Generation of safe lanes

In this report, we explain how to implement an algorithm that creates a coherent mesh of safe lanes that protects roads between assets. In this approach, the traffic problem is solved as a linear impedance problem between distant nodes, as it is classically done in economics. The mesh of lane that is generated aims at optimizing some cost-function that aims at measuring how hard it is to travel along the lanes. To perform this, we compute the gradient of this cost-function by the adjoint-state method and at each iteration, the best lane in each system is generated. Even if the solution it converges to is very probably sub-optimal, this algorithm shows good general behaviour and seems to be usable as a safe lane generator.

1 Introduction

When seen as a topological optimization problem, the generation of safe lanes needs to be based on a direct path-finding problem. The Dijkstra shortest-path algorithm could seem to be the best choice in that case. However, it suffers from one very penalizing flaw : as far as the author knows, there is no simple way to compute the gradient of the operator with respect to the impedance of the paths [1]. For that reason, it is preferred to solve the path-finding problem as an impedance problem.

In this context, the computation of the gradients can be done at very low computational cost via the adjoint method, and a gradient-based optimization method can be carried up. However, in the present case, this minimization problem is very susceptible to have many constraints (mostly related to the interaction between lanes of different factions). The constraints will probably cause the cost-function to be not convex with respect to the parameters of the optimization, and cause difficulties to a minimization algorithm. For this reason, it was chosen to use a much simpler algorithm, as described in the abstract, that however is very likely to converge to a sub-optimal solution.

This report is divided as follows : the direct impedance problem and its resolution *via* sparse linear algebra is first described. Then we introduce the cost-function that we will minimize, as well as the computation of its gradients. Finally, the chosen algorithm is presented.

2 The direct impedance problem

The discrete impedance problem arises naturally in physics from elliptic PDEs that model electricity, diffusive phenomena and mechanics. It is also used in economics to model the traffic of goods.

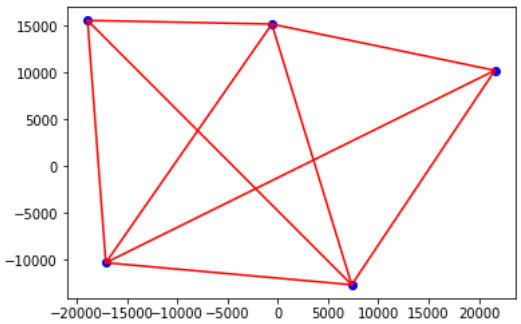

FIGURE 1 – A mesh

Let us suppose one wants to solve such a problem on the mesh presented in figure 1. Each node (blue) has un unknown potential u, and on each edge (red) there is an unknown oriented flux f. For

1. AFAIK, the shorter-path problem is not even well-posed in sense of Hadamard (because of lack of continuity)

each edge, there is also a given impedance k. On table 1, the physical quantities represented by u, f and k are listed.

Domain	u	f	k	Remark
Economy	Quantity of goods	Flux of goods	?	
Traffic modelling	Population	Traffic	?	It's our case
Thermic	Temperature	Heat flux	Thermal conductivity	
Diffusion	Quantity of particles	Flux of particles	Conductivity	
Electricity	Electric potential	Intensity	Inverse of resistance	
Mechanics	Displacement	Stress	Stiffness	u is a vector

TABLE 1 – Potential and Flux in different domains

One can impose three types of boundary conditions :

— Dirichlet BC consist at imposing some values of u at some nodes.
— Neumann BC consist at imposing an input flux b at some nodes. This leads to $\sum_j f_j = b$ for all edges that start from the given node.
— Robin BC consist at imposing a linear relation between u and b at some node. This leads to $\sum_j f_j = \mu u$.

Solving the impedance problem consists in determining the potential at each node and the flux at each edge of the graph from the given boundary conditions.

2.1 Numerical resolution of an impedance problem

2.1.1 Impedance matrix

From the potentials u_i, one can build a vector U. This vector is the unknown of the problem, and the fluxes f_j are computed by post processing U. For example, if the edge j links the node m to the node n, one has :

$$f_j = k_j(u_m - u_n) \tag{1}$$

From the given impedances k, one can build an impedance matrix K, by assembly of elementary matrices K_j^e build on each edge.

$$K_j^e = \begin{pmatrix} k_j & -k_j \\ -k_j & k_j \end{pmatrix} \tag{2}$$

For example, on the given mesh of figure 1, the impedance matrix would be :

$$K = \begin{pmatrix} k_1+k_2+k_3 & -k_1 & 0 & -k_2 & -k_3 \\ -k_1 & k_1+k_4+k_5+k_6 & -k_4 & -k_5 & -k_6 \\ 0 & -k_4 & k_4+k_7+k_8 & -k_7 & -k_8 \\ -k_2 & -k_5 & -k_7 & k_2+k_5+k_7+k_9 & -k_9 \\ -k_3 & -k_6 & -k_8 & -k_9 & k_3+k_6+k_8+k_9 \end{pmatrix} \tag{3}$$

This matrix is symmetric and the sum of coefficients on all lines and all columns is 0. One can show that this matrix is also semi-definite positive. The kernel of this matrix is composed of uniform vectors on the connected parts of the graph, and there are as many zero eigenvalues as there are disconnected sub-graphs. This point is very important in our context as the universe of Naev composes

a disconnected graph. For example, if we have a graph composed of 2 disconnected regions, the eigenvectors will be :

$$V_1 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} ; \quad V_2 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} \tag{4}$$

2.1.2 Taking into account the boundary conditions

There are 3 ways to take into account Dirichlet boundary conditions : penalization, substitution and Lagrange multipliers. All these operations lead to modifications of the matrix K (and of the right hand side B). If there is at least one node with a Dirichlet BC on each connected part of the graph, the matrix K becomes definite positive, which means that it has no more kernel and the linear system has a unique solution. In this report, we won't develop further on Dirichlet BCs

For Neumann BC, one assembles the right hand side F, that has the same size as U, and that concatenates the Neumann boundary conditions. The solution of the impedance problem is U such that :

$$KU = F \tag{5}$$

Robin boundary conditions are taken into account by adding μ to component (i, i) of the matrix K, where i is the index of the node where the Robin BC is applied. If there is at least one node with a Robin BC on each connected part of the graph, the matrix K becomes definite positive, which means that it has no more kernel and the linear system has a unique solution.

2.2 Impedance matrix kernel and resolution of the problem

The kernel of the operator has a great importance as it makes the matrix singular, and consequently, it is not possible to solve straightforwardly a linear problem that implies this matrix. Actually, the solution of such a problem is not unique and any vector that is inside the kernel can be added to a solution.

Once one Robin BC has been added on each connected part of the graph, the system has an unique solution. This solution is also one of the solutions of the problem without the Robin BCs if there is only one Robin BC per connected part of the mesh, and if the right hand side is orthogonal to the kernel.

There are a few interesting properties that will be useful to determine automatically during computation if two nodes are connected by the graph :

— If the right hand side is non-zero only in one connected sub-graph, the solution of the system vanishes in all the other sub-graphs.
— If the right hand side is orthogonal to the kernel of the matrix K (ie. with zero mean), the solution at a node with Robin BCs is zero.
— If the right hand side is non-zero only in one connected sub-graph, and each of its terms is greater or equal to zero, the solution of the system vanishes only in the other sub-graphs, and is strictly greater than zero in the sub-graph.

By virtue of the last property, if the right hand side is $b_i = \delta_{ij}$ (δ being the Kronecker symbol, which means 0 everywhere and 1 on node j), the only nodes connected to j are such that the corresponding component of $K^{-1}F$ is nonzero.

2.3 Large sparse impedance problems

In practice, when the problem get larger, each node tends to be linked to only a small part of all the other nodes. As a consequence, the matrix K has many zero components. We refer at such a matrix as sparse. Performing linear algebra operations for sparse matrices is mathematically equivalent to performing the same operations on full matrices, but the computational cost can be vastly reduced if a spasre linear algebra library is used.

3 Topological optimization problem

The problem we are interested in consists in designing the best graph of lanes in some sense, given by a cost-function. In our application, the connectivity of the graph does not change during the optimization process. What changes is the impedance of each edge of the graph, that is influenced by the presence of a safe lane.

A node is created for each inhabited asset of each system, as well as for each jump point, and internal edges are considered between each couple of nodes in the same system. The impedance of an internal edge is taken as equal to the invert of its length, and is changed when a safe lane is activated. External edges are considered between jump points of different systems, and their impedance is fixed. This creates a sparse graph that links the assets. A few edges are then suppressed in case they are too close from an other shorter edge.

3.1 Parameter-dependant impedance

We denote by $L_j \in \mathbb{R}_+^*$ the length of an edge and $\theta_j^k \in \{0, 1\}$ is 1 if there is a safe lane of faction k and 0 otherwise. $\sum_k \alpha\theta_j^k$ can not be greater than 1, which means that there can't be more than one lane per edge. α is the performance parameter of safe lanes. Here is how the impedance of an internal edge is computed :

$$k_j = \frac{1 + \sum_k \alpha\theta_j^k}{L_j} \tag{6}$$

Remark : For now, it is assumed that the impedance is the same for all factions, which is wrong at least in the case where these factions are enemies. Removing this hypothesis would mean that there is one impedance matrix per faction, and so one direct and adjoint problem per faction. This can be done, but will definitely have a big computational cost.

One finally ends up with an impedance matrix that is dependant on the set of parameters θ, and that will from now be denoted by $K(\theta)$.

3.2 Boundary conditions of the direct problem

We choose to add a Robin boundary condition on an arbitrary node in each connected part of the graph. We refer as such nodes as anchors. If there is only one anchor per connected pert of the graph, and if the right-hand side is orthogonal to the kernel of K, one can show that the choice of the particular node has no impact on the solution of the problem.

If one wants to compute the path between two nodes, one has to put a Neumann condition of 1 on the first node and -1 on the second node. The result is the flux of travellers when people want to go from one node to the other one.

The resolution of the entire impedance problem requests to solve the previous problem for any couple of assets, which would mean to solve and store the solution for $n(n-1)/2$ linear problems, where n is the number of inhabited assets. This could lead to cause severe performance issues. However, thanks to the property of linearity of this problem, one can store only the n solutions to problems with a Neumann condition on one node. This set of solutions is stored as a full matrix $\tilde{U}$ for which every column is the solution of one of these problems.

We introduce $\tilde{F}$, the multiple right hand side associated to the previous Neumann problems, and we get :

$$K(\theta)\tilde{U} = \tilde{F} \tag{7}$$

If we introduce the multi-index $\eta = (i, j)$, one can also write the following relation :

$$f_{k,\eta} = \tilde{f}_{k,i} - \tilde{f}_{k,j}, \quad \forall k \tag{8}$$

In a matricial notation, this writes :

$$F = \tilde{F}P^T \tag{9}$$

Where P is a sparse matrix of size $n(n-1)/2 \times n$. This matrix can be assembled thanks to its sparsity. By linearity, one has also :

$$U = \tilde{U}P^T \tag{10}$$

If i and j are not connected by the studied graph, which can be detected using the properties of part 2.2, then the couple (i, j) should not be considered in P. The corresponding line should be removed.

3.3 Topological optimization procedure

The principle of a topological optimization procedure is to build a cost-function $\phi(\theta)$ and find the best values of θ for which ϕ is minimal. One can introduce a function $\varphi(U, \theta)$ such that :

$$\phi(\theta) = \varphi(U, \theta), \text{ with } K(\theta)U = F \tag{11}$$

The sketch of the optimization process is as follows :

— Initialization : θ_0 is zero except for the lanes that are imposed as active
— At each iteration, compute U by solving the multiple right hand side linear system $K(\theta_i)U = F$
— Find the next iterate θ_{i+1}

4 Cost-function and gradients

4.1 Minimization problem

The cost-function should measure how hard it is to travel along the graph. It will be computed from the sum on all edges of the flux on this particular edge by the invert of the edge's impedance. On a given edge, one introduce $g_{j,\eta} = f_{j,\eta}/k_j$. One then has, according to (1) the following result :

$$g_{j,\eta} = u_{m,\eta} - u_{n,\eta} \tag{12}$$

This means that in our particular application, the cost-function φ does not depend on θ directly. In a matricial notation, we introduce the sparse difference matrix Q of size ?? :

$$G = QU = Q\tilde{U}P^T \tag{13}$$

We introduce now the fact that there are different faction, each of them has its own weight for a given couple of nodes. This is taken into account with the diagonal weighting matrix D_p :

$$G_p = Q\tilde{U}P^T D_p \tag{14}$$

One introduces $P_p = D_p P$, and then we have :

$$G_p = Q\tilde{U}P_p^T \tag{15}$$

With this notation, U, which size is too big, has been replaced by $\tilde{U}$, but for clarity, we will still write $\varphi(\tilde{U})$. This cost-function itself is computed as the Frobenius norm of the matrix G, that is :

$$\varphi(\tilde{U}) = \frac{1}{2}\|Q\tilde{U}P_p^T\|_f^2 \tag{16}$$

Where :

$$\|G\|_f = \left(\sum_{i,j} g_{ij}^2\right)^{1/2} \tag{17}$$

The minimization problem then reads :

$$\min_{K(\theta)\tilde{U}=\tilde{F}} \frac{1}{2}\|Q\tilde{U}P_p^T\|_f^2 \tag{18}$$

4.2 The adjoint method

Let us introduce the matrix term to term product : $c = A \cdot B$ means $\forall(i,j), c = \sum_{i,j} a_{ij}b_{ij}$.

In this part, we seek to determine the gradient of the cost-function ϕ with respect to θ. In order to take into account the constraint $K(\theta)\tilde{U} = \tilde{F}$, one introduces the Lagrange multiplier Λ and the Lagrangian $\psi(\theta, \tilde{U}, \Lambda)$:

$$\psi(\theta, \tilde{U}, \Lambda) = \frac{1}{2}\|Q\tilde{U}P_p^T\|_f^2 + \Lambda \cdot \left(K(\theta)\tilde{U} - \tilde{F}\right) \tag{19}$$

One can show that the solution of the minimization problem is also the saddle-point of ψ. One can compute the gradients of ψ :

$$\begin{cases} \nabla_u \psi = Q^T Q\tilde{U}P_p^T P_p + K(\theta)^T \Lambda \\ \nabla_\lambda \psi = K(\theta)\tilde{U} - \tilde{F} \\ \nabla_\theta \psi = \Lambda \cdot \nabla K(\theta)\tilde{U} \end{cases} \tag{20}$$

See annex A for the computation of $\nabla_u \psi$. The computation of $\nabla_\lambda \psi$ is not classic either, but quite straightforward.

For $\tilde{U}$ and λ such that $\nabla_u \psi = 0$ and $\nabla_\lambda \psi = 0$, one has $\nabla \phi = \nabla_\theta \psi$.

Remark : As ∇K is symmetric, one can show that $\Lambda \cdot \nabla K(\theta) \tilde{U} = \tilde{U} \cdot \nabla K(\theta) \Lambda$

As a conclusion, in order to compute $\nabla \phi$, which is an essential point in the algorithm, one has to do the following computations :

— Solve MRHS problem $K(\theta)\tilde{U} = \tilde{F}$ to obtain $\tilde{U}$
— Solve MRHS problem $K(\theta)\tilde{\Lambda} = -Q^T Q \tilde{U}$ to obtain $\tilde{\Lambda}$
— Compute $\Lambda = \tilde{\Lambda} P_p^T P_p$
— Evaluate $\nabla \phi = \Lambda \cdot \nabla K(\theta) \tilde{U}$, as detailed in annex C.

5 The algorithm

An idea could be to use a gradient-based minimization algorithm (BFGS or steepest descent) to minimize ϕ under a few additional constraints, with respect to $\theta \in [0; 1]$, and to use a regularization function, or any other procedure that tends to impose that θ is 0 or 1. However, this approach is quite complex and not straightforwardly compatible with the fact that there are different factions, that minimize different cost-functions.

In this report, an other approach is proposed. It consists in activating at each iteration one lane per system and per faction. This lane is the one with the strongest gradient (in absolute value). It is presented at algorithm 1.

Algorithm 1: Lanes building algorithm

Compute Q and $Q^T Q$
For each faction, compute P_p and $P_p^T P_p$
Assemble the MRHS $\tilde{F}$
Initialize the vector θ_1
for $i = 1, 2, \ldots, n$ **do**
  Compute $K(\theta_i)$
  Solve $K(\theta_i)\tilde{U} = \tilde{F}$
  Solve $K(\theta_i)\tilde{\Lambda} = -Q^T Q \tilde{U}$
  For each faction, compute $\Lambda_p = \tilde{\Lambda} P_p^T P_p$
  Compute for each faction
  $\nabla \phi = \Lambda_p^T \cdot \nabla K(\theta) \tilde{U} = \left((\Lambda_p \tilde{U}^T)_{ii} + (\Lambda_p \tilde{U}^T)_{jj} - (\Lambda_p \tilde{U}^T)_{ij} - (\Lambda_p \tilde{U}^T)_{ji} \right) k_{ij}$
  for Each system s **do**
    for Each faction p **do**
      Activate the affordable lane for which the ratio $\nabla \phi_\eta / L_\eta$ is the greatest.
    end for
  end for
end for

6 Conclusion

In this report, an algorithm was presented, that aims at generating a mesh of safe lanes via the resolution of a topological optimization problem based on the impedance problem.

Annex A : computation of derivative for the Frobenius norm

We develop the computation of $\nabla_u \varphi$

$$\varphi = \frac{1}{2} \sum_{i,l} \sum_{j,k} q_{ij} \tilde{u}_{jk} p_{lk} \tag{21}$$

$$\nabla_u \varphi|_{mn} = \frac{\partial \varphi}{\partial u_{mn}} = \sum_{i,l} q_{im} p_{ln} \sum_{j,k} q_{ij} \tilde{u}_{jk} p_{lk} \tag{22}$$

This leads to :

$$\nabla_u \varphi = Q^T Q \tilde{U} P^T P \tag{23}$$

Annex C : Evaluation of the gradient

We evaluate in this part $\nabla \phi = \Lambda^T \cdot \nabla K(\theta) \tilde{U}$. As $\nabla K(\theta)$ is an order 4 tensor, (because θ is indexed by a multi-index), it is absolutely out of question to assemble it. For that reason, a smarter approach is needed.

We introduce $k_{ij} = \alpha / L_{ij}$, and the Kroneker symbol δ. From (6), one has :

$$\nabla K(\theta_{ij})|_{kl} = \delta_{ik}\delta_{il}k_{ij} + \delta_{jk}\delta_{jl}k_{ij} - \delta_{ik}\delta_{jl}k_{ij} - \delta_{jk}\delta_{il}k_{ij} \tag{24}$$

$$\begin{aligned} \left(K(\theta_{ij})\tilde{U}\right)|_{km} &= \sum_{l} \left(\delta_{ik}\delta_{il}k_{ij} + \delta_{jk}\delta_{jl}k_{ij} - \delta_{ik}\delta_{jl}k_{ij} - \delta_{jk}\delta_{il}k_{ij}\right) u_{lm} \\ &= \delta_{ik}k_{ij}u_{im} + \delta_{jk}k_{ij}u_{jm} - \delta_{ik}k_{ij}u_{jm} - \delta_{jk}k_{ij}u_{im} \end{aligned} \tag{25}$$

$$\begin{aligned} \nabla\phi|_{\eta} = \nabla\phi|_{ij} &= \sum_{k,m} \lambda_{km} \left(\delta_{ik}k_{ij}u_{im} + \delta_{jk}k_{ij}u_{jm} - \delta_{ik}k_{ij}u_{jm} - \delta_{jk}k_{ij}u_{im}\right) \\ &= \sum_{m} \left(\lambda_{im}k_{ij}u_{im} + \lambda_{jm}k_{ij}u_{jm} - \lambda_{im}k_{ij}u_{jm} - \lambda_{jm}k_{ij}u_{im}\right) \\ &= \left((\Lambda\tilde{U}^T)_{ii} + (\Lambda\tilde{U}^T)_{jj} - (\Lambda\tilde{U}^T)_{ij} - (\Lambda\tilde{U}^T)_{ji}\right) k_{ij} \end{aligned} \tag{26}$$