Step size selection in Frank-Wolfe method

Ignat Romanov, [1] **Elfat Sabitov,** [1] **Petr Sychev,** [1] **Boris Mikheev** [1]

Higher School of Economics

Abstract

This project aims to explore, evaluate and conduct numerical comparison of the efficiency of step selection strategies in the Frank-Wolfe method on different problems with different dimensions and condition numbers. We consider four strategies for step size selection and compare corresponding Frank-Wolfe method performance on both real and synthetic data with different properties and constraints on the solution.

1. Introduction

The Frank-Wolfe (FW) or conditional gradient algorithm is one of the oldest methods for nonlinear constrained optimization and has seen an impressive revival in recent years due to its low memory requirement and projection-free iterations. It can solve problems of the form

$$\min_{x \in Q} f(x), \tag{1}$$

where f is differentiable with L-Lipschitz gradient and the domain Q is a convex and compact set. Frank-Wolfe is a remarkably simple algorithm that given an initial guess x_0 constructs a sequence of estimates $x_1, x_2, ...$ that converges towards a solution of the optimization problem. The algorithm itself is provided in Algorithm 1.

Contrary to other constrained optimization algorithms like projected gradient descent, the Frank-Wolfe algorithm does not require access to a projection, hence why it is sometimes referred to as a projection-free algorithm. It instead relies on a routine that solves a linear problem over the domain, i. e. on the step $s^k \in \arg\max_{s \in Q} \langle -\nabla f(x^k), s \rangle$. This routine is commonly referred to as a *linear minimization oracle* (LMO). The rest of the algorithm mostly concerns finding the appropriate step size to move in the direction dictated by the linear minimization oracle $d^k = s^k - x^k$, where s^k is the result of the linear minimization oracle.

Algorithm 1 Frank-Wolfe algorithm

Input: initial guess x_0, gap tolerance $\delta > 0$
for $k = 0, 1, ...$ **do**
  $s^k \in \arg\max_{s \in Q} \langle -\nabla f(x^k), s \rangle$
  $d^k = s^k - x^k$
  $g^k = \langle -\nabla f(x^k), d^k \rangle$
  if $g^k < \delta$ **then**
    return x^k
  end if
  Set step size γ^k by the certain selection strategy
  $x^{k+1} = x^k + \gamma^k d^k$
end for
return x^k

One can see the Frank-Wolfe algorithm is as an algorithm that solves a potentially non-linear problem by solving a sequence of linear ones. The effectiveness of this approach is then tightly linked to the ability to quickly solve the linear subproblems. As it turns out, for a large class of problems, of which the ℓ_1 or nuclear (also known as trace) norm ball are the most widely known examples, the linear subproblems have either a closed form solution or efficient algorithms exist. Compared to a projection, the use of a linear minimization oracle has other important consequences. For example, the output of this linear minimization oracle is always a vertex of the domain and so by the update rule $x^{k+1} = x^k + \gamma^k d^k$ the iterate is expressed as a convex combination of vertices. This feature can be very advantageous in situations with a huge or even infinite number of features, such as architecture optimization in neural networks or estimation of an infinite-dimensional sparse matrix arising in multi-output polynomial network.

2. Step size selection methods

As other gradient-based methods, the FW algorithm depends on a step size parameter γ^k. The following step size selection approaches were considered:

- **Predefined decreasing sequence (trivial)**. The simplest choice is to choose the step-size according to the pre-defined decreasing sequence:

$$\gamma^k = \frac{2}{k+2} \tag{2}$$

This choice of step size is straightforward and cheap to compute. However, in practice mostly it performs worst than the alternatives, although it enjoys the same worst-case complexity bounds.

- **Exact line-search.** Another alternative is to take the step-size that maximizes the decrease in objective along the update direction:

$$\gamma^k = \arg\min_{\gamma \in [0,1]} f(x^k + \gamma d^k) \tag{3}$$

By definition, this step-size gives the highest decrease per iteration. However, solving such additional objective can be a costly optimization problem, so this variant is not practical except on a few specific cases where the above problem is easy to solve (for instance, in quadratic objective functions).

- **Demyanov-Rubinov step-size.** A less-known but highly effective step-size strategy for FW in the case in which we have access to the Lipschitz constant of ∇f, denoted L, is the following:

$$\gamma^k = \min\left(\frac{g^k}{L||d^k||^2}, 1\right) \tag{4}$$

Note this step-size naturally goes to zero as we approach the optimum, which as we'll see in the next section is a desirable property. This is because the step size is proportional to the Frank-Wolfe gap g^k, which is a measure of problem suboptimality.

- **Armijo rule.** This method considers more general gradient-related descent directions which can be expressed in the following form:

$$d^k(x) = -M\nabla f(x^k), \tag{5}$$

where $M \in \mathcal{S}^d$ is positive definite. In general, the Armijo rule is a condition which ensures a sufficient descent in the following sense:

$$f(x^k + \gamma^k d^k) \leq f(x^k) + \sigma\gamma^k \nabla f(x^k)^T d^k \tag{6}$$

for fixed $\sigma \in (0, 1)$. Such requirement can be interpreted as a restriction on the step size γ^k, and it is used to adjust the step size. So, in this step size selection strategy the initial step size is chosen as $\gamma^0 = 1$, and then at each step k, while (6) is not satisfied, we take

$$\gamma^k :\in [\nu_1\gamma_k, \nu_2\gamma^k], \tag{7}$$

where $0 < \nu_1 < \nu_2 < 1$, or, in simpler version, which will be used further,

$$\gamma_k := \frac{\gamma_k}{2} \tag{8}$$

each time until the condition (6) met.

- **Backtracking line-search.** The main drawback of the Demyanov-Rubinov step-size is that it requires knowledge of the Lipschitz constant. This limits its usefulness because, first, it might be costly to compute this constant, and secondly, this constant is a global upper bound on the curvature, leading to suboptimal step-sizes in regions where the local curvature might be much smaller.

But there's a way around this limitation. It's possible estimate a step-size such that it guarantees a certain decrease of the objective without using global constants.

In this algorithm, instead of relying on the quadratic upper bound given by the Lipschitz constant, a local quadratic approximation at x^k is constructed. This quadratic function is the following:

$$Q^k(\gamma^k, M^k) = f(x^k) - \gamma^k g^k + \frac{(\gamma^k)^2 M^k}{2}||d^k||^2, \tag{9}$$

where M^k is the local constant which is smaller than global Lipschitz constant L.

Also there's a way to estimate M^k, that is both convenient and also gives strong theoretical guarantees. For this, it is possible to choose the M^k that makes the quadratic approximation an upper bound at the next iterate: $f(x^{k+1}) \leq Q^k(^k, M^k)$. This way, we can guarantee that the backtracking line-search step-size makes at least as much progress as exact line-search in the quadratic approximation. There are many values of M^k that verify this condition. For example, from the L-smooth inequality it follows that any $M^k \geq L$ will be a valid choice. However, values of M^k that are much smaller than L will be the most interesting, as these will lead to larger step-sizes.

In practice there's little value in spending too much time finding the smallest possible value of M^k, and the most common strategy consists in initializing this value a bit smaller than the one used in the previous iterate (for example $0.99M^{k-1}$), and correct if necessary.

Algorithm 2 Frank-Wolfe algorithm with backtracking line-search

Input: initial guess x_0, gap tolerance $\delta > 0$, backtracking line-search parameters $\tau > 1$, $\eta \leq 1$, initial guess for M^{-1}.
for $k = 0, 1, ...$ **do**
  $s^k \in \arg\max_{s \in Q} \langle -\nabla f(x^k), s \rangle$
  $d^k = s^k - x^k$
  $g^k = \langle -\nabla f(x^k), d^k \rangle$
  $M^k = \eta M^{k-1}$
  $\gamma^k = \min(\frac{g^k}{M^k ||d^k||^2}, 1)$
  while $f(x^k + \gamma^k d^k) > Q^k(\gamma^k, M^k)$ **do**
    $M^k = \tau M^k$
  end while
  $x^{k+1} = x^k + \gamma^k d^k$
end for
return x^k

The full Frank-Wolfe algorithm with backtracking line-search is provided in Algorithm 2.

The block starts by choosing a constant that is a factor of η smaller than the one given by the previous iterate. If $\eta = 1$, then it will be exactly the same than the previous iterate, but if η is smaller than 1 (practically $\eta = 0.9$ is a reasonable default), this will result in a candidate value of M^k that is smaller than the one used in the previous iterate. This is done to ensure this constant can decrease if we move to a region with smaller curvature. The next line the algorithm sets a tentative value for the step-size based on the formula using the current (tentative) value for Mt. The next line is a while loop that increases M^k until it verifies the sufficient decrease condition. The algorithm is not fully agnostic to the "local" Lipschitz constant, as it still requires to set an initial value for this constant, M^{-1}. One of the working well in practice heuristics is to initialize it to the (approximate) local curvature along the update direction. For this, select a small ε, e. g. $\varepsilon = 10^{-3}$ and set $M^{-1} = \frac{||\nabla f(x^0) - \nabla f(x^0 + \varepsilon d^0)||}{\varepsilon ||d^0||}$

3. Experiments

3.1. Datasets

For experiments the following datasets were used:

- **UCI Mushrooms**: binary classification dataset, contains descriptions of mushrooms (poisonous/edible), 8124 objects, 22 features
- **UCI Gisette**: binary classification dataset, contains engineered features of handwritten digits, the task is to disambiguate the digits 4 and 9, 13500 objects, 5000 features
- **UCI Covertype**: binary classification dataset, contains cartographic variables, the task is to determine the forest cover type, 581012 objects, 54 features.
- **Synthetic data:** binary classification data generated using the sklearn package, three setups were considered: standard (5000 samples, 50 features), high-dimensional (5000 samples, 1024 features), and ill-conditioned (5000 samples, 50 features)
- **Rosenbrock function**: common test function for optimization performance testing, recieves the array of points $x_1, ..., x_n$ as input, has the following form:

$$f(x) = \sum_{i=1}^{n-1} (100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2) \quad (10)$$

3.2. Constraint sets

For constraint sets on the desired solution we considered ℓ_1 and ℓ_2 balls centered at 0, of radiuses $R = 10, 100, 500$

3.3. Experiments setup

All the datasets were used without features changes, except instances labels were transformed into 1 and -1 values for positive and negative objects respectively. All the datasets were split on the train and test parts in proportion 8/2. The standard logistic regression objective was considered for the optimization convergence and performance criterion:

$$f(X, Y, w) = \sum_{i=1}^{n} \ell(w^T x_i, y_i), \quad (11)$$

$$\ell(z, y) = ln(1 + e^{-yz}), \quad (12)$$

$x_1, ..., x_n \in \mathbb{R}^d$ - data objects features, $y_1, ..., y_n \in \{-1, 1\}$ - corresponding labels, $w \in \mathbb{R}^d$ - logistic regression models weights.

4. Results

The results of conducted experiments are provided on the following obtained plots below in Section 7.

5. Conclusion

- Backtracking line search has the best performance
- Armijo has the same performance
- Demyanov Rubinov method sometimes even worse that trivial, but on datasets with low L-Lipschitz and low features amount
- High-dimensional dataset leads to unstable convergence
- Functions with complex landscape leads to the bad performance with the trivial approach

6. Repository

Our GitHub

7. Appendix

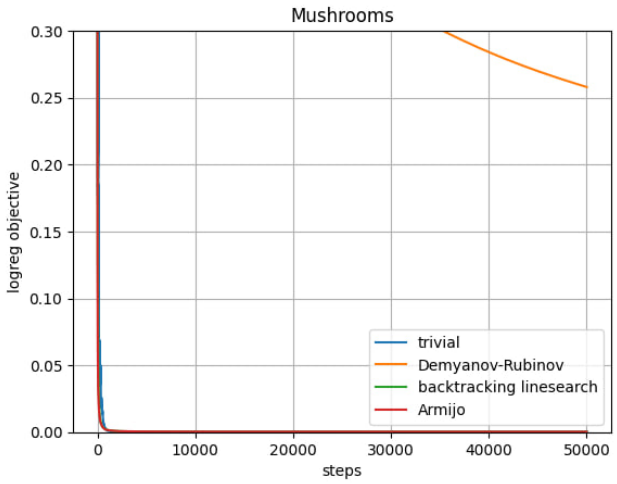

Figure 1. Values of convergence criterion (logreg objective) by iteration number for different Frank-Wolfe method step size values, Mushrooms dataset, constraint on the ℓ_1 ball of radius $R = 100$

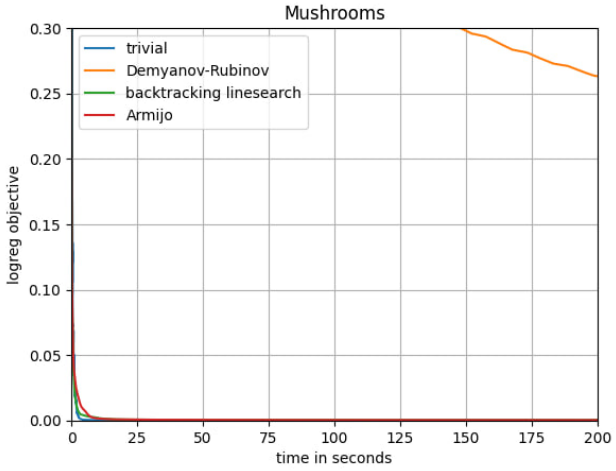

Figure 2. Values of convergence criterion (logreg objective) by time for different Frank-Wolfe method step size values, Mushrooms dataset, constraint on the ℓ_1 ball of radius $R = 100$

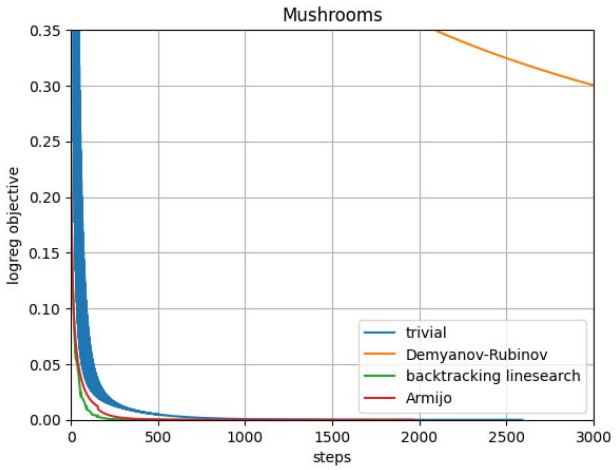

Figure 3. Values of convergence criterion (logreg objective) by iteration number for different Frank-Wolfe method step size values, Mushrooms dataset, constraint on the ℓ_2 ball of radius $R = 100$

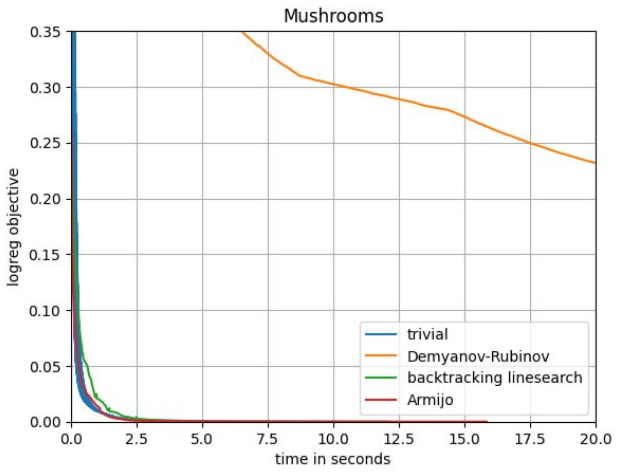

Figure 4. Values of convergence criterion (logreg objective) by time for different Frank-Wolfe method step size values, Mushrooms dataset, constraint on the ℓ_2 ball of radius $R = 100$

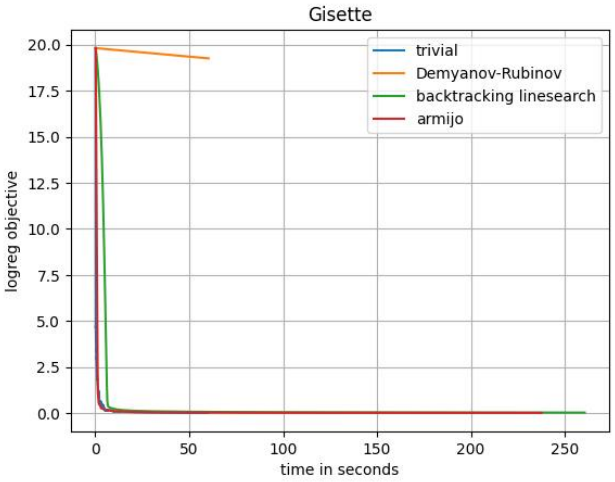

Figure 6. Values of convergence criterion (logreg objective) by time for different Frank-Wolfe method step size values, Gisette dataset, constraint on the ℓ_1 ball of radius $R = 100$

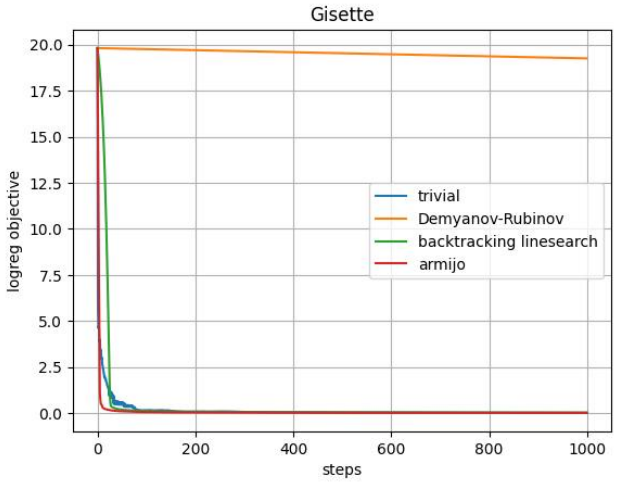

Figure 5. Values of convergence criterion (logreg objective) by iteration number for different Frank-Wolfe method step size values, Gisette dataset, constraint on the ℓ_1 ball of radius $R = 100$

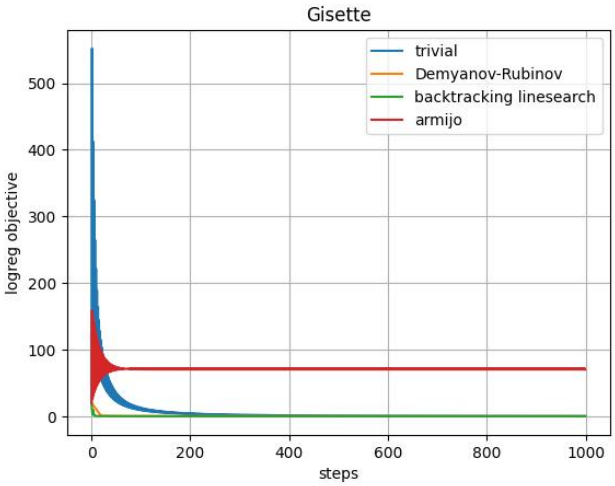

Figure 7. Values of convergence criterion (logreg objective) by iteration number for different Frank-Wolfe method step size values, Gisette dataset, constraint on the ℓ_2 ball of radius $R = 100$

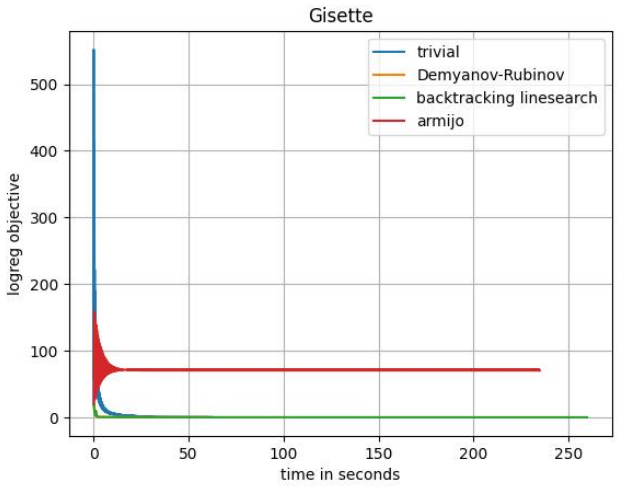

Figure 8. Values of convergence criterion (logreg objective) by time for different Frank-Wolfe method step size values, Gisette dataset, constraint on the ℓ_2 ball of radius $R = 100$

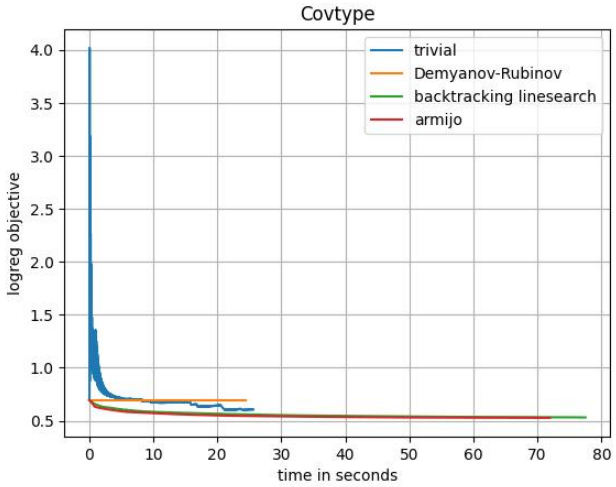

Figure 10. Values of convergence criterion (logreg objective) by iteration number for different Frank-Wolfe method step size values, Covertype dataset, constraint on the ℓ_1 ball of radius $R = 100$

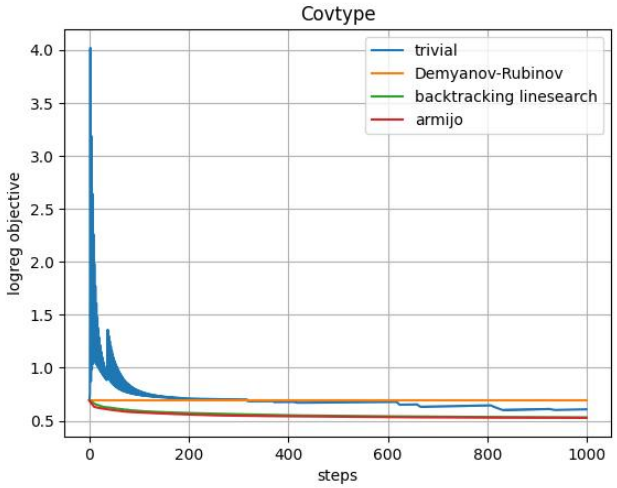

Figure 9. Values of convergence criterion (logreg objective) by iteration number for different Frank-Wolfe method step size values, Covertype dataset, constraint on the ℓ_1 ball of radius $R = 100$

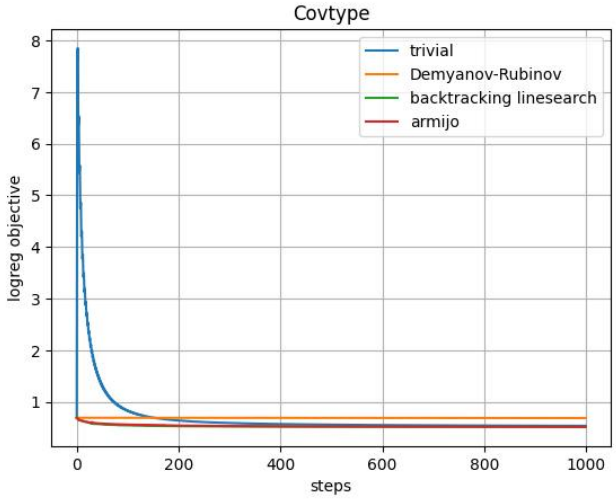

Figure 11. Values of convergence criterion (logreg objective) by iteration number for different Frank-Wolfe method step size values, Covertype dataset, constraint on the ℓ_2 ball of radius $R = 100$

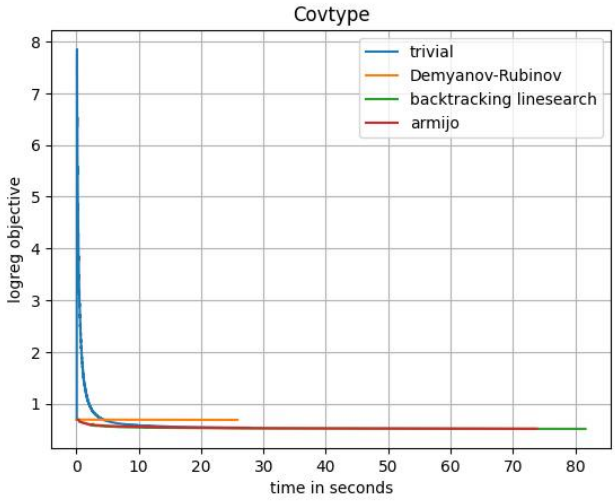

Figure 12. Values of convergence criterion (logreg objective) by time for different Frank-Wolfe method step size values, Covertype dataset, constraint on the ℓ_2 ball of radius $R = 100$

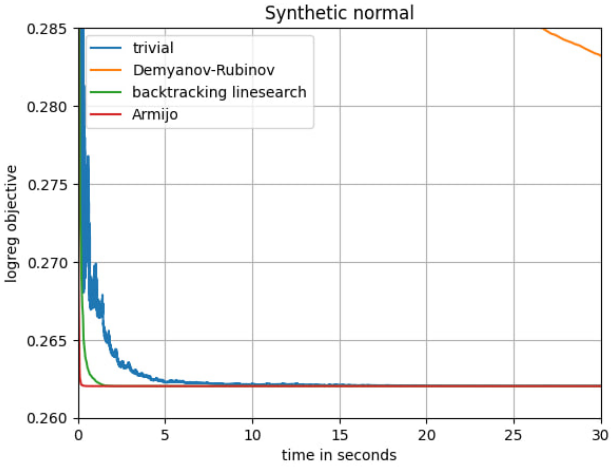

Figure 14. Values of convergence criterion (logreg objective) by time for different Frank-Wolfe method step size values, synthetic normal dataset, constraint on the ℓ_1 ball of radius $R = 100$

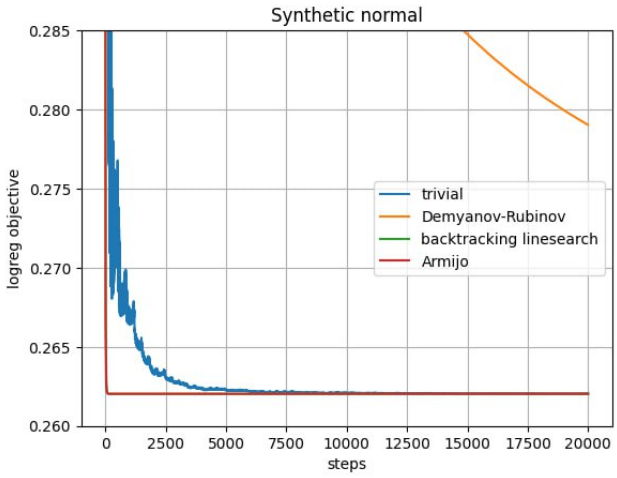

Figure 13. Values of convergence criterion (logreg objective) by iteration number for different Frank-Wolfe method step size values, synthetic normal dataset, constraint on the ℓ_1 ball of radius $R = 100$

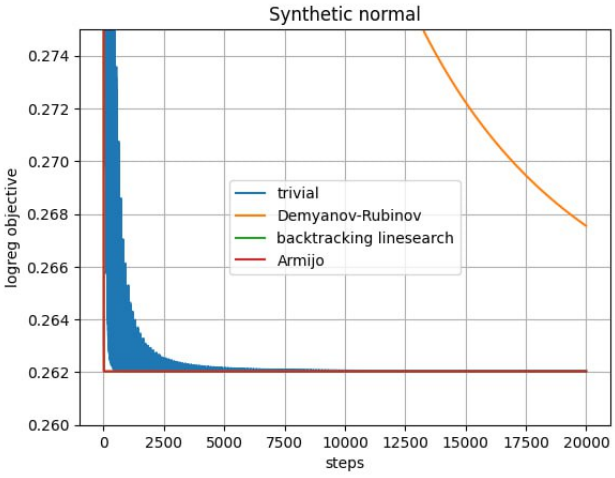

Figure 15. Values of convergence criterion (logreg objective) by iteration number for different Frank-Wolfe method step size values, synthetic normal dataset, constraint on the ℓ_2 ball of radius $R = 100$

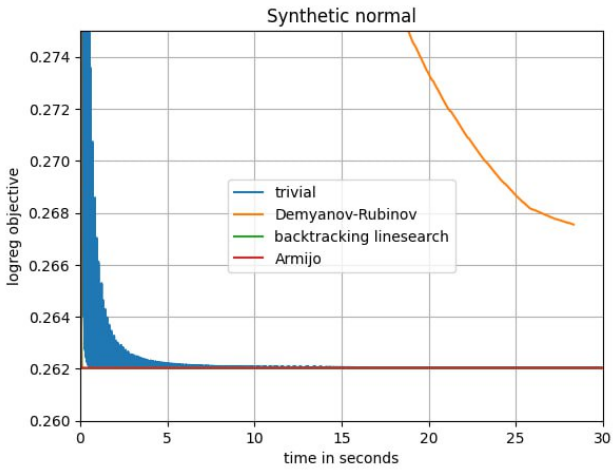

Figure 16. Values of convergence criterion (logreg objective) by time for different Frank-Wolfe method step size values, synthetic normal dataset, constraint on the ℓ_2 ball of radius $R = 100$

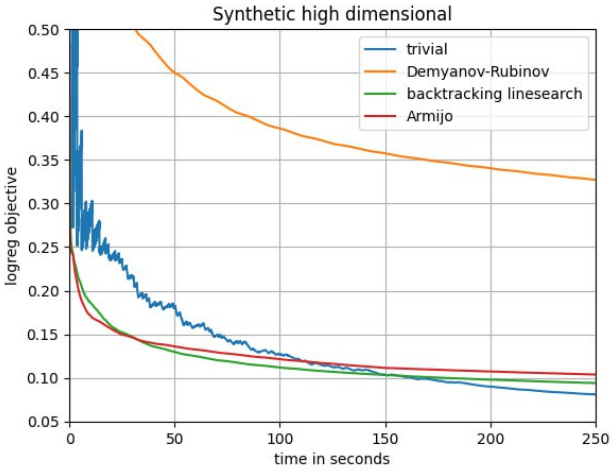

Figure 18. Values of convergence criterion (logreg objective) by time for different Frank-Wolfe method step size values, synthetic high-dimensional dataset, constraint on the ℓ_1 ball of radius $R = 100$

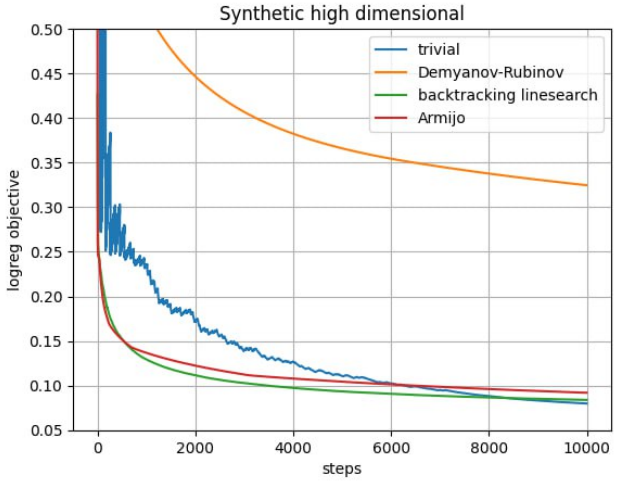

Figure 17. Values of convergence criterion (logreg objective) by iterations number for different Frank-Wolfe method step size values, synthetic high-dimensional dataset, constraint on the ℓ_1 ball of radius $R = 100$

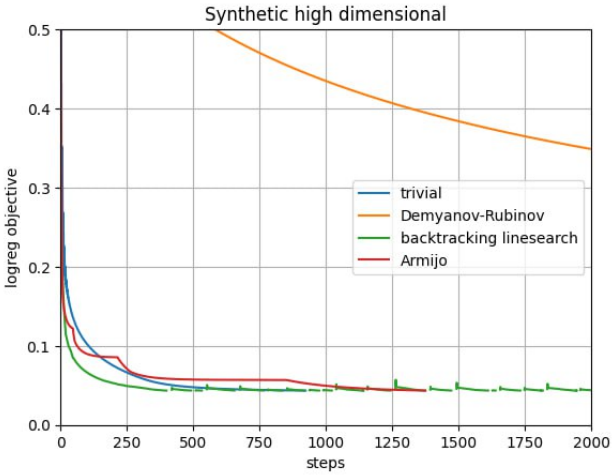

Figure 19. Values of convergence criterion (logreg objective) by iterations number for different Frank-Wolfe method step size values, synthetic high-dimensional dataset, constraint on the ℓ_2 ball of radius $R = 100$

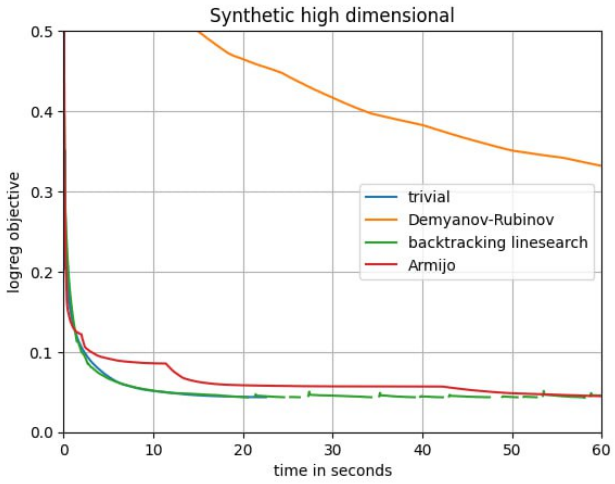

Figure 20. Values of convergence criterion (logreg objective) by time for different Frank-Wolfe method step size values, synthetic high-dimensional dataset, constraint on the ℓ_2 ball of radius $R = 100$

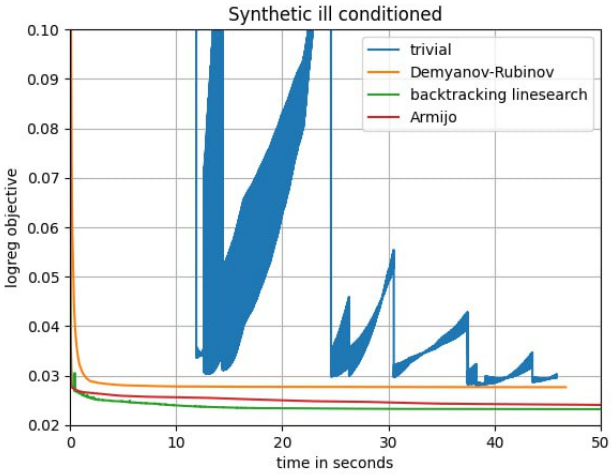

Figure 22. Values of convergence criterion (logreg objective) by time for different Frank-Wolfe method step size values, synthetic ill-conditioned dataset, constraint on the ℓ_1 ball of radius $R = 100$

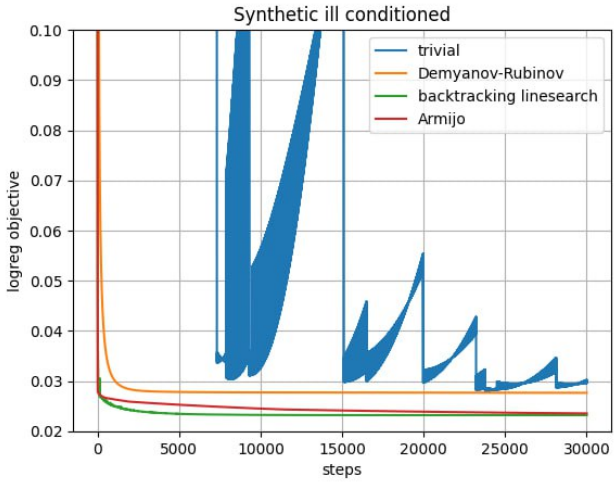

Figure 21. Values of convergence criterion (logreg objective) by iterations number for different Frank-Wolfe method step size values, synthetic ill-conditioned dataset, constraint on the ℓ_1 ball of radius $R = 100$

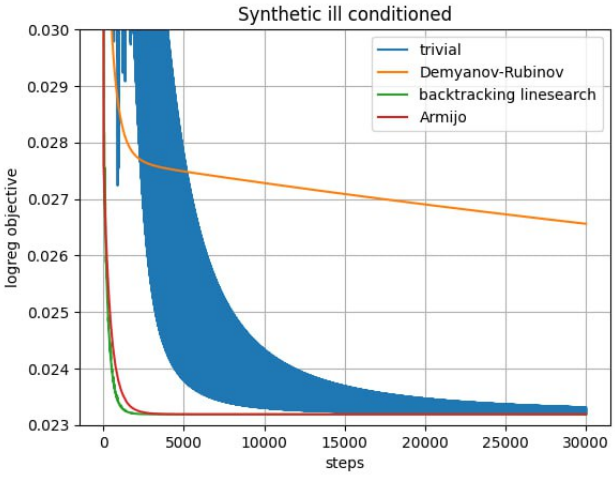

Figure 23. Values of convergence criterion (logreg objective) by iterations number for different Frank-Wolfe method step size values, synthetic ill-conditioned dataset, constraint on the ℓ_2 ball of radius $R = 100$

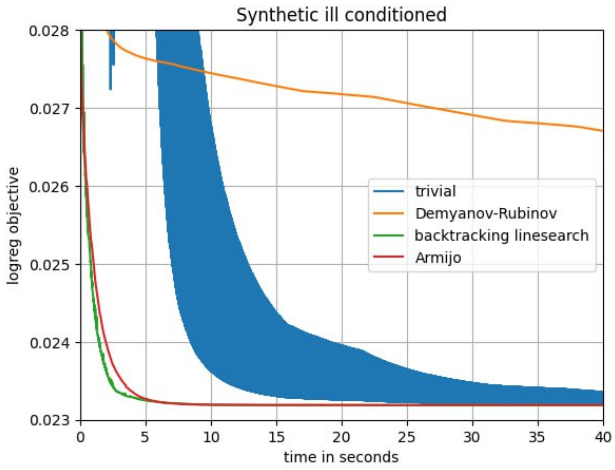

Figure 24. Values of convergence criterion (logreg objective) by time for different Frank-Wolfe method step size values, synthetic ill-conditioned dataset, constraint on the ℓ_2 ball of radius $R = 100$

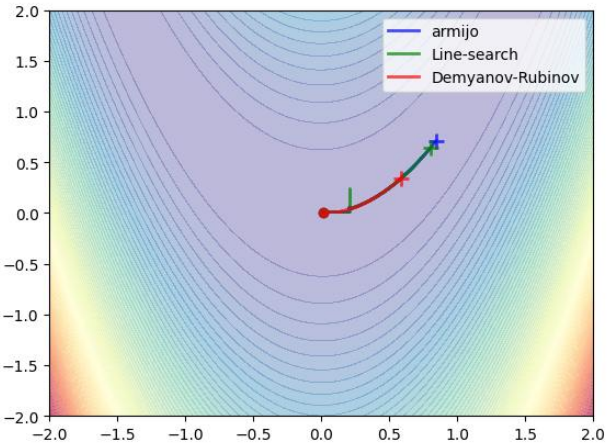

Figure 25. Landscape of convergence for Rosenbrock function for different Frank-Wolfe method step size values, constraint on the ℓ_1 ball of radius $R = 100$

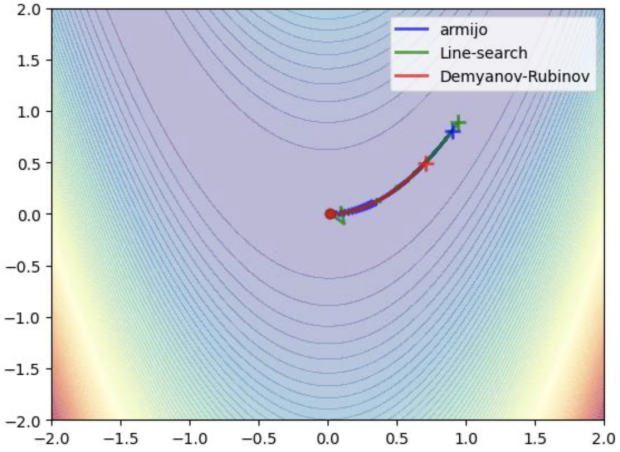

Figure 26. Landscape of convergence for Rosenbrock function for different Frank-Wolfe method step size values, constraint on the ℓ_2 ball of radius $R = 100$

8. Team member's contributions

Ignat Romanov (25% of work)

- Core algorithms development
- Main experiments
- Datasets preparation

Elfat Sabitov (25% of work)

- Core algorithms development
- Additional experiments
- Github repository

Petr Sychev (25% of work)

- Theory research
- Armijo method
- Presentation

Boris Mikheev (25% of work)

- Project description,
- Literature review
- Experiments