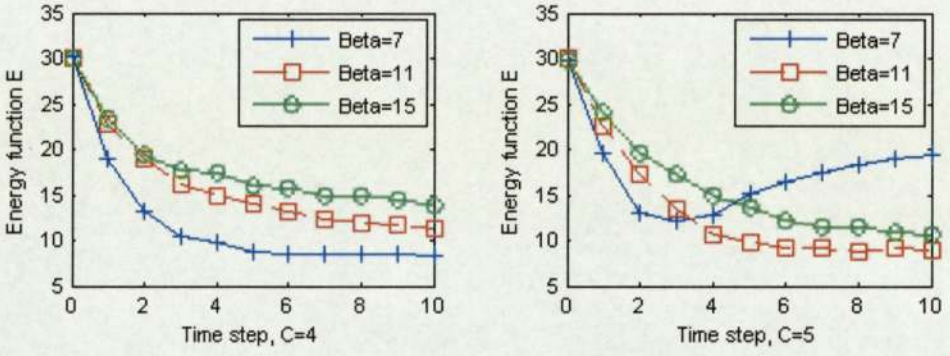

Figure 7.1: Energy function for different β values, $C = 4$ and $C = 5$

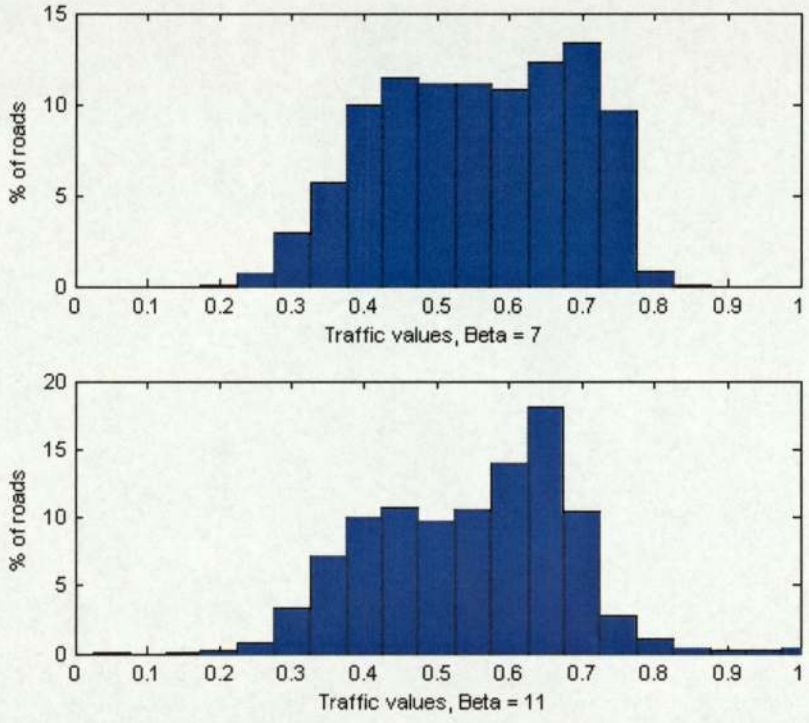

Figure 7.2: Final load distribution for $C = 5$

In the case of $C = 5$, soft constraints do not lead to successful solutions arguably as there are many competing solutions. Starting from time step 3, the congestion is sufficiently reduced, and the loads tend to accumulate around the threshold value (figure 7.2). By using a higher value $\beta = 11$ the difference between values above and below the threshold is emphasized. We somehow help the algorithm to strengthen decision at this blurry border.

For $C = 6$, it is similar to the previous case. As there are more roads connected to each junction, the traffic spreads even faster. From figure 7.3 and 7.4, we can see that the increase of congested roads happens earlie than for $C = 5$, and even occurs

for $\beta = 11$. As the traffic is distributed faster, it takes less time to uncongest roads and therefore loads tend be around the threshold value sooner, figure 7.4. It becomes more difficult for the algorithm to take a decision, even for $\beta = 11$, as there are more roads connected to each junction. Constraints have to be stricter than in the $C = 5$ case. Only with harder constraints $\beta = 15$ does the algorithm manage to converge.

In this last case, I suggest another strategy to define β.

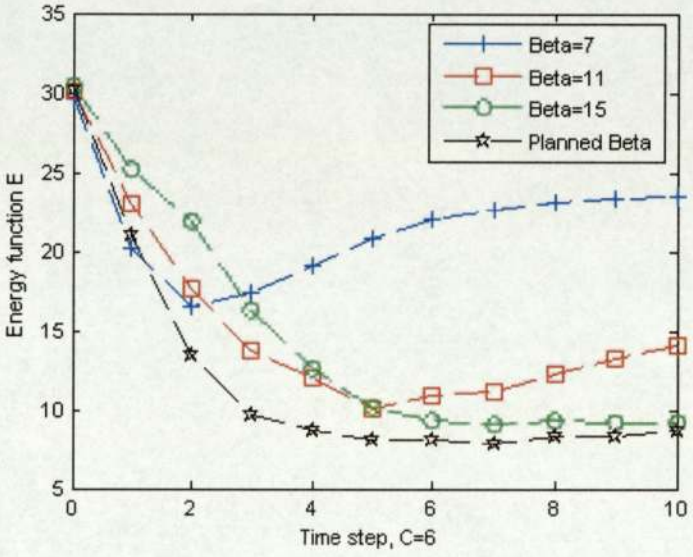

Figure 7.3: Energy function for different β values, $C = 6$

7.1.2 Adaptive β values

From figure 7.3, we can notice that depending on the values of β, the algorithm behaves better for different stages of the simulation.

Instead of using a fixed value of β, I decided to use predefined values of β for each time step. By increasing the value of β, I tighten the constraints in order to guide the BP algorithm as it is more and more difficult for the algorithm to converge. The convergence of the algorithm is definitely faster and the number of congested roads is even smaller.

Time step	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
β	7	11	12	13	14	15	18	18	18	19

To choose the values of β, I observed the evolution of the cost function for different β and choose the value which gave me the lowest number of congested roads at a given time step.

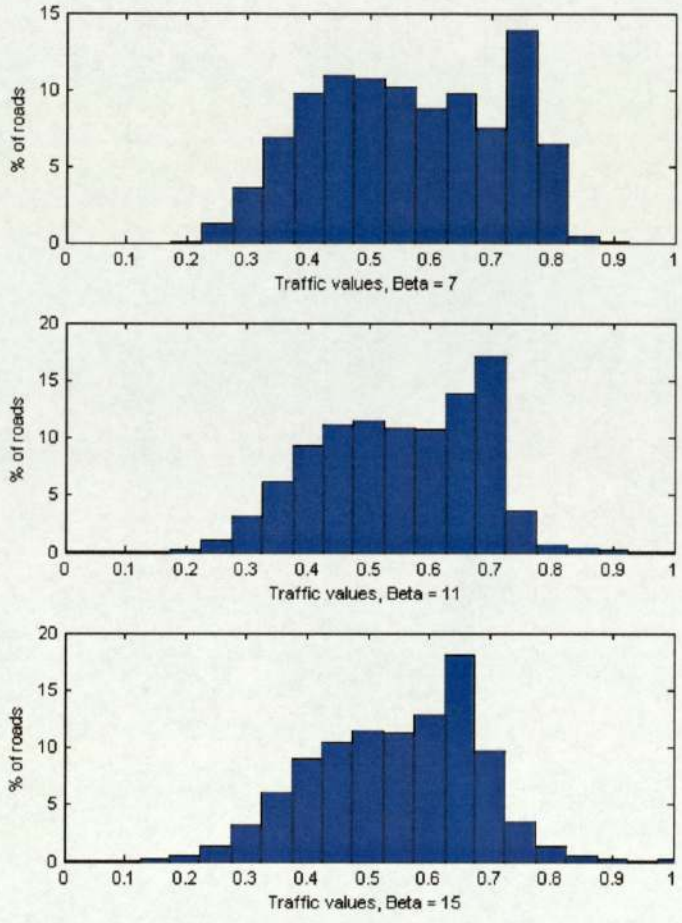

Figure 7.4: Final load distribution for $C = 6$

7.2 Convergence

From the initial traffic distribution, the algorithm reduces the number of congested roads until it stabilizes. We do not achieve an absolutely uncongested network, because of the approximation taken, by using the erfc(x) function, we tolerate some values above the threshold. It is particulary clear on figure 7.5 that the final state is representative of our model.

The algorithm is not time-consuming, as all the computations are carried out locally. By using BP, we manage to reduce the number of congested roads, while respecting the capacity constraints. The objectives expressed in the cost function are clearly satisfied, even if we do not reach the optimal solution.

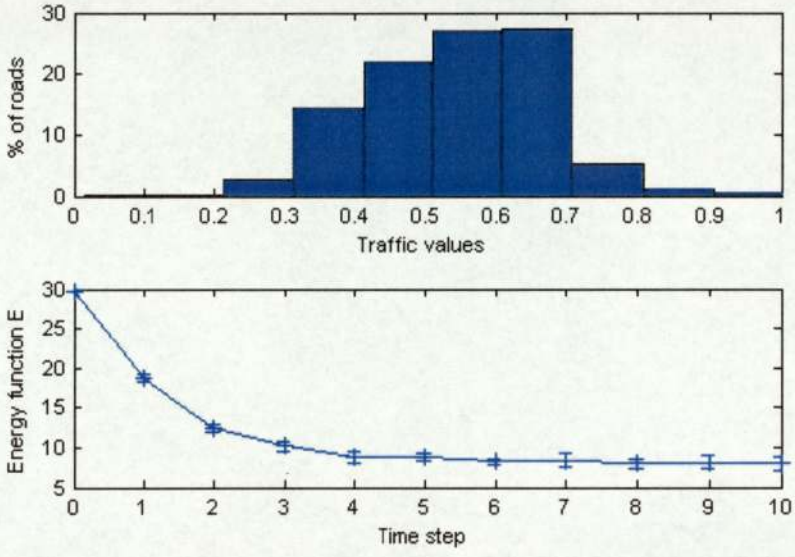

Figure 7.5: Histogram of the initial and final states of the system, and cost function ($C = 4$)

Chapter 8

Conclusion

8.1 Achievement

This project aims at devising a probabilistic algorithm for optimal traffic lights control given some measure if success.

The fist part of this project focused on balancing loads on irrespective of road capacity restrictions, modelling traffic networks with an approach similar to fluid flow. Our model is similar to information networks, where the traffic is routed at each junctions given some simple rules. It was a really simple model, considering only the capacities of the roads. We did not take into account any practical properties as driver behavior, limited speed, traffic build-up, but concentrated on balancing loads. The analogy with fluid dynamics is particulary clear as we assume conservative flows. Although it is simple, the probabilistic BP algorithm we devised managed to give good results.

With the first experiments, we demonstrated that the algorithm is capable of balancing the loads on a random network and offers an improvement compared to an equal-time assignement of the traffic lights state. We observed a higher improvement as the problem became more difficult, i.e., at lower connectivity networks. The algorithm concentrates on the highly loaded roads to accelerate the traffic distribution. The number of iterations the algorithm needed to find a solution was small, especially in the last steps of the simulation when the system was close to the optimal distribution. It showed dependence on the difficulty of the problem and the connectivity.

The second set of experiments we carried out was based on an algorithm aimed at reducing congestion. For that, we firstly added capacity constraints to the model and modified the factor nodes to fit these constraints using the ratio of congested roads as a cost function. We noticed that the performance of the algorithm is highly dependent on the β parameter we use; it reflects how strict are the constraints and has to be adapted to the problem. For the algorithm to work in difficult cases, the constraints has to be soften, using a lower value for β. But as there is more choice to distribute the traffic,

and less congested roads, restricting the constraints allows to guide the algorithm when it is indecisive. Once β is correctly set, the algorithm demonstrated that it was able of reducing the number of congested roads, while respecting the capacity limits.

8.2 Future directions

The current work should be used in conjunction with a better model to represent more realistic traffic scenarios. It would be interesting to observe how it behaves with more complex models. A good start would be to use OD-networks to have information on road loads, and then use it to define road priority indexes. More complex models can also give us a better simulation of the traffic, the cars will not necessarily follow the routing rules and may have their own destination and behavior. More realistic traffic lights state should also be considered, with multi-road routing and different cycle durations. Once all of this done, one can look at the results on real networks.

However I would not extend this approach to others networks like the Web or power-grids, as they have higher connectivity and would be really time-consuming as it is now. It is definitely not appropriate in these cases. Kabashima [13] studied the application of BP in dense graphs, and should be studied further.

Bibliography

[1] D. MacKay. *Information Theory, Inference, and Learning Algorithms.* Cambridge University Press, 2003.

[2] D. de Oliveira Ana Bazzan and V Lesser. Using Cooperative Mediation to Coordinate Traffic Lights: a Case Study. In *Proceedings of Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 463–469, July 2005.

[3] J. Pearl. *Probabilistic reasoning in intelligent systems : networks of plausible inference.* Morgan Kaufmann Publishers Inc., San Francisco, 1988.

[4] J. Yedidia, W.T. Freeman, and Y. Weiss. *Constructing Free Energy Approximations and Generalized Belief Propagation Algorithms.* TR-2004-040, 2004.

[5] F.R. Kschischang, B.J. Frey, and H-A. Loeliger. *Factor Graphs and the Sum-Product Algorithm.* IEEE Transactions on Information Theory vol 47, 2001.

[6] T. Bellemans, B. De Schutter, and B. De Moor. Models for traffic control, 2002.

[7] B. Chopard, P. Queloz, and P. Luthi. Traffic models of a 2d road network.

[8] Kai Nagel. Traffic networks, 2002.

[9] R. G. Gallager. *Low Density Parity-Check Codes.* MIT Press, Cambridge, MA, 1963.

[10] Y. Weiss. *Correctness of Local Probability Propagation in Graphical Models with Loops.* Neural Computation 12, 2000.

[11] K.Y.M. Wong, C.H. Yeung, and D. Saad. Distributed algorithms for global optimization on sparse networks of arbitrary bandwidths, 2006.

[12] Abramowitz, M., and Stegun. *Repeated integrals of the error function, Handbook of mathematical functions with formulas, graphs, and mathematical tables, 9th printing, pp. 299-300.* New York: Dover, 1972.

[13] Yoshiyuki Kabashima. *A CDMA multiuser detection algorithm on the basis of belief propagation.* J. Phys. A **36** page 11111, 2003.