Variants of the Frank-Wolfe Algorithm for Optimal Transport
Optimization for Data Science

Federico Bottarelli
federico.bottarelli@studenti.unipd.it

Jan Elfes
jan.elfes@studenti.unipd.it

Ivan Dragomirov Padezhki
ivandragomirov.padezhki@studenti.unipd.it

14 June 2022

1 Introduction

The first projection-free method was invented by Marguerite Straus-Frank and Philip Wolfe 66 years ago. Now known as the Frank-Wolfe (FW) method, it is an iterative first-order optimization algorithm for constrained convex optimization. The algorithm has recently been subject to interest from researchers because many modern problems can be traced back to optimization problems over convex hulls of an atomic set. In addition, its good sparsity proprieties prove to be useful in many applications.

We begin by summarizing the main concepts present in three relevant papers, and following the example of Fukunaga & Kasai, 2021, we implement three algorithms to resolve an optimal transport problem, specifically about color transfer between images.

1.1 Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization

The first paper to be considered is Jaggi, 2013 which concentrates on extending the original Frank-Wolfe algorithm in the context of more recent problems. Additionally, the efficacy of the algorithm is analyzed and its wide applications are discussed. The novelty of the paper is proving duality gap convergence certificates and the affine invariance of the algorithm.

The Frank-Wolfe algorithm (abbreviated as FW), else known as "conditional gradient method", is applied to constrained convex optimization prob-

lems, which aim to minimize a continuously differentiable convex function over a compact convex set. The set is a combination of atoms and also the iterates of the FW algorithm remain in this set. According to the author, this produces sparse solutions, which is an additional strength of the algorithm when comparing it to other methods, such as Projection Methods. The latter are said to be more expensive since they do not utilize the linear approximation of the function.

Assuming the objective function is convex, the paper proves that the duality gap, formulated as $g(x) := \max_{x \in \mathcal{D}} \langle x - s, \nabla f(x) \rangle$, can be used as a certificate for the approximation quality. In the above definition, x is a feasible point in the domain and $x - s$ represents a feasible direction. In addition, monitoring the gap is a useful condition for stopping the algorithm.

Necessary concepts for the convergence of the Frank-Wolfe algorithm and its variants include the approximation of the linear sub-problems (with an arbitrary quality), the line-search of the step-size (which can replace the decaying step-size defined as $\gamma = 2/(k+2)$ with k as the current iteration), and the curvature of the function. The latter measures the non-linearity of f over the given compact domain and is related to the Lipschitz continuous gradient.

For an Away-Step Frank-Wolfe (from now on AFW), it is relevant to note that in each iteration, an atom may get removed from the active set leading to potentially sparser iterates and faster linear convergence (provided a specific class of problems). Furthermore, the Fully Corrective Frank-Wolfe (FCFW) variant uses a re-optimization of the objective function over the set of atoms. While it can lead to sparser solutions, this version can also become expensive due to this additional operation.

The algorithms described in the paper are proven to have primal convergence of the form:

$$f(x^{(k)}) - f(x^*) \leq \frac{2C_f}{k+2}(1+\delta)$$

The above primal convergence is however hard to compute since the optimal value and the curvature constant are often unavailable. Therefore, the authors provide an upper bound for the dual gap for the case of $K \geq 2$ iterations and for the specific iterate $x^{(\hat{k})}$:

$$g(x^{(\hat{k})}) \leq \frac{2\beta C_f}{K+2}(1+\delta)$$

where $\beta = 3.375$ and $\delta \geq 0$ represents the accuracy of the solutions of the linear sub-problems.

With these results in mind, the author provides proof that the convergence remains valid under affine transformations of the domain since the curvature constant is invariant. Furthermore, the solutions obtained from Frank-Wolfe algorithms are characterized by their sparsity and approximation quality.

Possible applications include optimization over atomic norms (the domain is the convex hull of another set) such as sparse vectors or matrices. For many specific cases of domains, there are complexity bounds given in the paper. One of

the reasons why FW is used more and more is that in various machine learning applications it is possible to structure the problems in terms of the convex hull of a set. In many atomic norms, an exact Frank-Wolfe iteration can have complexity in the order of $O(n)$. The author also shows that the same convergence can be reached by using an inexact gradient.

1.2 On the Global Linear Convergence of Frank-Wolfe Optimization Variants

The algorithms from the first paper are discussed in more detail by Lacoste-Julien & Jaggi, 2015, in particular the away-steps, pairwise and fully-corrective Frank-Wolfe, and the minimum norm point algorithms. Similar to previous analyses, the paper proves global linear convergence for the case of a strongly convex objective function. Building on top of this, linear convergence is shown for functions that are not strongly convex using the condition number of the function and a condition number of the constraint set played by a geometric quantity called the pyramidal width. The objective function is minimized over the convex hull of a finite set of atoms, so over a polytope which is convex and bounded.

$$\min_{\boldsymbol{x}\in\mathcal{M}} f(\boldsymbol{x}), \quad \mathcal{M} = \text{conv}(\mathcal{A}), \qquad \text{with only access to:} \qquad \text{LMO}_{\mathcal{A}}(\boldsymbol{r}) \in \arg\min_{\boldsymbol{x}\in\mathcal{A}} \langle \boldsymbol{r}, \boldsymbol{x} \rangle$$

$\mathcal{A} \subseteq \mathbb{R}^d$ is a finite set of vectors called atoms, which can represent combinatorial objects of an arbitrary type. $\mathcal{M}$ is a (convex and bounded) polytope. The linear minimization oracle $\text{LMO}_{\mathcal{A}}(.)$ is associated with the domain $\mathcal{M}$ through a generating set of atoms $\mathcal{A}$. The variable r represents a direction in this paper.

The description of the algorithms begins with the original formulation of the FW algorithm, which suffers from a zig-zagging phenomenon if the optimal solution lies on a boundary of the polytope. This can be resolve by the AFW method which monitors the active set of atoms $\mathcal{S}^{(k)} \subseteq \mathcal{A}$ for each iteration k. This allows to reduce the influence of or fully remove a "bad" atom (high gradient).

Another way of dealing with the zig-zagging behavior of the base FW algorithm is adding pairwise steps (pairwise Frank-Wolfe, or PFW). Instead of removing an atom or adding a new one at each step, weight is moved from the old to thew new one. If all the weight is moved to the new atom, this is called a swap-step. Due to the unspecified number of problematic swap-steps which can occur, the convergence rate for the PFW is less tight compared to other variants.

Lastly, one can introduce another change to the algorithm and optimize the objective function at every iteration over the convex hull of the set of atoms and thus obtain the new iterate. This is especially useful if each gradient step is costly compared to this optimization of the objective function. Instead of tracking the active set $\mathcal{S}^{(k)}$ the algorithm keeps a larger one $\mathcal{A}^{(k)}$ that defines a correction polytope and then at each iteration re-optimizes the objective function f over this polytope defined by $\text{conv}\left(\mathcal{A}^{(k)}\right)$. Based on how the polytope

correction is defined and maintained there can be many variations of the algorithm. The general name for this type of algorithm is Fully-Corrective Frank-Wolfe, with a subtype known as the Min-Norm Point Algorithm (MNP). The FCFW does not suffer from drop steps.

1.2.1 Global Linear Convergence Analysis

For all of the above algorithms, the authors provide a novel proof of global linear convergence that does not rely on as many constants as past research papers but rather on a constant that not only is affine invariant but also a combination of the complexity of the domain and the curvature of the objective function. This inverse of the rate constant ρ is defined as $\rho := \frac{\mu}{4L}(\frac{\delta}{M})^2$ combines the condition number of the objective function and the condition number of the domain. The constant $\delta = \text{PWidth}(\mathcal{A})$ is defined as the smallest pyramidal width of all of the faces of the set and M is the diameter of the set. With the definitions above and the suboptimality of the iterates defined as $h_t := f\left(\boldsymbol{x}^{(t)}\right) - f\left(\boldsymbol{x}^*\right)$, the authors show that h_t decreases as

$$h_{t+1} \leq (1-\rho)h_t$$

in the case of a good step for the AFW and PFW (a good step is $\gamma_t < \gamma_{max}$). Writing the number of good steps as $k(t)$, all algorithms have a global linear convergence rate of $h_t \leq h_0\, exp(-\rho k(t))$.

With this in mind, the FW duality gap g_t^{FW} is also proven to converge linearly and is upper bounded by the primal error h_t so that it becomes

$$g_t^{FW} \leq h_t + \frac{LM^2}{2} \text{ when } h_t > \frac{LM^2}{2},\ and\ g_t^{FW} \leq M\sqrt{2h_tL} \text{ otherwise.}$$

This global linear convergence result can be generalized in the case where $f(\boldsymbol{x}) := g(\boldsymbol{A}\boldsymbol{x}) + \langle \boldsymbol{b}, \boldsymbol{x}\rangle$, for $\boldsymbol{A} \in \mathbb{R}^{p\times d}, \boldsymbol{b} \in \mathbb{R}^d$ where g is μ_g-strongly convex and continuously differentiable over $\boldsymbol{AM}$ In this case, f is not necessarily strongly convex anymore at the cost of changing the parameter μ in ρ with the generalized strong convexity constant for f .

1.2.2 Results

To conclude, the authors show some numerical results of the algorithm variants applied to two cases: In a constrained Lasso problem in which the objective is $\min_{\boldsymbol{x}\in\mathcal{M}} f(\boldsymbol{x}) = \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|^2$ with $\mathcal{M} = 20 \cdot L_1$ a scaled L_1-ball. The second example is video-localization formulated as a quadratic program over a flow polytope. In both cases, the FW variants outperform the original FW algorithm and the linear convergence is visible.

1.3 Fast Block-Coordinate Frank-Wolfe Algorithm for Semi-relaxed Optimal Transport

The final paper concerns an efficient implementation of Frank-Wolfe's algorithm, in particular in the case of the optimal transport (OT) problem, formalized by

Gaspard Monge in 1781. Given two distributions of mass equal to $a(x), b(x)$, the goal is to find the transport matrix that minimizes the total cost of transport.

Having two sets of features X and Y both in a d-dimensional space with distributions

$$\boldsymbol{\nu} = \sum_{i=1}^{m} a_i \boldsymbol{\delta}_{x_i}, \quad \boldsymbol{\mu} = \sum_{i=1}^{n} b_i \boldsymbol{\delta}_{y_i}$$

means the problem is a bipartite weighted matching problem between two discrete distributions. δ_{x_i} is the Dirac delta function located at x_i a and b are non-negative weights.

We consider the transfer map $\mathrm{T} : \mathbb{R}^d \to \mathbb{R}^d$ between X and Y that

$$\min_{\mathrm{T}} \sum_{i=1}^{m} d\left(\boldsymbol{x}_i, \mathrm{T}\left(\boldsymbol{x}_i\right)\right)$$
$$\text{subject to } b_j = \sum_{i:\mathrm{T}(\boldsymbol{x}_i)=\boldsymbol{y}_j} a_i, \quad \forall j \in \{1, ..., m\}$$

where $d(\cdot, \cdot)$ is the cost function between two points.

Using the Kantorovich formulation of the problem in which the constraints are continuous allows the problem to be represented as a convex linear programming (LP) problem for which FW and its variants can be applied. The problem is defined as:

$$\min_{\mathrm{T} \in \mathcal{U}(a,b)} \langle \mathrm{T}, \mathrm{C} \rangle$$

with the domain

$$\mathcal{U}(a, b) = \left\{ \mathrm{T} \in \mathbb{R}+^{m \times n} : \mathrm{T}1_n = a, \mathrm{T}^T 1_m = b \right\}.$$

In this definition, T is the transport matrix, C is the cost matrix and 1_m is an m-dimensional vector of ones.

Previous algorithms (e.g. entropy-regularizing or smooth-regularized approaches) are assumed to not be adaptable to cases with relaxed mass-conservation constraints. Therefore, as the authors consider a relaxed version of the OT problem, they aim to make the formulation applicable to the color transfer problem.

Moreover, three fast variants of the BCFW are developed and evaluated: pairwise-step, away-step, and gap-adaptive sampling. Upper bounds for the worst case convergence iterations are found, and the equivalence between the linearization duality gap and the Lagrangian duality gap is proven for the case of the semi-relaxed OT.

The authors define semi-relaxed OT as

$$\min_{\mathrm{T} \geq 0, \mathrm{T}^T 1_m = b} \langle \mathrm{T}, \mathrm{C} \rangle + \Phi(\mathrm{T}1_n, a),$$

where Φ is a smooth divergence function. This formulation is useful in the case of color transfer as is reflects differences in the colors of an input and a reference image.

The specific semi-relaxed problem considered in the paper uses an objective function defined as

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

where Φ is convex and smooth and λ is a relaxation parameter. Additionally, the domain is transformed to $\mathcal{M} = b_1\Delta_m \times b_2\Delta_m \times ... \times b_n\Delta_m$, where a component indicates the simplex of the summation b_i and is considered as a variable block. To apply BCFW, it is necessary that the domain can be separated into the Cartesian product of convex blocks. For each variable block, the BCFW algorithm solves

$$s_i = b_i \underset{e_k \in \Delta_m,\, k\in\{1,\ldots,m\}}{\operatorname{argmin}} \langle e_k, \nabla_i f(\mathrm{T}^{(k)})\rangle$$

with e_i being the unit vector with 1 in position i. After selecting a block either using uniform random or cyclic order on a permuted index, the sub-problem is solved on the block. The step-size is computed either by line-search or using the decaying step-size $\gamma = \frac{2n}{(k+2n)}$ with n the number of columns. After the update of the iterate, the duality gap is monitored as a stopping criterion, however only every n iterations. The paper proves that the linearization duality gap defined as

$$g(\mathrm{T}) = \langle \mathrm{T} - \mathrm{S}, \mathrm{C}\rangle + \frac{1}{\lambda}\langle \mathrm{T}1_n - \mathrm{S}1_n,\, \mathrm{T}1_n - a\rangle$$

is the same as the Lagrangian duality gap. In the equation, S is the solution to the sub-problem defined above.

In order to achieve a linear convergence rate, the authors introduce fast variants of the BCFW. Away-steps and pairwise steps, as described in the previous two papers, are implemented. In the case of these two algorithms, a line-search is used for the update. The line search is performed as follows:

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

where d is the direction and c_i is a column of the cost matrix. Afterward, not only the randomly chosen column vector is updated, but also the selected set of active atoms.

Additionally, the authors attempt to improve the convergence rate by adding a gap-adaptive sampling strategy, which selects columns with larger duality gaps more often for updating. Choosing them can lead to larger improvement in terms of the objective function value. In other words, the random choice of column i is weighted by the probability generated by the column-wise duality gap.

1.3.1 Convergence analysis

In sum, the analysis of the algorithms shows worse complexity ($\mathcal{O}(n(m+n))$) for the gap-adaptive method compared to the BCFW (which according to the

authors has a complexity of $\mathcal{O}(m)$ per iteration), since computations of the probabilities are needed at every iteration. The BCAFW and BCPFW have only the away and pairwise steps as additional complexity, which leads to complexity per iteration equal to $\mathcal{O}(m + |S_i|)$, where S_i is the cardinality of the set for the current block. In the case of $1 \leq |S_i| \leq m$, then the two algorithms are said to have a computational cost of $\mathcal{O}(n)$.

Following a series of numerical evaluations on the semi-relaxed OT problem in the context of color transfer, it is concluded that BCFW converges faster for large λ, yet the images show artifacts. With a small relaxation parameter, the algorithm produces a rather natural image, however with a slower convergence rate. Therefore, the authors advise stopping the algorithm prior to convergence.

2 Methods

Following the summary of the literature on the topic of Frank-Wolfe variants, the current project implements three versions of the Frank-Wolfe algorithm. The goal is to test them on the Optimal Transport problem as described in the third available paper.

2.1 Algorithms

2.1.1 Frank-Wolfe

The first algorithm is the base Frank-Wolfe, which closely follows the description in Fukunaga & Kasai, 2021. In the context of the optimal transport problem, the Frank-Wolfe algorithm is widely used due to being projection-free while also allowing for sparse solutions. The algorithm calls a linear oracle to optimize a linear approximation of the objective function over the convex feasible set, which we denote as $\mathcal{M}$.

In practice, in each iteration, the linear oracle is called for every column of the transport matrix. The requirement at the beginning of the algorithm aims to ensure that the iterates stay inside the feasible set. The gradient is computed as

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

which is the gradient for all variable blocks $b_i \Delta_m$. First, the minimization problem $s = \underset{s' \in \mathcal{M}}{\operatorname{argmin}} \langle s', \nabla f(\mathrm{x}^{(k)}) \rangle$ is solved for the current point $x^{(k)}$ in each iteration. The atom s is the feasible atom to move towards. Afterwards, the next iterate is computed as a convex combination of $x^{(k+1)} = (1 - \gamma)x^{(k)} + \gamma s$ with γ being the step-size.

For the current implementation of the algorithm, the decay step-size is chosen. This is because the analytical evaluations in Fukunaga & Kasai, 2021 use the decay step size, and also the numerical evaluations show longer computational time for FW and BCFW with exact line-search compared to decay step size. The latter only depends on the number of the iteration k. In the algorithmic representation, S indicates the matrix consisting of s_i as columns.

The stopping criterion for the algorithm is based on monitoring the duality gap $g(\mathrm{T})$, comparing it to a threshold ϵ.

Algorithm 1 Frank-Wolfe

Require: $T^{(0)} \in b_1\Delta_m \times b_2\Delta_m \times ... \times b_n\Delta_m$

for $k = 0 ... K$ **do**
  for $i = 0, ..., n$ **do**
    Compute $s_i = b_i \underset{e_k \in \Delta_m,\, k \in \{1,...,m\}}{\operatorname{argmin}} \langle e_k, \nabla_i f(\mathrm{T}^{(k)}) \rangle$
  end for
  if $g(T^{(k)}) \leq \epsilon$ **then**
    break
  end if
  Compute step-size $\gamma = \frac{2}{k+2}$
  Update $T^{(k+1)} = (1-\gamma)T^{(k)} + \gamma S$
end for

2.1.2 Block-Coordinate Frank-Wolfe

Next, we describe the block-coordinate version of the Frank-Wolfe algorithm. When dealing with large transport matrices, the cost of the base Frank-Wolfe algorithm grows due to the increasing number of linear oracle calls. To deal with this issue, a combination with the block-coordinate gradient approach makes a decrease in computational cost as well as a faster convergence rate possible. Therefore, assuming the domain $\mathcal{M}$ allows separation into blocks, the update of the iterate can be performed on a single convex block of $\mathcal{M}^{(i)}$.

Algorithm 2 Block-Coordinate Frank-Wolfe

Require: $T^{(0)} = (t_1^{(0)}, t_2^{(0)}, ..., t_n^{(0)}) \in b_1\Delta_m \times b_2\Delta_m \times ... \times b_n\Delta_m$

for $k = 0 ... K$ **do**
  Select index $i \in \{1, ..., n\}$ randomly
  Compute $s_i = b_i \underset{e_k \in \Delta_m,\, k \in \{1,...,m\}}{\operatorname{argmin}} \langle e_k, \nabla_i f(\mathrm{T}^{(k)}) \rangle$
  Compute step-size $\gamma = \frac{2n}{k+2n}$
  Update $t_i^{(k+1)}$ $\forall j \in \{1...n\}$ as $t_j^{(k+1)} = \begin{cases} \mathrm{t}^{(k)}, & \text{for } j \neq i \\ (1\text{-}\gamma)t_i^{(k)} + \gamma s_i, & \text{otherwise.} \end{cases}$
end for

The problem being solved here is the same as in the previous algorithm, however, it only refers to a single column. The choice of a column is random. Line 4 of the algorithm shows that the decay step-size is calculated the same way as before, yet it takes into consideration how many variable blocks there are.

The algorithm compares the value of the duality gap only every n iterations.

2.1.3 Block-Coordinate Away-Step Frank-Wolfe

The final algorithm is the Block-Coordinate version of the Away-Step Frank-Wolfe Algorithm. As described in the Introduction section, the Frank-Wolfe algorithm suffers from zig-zagging behavior if the optimal solution lies on on the boundary of the domain, leading to a sublinear rate of convergence.

A possible solution is the addition of away-steps, which aim to avoid the selection of unwanted atoms in the set by having the possibility to move away from an atom. After an update of the iterate, the set of active atoms is also updated, removing necessary ones (Line 14 in the algorithm). This procedure does not require an oracle. In order to do this, the coefficients of the extreme points (α_{e_j}) are selected if they are larger than 0 - which works since the active set is $\mathcal{S}_i \subset \{e_1, e_2, ..., e_n\}$.

The step-size and the choice of direction both depend on whether the FW direction or the away-step direction leads to a larger reduction in the objective function value (Line 7). The algorithm also implements a maximum step-size, which has the goal of maintaining the iterates inside the convex domain $\mathcal{M}$. The special case when the mass is fully transferred from one atom to another (if $\gamma = \gamma_{max}$) is called a drop step because the first atom is then removed from the active set.

Finally, monitoring the duality gap every n iterations allows the algorithm to stop if the gap becomes smaller than a certain threshold.

2.2 Color transfer problem

The color transfer problem to which the algorithms are applied to takes two images: an input and a reference. The goal is to transfer the input image into the color set of the reference image.

To apply the algorithms to the problem, the distributions of pixels according to color in RGB dimensions are extracted from each image, and 32 color centroids are computed via K-Means. These are taken as the distributions required for the constraints of the semi-relaxed OT problem. The cost matrix is also calculated.

In the semi-relaxed optimal transport problem there is a hyper parameter called the relaxation parameter which serves to adjust the strength of the constraint. Larger λ's cause smaller relaxation term and vice versa. After some experiments, we decided to fix $\lambda = 10^{-6}$ which is also considered a good choice in the practical results of Fukunaga & Kasai, 2021.

Algorithm 3 Block-Coordinate Away-Step Frank-Wolfe

Require: $T^{(0)} = (t_1^{(0)}, t_2^{(0)}, ..., t_n^{(0)}) \in b_1\Delta_m \times b_2\Delta_m \times ... \times b_n\Delta_m$ $\mathcal{S}_i = \{e_1\}$, $\forall i \in \{1, ..., m\}$

for $k = 0 \dots K$ **do**
  Select index $i \in [n]$ randomly
  Compute $s_i = b_i \underset{e_k \in \Delta_m,\, k \in \{1,...,m\}}{\operatorname{argmin}} \langle e_k, \nabla_i f(\mathrm{T}^{(k)}) \rangle$
  Set $d_{FW} = s_i - t_i^{(k)}$
  Compute $v_i = b_i \underset{e_j \in \mathcal{S}_i,\, j \in \{1,...,m\}}{\operatorname{argmax}} \langle e, \nabla_i f(\mathrm{T}^{(k)}) \rangle$
  Set $d_{AW} = t_i^{(k)} - v_i$
  if $\langle -\nabla f(T^{(k)}), d_{FW} \rangle \geq \langle -\nabla f(T^{(k)}), d_{AW} \rangle$ **then**
    $d = d_{FW}, \gamma_{max} = 1$
  else
    $d = d_{AW}, \gamma_{max} = \frac{\alpha_{v_i}}{1 - \alpha_{v_i}}$
  end if
  if $k \mod n = 0$ **then**
    if $\langle -\nabla f(\mathrm{T}^{(k)}), d \rangle \leq \epsilon$ **then**
      break
    end if
  end if
  Compute step-size $\gamma = \gamma_{LS} = -\frac{\lambda \langle d, c_i \rangle + \langle d, \mathrm{T}^{(k)} 1_n - a \rangle}{||d||^2}$
  Update $t_i^{(k+1)}$ $\forall j \in \{1...n\}$ as $t_j^{(k+1)} = \begin{cases} \mathrm{t}^{(k)}, & \text{for } j \neq i; \\ (1\text{-}\gamma) t_i^k + \gamma s_i, & \text{otherwise.} \end{cases}$
  Update $S_i^{(k+1)} = \{e \in \Delta_m : \alpha_e^{(t+1)} > 0\}$
end for

3 Results: evaluation of the algorithms on the problem

For our implementation of the color transfer problem described in the previous section, we selected two paintings by Pablo Picasso (figure 1). As input picture we use *The Blue Room*, a painting from his Blue Period[1]. As reference we use a later work: *Reading* from 1932. Besides testing the different algorithms described in section 2.1, we also want to find out how it would have looked if he painted the first picture not in his Blue Period but much later.

In the following, we will describe the experiments that we have done and their results. For further details on the implementation of the different algorithms, please refer to the Jupyter Notebook. The outputs are shown in figure 2, 3, 4 and 5.

Figure 2 shows the output of our experiment for the FW algorithm. The

[1]The Blue Period refers to a time between 1901 and 1904 when Picasso mainly painted in blue and only little other colors [source].

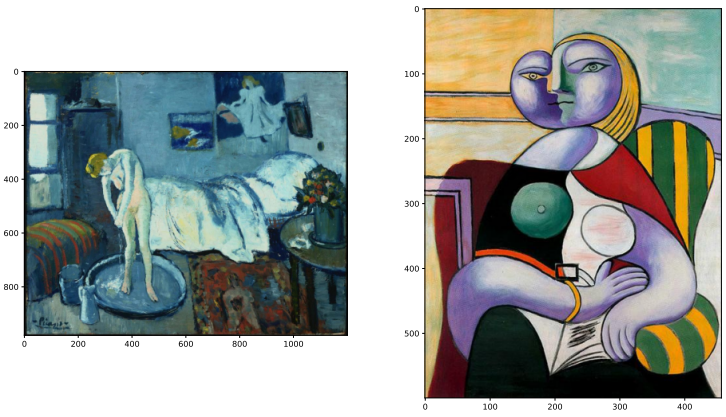

Figure 1: The two paintings for the implementation of the Color Transfer problem: The Blue Room from 1901 (left) and Reading from 1932 (right) both from Pablo Picasso;

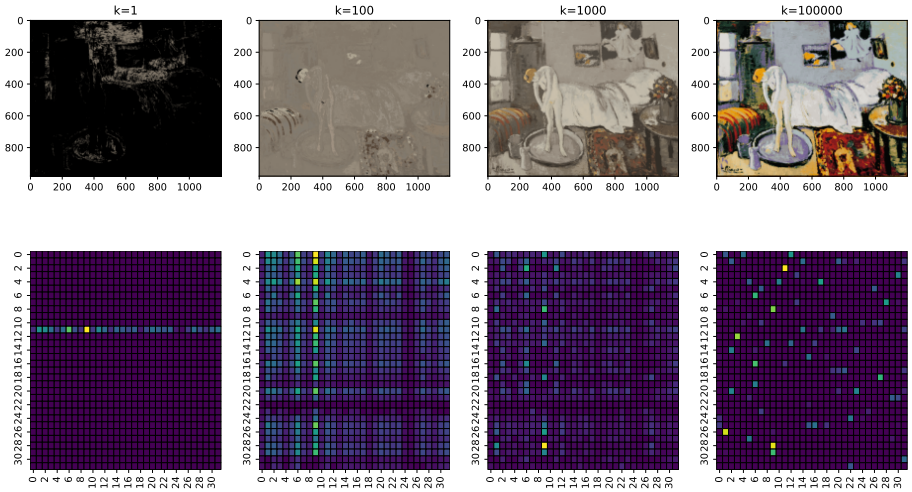

Figure 2: Output of the FW algorithm for the Color Transfer problem at different iterations: Color transferred pictures (top) and the heat-map of the transport matrix (bottom);

top row contains the color transferred input picture at different iterations. In the bottom row are the heat-maps of the respective transfer matrices. After $k = 1000$ iterations the picture is again recognizable. This corresponds with the transport matrix losing its structure from the first couple of 100 iterations. After $k = 1 \times 10^5$ iterations the contrasts of the colors are better and look more like the reference picture. This can also be seen in the transport matrix. For $k = 1000$ it contains many small, close to zero elements. Most of those are zero in the final result on the right leading to the observed sharper contrasts.

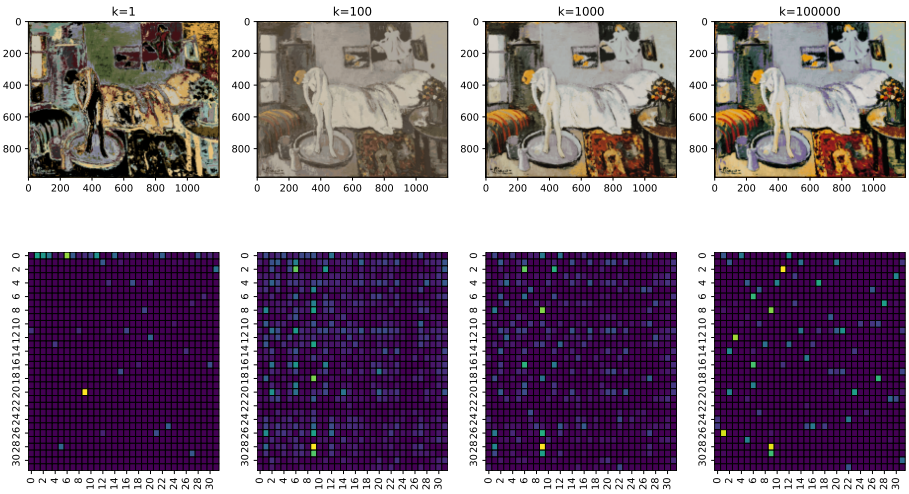

Figure 3: Output of the BCFW algorithm for the Color Transfer problem at different iterations: Color transferred pictures (top) and the heat-map of the transport matrix (bottom);

The BCFW method converges faster to a good solution. After 1000 iterations the picture already looks quite similar to the final one of FW. After $k = 1 \times 10^5$ the results are nearly identical. This is especially due to the higher flexibility of the BCFW algorithm in the early epochs. For $k = 1$ (which for BCFW corresponds to 32 partial gradient steps) the transport matrices for FW and BCFW already look quite different. FW calculates the full gradient at once. This leads to the situation that the second term of equation 1 is the same for all columns of the transport matrix. The only difference lies in the cost matrix. Combined with a small value for $\lambda = 10^{-6}$ this leads to the second term dominating in the beginning and therefore, the same gradient direction (row) for each column. We can see this trend also in the second picture of FW where the horizontal trend is more distributed. When it is finally broken in the third picture the result looks much better. For the BCFW the horizontal symmetry in T is already broken after the first epoch. This is due to the fact that each new partial gradient uses the updated result from the one before and

not the same initial one.

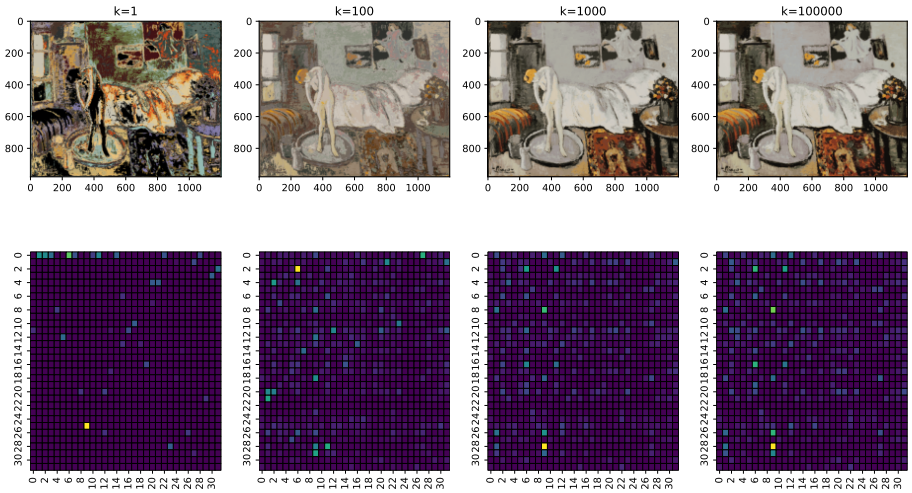

Figure 4: Output of the BCAFW-ELS algorithm for the Color Transfer problem at different iterations: Color transferred pictures (top) and the heat-map of the transport matrix (bottom);

The BCAFW-ELS performs similar to the BCFW (figure 4). The main difference is that the final result on the right is less sparse than the previous algorithm. This is a bit counterintuitive, as we would expect that the drop-steps would make our solution sparser. A reason for this could be the away-steps. For example, for a given run of 1000 epochs ($32 * 1000$ steps in total), the algorithm performed 1397 away-steps of which 813 were drop-steps. This means 581 away-steps reduce the coefficient for one of the atoms but do not remove it. This could be an explanation for the increase of atoms with low coefficients in the solution.

Finally, we compare the three algorithms regarding their dual gap, gradient norm and running time (figure 5). To get comparable results for the block-coordinate methods to the standard FW we calculate a full gradient and FW step each n iterations for these methods. Both, BCFW and BCAFW-ELS improve quickly in the beginning. The away-step algorithm improves even faster for the first 100 iterations, but then falls behind as it shows unstable behaviour for higher iterations. The standard FW converges slower than the two block-coordinate methods and keeps a higher duality gap during the considered 1000 epochs. Regarding the gradient norm, BCFW has the best performance. BCAFW-ELS lags behind and FW seems to be catching up over time. The running times show clear distinctions between the three algorithms, with FW being the fastest and BCAFW-ELS the slowest. Keep in mind that one epoch corresponds to one FW step but n block-coordinate steps. As shown in equation 1, the main cost of calculating the gradient is the row sum of the transport ma-

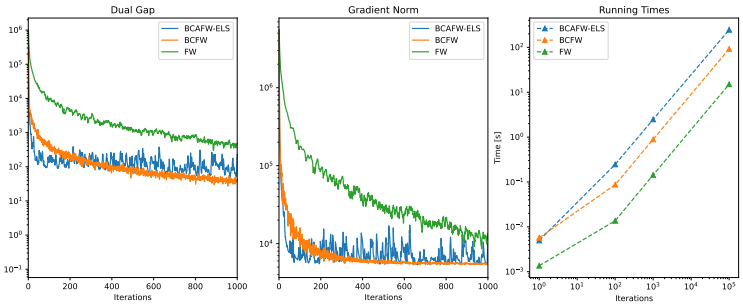

Figure 5: Dual-gap (left), matrix norm of the transport matrix (middle) and running times (right) for the three Frank-Wolfe algorithms over;

Algorithm	iterations	time [s]	error
FW	8434	1.28 s	9.99
BCFW	2788	2.56 s	9.02
BCAFW-ELS	1×10^4	24.38 s	118.06

Table 1: Running times of the three different algorithms for the color transfer problem (threshold = 10 and maximum iterations = 1×10^4);

trix in the second term. This operation is the same for each block. Therefore, the calculation of a full FW step and a partial gradient step are nearly the same. Because of this, one epoch of BCFW or BCAFW-ELS includes n times more matrix operation as one epoch of FW, which leads to a slower running time.

Drawing from the insights before, as a final test we compare how long the algorithms need to converge to a fixed threshold (table 1). Although BCFW and BCAFW-ELS need less iterations to reach smaller errors than FW, this effect is negated when comparing the running times. For a threshold of 10 and a maximum number of iterations 1×10^4 FW converges after 1.28 s. The BCFW needs 2.56 s and BCAFW-ELS does not converge but reaches the maximum number of iterations after 24.38 s.

4 Discussion

Building on top of Fukunaga & Kasai, 2021, the current report evaluates the performance of the proposed block-coordinate extensions of known Frank-Wolfe algorithms.

Both the FW and away-step FW have wide applications in various convex constrained optimization problems and according to the analyses provided by the three publications, they enjoy linear convergence rates for some of the applications. The away-step version can improve the performance in some ap-

plications.

Combining these algorithms with a block-coordinate approach can additionally improve their performance. In our implementation of the semi-relaxed Optimal Transport problem, this hypothesis proves to be partially true. In terms of iterations, both BCFW and BCAFW-ELS perform better than standard FW. However, due to the structure of the problem, the block-coordinate approaches do not increase the speed of each iteration, as they introduce additional computations. Moreover, the away-steps algorithm requires the storage and update of the active set of atoms, which can be costly. This is a potential point to improve in future implementations of the BCAFW - using faster data structures could partially alleviate this issue. As shown in our final experiment, when given a specific threshold for the duality gap FW still outperforms the other two algorithms.

Also, as discussed in the lectures, there might be different ways to represent the current iterate as a combination of vertices and in the case of our away-step algorithm, the presence of drop steps might lead to a different representation of the transport matrix compared to the FW and BCFW method (as seen in our result section).

The extent to which we can compare our results with the evaluations of the paper is limited since the paper does not look into the differences between FW, BCFW and BCAFW on a single problem but rather on subsets of problems.

Overall, the variants of the Frank-Wolfe algorithm prove to lead to good results when applied to the Color Transfer problem, thus showing that the formulation of the semi-relaxed OT problem of Fukunaga & Kasai, 2021 is solvable by the FW algorithms.

5 References

Lacoste-Julien, Simon, and Martin Jaggi. "On the global linear convergence of Frank-Wolfe optimization variants." Advances in neural information processing systems 28 (2015).

Fukunaga, Takumi, and Hiroyuki Kasai. "Fast block-coordinate Frank-Wolfe algorithm for semi-relaxed optimal transport." arXiv preprint arXiv:2103.05857 (2021).

Jaggi, Martin. "Revisiting Frank-Wolfe: Projection-free sparse convex optimization." International Conference on Machine Learning. PMLR, 2013.

The Blue Room https://en.wikipedia.org/wiki/The_Blue_Room_(Picasso)/media/File:Picasso's_Blue_Room_1901.jpg, first accessed on 08/06/2022

Reading https://www.wikiart.org/en/pablo-picasso/reading-1932/, first accessed on 08/06/2022