SPOT : Sliced Partial Optimal Transport

Quentin Spinat

We want to find the optimal injective assignment $a : [\![1; m]\!] \to [\![1; n]\!]$ between two point sets X and Y of different sizes $m \leqslant n$.

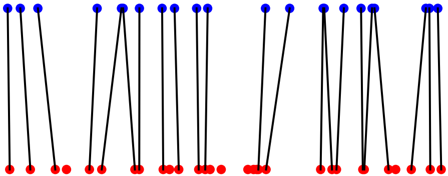


Figure – Partial Optimal Assignment in 1D

- Efficient 1D injective assignment algorithm
- Sliced optimal transport
- Combined with other methods if needed (Iterative Closest Point,...)

- Color Transfer for images with content in different proportion

Figure – Color Transfer

- Point cloud registration for point clouds of different sizes

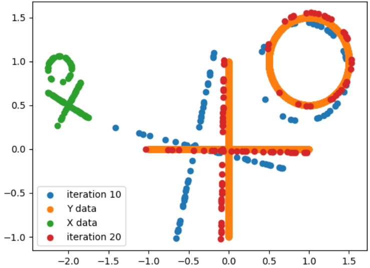


Figure – Point cloud Registration

Partial Transport in 1-D

The core of the method relies on the nearest neighbor assignment t, which is easy to compute : the algorithm consists in scanning X and Y simultaneously from left to right and comparing their value.

Algorithm 1 Nearest Neighbor Assignment

Input : sorted X, Y
Output : t

$i \leftarrow 1$
$j \leftarrow 1$
while $i < m$ **do**
    if $x_i \leqslant y_j$ **then**
        $t[i] \leftarrow j$
        $i \leftarrow i + 1$
    else if $j = n$ **then** :
        $t[i] \leftarrow n$
        $i \leftarrow i + 1$
    else if $y_{j+1} < x_i$ **then**
        $j \leftarrow j + 1$
    else if $|x_i - y_j| < |x_i - y_{j+1}|$ **then** :
        $t[i] \leftarrow j$
        $i \leftarrow i + 1$
    else
        $t[i] \leftarrow j + 1$
        $j \leftarrow j + 1$
        $i \leftarrow i + 1$
return t

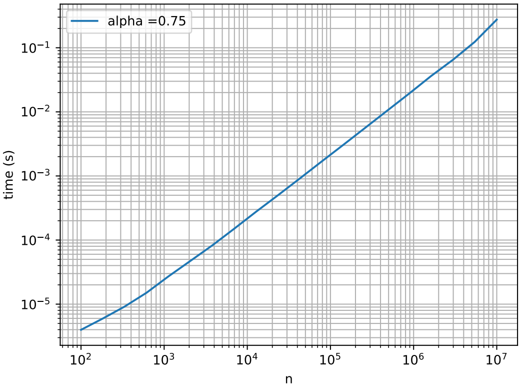


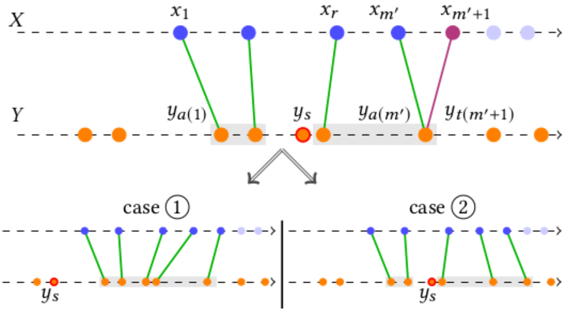
Figure – Mean time over 100 simulations for Nearest Neighbor Assignment with respect to n, $\alpha = \dfrac{m}{b} = 0.75$, logscale

Nearest Neighbor assignment t is not injective

- We need to resolve issues where $t[i] = t[j]$

Method :

- Find $a_{m'}$ optimal injective assignment between $X' = \{x_i\}_{i \in [\![1;m']\!]}$ and Y thanks to t and $a_{m'-1}$, progressively increasing m' from 1 to m.



Algorithm 2 Quadratic Partial Optimal Assignment

Input : sorted X, Y
Output : a

compute t
$a[1] \leftarrow t[1]$
for i from 1 to $m-1$ **do**
    if $t[i+1] > a[i]$ **then**
        $a[i+1] \leftarrow t[i+1]$
        update r
    else
        $w_1 \leftarrow \sum_{k=r}^{i} (x_k - y_{a[k]})^2 + (x_{i+1} - y_{a[i]+1})^2$
        $w_2 \leftarrow \sum_{k=r}^{i} (x_k - y_{a[k]-1})^2 + (x_{i+1} - y_{a[i]})^2$
        if $w_1 \leqslant w_2$ **then** ▷ Case 1
            $a[i+1] \leftarrow a[i] + 1$
        else ▷ Case 2
            $a[i+1] \leftarrow a[i]$
            $a[r:i] \leftarrow a[r:i] - 1$
            update r
return a

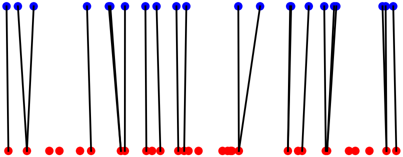


Figure – Nearest Neighbor Assignment

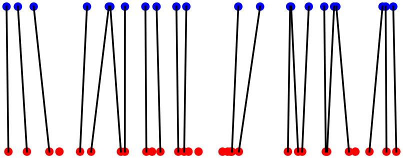


Figure – Partial Optimal Assignment

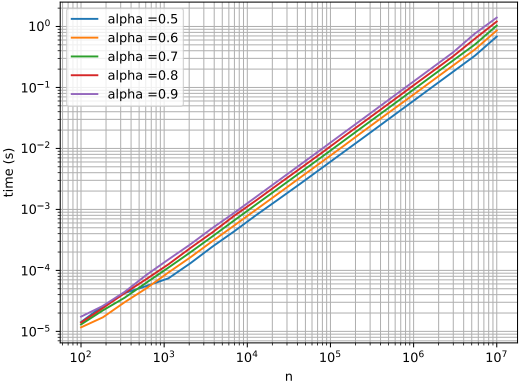


Figure – Mean time over 100 simulations for Quadratic Partial Optimal Assignment with respect to n, for different values of $\alpha = \frac{m}{n}$, logscale

There are easy sub-cases that can accelerate the algorithm :

- If $m = n$ or $m = n - 1$
- If there exists i such that $X[1 : i] \leqslant Y[1 : i]$
- If there exists i such that $Y[i : n] \leqslant X[i : m]$
- If t is injective
- We can decrease the size of Y according to the number of non-injective values of t

Finally we don't need to compute w_1 and w_2 entirely when we choose the case 1

Goal :

- decompose the original problem into many easier subproblems, which can be solved independently (hence in parallel) with the previous partial optimal assignment algorithm.

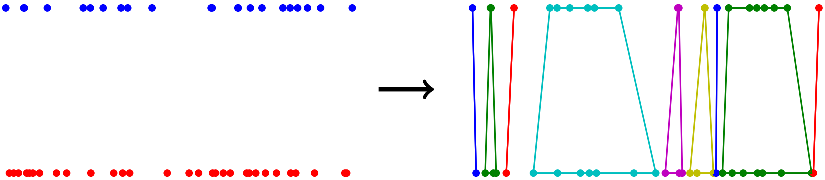


Figure – Assignment problem decomposition

Algorithm 3 Decomposition of the assignment problem

Input : sorted X, Y
Output : A

Compute t
for m' from 1 to m **do**
  if $t[m']$ not considered by any subproblem **then**
    Create new subproblem with $x_{m'}$ and $y_{t[m']}$
  else
    Consider last subproblem $A_{k'}$
    if $t[m'] \neq t[m'-1]$ **then**
      Add $x_{m'}$ to $X_{k'}$ and expand $Y_{k'}$ on the right only
    else
      while first point before $Y_{k'}$ is considered by previous subproblem **do**
        Merge $A_{k'}$ with previous subproblem
      Add $x_{m'}$ to $X_{k'}$ and expand $Y_{k'}$ on the right and on the left
return A

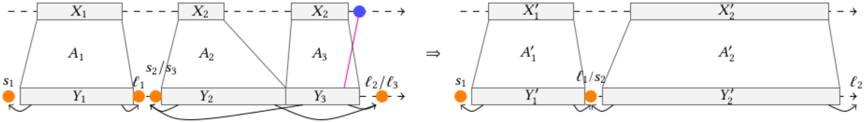


Figure – Problem decomposition : Merging two subproblems to expand on the right and on the left

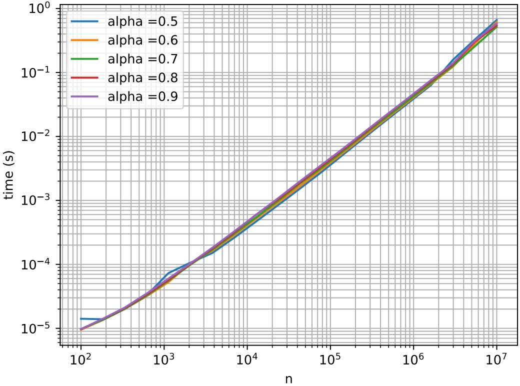


Figure – Mean time over 100 simulations for the decomposition of the assignment problem with respect to n, for different values of $\alpha = \frac{m}{n}$, logscale

Sliced Partial Transport

Goal :

- Use 1-D partial optimal transport to solve d-D partial optimal transport

Instead of directly solving :

$$\min_{\tilde{X}} W_S(\tilde{X}, Y) \tag{1}$$

We do the minimization :

$$\min_{\tilde{X}} \int_{S^{d-1}} W_S(Proj_\omega(\tilde{X}), Proj_\omega(Y))\, d\omega \tag{2}$$

We use a stochastic gradient descent

The gradient of $E_\omega(X_k) = W_S(Proj_\omega(X_k), Proj_\omega(Y))$ is

$$\nabla_X E_\omega(X_k) = Proj_\omega(X_k) - Proj_\omega(Y \circ a) \tag{3}$$

Algorithm 4 Stochastic Gradient Descent

Input : sorted X, Y
Output : X^*

Initialize $X_0 = X$
2: **for** k from 0 to $n_{iter} - 1$ **do**
  Choose ω random direction
4: Compute the optimal Partial assignment a between $Proj_\omega(X_k)$ and $Proj_\omega(Y)$
  Update $X_{k+1} \leftarrow X_k - \eta_k \nabla_X E_\omega(X_k)$
6: **return** $X_{n_{iter}}$

Results : Color Transfer

Goal : Transferring the colors of an image to second image.

Slices Partial Optimal Transport can be useful for color transfer between images with their content not in the same proportion.

A classical color transfer algorithm between an image with a lot of trees and an image with a lot of sky will result in an image where the trees are blue.

Idea :

- Upscale the target image and use sliced partial optimal transport

Figure – Color transfer for different upscaling values

Figure – Color transfer for different upscaling values

Figure – Color transfer for different upscaling values

Results : Point Cloud Registration

- Goal : Match a deformed subset of a point cloud to the original one, and find the transformation.
- Method : Fast Iterative Sliced Transport, a variation of Iterative Closest Point.

Algorithm 5 Fast Iterative Closest Point

Input : sorted X, Y
Output : X^*

Initialize $X_0 = X$
for k from 0 to $n_{iter} - 1$ **do**
    Choose ω random direction
    Compute the optimal Partial assignment a between $Proj_\omega(X_k)$ and $Proj_\omega(Y)$
    Find the best transformation T that transforms X_k into $Y \circ a$ inside the set of allowed transformations
    Update $X_{k+1} \leftarrow T(X_k)$
return $X_{n_{iter}}$

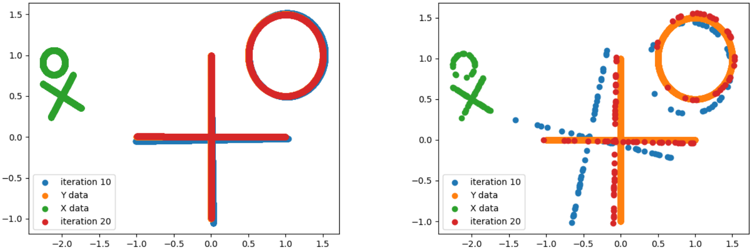


Figure – Point cloud Registration, $n = 10000$, $m_1 = 8000$, $m_1 = 100$

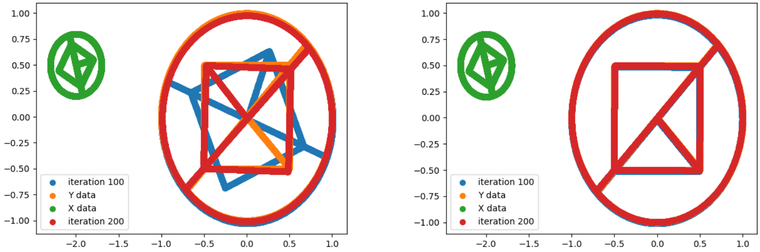


Figure – Point cloud Registration, $n = 10000$, $m = 8000$

Conclusion

In this project I have :

- Made a fast python implementation of Sliced Partial Optimal Transport. This method is fast and efficient in time and memory.
- Implemented a color transfer method that works with images that have content with different proportions.
- Implemented the FIST algorithm for point cloud registration, which avoid the zero convergence problem.

Thank you for your attention !