Frank-Wolfe Variants for Semi-Relaxed Optimal Transport

Alberto Rossetto, Cecile Obeid

Abstract

The main task of this work is solving an *Optimal Transport* (OT) problem applied to the color transfer field. Due to the various issues that can be encountered while solving it, we tackle the relaxation approach of the OT problem by relaxing one of its constraints. We solve it using the *Frank-Wolfe* algorithm and two of its variants: *Block-Coordinate Frank-Wolfe*, and *Block-Coordinate Away Frank-Wolfe*, which is supposed to give even better results. We employ different values of the relaxation parameter and/or the number of iterations in order to improve the baseline results and compare the outputs of the different values used. The assessment was done by analyzing the objective function and by observing the results of the color-transferred images in each case.

1 Introduction

The Optimal Transport problem derives from the *Monge* problem, which is one of the most efficient ways that map two probability distributions, by finding an optimal transport matrix. Of the many formulations for the *Monge* problem, we have the *Kantorivich* formulation, that has two continuous constraints, and which we will be solving. Many algorithms were tested with it, raising issues including high computational cost with large scale problems, slow convergence, dense transport matrix, etc. This has led to the introduction of the relaxation of the constraints by adding regularizers to the objective function.

Our work focuses on the transport problem applied with *Regularized Constraint Relaxation* on one of the two constraints we have, through the algorithm *Frank-Wolfe* and some of its variants such as *Block-Coordinate Frank-Wolfe* and *Block-Coordinate Away Step Frank-Wolfe*, in order to improve the baseline results.

Each of these algorithms is applied to the color transfer task. In this paper, we discuss the results of the different algorithms, compare and interpret them through some experiments done on the images using specific evaluation metrics.

2 Dataset

After implementation, the algorithms need to be tested. Therefore, in order to check the performance of the 3 proposed solutions, it is important to mention that the experiments are done on the two following images, having 3 dimensions of RGB:

- Source Image: *"Gangshan District"* by *Boris Smokrovic*, *Figure 1*
- Reference Image: *"Minesota Landscape Arboretum"* by *Shannon Kunkle*,*Figure 2*

Figure 1: Source Image

Figure 2: Reference Image

2.1 Preprocessing

These source and reference images are loaded as RGB images of shape (427, 640, 3). They are reshaped to become arrays of the shape (273280, 3) in order to get easier process.

2.2 K-Means Algorithm

K-Means algorithm is used for data quantization. It is an unsupervised clustering algorithm that helps group the image's pixels together in a certain number of classes. After executing k-means with a predefined number of classes k, all the pixels of the image are assigned into each class. Averaging all the pixel values yields weight vectors, the centroids.

The m centroids for the source image are stored as vector $x_1, x_2, ..., x_m \in \mathbb{R}^3$, and the n centroids for the reference image as vector $y_1, y_2, ..., y_n \in \mathbb{R}^3$. Additionally, we obtain a color histogram by counting of the assigned pixels for m classes: $a \in \Delta_m$ and $b \in \Delta_n$.

3 Color Transfer Problem

Nowadays, many software are used in images editing, which, by itself, has been really challenging recently. Color transfer problem is about modifying the image's colors. It plays a key role in the media field and has had a lot of attention and improvements in the past years. It may be time-consuming or may have some segmentation issues, for example. In color transfer, we can try to remove some colors or decrease their strongness compared to other ones. An example of its usage is removing the yellow in some shots taken in a very bright and illuminated place. With color transfer we will be able to decrease this domination.

4 Implementation

Our aim from this project is to find the Optimal Transport Matrix $T \in \mathbb{R}^{m \times n}$ between our source and destination probability mass functions a $\in \mathbb{R}^m$ and b $\in \mathbb{R}^n$ in order to reflect the color transfer problem. The color transferred image may depend on many factors such as, in our cases, the relaxation parameter value in all 3 algorithms, number of iterations, the block chosen in the block-coordinate algorithms and many other factors as we will discuss below. All these may contribute to better results in the image and in the convergence of the algorithm.

4.1 Objective Function

The initial *Kantorovich* problem is of the form:

$$\begin{aligned} \min_{T \in U(a,b)} \quad & \langle T, C \rangle \\ \text{s.t.} \quad & U(a,b) = \{T \in \mathbb{R}_+^{m \times n} : T1_n = a, T^T 1_m = b\} \end{aligned} \tag{1}$$

As mentioned previously, some modifications should be done on the domain U(a,b), with a and b being two probabilities, because the mass-conservation constraints might cause dreadful degradation of the performance in an application like color transfer. Therefore, the best way to relax these constraints, in our case, is to use the *Regularized Constraint Relaxation*, which makes our problem of the form:

$$\min_{T \geq 0} \quad \langle T, C \rangle + \frac{1}{2}\phi(T1_n, a) + \frac{1}{2}\phi(T^T 1_m, b) \tag{2}$$

As for the color transfer problem, it would be better if we have close ratios of both source and reference images. Our *Semi-Relaxed Optimal Transport Problem* then becomes:

$$\begin{aligned} \min_{T \geq 0, T^T 1_m = b} \quad & \langle T, C \rangle + \frac{1}{2}\phi(T1_n, a) \\ \text{where} \quad & \phi(x,y) = \frac{1}{2\lambda}||x-y||_2^2 \end{aligned} \tag{3}$$

Which is equivalent to:

$$\min_{T \geq 0, T^T 1_m = b} \quad \{f(T) := \langle T, C \rangle + \frac{1}{2\lambda} ||T1_n - a||_2^2\} \tag{4}$$

Our objective function shown above takes the value of *the relaxation parameter* λ, a and the matrices T and C in order to give an output of the corresponding Objective Function.

4.2 Cost Function

We can find the cost matrix's elements C $\in \mathbb{R}_+^{m \times n}$ from the values calculated as $C_{i,j} = ||x_i - y_j||_2$, where x_i and y_j represent the centroids of the source and reference images.

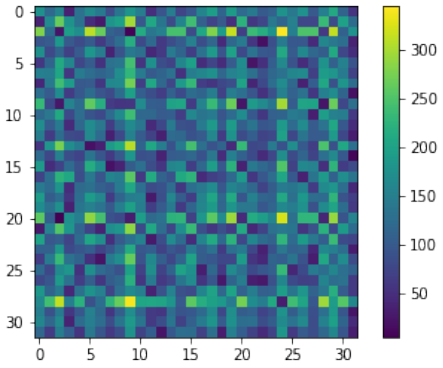


Figure 3: Heatmap of the Cost Matrix

4.3 Transport Matrix

Our goal from this problem is to find the optimal transport matrix $T \in \mathbb{R}_+^{m \times n}$. Therefore, we need to find the total minimum transport cost. As mentioned, we start with $T_0 \in \mathbb{R}_+^{m \times n}$ that is all zeros, except for the first row which is initialized to the vector b.

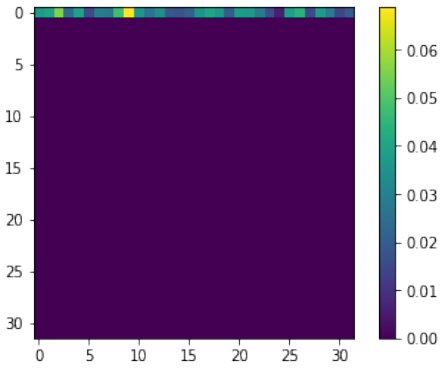


Figure 4: Heatmap of the Transport Matrix T_0

4.4 Solutions

4.4.1 Frank-Wolfe

The Frank Wolfe (FW) algorithm is a constraint convex optimization method. It is known to be a linear approximation algorithm that uses conditional gradient. At every iteration, the feasible point s is found first by minimizing the linearization of the objective function f over the convex feasible set

Algorithm A.1 Frank-Wolfe (FW) algorithm for semi-relaxed OT

Require: $\mathbf{T}^{(0)} \in b_1\Delta_m \times \cdots \times b_n\Delta_m$

for $k = 0 \dots K$ **do**
  for $i = 0 \dots n$ **do**
    Compute $s_i = b_i \underset{e_k \in \Delta_m, k \in [m]}{\arg\min} \langle e_k, \nabla_i f(\mathbf{T}^{(k)}) \rangle$
  end for
  if $g(\mathbf{T}^{(k)}) \leq \epsilon$ **then**
    break
  end if
  Compute stepsize γ as

$$\gamma = \begin{cases} \gamma_{\text{LS}}, & (\text{line} - \text{search in } (A.1)) \\ \dfrac{2}{k+2}, & (\text{decay rule}) \end{cases}$$

  Update $\mathbf{T}^{(k+1)} = (1-\gamma)\mathbf{T}^{(k)} + \gamma\mathbf{S}$
end for

$\mathcal{M}$. This is where the linear minimization oracle LMO_A is used. To find the feasible point s, we solve the following subproblem:

$$s = arg \min_{s' \in \mathcal{M}} \langle s', \nabla f(x^{(k)}) \rangle \tag{5}$$

In the equation, $x^{(k)}$ represents the k-th current point. Because the domain $\mathcal{M}$ is the convex set and the objective function is linear for s. Finally, the next iterate $x^{(k+1)}$ is obtained by a convex combination as $x^{(k+1)} = (1-\gamma)x^{(k)} + \gamma s$, where γ is the stepsize. Regarding the stepsize calculation, we consider the *Decay Rule* where $\gamma = \frac{2}{(k+2)}$ or the *Exact Line Search* where γ is computed as follows:

$$\gamma_{LS} = \frac{\lambda\langle T^{(k)} - S, C\rangle_F + \langle T^{(k)}1_n - S1_n, T^{(k)}1_n - a\rangle}{||T^{(k)}1_n - S1_n||^2} \tag{6}$$

In our problem the gradient $\nabla f(T) \in \mathbb{R}^{m \times n}$ is given as

$$\nabla f(T) = \begin{bmatrix} c_1 \\ \vdots \\ c_i \\ \vdots \\ c_n \end{bmatrix} + \frac{1}{\lambda} \begin{bmatrix} T1_n - a \\ \vdots \\ T1_n - a \\ \vdots \\ T1_n - a \end{bmatrix}$$

where the $\nabla f_i(T) = c_i + \frac{1}{\lambda}(T1_n - a) \in \mathbb{R}^m$
The subproblem (5) is equivalent to

$$s_i = b_i e_j = b_i arg \min_{e_k \in \Delta_m, k \in [m]} \langle e_k, \nabla_i f(T^{(k)}) \rangle \tag{7}$$

where $j \in [m]$ and e_j is the extreme point on the probability simplex. In other words we just find the index of the minimal elements of the gradient of the variable blocks.

4.4.2 Block-Coordinate Frank-Wolfe

On large-scale problems, the Frank-Wolfe is costly and takes more time to solve the minimization problem. If the domain $\mathcal{M}$ can be block-separable as a cartesian product $\mathcal{M} = \mathcal{M}^{(1)} \times \mathcal{M}^{(2)} \times ... \times \mathcal{M}^{(n)} \subset \mathbb{R}^m$, then we can perform a single cheaper iteration of only $\mathcal{M}^{(i)}$. We assume that each block $\mathcal{M}^{(i)}$ is convex. At each iteration the algorithm selects randomly a single column of T and uses it for solving the subproblem identical to (7). Then, all the other columns of T remain the same. The

stepsize is computed in a similar way of Frank-Wolfe. We consider the *Decay Rule* where $\gamma = \frac{2n}{(k+2n)}$ and the *Exact Line Search* where γ is computed as follows:

$$\gamma_{LS} = \frac{\lambda\langle t_i^{(k)} - s_i, c_i\rangle + \langle t_i^{(k)} - s_i, T^{(k)}1_n - a\rangle}{||t_i^{(k)} - s_i||^2} \tag{8}$$

Algorithm 1 Block-coordinate Frank-Wolfe (BCFW) for semi-relaxed OT

Require: $\mathbf{T}^{(0)} = (\boldsymbol{t}_1^{(0)}, \dots, \boldsymbol{t}_n^{(0)}) \in b_1\Delta_m \times \cdots \times b_n\Delta_m$

1: **for** $k = 0 \dots K$ **do**

2: Select index $i \in [n]$ randomly

3: Compute $\boldsymbol{s}_i = b_i \underset{e_k \in \Delta_m, k \in [m]}{\arg\min} \langle \boldsymbol{e}_k, \nabla_i f(\mathbf{T}^{(k)})\rangle$

4: Compute stepsize γ as

$$\gamma = \begin{cases} \gamma_{\text{LS}}, & (\text{line} - \text{search in } (8)) \\ \dfrac{2n}{k+2n}, & (\text{decay rule}) \end{cases}$$

5: Update $\boldsymbol{t}_j^{(k+1)}$ $\forall j \in [n]$ as

$$\boldsymbol{t}_j^{(k+1)} = \begin{cases} \boldsymbol{t}_j^{(k)}, & (\text{for } j \neq i) \\ (1-\gamma)\boldsymbol{t}_i^k + \gamma \boldsymbol{s}_i, & (\text{otherwise}) \end{cases}$$

6: **end for**

4.4.3 Block-Coordinate Away-Step Frank-Wolfe

In order to improve the convergence rate of the Frank-Wolfe algorithm, a variant is being investigated. In particular we focus on a modification of the original algorithm that takes into account a strategy that replaces the Frank-Wolfe direction with a different direction called Away-Step. The Block-Coordinate Frank-Wolfe algorithm generates a convex combination point from atoms in each variable block. this can lead to selection of non-desirable atoms, which causes the sublinear convergence rates of the Frank-Wolfe. In order to avoid this shortcoming, the Block-Coordinate Away-Step Frank-Wolfe algorithm removes unnecessary atoms from an active set on each variable block. Let S_i be the active set on the i-th ($i \in [n]$) variable block, which is defined as:

$$S_i = \{e_j \in \Delta_m : \alpha_{e_j} > 0, j \in [n]\} \tag{9}$$

where α_{e_j} is the coefficient of the j-th extreme point e_j.

We consider a new subproblem in order to remove the atoms

$$v_i = b_i e_j = b_i arg \max_{v' \in S_i} \langle v', \nabla_i f(T^{(k)})\rangle \tag{10}$$

Defining two directions, the FW direction $d_{FW} = s_i - t_i^k$ and the Away Direction $d_{AW} = t_i^k - v_i$, the procedure selects the one that reduces the objective function value more. The stepsize is then calculated as:

$$\gamma_{LS} = \frac{\lambda\langle d, c_i\rangle + \langle d, T^{(k)}1_n - a\rangle}{||d||^2} \tag{11}$$

where d can be d_{FW} or d_{AW}. Then we update the column vector t_i^k and the active set S_i^k.

5 Experiments

Many experiments are carried out using the algorithms discussed above, where we choose $m = n = 32$. We compare the algorithms having the same number of iterations and value of relaxation parameter,

Algorithm A.2 Block-Coordinate Away-step Frank-Wolfe (BCAFW) for semi-relaxed OT

Require: $\mathbf{T}^{(0)} = (\boldsymbol{t}_1^{(0)}, \boldsymbol{t}_2^{(0)}, \dots, \boldsymbol{t}_n^{(0)}) \in b_1\Delta_m \times b_2\Delta_m \times \cdots \times b_n\Delta_m$ $\mathcal{S}_i = \{\boldsymbol{e}_1\}, \forall i \in [n]$.

for $k = 0 \dots K$ **do**
  Select index $i \in [n]$ randomly
  Compute $\boldsymbol{s}_i$ ▷ Line 2 in **Algorithm 1**
  Set $\boldsymbol{d}_{\text{FW}} = \boldsymbol{s}_i - \boldsymbol{t}_i^{(k)}$
  Compute $\boldsymbol{v}_i = b_i \underset{\boldsymbol{e}_j \in \mathcal{S}_i, j \in [m]}{\arg\max} \langle \boldsymbol{e}, \nabla_i f(\mathbf{T}^{(k)}) \rangle$
  Set $\boldsymbol{d}_{\text{Away}} = \boldsymbol{t}_i^{(k)} - \boldsymbol{v}_i$
  if $\langle -\nabla f(\mathbf{T}^{(k)}), \boldsymbol{d}_{\text{FW}} \rangle \geq \langle -\nabla f(\mathbf{T}^{(k)}), \boldsymbol{d}_{\text{Away}} \rangle$ **then**
    $\boldsymbol{d} = \boldsymbol{d}_{\text{FW}}, \gamma_{\max} = 1$
  else
    $\boldsymbol{d} = \boldsymbol{d}_{\text{Away}}, \gamma_{\max} = \dfrac{\alpha_{\boldsymbol{v}_i}}{1 - \alpha_{\boldsymbol{v}_i}}$
  end if
  Compute stepsize $\gamma = \gamma_{\text{LS}}$ using $\boldsymbol{d}$ ▷ (10)
  Update $\boldsymbol{t}_j^{(k+1)} \quad \forall j \in [n]$ ▷ Line 5 in **Algorithm 1**
  Update $\mathcal{S}_i^{(k+1)} = \{\boldsymbol{e} \in \Delta_m : \alpha_{\boldsymbol{e}}^{(t+1)} > 0\}$
end for

as well as evaluate the performance of each algorithm alone when we change these values. We also try to understand the impact of the relaxation parameter λ on the color transfer problem.

5.1 Frank-Wolfe

With the *Frank-Wolfe* algorithm, we first try the color-transfer problem on the source image, with different combinations of λ and iterations number. We used decay stepsize rule.

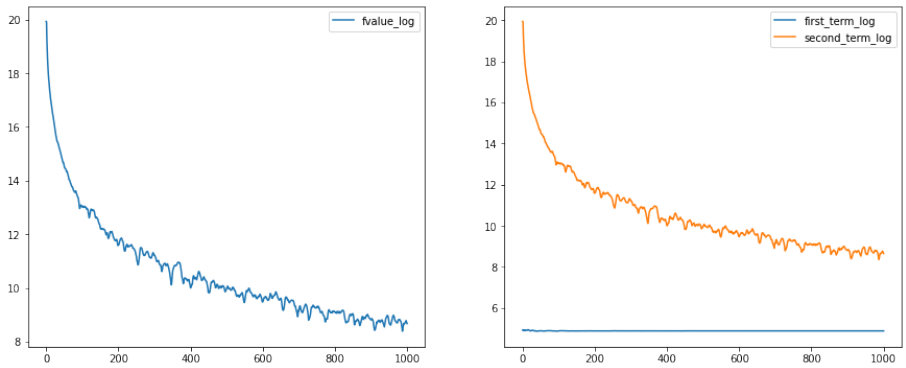
As a result of the experiments carried out, we can see in *Figures (6)* and *(7)* that we have naturally color-transferred images produced and only a single color that is scattered when we use $K = 10^5$ and $\lambda = 10^{-9}$ or $K = 1000$ and $\lambda = 10^{-6}$. On the other hand, also with $K = 1000$ but with a smaller $\lambda = 10^{-9}$, we have an image in *Figure (5)* that displays nearly one single brown color, which represents the b-weighted mixed color of the 32 reference colors. As for the result of *Figure (8)* which shows the results of $K = 10^4$ and $\lambda = 10^{-6}$, the image produced contains artificial gray or blue pixels in the background. The vertical structure in the heatmap is the most obvious in the cases shown in *Figure (5)*. However, this vertical structure disappears as the iteration proceeds. This structure is less strong in the other cases and disappears completely in the case of *Figure (8)* where there is no pattern in the heatmap.

5.2 Block-Coordinate Frank-Wolfe

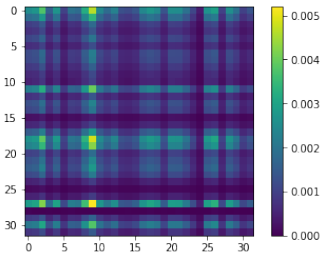
From the experiments using the Block-Coordinate Frank Wolfe, we can notice that the results obtained are similar to the ones using Frank Wolfe. In particular using a value of $\lambda = 10^{-9}$, the optimization process needs many iteration in order to produce a naturally color-transferred image (*Figure (10)*). On the other hand, when $\lambda = 10^{-6}$, the obtained image is quite satisfactory when the number of iteration is K=1000 but when the process proceeds further, it can be seen that the image contains artificial grey (*Figure (12)*).

5.3 Block-Coordinate Away Frank-Wolfe

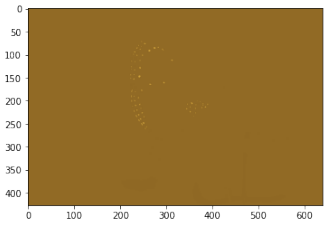
For what concerns the experiments done with the Block-Coordinate Away Frank-Wolfe, we decided to perform only a test using $K = 1000$ and $\lambda = 10^{-6}$. The results are shown in *Figure (13)*



(a) Objective Function and Terms of Objective Function

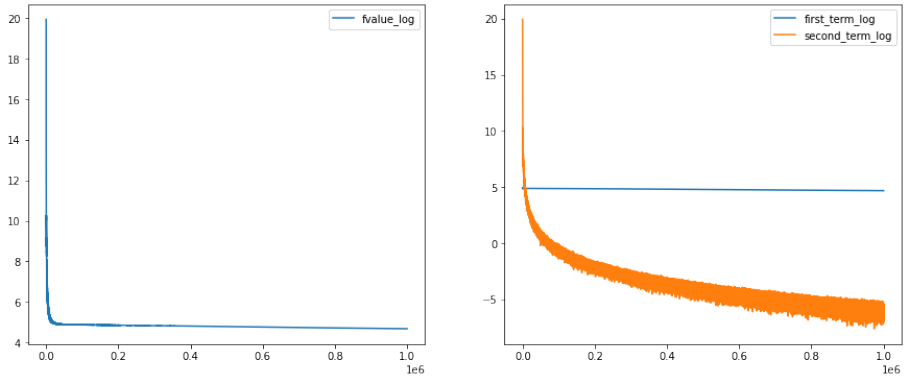


(b) Heatmap

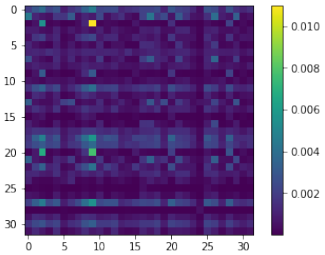


(c) Result Image

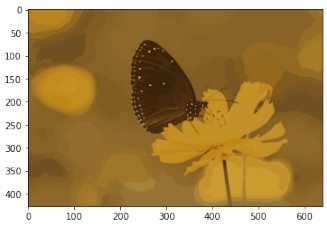
Figure 5: Frank-Wolfe with K=1000 and $\lambda = 10^{-9}$



(a) Objective Function and Terms of Objective Function

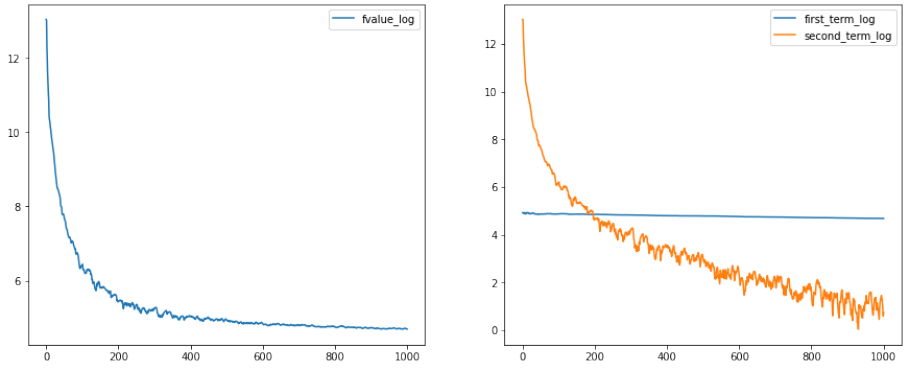


(b) Heatmap

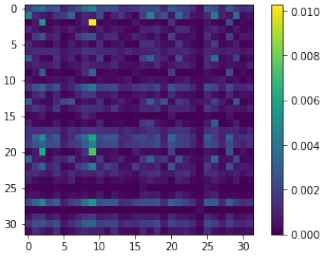


(c) Result Image

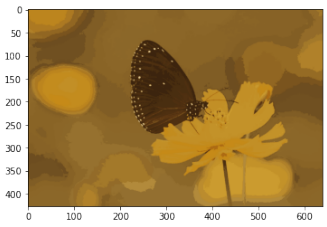
Figure 6: Frank-Wolfe with $K = 10^5$ and $\lambda = 10^{-9}$



(a) Objective Function and Terms of Objective Function

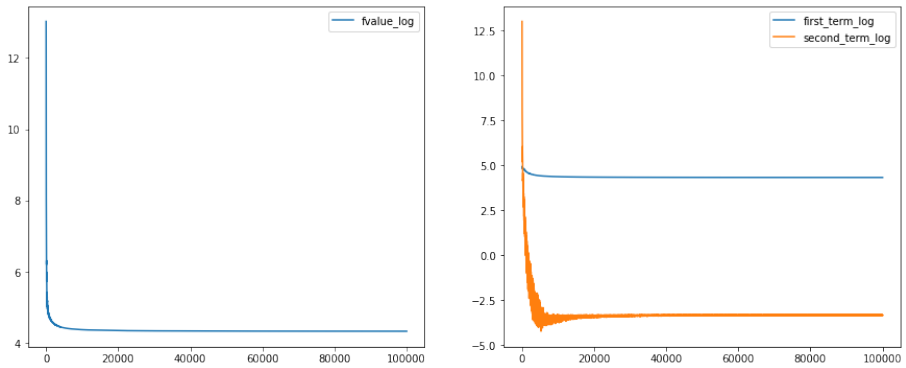


(b) Heatmap

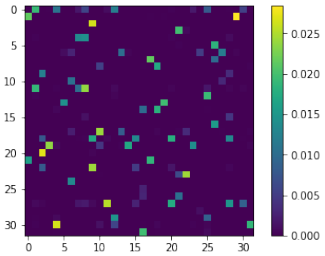


(c) Result Image

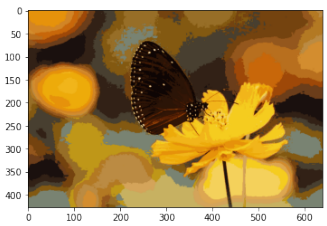
Figure 7: Frank-Wolfe with K=1000 and $\lambda = 10^{-6}$



(a) Objective Function and Terms of Objective Function

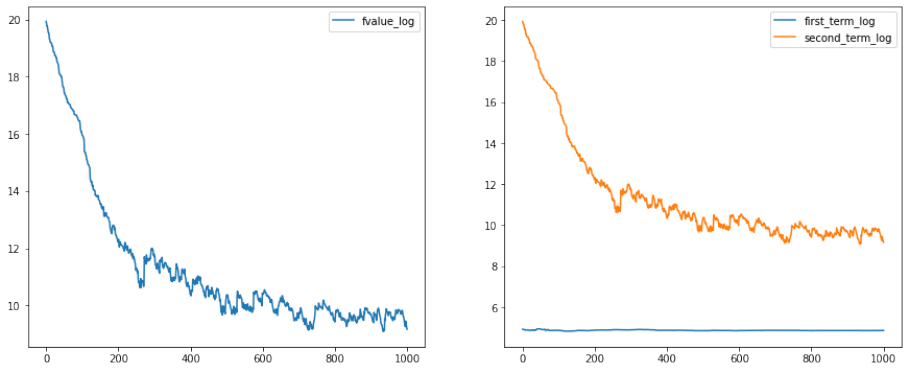


(b) Heatmap

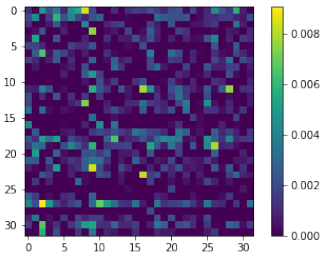


(c) Result Image

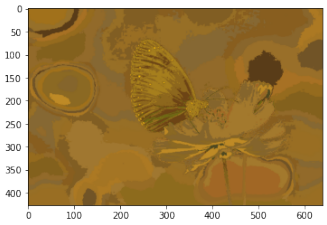
Figure 8: Frank-Wolfe with $K = 10^4$ and $\lambda = 10^{-6}$



(a) Objective Function and Terms of Objective Function

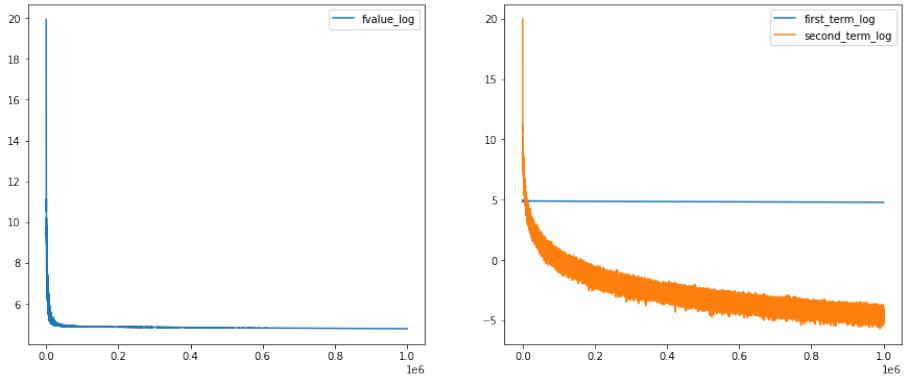


(b) Heatmap

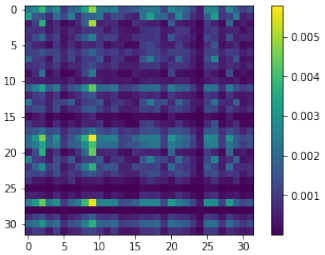


(c) Result Image

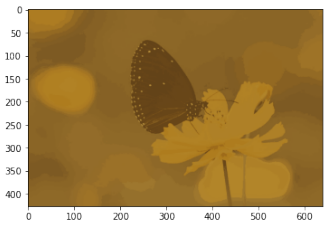
Figure 9: Block-Coordinate Frank-Wolfe with $K = 1000$ and $\lambda = 10^{-9}$



(a) Objective Function and Terms of Objective Function

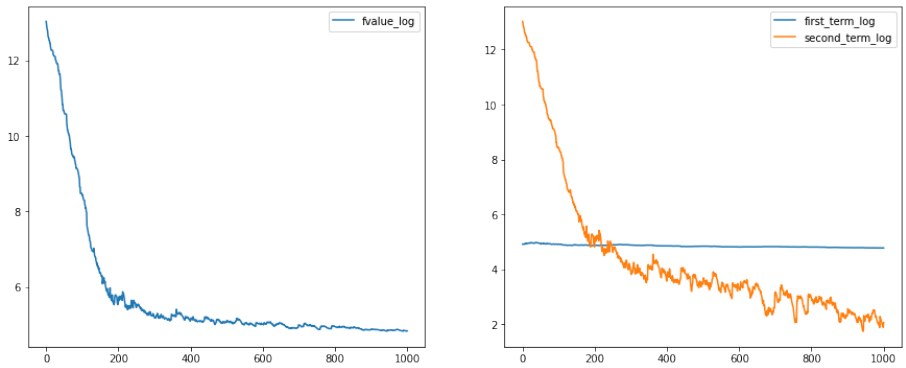


(b) Heatmap

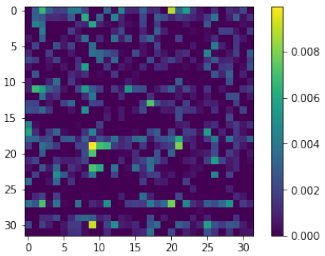


(c) Result Image

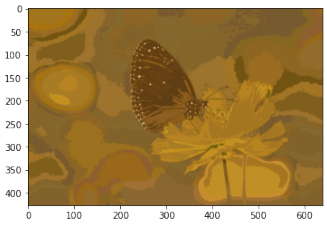
Figure 10: Block-Coordinate Frank-Wolfe with $K = 10^5$ and $\lambda = 10^{-9}$



(a) Objective Function and Terms of Objective Function

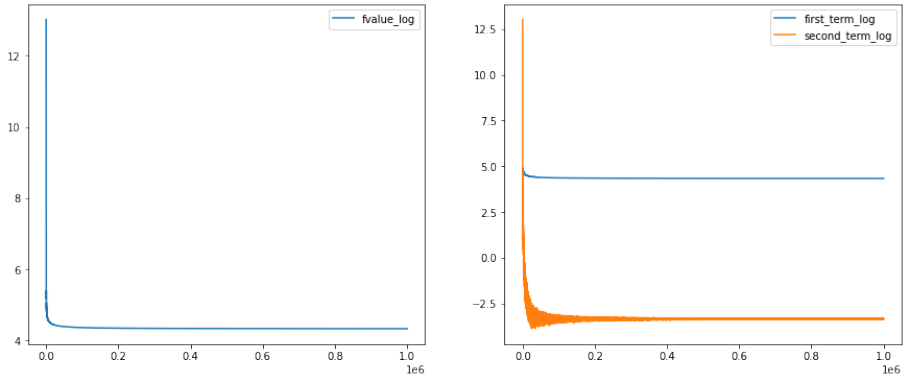


(b) Heatmap

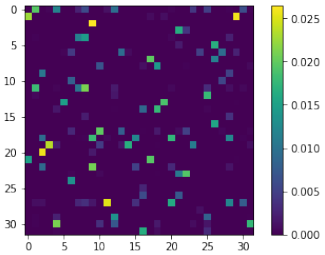


(c) Result Image

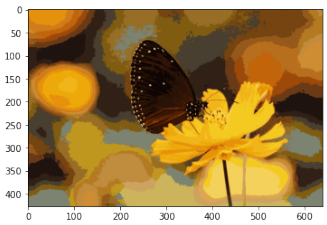
Figure 11: Block-Coordinate Frank-Wolfe with K=1000 and $\lambda = 10^{-6}$



(a) Objective Function and Terms of Objective Function

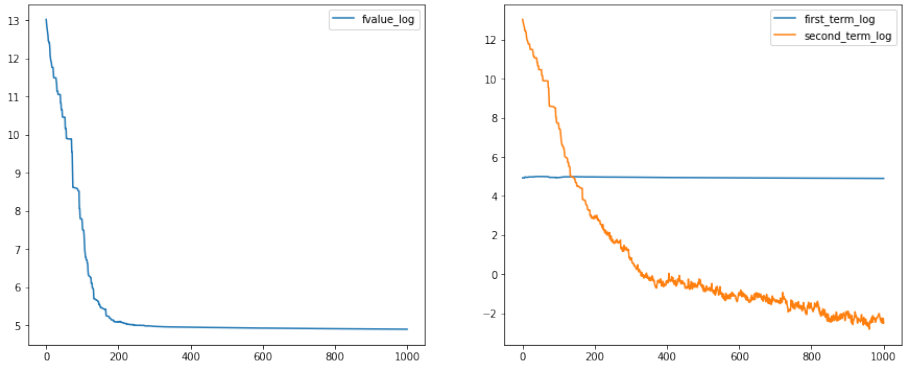


(b) Heatmap

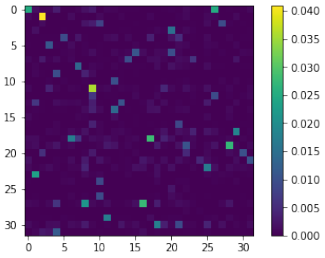


(c) Result Image

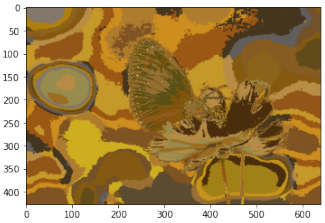
Figure 12: Block-Coordinate Frank-Wolfe with $K = 10^5$ and $\lambda = 10^{-6}$



(a) Objective Function and Terms of Objective Function



(b) Heatmap



(c) Result Image

Figure 13: Block-Coordinate Away Frank-Wolfe with $K = 1000$ and $\lambda = 10^{-6}$

5.4 Comparison Between Frank-Wolfe and Block-Coordinate Frank-Wolfe

After the experiments, we want to analyze which algorithm performs better, when having the same number of iterations. As per the *Figures (14))* and *(15)*, when λ is the same, either big or small, the Block- Coordinate Frank-Wolfe performs generally better in terms of decrease of the objective function. It is clear that Block-Coordinate Frank-Wolfe requires less computational time. As in terms of number of iterations needed, the two algorithms show the same behaviour.

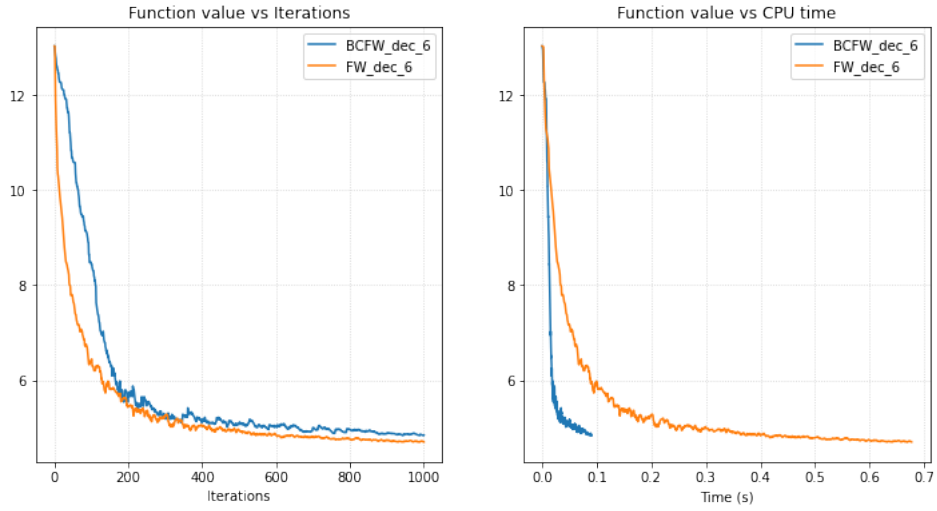


Figure 14: FW and BCFW Iterations and CPU Times, K=1000 $\lambda = 10^{-6}$

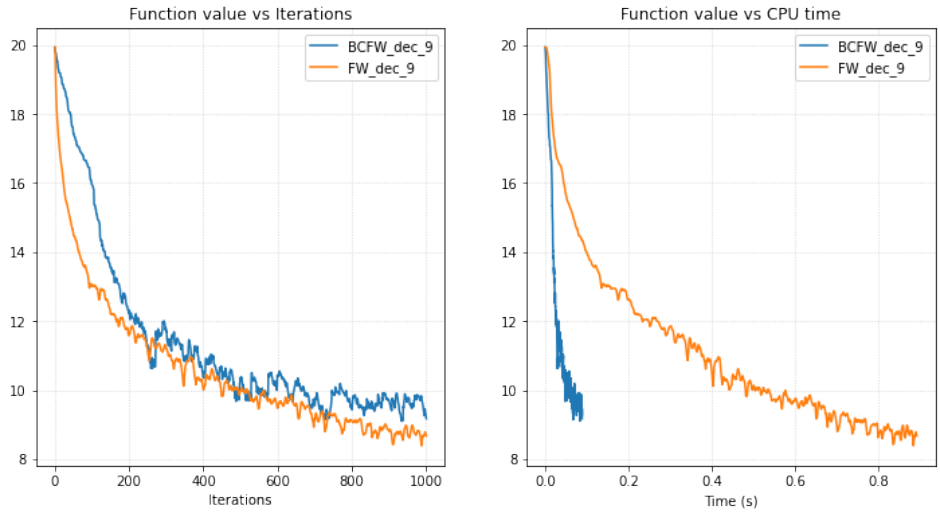


Figure 15: FW and BCFW Iterations and CPU Times, K=1000 $\lambda = 10^{-9}$

5.5 Comparison of Block-Coordinate Frank-Wolfe with different λ

When the relaxation parameter is large ($\lambda = 10^{-6}$), the Block-Coordinate Frank-Wolfe converges faster, but the color-transferred image is not satisfactory. This is due to the fact that artificial grey is produced *(Figure (12))*.

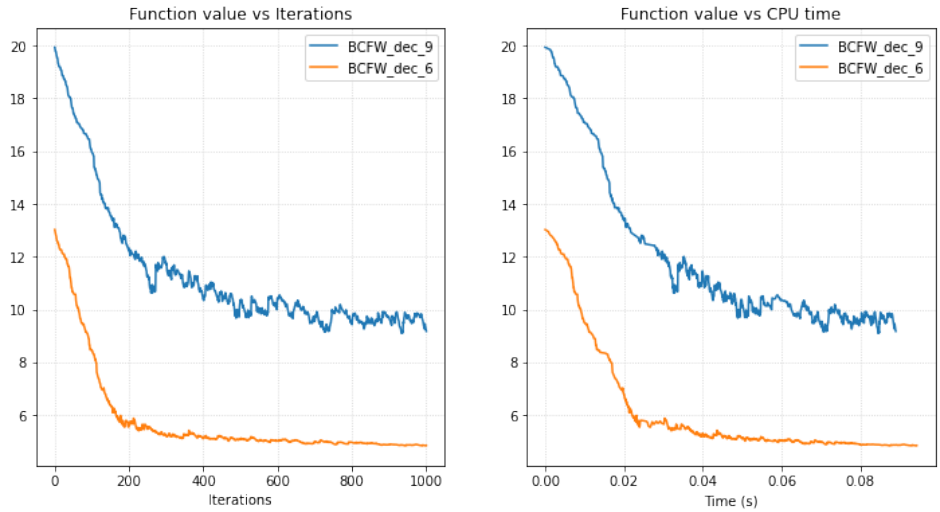


Figure 16: Block-Coordinate Frank-Wolfe with $\lambda = 10^{-6}$ and $\lambda = 10^{-9}$

5.6 Comparison between Block-Coordinate Frank-Wolfe and Block-Coordinate Away-Step Frank-Wolfe

Comparing the performance of the Block-Coordinate with Away Steps with the Block-Coordinate Frank-Wolfe, we can notice that the objective function decreases faster in terms of iterations and CPU time in *(Figure (17))*.

6 Conclusion

We tested in this paper the Frank-Wolfe algorithm, with its two variants Block-Coordinate Frank-Wolfe and Block-Coordinate Away-Step Frank-Wolfe applied to a convex semi-relaxed optimal trans-

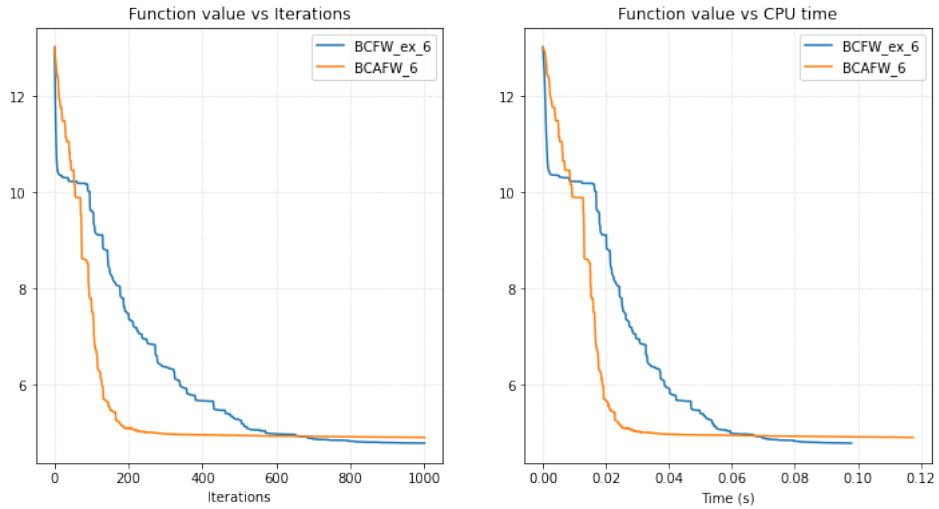


Figure 17: Block-Coordinate Frank-Wolfe vs Block-Coordinate Away Frank-Wolfe

port problem. We compared different combinations of parameters. After comparing the algorithms with different combinations of the parameters used, the tests showed that the global behaviour of the *Block-Coordinate Frank-Wolfe* algorithm reflects what was expected: it is faster than the *Frank-Wolfe* algorithm. On the other hand, the *Block-Coordinate Away-Step Frank-Wolfe* shows performance improvements that was theoretically supposed.