



Panos Pardalos · Michael Khachay ·
Alexander Kazakov (Eds.)


Mathematical Optimization Theory and Operations Research

20th International Conference, MOTOR 2021
Irkutsk, Russia, July 5–10, 2021
Proceedings

Editors

Panos Pardalos 
University of Florida
Gainesville, FL, USA

Alexander Kazakov 
Matrosov Institute for System Dynamics
and Control Theory
Irkutsk, Russia

Michael Khachay 
Krasovsky Institute of Mathematics
and Mechanics
Ekaterinburg, Russia

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-030-77875-0 ISBN 978-3-030-77876-7 (eBook)
<https://doi.org/10.1007/978-3-030-77876-7>

LNCS Sublibrary: SL1 – Theoretical Computer Science and General Issues

© Springer Nature Switzerland AG 2021

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland



On the Computational Efficiency of Catalyst Accelerated Coordinate Descent

Dmitry Pasechnyuk^(✉) and Vladislav Matyukhin

Moscow Institute of Physics and Technology (National Research University),
Dolgoprudny, Russia
{pasechniuk.da,matyukhin}@phystech.edu

Abstract. This article is devoted to one particular case of using universal accelerated proximal envelopes to obtain computationally efficient accelerated versions of methods used to solve various optimization problem setups. We propose a proximally accelerated coordinate descent method that achieves the efficient algorithmic complexity of iteration and allows taking advantage of the data sparseness. It was considered an example of applying the proposed approach to optimizing a SoftMax-like function, for which the described method allowing weaken the dependence of the computational complexity on the dimension n in $\mathcal{O}(\sqrt{n})$ times and, in practice, demonstrates a faster convergence in comparison with standard methods. As an example of applying the proposed approach, it was shown a variant of obtaining on its basis some efficient methods for optimizing Markov Decision Processes (MDP) in a minimax formulation with a Nesterov smoothed target function.

Keywords: Proximal accelerated method · Catalyst · Accelerated coordinate descent method · SoftMax · Markov decision processes

1 Introduction

One of the most important theoretical results in convex optimization was the development of accelerated optimization methods [23]. At the initial stage of implementation of this concept, many accelerated algorithms for different problem setups were proposed. But each such case required special consideration of the possibility of acceleration. Therefore, the proposed designs were significantly different and did not allowing assume a way to generalize them. A significant step towards the development of a universal scheme for accelerating optimization

D.A. Pasechnyuk's research was supported by the A.M. Raigorodsky Scholarship in the field of optimization and RFBR grant 19-31-51001 (Scientific mentoring). The work of V.V. Matyukhin was supported by the Ministry of Science and Higher Education of the Russian Federation (state assignment) No. 075-00337-20-03, project number 0714-2020-0005.

methods was the paper in which an algorithm called Catalyst proposed, based on the idea of [26, 27] and allowing to accelerate other optimization methods, using them for the sequential solving of several Moreau–Yosida regularized auxiliary problems [18, 19]. Following these ideas, many variants of the applications of this method and its modifications [14, 17, 25] were proposed. Among the most recent results until the time of writing this paper, the generalizations of the discussed approach to tensor methods [5, 10, 11, 20] were also described. The corresponding form of the accelerated proximal envelope to the authors’ knowledge is the most general of those described in the literature, and therefore, this paper is focused primarily on the methods proposed in the papers [10, 11].

main motivation of this paper is to describe the possibilities of the practical application of universal accelerated proximal envelopes for constructing computationally and oracle efficient optimization methods. Let us consider the classical coordinate descent method [4], the iteration of which for the convex function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is of the form:

$$x_{k+1}^i = x_k^i - \eta \nabla_i f(x_k), \quad i \sim \mathcal{U}\{1, \dots, n\}, \quad \eta > 0.$$

One of the many applications of this method is the optimization of functions, for which the calculation of the one component of the gradient is significantly more efficient than the calculation of the full gradient vector (in particular, many problems in the case of sparse formulations satisfy this condition). The oracle complexity of this method, provided that the method stops when the ε -small function value residual is reached, is $\mathcal{O}\left(n \frac{\bar{L} R^2}{\varepsilon}\right)$, where $R^2 = \|x_0 - x_*\|_2^2$, $\bar{L} = \frac{1}{n} \sum_{i=1}^n L_i$ is the average of the Lipschitz constants of the gradient components. However, this estimate is not optimal for the class of convex problems. Let us now consider the accelerated coordinate descent method proposed by Yu.E. Nesterov [24]. The oracle complexity of this method corresponds to the optimal bound: $\mathcal{O}\left(n \sqrt{\frac{\tilde{L} R^2}{\varepsilon}}\right)$, where $\sqrt{\tilde{L}} = \frac{1}{n} \sum_{i=1}^n \sqrt{L_i}$ is the mean of square roots of the Lipschitz constants of the gradient components. At the same time, the situation changes drastically when the algorithmic complexity of the method considered. Namely, even if the computation of one component of the gradient has the complexity $\mathcal{O}(s)$, $s \ll n$, the complexity of the whole iteration of the accelerated coordinate descent method will be $\mathcal{O}(n)$, unlike the standard method, the iteration complexity of which is $\mathcal{O}(s)$. It means that the sparseness of the problem when using the accelerated coordinate descent method does not significantly affect the complexity of the algorithm, and besides, the complexity in this case quadratically depends on the dimension of the problem. Together, this somewhat devalues the use of the coordinate descent method in this case. Thus, an interesting problem is the construction of an accelerated coordinate descent method, the iteration complexity of which, as in the standard version of the method, is $\mathcal{O}(s)$. This is possible due to the application of the universal accelerated proximal envelope “Accelerated Meta-algorithm” [11].

This article consists of an introduction, conclusion and the main Sect. 2. It describes the theoretical results on the convergence and algorithmic complexity

of the coordinate descent method, accelerated by using the “Accelerated Meta-algorithm” envelope (Sect. 2.1). Using the example of the SoftMax-like function optimization problem, it was experimentally tested method’s effectiveness with relation to its working time. There were also described the possibilities of its computationally efficient implementation, and carried out a comparison with standard methods (Sect. 2.2). Further, as an example of applying the proposed approach, it was provided a method for optimizing Markov Decision Processes in a minimax formulation, based on applying the method introduced in this paper to the Nesterov smoothed target function. The proposed approach obtains estimates close to that for several efficient and practical methods for optimizing the discounted MDP and matches the best estimates for the averaged MDP problem (Sect. 2.3).

2 Accelerated Meta-algorithm and Coordinate Descent

2.1 Theoretical Guarantees

Let us consider the following optimization problem of the function $f : \mathbb{R}^n \rightarrow \mathbb{R}$:

$$\min_{x \in \mathbb{R}^n} f(x),$$

subject to:

1. f is differentiable on \mathbb{R}^n ;
2. f is convex on \mathbb{R}^n ;
3. $\nabla_i f$ is component-wise Lipschitz continuous, i.e. $\forall x \in \mathbb{R}^n$ and $u \in \mathbb{R}$, $\exists L_i \in \mathbb{R}$ ($i = 1, \dots, n$), such that

$$|\nabla_i f(x + ue_i) - \nabla_i f(x)| \leq L_i |u|,$$

where e_i is the i -th unit basis vector, $i \in \{1, \dots, n\}$;

4. ∇f is L -Lipschitz continuous.

Let us turn to the content of the paper [11], where a general version of the “Accelerated Meta-algorithm” for solving convex optimization problems for composite functionals in form of $F(x) = f(x) + g(x)$ was proposed. For the considered formulation of the problem, such generality is not required. It is sufficient to apply a special case of the described scheme for $p = 1$, $f \equiv 0$ (using the designations of the corresponding paper), in which the described envelope takes the form of an accelerated proximal gradient method. The method used is listed as Algorithm 1.

Before formulating any results on the convergence of the proposed accelerated coordinate descent method described below, it is necessary to start with a detailed consideration of the process of solving the auxiliary problems, where its analytical solution is available only in rare cases. Therefore, one should apply numerical methods to find its approximate solution, and that is inaccurate. The

Algorithm 1: Accelerated Meta-algorithm for First-order Method \mathcal{M}

Input: $H > 0, x_0 \in \mathbb{R}^n$;
 $\lambda \leftarrow 1/2H$;
 $A_0 \leftarrow 0; v_0 \leftarrow x_0$;
for $k = 0, \dots, \tilde{N} - 1$ **do**
 $a_{k+1} \leftarrow \frac{\lambda + \sqrt{\lambda^2 + 4\lambda A_k}}{2}$;
 $A_{k+1} \leftarrow A_k + a_{k+1}$;
 $\tilde{x}_k \leftarrow \frac{A_k v_k + a_{k+1} x_k}{A_{k+1}}$;
 By running the method \mathcal{M} ,
 find the solution of the following auxiliary problem
 with an accuracy ε by the argument:
 $v_{k+1} \in \text{Arg}^\varepsilon \min_{y \in \mathbb{R}^n} \left\{ f(y) + \frac{H}{2} \|y - \tilde{x}_k\|_2^2 \right\}$;
 $x_{k+1} \leftarrow x_k - a_{k+1} \nabla f(v_{k+1})$;
end
return $v_{\tilde{N}}$;

process of solving the auxiliary problem should continue until the following stopping condition is satisfied ([16], Appendix B):

$$\left\| \nabla \left\{ F(y_\star) := f(y_\star) + \frac{H}{2} \|y_\star - \tilde{x}_k\|_2^2 \right\} \right\|_2 \leq \frac{H}{2} \|y_\star - \tilde{x}_k\|_2, \tag{1}$$

where y_\star is an approximate solution of the auxiliary problem, returned by internal method \mathcal{M} . Due to the $\|\nabla F(y_\star)\|_2 = 0$ (where y_\star denotes an exact solution of the considered problem), and due to the $(L + H)$ -Lipschitz continuity of ∇F , we have got:

$$\|\nabla F(y_\star)\|_2 \leq (L + H) \|y_\star - y_\star\|_2. \tag{2}$$

Writing out the triangle inequality: $\|\tilde{x}_k - y_\star\|_2 - \|y_\star - y_\star\|_2 \leq \|y_\star - \tilde{x}_k\|_2$, and using it together the inequalities (1) and (2), we have got the final form of the stopping condition:

$$\|y_\star - y_\star\|_2 \leq \frac{H}{3H + 2L} \|\tilde{x}_k - y_\star\|_2. \tag{3}$$

This implies that the required argument accuracy of solving the auxiliary problem does not depend on the accuracy required for the main problem solution. That makes it possible to significantly simplify the inference of further results.

Let us now consider the main method used for solving auxiliary problems. Coordinate Descent Method in version of [22] (in the particular case, when $\gamma = 1$) is listed as Algorithm 2. For this method, in the case of the considered auxiliary problems, the following result holds:

Algorithm 2: Coordinate descent method

Input: $y_0 \in \mathbb{R}^n$;
 $Z \leftarrow \sum_{i=1}^n (H + L_i)$;
 $p_i \leftarrow (H + L_i)/Z$, $i \in \{1, \dots, n\}$;
Discrete probability distribution π
with probabilities p_i ;
for $k = 0, \dots, N - 1$ **do**
 $i \sim \pi\{1, \dots, n\}$;
 $y_{k+1} \leftarrow y_k$;
 $y_{k+1}^i = y_k^i - \frac{1}{H + L_i} \nabla_i F(y_k)$;
end
return y_N ;

Theorem 1 ([4], theorem 6.8). *Let F be H -strongly convex function with respect to $\|\cdot\|_2$. Then for the sequence $\{y_k\}_{k=1}^N$ generated by the described coordinate descent algorithm 2, it holds the following inequality:*

$$\mathbb{E}[F(y_N)] - F(y_*) \leq \left(1 - \frac{1}{\kappa}\right)^N (F(y_0) - F(y_*)), \quad (4)$$

$$\text{where } \kappa = \frac{H}{Z}, \quad Z = \sum_{i=1}^n (H + L_i), \quad (5)$$

where $\mathbb{E}[\cdot]$ denotes the mathematical expectation of the specified random variable with respect to the randomness of methods trajectory induced by a random choice of components i at each iteration.

Using this result, lets formulate the following statement on the number of iterations of the coordinate descent method sufficient to satisfy the stopping condition (3).

Corollary 1. *The expectation $\mathbb{E}[y_N]$ of the point resulting from the coordinate descent method (Algorithm 2) satisfies the condition (3) if the following inequality on iterations number holds:*

$$N \geq N(\tilde{\varepsilon}) = \left\lceil \frac{Z}{H} \ln \left\{ \left(1 + \frac{L}{H}\right) \left(3 + \frac{2L}{H}\right)^2 \right\} \right\rceil, \quad (6)$$

$$\text{where } \tilde{\varepsilon} = \frac{H}{2} \left(\frac{H}{3H + 2L} \right)^2 \|y_0 - y_*\|_2^2. \quad (7)$$

Proof. Is in Appendix A¹.

¹ For the detailed proofs see appendices in full paper version on <https://arxiv.org/abs/2103.06688>.

Now that the question of the required accuracy and oracle complexity of solving the auxiliary problem using the proposed coordinate descent method is clarified, we can proceed to the results on the convergence of the Accelerated Meta-algorithm. For the used stopping condition (3), the following result on the convergence of the Accelerated Meta-algorithm holds.

Theorem 2 ([11], theorem 1). *For $H > 0$ and the sequence $\{v_k\}_{k=1}^{\tilde{N}}$ generated by the Accelerated Meta-algorithm with some non-stochastic internal method, it holds the following inequality:*

$$f(v_{\tilde{N}}) - f(x_*) \leq \frac{48}{5} \frac{H \|x_0 - x_*\|_2^2}{\tilde{N}^2}. \tag{8}$$

Based on the last statement, one can formulate a theorem on the convergence of the Accelerated Meta-algorithm in the case of using the stochastic method and, in particular, coordinate gradient descent method.

Theorem 3. *For $H > 0$ and some $0 < \delta < 1$, the point $v_{\tilde{N}}$ resulting from the Accelerated Meta-algorithm using coordinate descent method to solve the auxiliary problem, solving it within N_δ iterations, satisfies the condition*

$$Pr(f(v_{\tilde{N}}) - f(x_*) < \varepsilon) \geq 1 - \delta,$$

where $Pr(\cdot)$ denotes the probability of the specified event, if

$$\tilde{N} \geq \left\lceil \frac{4\sqrt{15}}{5} \sqrt{\frac{H \|x_0 - x_*\|_2^2}{\varepsilon}} \right\rceil, \tag{9}$$

$$N_\delta \geq N \left(\frac{\tilde{\varepsilon}\delta}{\tilde{N}} \right) = \left\lceil \frac{Z}{H} \ln \left\{ \frac{\tilde{N}}{\delta} \left(1 + \frac{L}{H} \right) \left(3 + \frac{2L}{H} \right)^2 \right\} \right\rceil. \tag{10}$$

Proof. The Corollary 1 presents an estimate of the number of iterations sufficient to satisfy the following condition for the expected value of the function at the resulting point of the method:

$$\mathbb{E}[F(y_{N(\tilde{\varepsilon})})] - F(y_*) \leq \tilde{\varepsilon}.$$

Lets use the Markov inequality and obtain the formulation of this condition in terms of the bound for the probability of large deviations [1]: deliberately choose the admissible value of the probability of non-fulfilment of the stated condition, so that $0 < \delta/\tilde{N} < 1$, where \tilde{N} expressed from (8); then

$$Pr \left(F \left(y_{N(\tilde{\varepsilon}\delta/\tilde{N})} \right) - F(y_*) \geq \tilde{\varepsilon} \right) \leq \frac{\delta}{\tilde{N}} \cdot \frac{\mathbb{E} \left[F \left(y_{N(\tilde{\varepsilon}\delta/\tilde{N})} \right) \right] - F(y_*)}{\tilde{\varepsilon} \cdot \delta/\tilde{N}} = \frac{\delta}{\tilde{N}}.$$

Since the probability that the obtained solution of some separately taken auxiliary problem will not satisfy the stated condition is equal to δ/\tilde{N} . It means that the probability that for \tilde{N} iterations of the Accelerated Meta-algorithm the condition will not be satisfied for at least one of the problems is $\tilde{N} \cdot \delta/\tilde{N} = \delta$, whence the proved statement follows.

Further, combining the estimates given in Theorem 3, we can obtain an asymptotic estimate for the total number of iterations of the coordinate descent method, sufficient to obtain a solution of the considered optimization problem with a certain specified accuracy, as well as an estimate for the optimal H value:

Corollary 2. *In order to the point $v_{\tilde{N}}$, which is the result of the Accelerated Meta-algorithm, to satisfy the condition*

$$\Pr(f(v_{\tilde{N}}) - f(x_*) < \varepsilon) \geq 1 - \delta,$$

it is sufficient to perform a total of

$$\hat{N} \geq \tilde{N} \cdot N_\delta = \mathcal{O} \left(\frac{Z \|x_0 - x_*\|_2}{\sqrt{H}} \cdot \frac{1}{\varepsilon^{1/2}} \log \left\{ \frac{1}{\varepsilon^{1/2} \delta} \right\} \right) \quad (11)$$

iterations of coordinate descent method to solve the auxiliary problem. In this case, the optimal value of the regularization parameter H of the auxiliary problem should be chosen as $H \simeq \frac{1}{n} \sum_{i=1}^n L_i$ (\simeq denotes equality up to a small factor of the log order).

Proof. The expression for \hat{N} can be obtained by the direct substitution of one of the estimates given in (9) into another, and their subsequent multiplication. If we exclude from consideration a small factor of order $\log(L/H)$, the constant in the estimate will depend on H as:

$$\sqrt{H} \cdot \frac{Z/n}{H} = \sqrt{H} \left(1 + \frac{\frac{1}{n} \sum_{i=1}^n L_i}{H} \right).$$

By minimizing the presented expression by H , we get the specified result.

A similar statement can be formulated for the expectation of the total iterations number without resorting to bound for the probabilities of large deviations, following the reasoning scheme proposed in [9]:

Theorem 4. *The expectation $\mathbb{E}[\hat{N}]$ of the sufficient total number of the iterations of the coordinate descent method, to obtain a point $v_{\tilde{N}}$ satisfying the following condition:*

$$f(v_{\tilde{N}}) - f(x_*) < \varepsilon$$

can be bounded as follows:

$$\mathbb{E}[\hat{N}] \leq \tilde{N} \cdot (N(\tilde{\varepsilon}) + 1) = \mathcal{O} \left(\sqrt{\frac{\bar{L} \|x_0 - x_*\|_2^2}{\varepsilon}} \right), \quad \text{where } \bar{L} = Z/n = \frac{1}{n} \sum_{i=1}^n L_i.$$

Proof. Is in Appendix B.

As we can see, the proposed scheme of reasoning allows us to reduce the logarithmic factor in estimate for the number of iterations of the method. However, this result is less constructive than the presented one in Corollary 2. Indeed, in

the above reasoning, we operated with the number of iterations N , after which the stopping condition for the internal method is satisfied. But in the program implementation, stopping immediately after fulfilment of this condition is not possible, if only because it is impossible to verify this criterion due to the natural lack of information about y_* . So the last result is more relevant from the point of view of evaluating the theoretical effectiveness of the method, while when considering specific practical cases, one should rely on the estimate (11).

Let us now consider in more detail the issue of the algorithmic complexity of the proposed accelerated coordinate descent method.

Theorem 5. *Let the complexity of computing one component of the gradient of f is $\mathcal{O}(s)$. Then the algorithmic complexity of the Accelerated Meta-algorithm with coordinate descent as internal method is*

$$T = \mathcal{O} \left(s \cdot n \cdot \sqrt{\frac{\bar{L} \|x_0 - x_*\|_2^2}{\varepsilon}} \log \left\{ \frac{1}{\varepsilon^{1/2} \delta} \right\} \right).$$

Proof. Is in Appendix C.

Note also that the memory complexity of the method is $\mathcal{O}(n)$, as well as the complexity of the preliminary calculations (for the coordinate descent method, there is no need to perform them again for every iteration).

Let us compare the estimates obtained for proposed approach (Catalyst CDM) with estimates for other methods that can be used to solve problems in the described setting: Fast Gradient Method (FGM), classical Coordinate Descent Method (CDM) and Accelerated Coordinate Descent Method in the version of Yu.E. Nesterov (ACDM). The estimates are shown in Table 1 below. As can be seen from the above asymptotic estimates of the computational complexity, the proposed method allows to achieve a convergence rate that is not inferior to other methods with respect to the dependence on the dimension n and the required accuracy ε (for a certain price, in the form of additional logarithmic factor). Note, in addition, that despite the significant similarity of estimates, between the two most efficient methods in the table (FGM and Catalyst CDM) there is also a difference in the constants characterising the smoothness of the function. Namely, L in FGM and \bar{L} in Catalyst CDM. Thus the behaviour of the considered method for various problems directly depends on the character of its component-wise smoothness.

2.2 Numerical Experiments

This section describes the character of the practical behaviour of the proposed method by the example of the following SoftMax-like function optimization problem:

$$\min_{x \in \mathbb{R}^n} \{f(x) = \gamma \ln \left(\sum_{j=1}^m \exp \left(\frac{[Ax]_j}{\gamma} \right) \right) - \langle b, x \rangle\}, \quad (12)$$

Table 1. Comparison of the effectiveness of methods

Algorithm	Iteration complexity	Comp. complexity	Source
FGM	$\mathcal{O}(s \cdot n)$	$\mathcal{O}\left(s \cdot n \cdot \frac{1}{\varepsilon^{1/2}}\right)$	[23]
CDM	$\mathcal{O}(s)$	$\mathcal{O}\left(s \cdot n \cdot \frac{1}{\varepsilon}\right)$	[4]
ACDM	$\mathcal{O}(n)$	$\mathcal{O}\left(n^2 \cdot \frac{1}{\varepsilon^{1/2}}\right)$	[24]
Catalyst CDM	$\mathcal{O}(s)$	$\tilde{\mathcal{O}}\left(s \cdot n \cdot \frac{1}{\varepsilon^{1/2}}\right)$	This paper

where $b \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, $\gamma \in \mathbb{R}_+$. Similar problems are essential for many applications. In particular, they arise in entropy-linear programming problems as a dual problem [8, 12], in particular in optimal transport problem, is also a smoothed approximation of the max function (which gave the function the name SoftMax) and, accordingly, of the norm $\|\cdot\|_\infty$, which may be needed in some formulations of the PageRank problem or for solving systems of linear equations. Moreover, in all the described problems, an important special case is the sparse setting, in which the matrix A is sparse, that is, the average number of nonzero entries in the row A_j does not exceed some $s \ll n$ (it will also be convenient to assume that the one of the rows A_j is non-sparse).

Let us formulate the properties possessed by the function f [13]:

1. f is differentiable;
2. ∇f satisfies the Lipschitz condition with the constant $L = \max_{j=1, \dots, m} \|A_j\|_2^2$;
3. $\nabla_i f$ satisfy the component-wise Lipschitz condition with the constants $L_i = \max_{j=1, \dots, m} |A_{ji}|$.

Let us write the expression for the i -th component of the gradient of the function f :

$$\nabla_i f(x) = \frac{\sum_{j=1}^m A_{ji} \exp([Ax]_j)}{\sum_{j=1}^m \exp([Ax]_j)}.$$

As we can see, the naive calculation of this expression can take time comparable to the calculation of the whole gradient and it will significantly affect the computational complexity and the working time of the method. At the same time, many terms in this expression can be recalculated either infrequently or in a component-wise manner, and used as additional sequences when performing a step of method. Using this approach, the complexity of the iteration will remain efficient, and the use of the coordinate descent methods will be justified. For the convenience of describing the computational methods used, we write the step of the coordinate descent algorithm in the form of

$$y_{k+1} = y_k + \eta e_i,$$

where η is the step size, multiplied by the corresponding gradient component, e_i is the i -th unit basis vector.

So, let us describe the additionally introduced computational procedures:

1. We will store a sequence of values $\left\{ \exp \left([Ay_k]_j \right) \right\}_{j=1}^m$, used to calculate the sum in the numerator. Updating these values after executing a method step takes $\mathcal{O}(s)$ algorithmic complexity, due to the $Ay_{k+1} = Ay_k + \eta A_i$ and that A_i has at most s nonzero components, which means that it will be necessary to calculate not more than s correcting factors and multiply the corresponding values from the sequence by them.
2. From the first point, it can be understood that the multiplication of sparse vectors should be performed in $\mathcal{O}(s)$, considering only nonzero components. In terms of program implementation, this means the need to use a sparse representation for cached values and for rows of the matrix A , that is, storing only index-value pairs for all nonzero elements. Then, obviously, the complexity of arithmetic operations for such vectors will be proportional to the complexity of a loop with elementary arithmetic operations, the number of iterations of which is equal to the number of nonzero elements (in the python programming language, for example, this storage format is implemented in the method `scipy.sparse.csr_matrix` [28]).
3. Similarly, we will store the value $\sum_{j=1}^m \exp \left([Ay_k]_j \right)$, which is the denominator of the presented expression. Its updating is carried out with the same complexity as updating a sequence (by calculating the sum of nonzero terms added to each value from the sequence).
4. Since evaluating the specified expression requires evaluating exponent values, the type overflow errors can occur. To solve this problem, it will be used the standard technique of exp-normalize trick [3]. However, to use it, one should also store the value $\max_{j=1, \dots, m} [Ay_k]_j$. At the same time, there is no need to maintain exactly this value. Indeed, it is sufficient to use only an approximation of it to keep the exponent values small, so this value can be recalculated much rarely: for example, once in m iterations (in this case, the amortized complexity will also be equal to $\mathcal{O}(s)$).

So, in the further reasoning, one can assume that the iteration of the coordinate descent algorithm for solving the corresponding auxiliary problem has amortized complexity $\mathcal{O}(s)$.

Further, let us consider in more detail the question of the values of the smoothness constants of this function. We can write down asymptotic formulas for L and $\bar{L} = \frac{1}{n} \sum_{i=1}^n L_i$:

$$L = \max_{j=1, \dots, m} \|A_j\|_2^2 = \mathcal{O}(n), \quad \bar{L} = \frac{1}{n} \sum_{i=1}^n \max_{j=1, \dots, m} |A_{ji}| = \mathcal{O}(1).$$

Using these estimates, let us refine the computational complexity of the FGM and Catalyst CDM (CCDM) methods as applied to this problem:

$$T_{FGM} = \mathcal{O}\left(s \cdot n^{3/2} \cdot \frac{1}{\varepsilon^{1/2}}\right), \quad T_{CCDM} = \tilde{\mathcal{O}}\left(s \cdot n \cdot \frac{1}{\varepsilon^{1/2}}\right).$$

Thus, in theory, the application of the Catalyst CDM method for solving this problem allows, in comparison with FGM, to reduce the factor of order $\mathcal{O}(\sqrt{n})$ in the asymptotic estimate of the computational complexity. In practice, this means that it is very reasonable to apply the proposed method to problems of large dimensions.

Let us now compare the performance of the proposed approach (Catalyst CDM) with a number of alternative approaches: Gradient Method (GM), Fast Gradient Method (FGM), Coordinate Descent Method (CDM) and Accelerated Coordinate Descent Method (ACDM), by the example of the problem (12) with an artificially generated matrix A in two different ways. Figures 1 and 2 present plots of the convergence of the methods under consideration: the x-axis shows the working time of the methods in seconds, and the y-axis shows the function value residual in a logarithmic scale (f_* calculated by searching for the corresponding point x_* using the FGM method, tuned for an accuracy that is obviously much higher than that possible to achieve at the selected time interval).

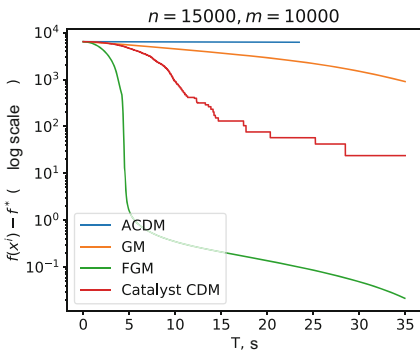


Fig. 1. Convergence of methods for the SoftMax problem (12) with a uniformly sparse random matrix.

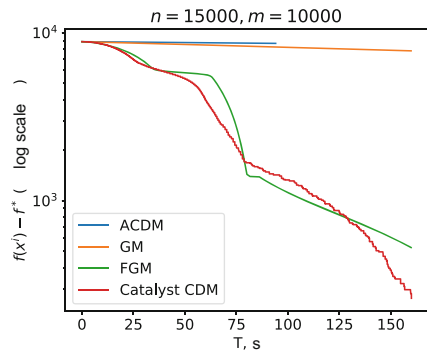


Fig. 2. Convergence of methods for the SoftMax problem (12) with heterogeneously sparse matrix.

In Fig. 1, the case is presented for which all elements of the matrix A are i.i.d. random variables from the discrete uniform distribution $A_{ji} \in \mathcal{U}\{0, 1\}$, the number of nonzero elements is $s \approx 0.2m$, and the parameter $\gamma = 0.6$ (as well as in the second case). In this setting, the proposed method demonstrates faster convergence compared to all methods under consideration, except the FGM. At the same time, in the setting shown in Fig. 2, in which the number of nonzero elements, in comparison with the first case, is increased to $s \approx 0.75 m$, and the

matrix is generated heterogeneously in accordance with the rule: $0.9m$ rows with $0.1n$ nonzero elements and $0.1m$ rows with $0.9n$ nonzero elements, and also one row of the matrix is completely nonsparse, the proposed method (Accelerated Meta-algorithm with coordinate descent as internal method) converges faster than FGM. This is explained by the fact that in this case $L = n$, but \bar{L} is still quite small and, as a result, the constant in the proposed method has a noticeably smaller effect on the computational complexity than in the case of FGM. From the results of the experiment, it can also be noted that the character of its componentwise smoothness affects the efficiency of the proposed method much more significantly than the sparseness of the problem.

2.3 Application to Optimization of Markov Decision Processes

We denote an MDP instance by a tuple $\mathcal{MDP} := (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$ with components defined as follows:

1. \mathcal{S} is a finite set of states, $|\mathcal{S}| = n$;
2. $\mathcal{A} = \bigcup_{i \in \mathcal{S}} \mathcal{A}_i$ is a finite set of actions that is a collection of sets of actions \mathcal{A}_i for states $i \in \mathcal{S}$, $|\mathcal{A}| = m$;
3. \mathcal{P} is the collection of state-to-state transition probabilities where $\mathcal{P} := \{p_{ij}(a_i) | i, j \in \mathcal{S}, a_i \in \mathcal{A}_i\}$;
4. r is the vector of state-action transitional rewards where $r \in [0, 1]^{\mathcal{A}}$, r_{i,a_i} is the instant reward received when taking action a_i at state $i \in \mathcal{S}$;
5. γ is the discount factor of MDP, by which one down-weights the reward in the next future step. When $\gamma \in (0, 1)$, we call the instance a discounted MDP (DMDP) and when $\gamma = 1$, we call the instance an average-reward MDP (AMDP).

Let us denote by $P \in \mathbb{R}^{\mathcal{S} \times \mathcal{S}}$ the state-transition matrix where its (i, a_i) -th row corresponds to the transition probability from state $i \in \mathcal{S}$ where $a_i \in \mathcal{A}_i$ to state j . Correspondingly we use \hat{I} as the matrix with a_i -th row corresponding to e_i , for all $i \in \mathcal{S}$, $a_i \in \mathcal{A}_i$. Our goal is to compute a random policy which determines which actions to take at each state. A random policy is a collection of probability distributions $\pi := \{\pi_i\}_{i \in \mathcal{S}}$, where $\pi_i \in \Delta_{|\mathcal{A}_i|}$, $\pi_i(a_j)$ denotes the probability of taking $a_j \in \mathcal{A}_i$ at state i . One can extend π_i to the set of Δ_m by filling in zeros on entries corresponding to other states $j \neq i$. Given an MDP instance $\mathcal{MDP} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$ and an initial distribution over states $q \in \Delta_n$, we are interested in finding the optimal π^* among all policies π that maximizes the following cumulative reward \bar{v}_π of the MDP:

$$\pi^* := \arg \max_{\pi} \bar{v}^\pi,$$

$$\bar{v}^\pi := \begin{cases} \mathbb{E}^\pi \left[\sum_{t=1}^{\infty} \gamma^{t-1} r_{i_t, a_t} \mid i_1 \sim q \right] & \text{in the case of DMDP,} \\ \lim_{T \rightarrow \infty} \frac{1}{T} \mathbb{E}^\pi \left[\sum_{t=1}^T r_{i_t, a_t} \mid i_1 \sim q \right] & \text{in the case of AMDP.} \end{cases}$$

For AMDP, \bar{v}^* is the optimal average reward if and only if there exists a vector $v^* = (v_i^*)_{i \in \mathcal{S}}$ satisfying its corresponding Bellman equation [2]:

$$\bar{v}^* + v_i^* = \max_{a_i \in \mathcal{A}_i} \left\{ \sum_{j \in \mathcal{S}} p_{ij}(a_i) v_j^* + r_{i,a_i} \right\}, \forall i \in \mathcal{S}.$$

For DMDP, one can show that at optimal policy π^* , each state $i \in \mathcal{S}$ can be assigned an optimal cost-to-go value v_i^* satisfying the following Bellman equation:

$$v_i^* = \max_{a_i \in \mathcal{A}_i} \left\{ \sum_{j \in \mathcal{S}} \gamma p_{ij}(a_i) v_j^* + r_{i,a_i} \right\}, \forall i \in \mathcal{S}.$$

One can further write the above Bellman equations equivalently as the following primal linear programming problems.

$$\begin{aligned} \min_{\bar{v}, v} \quad & \bar{v} & \text{s.t.} \quad & \bar{v} \mathbf{1} + (\hat{I} - P)v - r \geq 0 \quad (\text{LP AMDP}) \\ \min_v \quad & (1 - \gamma)q^\top v & \text{s.t.} \quad & (\hat{I} - \gamma P)v + r \geq 0 \quad (\text{LP DMDP}) \end{aligned}$$

By standard linear duality, we can recast the problem formulation using the method of Lagrangian multipliers, as bi-linear saddle-point (minimax) problem. The equivalent minimax formulations are

$$\min_{v \in \mathbb{R}^n} \left\{ F(v) = \max_{\mu \in \Delta_m} \left(\mu^\top ((P - \hat{I})v + r) \right) \right\} \quad (\text{AMDP})$$

$$\min_{v \in \mathbb{R}^n} \left\{ F_\gamma(v) = \max_{\mu \in \Delta_m} \left((1 - \gamma)q^\top v + \mu^\top ((\gamma P - \hat{I})v + r) \right) \right\} \quad (\text{DMDP})$$

Then, one can apply the Nesterov smoothing technique to the presented max-type function, according to [21]. (The calculation is presented for the case of DMDP, as a more general one. A smoothed version of the AMDP function is obtained similarly with the re-designations $A := P - \hat{I}$ and $\gamma = 1$):

$$\begin{aligned} F_\gamma(v) &= \max_{\mu \in \Delta_m} \left(\underbrace{(1 - \gamma)q^\top v}_b + \mu^\top \underbrace{((\gamma P - \hat{I})v + r)}_A \right) = \max_{\mu \in \Delta_m} \left(\sum_{j=1}^m \mu_j ([Av]_j + r_j) \right) + \langle b, v \rangle \\ &\rightarrow \max_{\mu \in \Delta_m} \left(\sum_{j=1}^m \mu_j ([Av]_j + r_j) - \sigma \sum_{j=1}^m \mu_j \ln \left(\frac{\mu_j}{1/m} \right) \right) + \langle b, v \rangle \\ &= \sigma \ln \left(\sum_{j=1}^m \exp \left(\frac{[Av]_j + r_j}{\sigma} \right) \right) - \sigma \ln m - \langle b, v \rangle =: f_\gamma(v), \text{ where } \sigma := \varepsilon / (2 \ln m). \end{aligned}$$

The resulting problem is of the form of SoftMax function, discussed in detail in the previous section. Taking into account the form of the matrix A, we can calculate the average component-wise Lipschitz constant:

$$P_{ji} \in [0, 1], \hat{I}_{ji} \in \{0, 1\} \implies \bar{L} = \frac{1}{\sigma} \frac{1}{n} \sum_{i=1}^n \max_{j=1, \dots, m} |A_{ji}| \leq \frac{\gamma}{\sigma} = \frac{2\gamma \ln m}{\varepsilon}$$

So, one can get the following estimates, which are also given in the Tables 2 and 3 (transition from the duality gap accuracy ε in matrix game setting to the $\tilde{\varepsilon}$ accuracy to obtain the $\tilde{\varepsilon}$ -approximate optimal policy satisfying the condition in expectation $\mathbb{E}\bar{v}^\pi \geq \bar{v}^* - \tilde{\varepsilon}$ is carried out according to the rules described in more detail in the paper [15]):

$$\begin{aligned} \varepsilon \sim \frac{1}{2} \cdot \frac{\tilde{\varepsilon}}{3} &\implies T_{CCDM} = \tilde{\mathcal{O}}\left(\text{nnz}(P)\sqrt{\log m} \cdot \tilde{\varepsilon}^{-1}\right), \\ \varepsilon_\gamma \sim \frac{1}{2} \cdot \frac{(1-\gamma)\tilde{\varepsilon}}{3} &\implies T_{CCDM,\gamma} = \tilde{\mathcal{O}}\left(\gamma^{1/2}(1-\gamma)^{-1} \cdot \text{nnz}(P)\sqrt{\log m} \cdot \tilde{\varepsilon}^{-1}\right), \end{aligned}$$

where $\text{nnz}(P) \leq n \cdot m$ denotes the number of nonzero elements in matrix P . In addition to the result corresponding to the considered method, the Tables 2 and 3 contain complexity bounds of other known approaches to solve matrix games and MDP problems (for the sake of compactness, it is used the notation $\text{nnz}'(P) = \text{nnz}(P) + (m+n) \log^3(mn)$). It can be seen that for the case of $\gamma = 1$, the described approach allowing to obtain one of the best among the known estimates, and in the case of $\gamma < 1$ it is close in efficiency to many modern approaches. Moreover, to describe the method used in this article, it was enough to apply only a special case of the universal accelerated proximal envelope for the classical coordinate descent method. This approach is conceptually much simpler than the other methods cited here (which, by the way, are often applicable only to very particular settings), and allows one to obtain complexity bounds for AMDP problem that notably competitive with the best alternatives.

Table 2. Comparison of the effectiveness of approaches ($\gamma = 1$ case)

Computational complexity	Source
$\tilde{\mathcal{O}}\left(\text{nnz}(P)\sqrt{\log m} \cdot \tilde{\varepsilon}^{-1}\right)$	This paper
$\tilde{\mathcal{O}}\left(\text{nnz}(P)\sqrt{m/n} \cdot \tilde{\varepsilon}^{-1}\right)$	[6]
$\tilde{\mathcal{O}}\left(\log^3(mn) \sqrt{\text{nnz}(P) \cdot \text{nnz}'(P)} \cdot \tilde{\varepsilon}^{-1}\right)$	[7]

Table 3. Comparison of the effectiveness of approaches ($\gamma \in (0, 1)$ case)

Computational complexity	Source
$\tilde{\mathcal{O}}\left(\gamma^{1/2}(1-\gamma)^{-1} \text{nnz}(P)\sqrt{\log m} \cdot \tilde{\varepsilon}^{-1}\right)$	This paper
$\tilde{\mathcal{O}}\left(\gamma(1-\gamma)^{-1} \text{nnz}(P)\sqrt{m/n} \cdot \tilde{\varepsilon}^{-1}\right)$	[6]
$\tilde{\mathcal{O}}\left(\gamma(1-\gamma)^{-1} \log^3(mn) \sqrt{\text{nnz}(P) \cdot \text{nnz}'(P)} \cdot \tilde{\varepsilon}^{-1}\right)$	[7]
$\tilde{\mathcal{O}}\left(nm (n + (1-\gamma)^{-3}) \cdot \log(\tilde{\varepsilon}^{-1})\right)$	[29]

3 Conclusion

In this paper, we propose a version of the Coordinate Descent Method, accelerated using the universal proximal envelope “Accelerated Meta-algorithm”. The performed theoretical analysis allows us to assert that the dependence of its computational complexity on the dimensionality and required solution accuracy is not inferior to other methods used to optimize convex Lipschitz smooth functions. Moreover, its computational complexity is comparable to that of the Fast Gradient Method. At the same time, the proposed scheme retains the properties of the classical Coordinate Descent Method, including the possibility of using the properties of component-wise smoothness of the function. The given numerical experiments confirm the practical efficiency of the method, and also emphasize the particular relevance of the proposed approach for the SoftMax-like function optimization problem that often arises in various applications. As an example of such an application, it was considered the problem of optimizing the MDP, and using the described approach, a method was proposed for solving the averaged version of the MDP problem, which gives a complexity bound that competes with the most efficient known ones.

References

1. Anikin, A., et al.: Modern efficient numerical approaches to regularized regression problems in application to traffic demands matrix calculation from link loads. In: Proceedings of International conference ITAS-2015. Russia, Sochi (2015)
2. Bertsekas, D.P.: Dynamic Programming and Optimal Control. Athena Scientific, Belmont, MA (1995)
3. Blanchard, P., Higham, D.J., Higham, N.J.: Accurately Computing the Log-Sum-Exp and Softmax Functions (2019)
4. Bubeck, S.: Convex optimization: algorithms and complexity. arXiv preprint [arXiv:1405.4980](https://arxiv.org/abs/1405.4980) (2014)
5. Bubeck, S., Jiang, Q., Lee, Y.T., Li, Y., Sidford, A.: Near-optimal method for highly smooth convex optimization. In: Conference on Learning Theory, pp. 492–507. PMLR (2019)
6. Carmon, Y., Jin, Y., Sidford, A., Tian, K.: Variance reduction for matrix games. In: Advances in Neural Information Processing Systems, pp. 11381–11392 (2019)
7. Carmon, Y., Jin, Y., Sidford, A., Tian, K.: Coordinate methods for matrix games. arXiv preprint [arXiv:2009.08447](https://arxiv.org/abs/2009.08447) (2020)
8. Chernov, A.: Direct-dual method for solving the entropy-linear programming problem. *Intell. Syst. Theor Appl.* **20**(1), 39–59 (2016)
9. d’Aspremont, A., Scieur, D., Taylor, A.: Acceleration methods (2021)
10. Doikov, N., Nesterov, Y.: Contracting proximal methods for smooth convex optimization. *SIAM J. Optim.* **30**(4), 3146–3169 (2020)
11. Dvinskikh, D., et al.: Accelerated meta-algorithm for convex optimization. arXiv preprint [arXiv:2004.08691](https://arxiv.org/abs/2004.08691) (2020)
12. Gasnikov, A., Gasnikova, E., Nesterov, Y., Chernov, A.: Efficient numerical methods for entropy-linear programming problems. *Comput. Math. Math. Phys.* **56**(4), 523–534 (2016)

13. Gasnikov, A.: Universal gradient descent. arXiv preprint [arXiv:1711.00394](https://arxiv.org/abs/1711.00394) (2017)
14. Ivanova, A., Pasechnyuk, D., Grishchenko, D., Shulgin, E., Gasnikov, A., Matyukhin, V.: Adaptive catalyst for smooth convex optimization. arXiv preprint [arXiv:1911.11271](https://arxiv.org/abs/1911.11271) (2019)
15. Jin, Y., Sidford, A.: Efficiently solving MDPs with stochastic mirror descent. In: International Conference on Machine Learning, pp. 4890–4900. PMLR (2020)
16. Kamzolov, D., Gasnikov, A., Dvurechensky, P.: On the optimal combination of tensor optimization methods. arXiv preprint [arXiv:2002.01004](https://arxiv.org/abs/2002.01004) (2020)
17. Kulunchakov, A., Mairal, J.: A generic acceleration framework for stochastic composite optimization. In: Advances in Neural Information Processing Systems. pp. 12556–12567 (2019)
18. Lin, H., Mairal, J., Harchaoui, Z.: A universal catalyst for first-order optimization. *Adv. Neural Inf. Process. Syst.* **28**, 3384–3392 (2015)
19. Lin, H., Mairal, J., Harchaoui, Z.: Catalyst acceleration for first-order convex optimization: from theory to practice. *J. Mach. Learn. Res.* **18**(1), 7854–7907 (2017)
20. Monteiro, R.D., Svaiter, B.F.: An accelerated hybrid proximal extragradient method for convex optimization and its implications to second-order methods. *SIAM J. Optim.* **23**(2), 1092–1125 (2013)
21. Nesterov, Y.: Smooth minimization of non-smooth functions. *Math. Program.* **103**(1), 127–152 (2005)
22. Nesterov, Y.: Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM J. Optim.* **22**(2), 341–362 (2012)
23. Nesterov, Y.: Lectures on Convex Optimization. SOIA, vol. 137. Springer, Cham (2018). <https://doi.org/10.1007/978-3-319-91578-4>
24. Nesterov, Y., Stich, S.U.: Efficiency of the accelerated coordinate descent method on structured optimization problems. *SIAM J. Optim.* **27**(1), 110–123 (2017)
25. Paquette, C., Lin, H., Drusvyatskiy, D., Mairal, J., Harchaoui, Z.: Catalyst acceleration for gradient-based non-convex optimization. arXiv preprint [arXiv:1703.10993](https://arxiv.org/abs/1703.10993) (2017)
26. Parikh, N., Boyd, S.: Proximal algorithms. *Found. Trends Optim.* **1**(3), 127–239 (2014)
27. Rockafellar, R.T.: Monotone operators and the proximal point algorithm. *SIAM j. Control Optim.* **14**(5), 877–898 (1976)
28. SciPy.org: Python Scipy documentation: `scipy.sparse.csr_matrix` (2020) https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.csr_matrix.html. Accessed 5 Jan 2021
29. Sidford, A., Wang, M., Wu, X., Ye, Y.: Variance reduced value iteration and faster algorithms for solving Markov decision processes. In: Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms. pp. 770–787. SIAM (2018)