Commun. Optim. Theory 2021 (2021), Article ID 13 https://doi.org/10.23952/cot.2021.13

MATHRES Communications in Optimization Theory
Available online at http://cot.mathres.org

NUMERICAL STUDY ON A CLASS OF POLYAK'S NONLINEAR RESCALING DECOMPOSITION ALGORITHMS

ABDELOUAHED HAMDI[1,*], AKRAM TAATI[2], TEMADHER A. ALMAADEED[1]

[1]Department of Mathematics, Statistics and Physics, College of Arts and Sciences, Qatar University, Doha, Qatar
[2]Department of Applied Mathematics, Faculty of Mathematical Science, University of Guilan, Rasht, Iran

Abstract. In this paper, we investigate the numerical efficiency of a decomposition algorithm to solve structured optimization problems. The proposed method is based on a class of modified Lagrangians combined with the nonlinear rescaling principle of Polyak and the allocation of resources decomposition algorithm. The influence of the involved parameters in the convergence of the algorithm is discussed and some comparisons with other famous decomposition algorithms are given.

Keywords. Augmented Lagrangian algorithm; Decomposition algorithm; Nonlinear rescaling; Numerical efficiency.

1. INTRODUCTION

There has been considerable recent interest in solving large-scale optimization models. Such problems arise in many fields, for instance, stochastic optimization, optimal structural design, transportation, telecommunication models, networks, and deep learning. Recently, the technology of fast computers providing parallel computations encouraged researchers to continue tackling large-scale problems. It is known that the classical coordination functions are in general non-smooth and therefore hard to be optimized. The computational cost is not the only drawback as the non-smoothness of the coordination function. It always implies that non-unique solutions are introduced in the sub-problems which reduces the possibility to decentralize them completely. These are the reasons why some penalization ideas, regularization techniques like the proximal point method look attractive. In addition to the possibility to treat non-smooth convex coordination problems efficiently, the introduction of the quadratic penalty terms in the subproblems can impose the unique solution to guarantee decentralized procedures. Classical primal penalty quadratic methods, for example, Kort and Bertsekas [1], and Polyak [2], lose the separability, which allows decomposition of the sub-problems' penalized objective when they are applied to large separable constrained models. The same diagnostic can be reached by

*Corresponding author.
E-mail addresses: abhamdi@qu.edu.qa (A. Hamdi), akram.taati@qu.edu.qa (A. Taati), t.alassiry@qu.edu.qa (T.A. Almaadeed)
Received August 23, 2020; Accepted October 31, 2021.

applying many other penalty schemes, in particular, the Hyperbolic penalty method introduced and studied by Xavier [3].

The paper is motivated by the effective theoretical and numerical results obtained by Polyak [2, 4], Polyak proposed a nonlinear re-scaling algorithm based on many kernels, for instance, the log-sigmoid, the modified barrier, the exponential and many other functionals. In 2012, Hamdi and Mukheimer [5] mixed the nonlinear rescaling concept with some decomposition algorithms developed by Hamdi et al. [6] and Hamdi [7]. These decomposition class of methods, which are known as separable augmented Lagrangian algorithms (SALA), can be derived from the resource directive subproblems associated with the coupling constraints. A more complete review of decomposition methods for convex and non-convex optimization problems can be found in Hamdi and Mishra [8] and the references therein. Our objective in this paper is to present a numerical study on the performance of φ-separable augmented Lagrangian algorithm (φ-SALA), proposed and studied theoretically in Hamdi and Mukheimer [8]. The algorithm, which is based on proximal-like techniques, is suitable for decentralized and parallelized computations. The φ-SALA can be seen as a separable version of the nonlinear rescaling principle method, and is closely related to the Separable Augmented Lagrangian Algorithm developed in Hamdi et al. [6] and Hamdi [7].

In addition, the proposed algorithm solves the second drawback associated with the multiplier methods when solving inequality-constrained problems in general. In other words, the classical augmented Lagrangian is once differentiable even when the data of the problem allow for higher differentiability, disabling the application of efficient Newton type methods. In fact, such a lack of continuity in the second derivative can significantly slow down the rate of convergence of these methods and thus cause algorithmic failure. Other ideas to cope with this difficulty can be found in, for example, Auslender and Teboulle [9]. Indeed, they suggested new Prox-like techniques, and the entropic proximal decomposition method (EPDM), which leads to a C^{∞}-Lagrangian for solving the structured convex minimization problems and some variational inequalities. In [10], Kyono and Fukushima proposed an extension of the Chen-Teboulle decomposition scheme combined with the Bregman-based proximal point algorithm for solving large-scale Variational Inequalities (VIPs). For more decomposition schemes to solve VIPs, we refer to some recent works in Bnouhachem and Hamdi [11] and Bnouhachem et al. [12] and the references therein. Recently, Jung et al. [13] proposed an efficient algorithm based on modified alternating multipliers method, which is favorable to decomposition. Their algorithm used the sub-gradient scheme with adaptive step sizes in updating the Lagrange multiplier and also kept the penalty parameter at an appropriate level not to cause oscillation. Recently, Palanduz and Growenwold [14] proposed and studied an iterative separable augmented Lagrangian algorithm for optimal structural design. Their decomposable multipliers method was for solving equality constrained models.

The remainder of this paper is organized as follows. In Section 2, we present the nonlinear rescaling principle of Polyak. In Section 3, we describe the developed and studied algorithm $\varphi SALA$ (Hamdi and Mukheimer [5]). Section 4 is dedicated to the main contribution of this paper: the numerical study of the $\varphi SALA$, the exponential version followed by some comparisons with the EPDM (Auslender and Teboulle [9]), the Hyperbolic Decomposition Algorithm (HDA) (Hamdi et al. [15]), and with the Inexact Dual fast Gradient-Projection Method (IDFGPM) for linearly coupled constraints (Li et al. [16]).

Finally, we need to mention that, in engineering, we encounter many concrete applications of structured problem, for instance, the multi-commodity flow problems with separable convex costs. These problems arise in the optimal routing of data networks and the cost function represents the average delay of the data packets when crossing the network. For example, the following model considers the arc-path formulation:

$$\begin{aligned}
&\text{Minimize} \quad \sum_{j=1}^{n} f_j(x_{0j}) \\
&\sum_{k=1}^{K}\sum_{p=1}^{N_k} \pi_{kp}(j)\,(x_k)_p = x_{0j},\ j=1,2,\cdots,n \\
&\sum_{p=1}^{N_k} x_{kx_p} = r_k,\ \ k=1,2,\cdots,K \\
&0 \le x_{0j} < c_j,\ \ j=1,2,\cdots,n \\
&0 \le x_{k_p}\ \ \forall k,\ \forall p.
\end{aligned}$$

where N_k is the number of paths of the kth commodity, x_{kp} is the quantity of commodity k using the pth path of k, and K is the total number of commodities. Also, $\pi_{kp}(j)=1$ if path p for commodity k uses arc j, and $\pi_{kp}(j)=0$, elsewhere; c_j is the capacity of arc j, x_{0j} is the total flow present on arc j, n is the number of arcs, and r_k is flow requirement for commodity k. The arc cost functions f_j are supposed to be convex and sufficiently smooth.

2. Polyak's Nonlinear Rescaling Principle

Let f be a convex real-valued function and let $g_1, g_2, \cdots, g_p$ be finite concave real-valued functions on $\Re^n$. Consider the convex programming problem:

$$\min\{f(x)\ :\ g_i(x)\ge 0,\ i=1,2,\cdots,p,\ x\in\Re^n\}. \tag{2.1}$$

The main idea of the nonlinear re-scaling principle (the NR method) is to consider a class of strictly concave and smooth enough scalar function with particular properties, and use it to transform the constraint terms of the classical Lagrangian. At each step, the unconstrained minimization of the classical Lagrangian for the equivalent problem alternates with the Lagrange multiplier update. It allows to generate a wide class of augmented Lagrangian methods, for instance, the exponential multiplier method (Kort and Bertsekas [1]), the modified log-barrier method (Polyak [2]), and the Log-Sigmoid multiplier method (Polyak [4]), etc. Let us consider the following class Φ of C^2 functions φ defined on $\Re$ with the following properties:

- **(P1)** $\psi'(t)>0$, for all $t\in\Re$;
- **(P2)** $\psi''(t)<0$, for all $t\in\Re$;
- **(P3)** $\psi(0)=0$;
- **(P4)** $\psi'(0)=1$;
- **(P5)** $m^{-1}\le\psi''(t)<0$, $\forall t\in\Re$ and $\psi''(t)\le -M^{-1}$, $\forall t<0$, where $M,m>0$;
- **(P6)** $(-\psi)_\infty(-1)\ge 0$;
- **(P7)** $(-\psi)_\infty(1)=0$.

Remark 2.1. we list some kernels ψ in the references as follows

Name	Expression	Ref.	Remarks
Exponential	$\psi(t) = 1 - e^{-t}$	Kort and Bertsekas [1]	No **(P5)-1**
Logarithmic (MBF)	$\psi(t) = \ln(1+t)$	Polyak [2]	No **(P5)-1**
Hyperbolic	$\psi(t) = \dfrac{t}{t+1}$	Polyak [2]	No **(P5)-1**
Log-Sigmoid	$\psi(t) = 2(\ln(2) + t - \ln(1+e^t))$	Polyak [4]	No **(P5)-2**
Modified (CHKS)	$\psi(t) = t - \sqrt{t^2+4\nu} + 2\sqrt{\nu},\ \nu > 0$	Polyak [4]	No **(P5)-2**

Many authors proposed to use the following re-built kernels that are twice continuously differentiable and satisfies the whole properties. Given $\tau \in (-1,\ 0)$,

$$\varphi(t) = \begin{cases} \psi(t), & \text{if} \quad t \geq \tau, \\ q_i(t) = a_i t^2 + b_i t + c_i, & \text{if} \quad t \leq \tau, \end{cases}$$

where the coefficients a_i, b_i and c_i can be determined by solving these equations

$$\begin{cases} \psi_i'(\tau) &= q_i(\tau), \\ \psi_i'(\tau) &= q_i'(\tau), \\ \psi_i''(\tau) &= q_i''(\tau). \end{cases}$$

Note that, from $(P1)-(P3)$, ψ and ψ' are one-to-one, and ψ is strictly concave. Let $\psi \in \Phi$. Then, for any $\lambda > 0$,

$$t \geq 0 \iff \frac{1}{\lambda}\varphi(\lambda t) \geq 0. \tag{2.2}$$

The nonlinear re-scaling principle is based on the idea of transforming the original problem (2.1) to an equivalent problem, namely, to one which has the same set of optimal solutions as (2.1). To this end, let us consider here a parameterized transformed problems written as follows:

$$\min\left\{ f(x) \ :\ \frac{1}{\lambda}\varphi(\lambda\, g_i(x)) \geq 0,\ i = 1,2,\cdots,p,\ x \in \Re^n \right\}. \tag{2.3}$$

Clearly, problem (2.3) is also convex and has the same feasible set as (2.1). The nonlinear re-scaling iterations are based on the classical Lagrangian function denoted here by $P_\lambda(x,u)$ associated with problem (2.3), that is,

$$P_\lambda(x,u) = f(x) - \sum_{i=1}^{p} \frac{1}{\lambda} u_i \varphi(\lambda\, g_i(x)),$$

and can be resumed as follows.

Algorithm 2.1. Nonlinear Rescaling (NR)
Initialization: Give $\varphi \in \Phi$, $u_0 > 0$, and $\lambda > 0$. `Do`

- **•:** $x^{k+1} \in \operatorname{argmin} P_\lambda(x, u^k)$.
- **•:** Update $u_i^{k+1} = u_i^k\, \varphi'\left(\lambda\, g_i(x^{k+1})\right),\ i = 1,2,\cdots,p.$

`Until` stopping criteria satisfied.

Remark 2.2. Note that the multipliers are non-negative for all k by (P_1). Also, it is worth to mention the possibility to change the penalty parameter λ at each iteration k. In Polyak [17], it was proposed that a dynamic scaling parameters updated as follows: $\lambda_i = \dfrac{\lambda}{u_i^{k+1}},\ i = 1,2,\cdots,p.$ This update will be used in our decomposition scheme in the numerical part of this paper.

The NR algorithm allows a generation of a wide class of augmented Lagrangian. In Section 4, we studied the exponential version of NR with $\widehat{\varphi}_1$ and $\varphi_1(t) = 1 - e^{-t}$ for the decomposition proposed algorithm numerically. The convergence analysis of Algorithm 2.1 is given in Polyak [2, 4, 17].

3. φ-Separable Augmented Lagrangian Algorithm

The generic model for the decomposition purpose is here a constrained separable problem with separable coupling inequality constraints:

$$\begin{aligned} \text{Minimize} \quad & \sum_{i=1}^{p} f_i(x_i) \\ & \sum_{i=1}^{p} g_{ij}(x_i) \geq 0, \ j = 1,2,\cdots,m, \end{aligned} \tag{3.1}$$

where $x_i \in \Re^{n_i}, i = 1,\cdots,p$, are the i'th block variables, $f_i : \Re^{n_i} \to \Re, i = 1,\cdots,p$ are the i'th block objective functions, and $g_i : \Re^{n_i} \to \Re^m, i = 1,\cdots,p$ are the i'th block constraint vector functions.

In this section, we only give the steps of the φ-separable augmented Lagrangian algorithm built, study theoretically in Hamdi and Mukheimer [5].

Algorithm 3.1. φ-Separable Augmented Lagrangian Algorithm (φ-SALA)
`Step 1.` Select $\varphi \in \Phi$, $\lambda > 0$, $u^0 \in \Re^m_+$ and $y^0 \in \Re^{pm}$, $y^0 = (y^0_1,\cdots,y^0_p)^\top$, s.t $\sum_{i=1}^{p} y^0_i = 0$, $\lambda^0_j = \lambda (u^0_j)^{-1}$, $\varepsilon_1, \varepsilon_2, \varepsilon_3 > 0$.
`Step 2.` Determine for each $i = 1,2,\cdots,p$

$$x^{k+1}_i := \arg\min_{x_i \in \Re^{n_i}} \left\{ f_i(x_i) - \sum_{j=1}^{m} \frac{1}{\lambda^k_j} u^k_j \varphi\left(\lambda^k_j \left(g_{ij}(x_i) + y^k_{ij}\right)\right) \right\}.$$

`If Stopping criteria satisfied` Stop.
`Otherwise` Go to next step

`Step 3.` Update and go back to the minimization in Step 2 (if $\delta^{k+1}_j = p^{-1} \sum_{i=1}^{p} g_{ij}(x^{k+1}_i)$)

$$\begin{cases} y^{k+1}_{ij} = -g_{ij}(x^{k+1}_i) + \delta^{k+1}_j, \ i = 1,2,\cdots,p, \ j = 1,2,\cdots,m \\ u^{k+1}_j = u^k_j \varphi'\left(\lambda^k_j \delta^{k+1}_j\right), \ i = 1,2,\cdots,p. \\ \lambda^{k+1}_j = \lambda (u^{k+1}_j)^{-1}, \ j = 1,2,\cdots,m \end{cases}$$

Remark 3.1. The φ-SALA is closely related to other decomposition methods, which appeared in the recent literature. The alternate direction method of multipliers (ADMM) first proposed by Gabay and Mercier [18] and later exploited in the context of distributed computations in Bertsekas and Tsitsiklis [19], Eckstein and Fukushima [20], Fukushima [21] and Jung et al. [13]. Some similar algorithms were applied to block-structured linear problems (see Gol'stein [22]) and Ha [23]. It is important to emphasize the role of the constraint structure in applying a decomposition procedure and its influence on the performance of the resulting algorithm. The φ-SALA above has the following properties:

(a) the sub-problem in Step 2 may be interpreted as minimizing the Lagrangian associated with the sub-problem of the classical resource-directive decomposition method. The difference stands in the specific updates of the primal and dual allocations.
(b) Step 2 is at the same time a stopping criterion and a coordination task as it must centralize the local proposals to test their global feasibility. All other steps are totally decentralized and parallelizable;
(c) the use of NR in φ-SALA avoids any requirement of indefinitely increase of the scaling parameter λ;
(d) since φ-SALA can be seen as an exterior, interior penalization in the primal, and dual spaces, it does not need any particular numerical effort to fulfil the initialization step.

4. Numerical Experiments

This section is devoted to some numerical tests, where we study the computational behaviour of the exponential version of φ-separable augmented Lagrangian algorithm (ESALA) where $\psi = 1 - e^{-t}$ and make a comparison with the inexact dual fast gradient-projection method (IDFGPM) (Li et al. [16]), the hyperbolic decomposition algorithm (HDA) (Hamdi et al. [15]), the entropic proximal decomposition method (EPDM) (Auslender and Teboulle [9]) and CVX, a package for specifying and solving convex programs (Grant and Boyed [24]). All experiments are performed in MATLAB R2019a on a laptop with 1.8 GHz CPU and 16 GB of RAM. Below, we recall IDFGPM, HDA and EPDM.

Algorithm 4.1. Inexact Dual Fast Gradient-Projection Method (IDFGPM)
`Step 1.` Given the outer target accuracy ε, set $\lambda_0 = \lambda_{-1} = 0 \in \Re^m$, $\theta_0 = \theta_{-1} = 1$, and $x_{-1} = 0$. Compute the number of outer iteration k_ε and the inner accuracy δ_i as in (4.1). $k = 0$.
`Step 2.` While $k < k_\varepsilon$, calculate $\mu^k = \lambda^k + \theta_k \left(\frac{1}{\theta_{k-1}-1}\right)(\lambda_k - \lambda_{k-1})$.
`Step 3.` Execute Inner loop:
For $i = 1$ *to* N,
choose $x_i^{k,0} = \bar{x}_i^{k-1}$ and Compute p_i as in (4.2).
For $p = 0, 1, \cdots, \lfloor p_i \rfloor$, apply Algorithm **(IFG)** to obtain: $\tilde{x_i^k} = x_i^{k,\lfloor p_i \rfloor}$.
`Step 4.` Compute approximate gradient $\tilde{\nabla} d(\mu^k) = \sum_{i=1}^N A_i \tilde{x_i^k} - b$, and update λ^{k+1} and θ_{k+1} by (4.3) and (4.4), respectively.
`Step 5.` Update averaged primal sequence $\bar{x}_i^k = (1-\theta_k)\bar{x}_i^{k-1} + \theta_k \tilde{x_i^k}$.

Inexact Fast Gradient Method (IFG)
Given μ_k, $\forall i = 1, \cdots, N$, choose $y_i^{k,0} = x^{k,0} \in X_i$ and set $\beta_i = \dfrac{\sqrt{L_i} - \sqrt{\sigma_i}}{\sqrt{L_i} + \sqrt{\sigma_i}}$. $k = 0$.
While (Stoping criteria unsatisfied)
For $1, \cdots, N$, execute in parallel:

$$\begin{cases} x_i^{k,p+1} = P_{X_i}\left[y^{k,p_i} - \frac{1}{L_i}\nabla_1 \mathscr{L}_i(y^{k,p}; \mu_k)\right], \\ y_i^{k,p+1} = x_i^{k,p+1} + \beta_i (x_i^{k,p+1} - x_i^{k,p}), \end{cases}$$

$$k_\varepsilon = \frac{2D_\lambda\sqrt{L_d}}{\sqrt{\varepsilon}} \quad , \quad \delta_i = \frac{\varepsilon\sqrt{\varepsilon}}{2ND_\lambda\sqrt{L_d}} \tag{4.1}$$

$$p_i = 1+\sqrt{\frac{L_i}{\sigma_i}}\ln\left(\frac{D_i^2(L_i+\sigma_i)}{\delta_i}\right) \tag{4.2}$$

$$\lambda^{k+1} = \left[\mu_k+\frac{1}{2L_d}\tilde{\nabla}d(\mu_k)\right]_+ \tag{4.3}$$

$$\theta^{k+1} = \frac{\sqrt{(\theta^k)^4+4(\theta^k)^2}-(\theta^k)^2}{2} \tag{4.4}$$

Algorithm 4.2. Hyperbolic Decomposition Algorithm (HDA)
`Step 1.` Select $\tau_0>0,\ \lambda_0>0,\ y^0=(y_1^0,\cdots,y_p^0),$ s.t $\sum_{i=1}^p y_i^0=0,\ \ \varepsilon_1>0,\ \varepsilon_2>0,\ r>1,\ 0<q<1,\ \ k=0$
`Step 2.` While Stopping criteria unsatisfied, determine for each $i=1,2,\cdots,p$

$$x_i^{k+1} := \arg\min_{x_i\in\Re^{n_i}}\left\{f_i(x_i)-\lambda_k\left(g_i(x_i)+y_i^k\right)+\sqrt{\tau_k^2+\lambda_k^2\left(g_i(x_i)+y_i^k\right)^2}\right\}$$

`Step 3.` If Stopping criteria satisfied, stop.
Otherwise, go to next step `Step 4` Update and go back to the minimization Step 1 (if $\delta^{k+1}=\frac{1}{p}\sum_{i=1}^p g_i(x_i^{k+1})$

$$\begin{cases} y_i^{k+1}=-g_i(x_i^{k+1})+\delta^{k+1}, \quad i=\overline{1,p}, \\ \\ \lambda_{k+1}=r\,\lambda_k, \ \ \tau_{k+1}=q\,\tau_k. \end{cases}$$

Algorithm 4.3. Entropic Proximal Decomposition Method (EPDM)
Given φ defined in $(E1)$, $(v_i^0,p_i^0,y_i^0)\in\Re^{n_i}\times\Re^m\times\Re^m,\ (u^0,w^0)\in\Re^{pm}_{++}\times\Re^M_{++}$ and $\lambda_k>0,\ k=0$. While Stopping criteria unsatisfied, generate for $i=1,\cdots,p$ the sequences $\{p_i^k;v_i^k;y_i^k\}$ and the positive sequences $\{w^k,u_i^k\}$ according to:
`Step 1.` Compute $p_i^{k+1}=y_i^k+(2\theta)^{-1}(w^k-u_i^k),\ \ i=1,2,\cdots,p.$
`Step 2.` For each $i=1,2,\cdots,p$, find the unique minimizer v_i^{k+1} solution of

$$v_i^{k+1}=\arg\min\left\{H_i^k(u^k,v_i),\ \ v_i\in\Re^{n_i}\right\},$$

where

$$H_i^k(u^k,v_i)=f_i(v_i)+\frac{1}{\lambda_k}\sum_{j=1}^m(u_{ji}^k)^2\,\varphi^*\left(\lambda_k\big(-g_{ij}(v_i)+p_{ji}^{k+1}\big)/u_{ji}^k\right).$$

`Step 3.` Compute

$$\begin{cases} w_j^{k+1}=w_j^k\big(\varphi^*\big)'\left(-\lambda_k\sum_{i=1}^p p_{ji}^{k+1}/w_j^k\right),\ j=1,2,\cdots,m, \\ u_{ji}^{k+1}=u_{ji}^k\big(\varphi^*\big)'\left(\lambda_k\big(-g_{ij}(v^{k+1}+p_{ji}^{k+1})\big)/u_{ji}^k\right),\ j=1,2,\cdots,m, \\ y_i^{k+1}=y_i^k+(2\theta)^{-1}(w^{k+1}-u_i^{k+1}),\ \ i=1,2,\cdots,p. \end{cases}$$

$$\varphi(t)=\begin{cases}\dfrac{\nu}{2}(t-1)^2+\mu(t-\ln(t)-1), & \text{if } t>0,\\ \infty, & \text{otherwise,}\end{cases} \tag{E1}$$

where $\nu>\mu>0$ are given fixed parameters.

4.1. **Problems and implementation.** In this subsection, we present some computational tests on the performance of the exponential version of Algorithm 2 (ESALA) when

$$\varphi(t) = \begin{cases} \psi(t), & \text{if } t \geq \tau, \\ at^2 + bt + c, & \text{if } t \leq \tau, \end{cases}$$

where $\tau = -0.5$, $\psi(t) = 1 - e^{-t}$, $a = \dfrac{-\sqrt{e}}{2}$, $b = \dfrac{\sqrt{e}}{2}$, and $c = 1 - \dfrac{5\sqrt{e}}{8}$, for solving the following convex separable problems.

- **Pb-1**

$$\begin{aligned} \min \quad & \sum_{i=1}^{p} c_i^T x_i \\ & \sum_{i=1}^{p} (x_i^T B_{ji} x_i + b_{ji}^T x_i + \alpha_{ji}) \geq 0, \quad j = 1, ..., m, \end{aligned}$$

  where, for $i = 1, ..., p$, $j = 1, ..., m$, $x_i \in \Re^{n_i}$ is the i'th block variable, $B_{ji} \in \Re^{n_i \times n_i}$ is a negative semidefinite matrix, $c_i, b_{ji} \in \Re^{n_i}$, $\alpha_{ji} \in \Re$, and $\sum_{i=1}^{p} n_i = n$.
- **Pb-2**

$$\begin{aligned} \min \quad & \sum_{i=1}^{p} (x_i^T D_i x_i + d_i^T x_i) \\ & \sum_{i=1}^{p} (x_i^T B_{ji} x_i + b_{ji}^T x_i + \alpha_{ji}) \geq 0, \quad j = 1, ..., m, \end{aligned}$$

  where , for $i = 1, ..., p$, $j = 1, ..., m$, $x_i \in \Re^{n_i}$ is the i'th block variable, $D_i \in \Re^{n_i \times n_i}$ is a positive semidefinite matrix, $B_{ji} \in \Re^{n_i \times n_i}$ is a negative semidefinite matrix, $d_i, b_{ji} \in \Re^{n_i}$, $\alpha_{ji} \in \Re$, and $\sum_{i=1}^{p} n_i = n$.
- **Pb-3**

$$\begin{aligned} \min \quad & \sum_{i=1}^{p} (x_i^T D_i x_i + d_i^T x_i) \\ & \sum_{i=1}^{p} A_i x_i \leq b, \end{aligned}$$

  where, for $i = 1, ..., p$, $x_i \in \Re^{n_i}$ is the i'th block variable, $D_i \in \Re^{n_i \times n_i}$ is a positive semidefinite matrix, $d_i \in \Re^{n_i}$, $A_i \in \Re^{m \times n_i}$, $b \in \Re^m$, and $\sum_{i=1}^{p} n_i = n$.
- **Pb-4**

$$\begin{aligned} \min \quad & a_1(x_1 - 0.5)^2 + \sum_{i=2}^{m} a_i(x_i + 1)^2 + \sum_{i=m+1}^{n} a_i(x_i - 1)^2 \\ & b_1(1 - x_1) + \sum_{i=2}^{m} b_i x_i^2 + \sum_{m+1}^{n} b_i(x_i - 1)^2 \geq 0, \end{aligned}$$

  where

$$a_i = \rho - \frac{\rho^2 - 1}{\rho(n-1)}(i-1), \quad b_i = \rho + \frac{\rho^2 - 1}{n-1}(i-1), \quad i = 1, 2, ..., n.$$

We generated random examples of problems Pb-1, Pb-2, and Pb-3 with diagonal and non-diagonal matrices. For the non-diagonal cases, the positive semidefinite matrix D_i was generated as $D_i = K^T K$, where $K \in \Re^{n_i \times n_i}$ was computed from the standard normal distribution via the command `K = randn(n_i)` in MATLAB. Similarly, the negative semi-definite matrix B_i was generated as $D_i = -R^T R$, where `R= randn(n_i)`. Moreover, the vectors were computed from the standard normal distribution. For the diagonal cases, we generated the diagonal of positive semi-definite matrix D_i uniformly from the interval $[1,5]$, and the diagonal of negative semi-definite matrix B_i was chosen uniformly from the interval $[-5,-1]$. Moreover, Matrix A_i in problem Pb-3, and the vectors were generated uniformly from interval $[-1,1]$.

4.2. **Algorithmic considerations.** To solve the unconstrained minimization problems in Step 2 of algorithms ESALA HDA and EPDM, we used `"fminunc"` function in MATLAB, amending its default setting to apply the `Quasi-Newton` method. In practice, the scaling parameter λ plays a fundamental role in the behavior and efficiency of the proposed algorithm ESALA and in general for any algorithms based on penalization and or on augmented (modified) Lagrangian. This parameter can be used to reach some accepted feasibility of the iterates. We used convenient stopping criteria, which are similar to those used by Breitfeld and Shanno [25]. In our study, we used the update formula suggested in Polyak [17], where the Lagrange multipliers are involved.

$$\lambda_j^{k+1} = \lambda \left(u_j^{k+1}\right)^{-1}, \ j = 1,2,\cdots,m. \tag{4.5}$$

The ESALA performs two types of iterations:

- an inner iteration for the solution of an unconstrained minimization subproblem;
- an outer iteration in which the Lagrange multipliers and the allocations are adjusted.

The inner iteration concerns an unconstrained minimization problem, where the objective contains the rescaled Lagrangian functional corresponding to (2.3). Moreover, we measured the accuracy and the violation of the constraints at the final solution for the compared algorithms by:

$$Ac(Algo) = \frac{|F(x^*) - F(x_{best})|}{|F(x_{best})|},$$

and

$$Vio(algo) = \sum_{i=1}^{m} -\min(g_i(x^*), 0),$$

respectively, where $F(x) = \sum_{i=1}^{p} f_i(x_i)$, x^* is the computed solution by each algorithm, and x_{best} is the solution of the smallest objective value among the algorithms. For each dimension and for each problem (Pb-1, Pb-2, and Pb-3) we generated 5 examples and we reported the accuracy (Ac) and CPU time of the algorithms in second (Time) averaged over the 5 examples.

4.3. **Computational results of ESALA.** To evaluate the performance of the exponential separable augmented Lagrangian algorithm (ESALA), we presented in Table 1 some partial feasibility results at each iteration for Pb-2 with $m = 1$ and $n = 100$. We gathered the variations in the objective function at each iteration in Figure 1. In Tables 2 and 3, we show the influence of the scaling factor λ on the iterations number denoted by "Iter", the CPU time, the optimal objective value denoted by $F(x^*)$ and the violation of the constraints at the optimal solution computed by

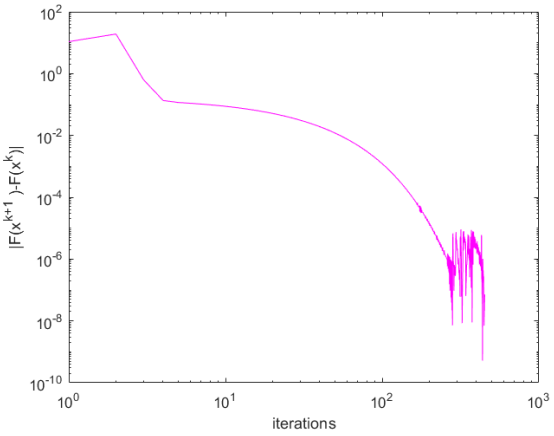

FIGURE 4.1. Behavior of ESALA where $n = 1000$

ESALA. The scaling factor λ plays an important role in the convergence of our algorithm. One may observe in Tables 2 and 3 that the number of iterations and consequently the CPU Time increases when the value of λ increases or when it is so small. Moreover, one may observe that the value of λ may affect the optimal solution returned by ESALA. In our testing, we found out that $\lambda = 0.5$ is the best for all problems tested in the paper.

TABLE 1. Pb-2–$m = 1$—ESALA–$n = 100$

Iter	**Feasibility**
1	2.3276e+01
2	1.6365e+01
3	5.2594e+00
4	1.1196e+00
5	2.0370e-01
6	3.5247e-02
⋮	⋮
42	6.2365e-07
43	5.8413e-07
44	4.5226e-07
45	4.7598e-07
46	5.1192e-06
47	2.3472e-06

4.4. **Comparison of ESALA with IDFGPM.** We implemented, tested, and compared the ESALA to IDFGPM. The comparison can be applied only on Pb-3 (with linear inequality constraints). The IDFGPM was not easy to implement since it needs a lot of parameters and constants. For each dimension, we generated 5 examples of problem Pb-3 with diagonal matrices and reported the averaged accuracy and CPU Time of ESALA and IDFGPM in Table 4. As we see, ESALA is too much faster and more accurate than IDFGPM.

TABLE 2. Influence of the scaling factor λ– ESALA- Pb-1, $m = 3$

	λ	$F(x^*)$	Vio(ESALA)	Iter	Time(ESALA)
$n = 500$					
	0.05	-9.9094	1.1962e-06	164	11.0752
	0.1	-9.9094	1.1357e-06	88	6.2858
	0.5	-9.9094	1.8168e-06	63	3.5496
	1	-9.9094	1.0193e-06	99	5.4722
	1.5	-9.9094	8.1130e-07	138	7.1549
	10	-9.9094	1.3672e-06	749	34.8462
	30	-9.9093	2.8366e-05	1000	49.3288
	50	-9.9091	8.1937e-06	1000	53.3762
$n = 1000$					
	0.05	-10.4518	1.7892e-06	161	26.5145
	0.1	-10.4518	1.5600e-06	86	16.2701
	0.5	-10.4518	6.2268e-08	62	9.2989
	1	-10.4518	1.8447e-06	100	13.2106
	1.5	-10.4518	2.0902e-06	157	19.9489
	10	-10.4518	2.4447e-06	705	80.9789
	30	-10.4517	0	1000	133.4646
	50	-10. 4518	1.7175e-05	1000	140.1813
$n = 3000$					
	0.05	-9.3718	7.2738e-07	161	43.3453
	0.1	-9.3718	2.2556e-06	110	37.5697
	0.5	-9.3718	0	71	17.9860
	1	-9.3718	0	107	21.5373
	1.5	-9.3718	2.1924e-06	110	23.6913
	10	-9.3716	0	1000	177.3332
	30	-9.3718	602e-05	1000	189.0496
	30	-9.3718	5.3451e-05	1000	188.9630
$n = 6000$					
	0.05	-10.0885	1.2415e-06	129	302.1915
	0.1	-10.0885	1.9484e-06	89	243.1712
	0.5	-10.0885	1.8451e-06	69	131.3021
	1	-10.0885	1.9868e-06	116	180.9875
	1.5	-10.0885	2.0179e-06	157	236.4721
	10	-10.0885	0	857	1198.0102
	30	-10.0884	0	1000	1569.4710
	50	-10.0883	0	1000	1708.7012

4.5. **Comparison of ESALA with EPDM and HDA.** In this subsection, we compared the performance of ESALA to EPDM, and to HDA. Since HDA is developed to handle the problems with only one constraint, we tested it on problems Pb-1, Pb-2 with $m = 1$ and Pb-4. Theoretically, in [9], it has been shown that the sequence w^k and u_i^k, $i = 1, ..., p$, generated by EPDM, are

TABLE 3. Influence of the scaling factor λ– ESALA– Pb-2, $m = 1$

	λ	$F(x^*)$	Vio (ESALA)	Iter	Time(ESALA)
$n = 500$					
	0.05	-7.0189	2.6796e-06	141	11.0679
	0.1	-7.0189	2.3336e-06	77	4.9750
	0.5	-7.0189	3.8881e-07	52	3.0997
	1	-7.0189	0	69	3.7724
	1.5	-7.0189	1.3112e-06	95	5.3242
	10	-7.0189	0	540	25.2853
	30	-7.0189	0	1000	45.7994
	50	-7.0189	1.0886e-06	1000	48.6099
$n = 1000$					
	0.05	-6.2053	2.1980e-06	141	26.7251
	0.1	-6.2053	3.3061e-06	93	15.8111
	0.5	-6.2053	8.3623e-07	59	8.0311
	1	-6.2053	0	77	10.3023
	1.5	-6.2053	1.5567e-06	110	13.7268
	10	-6.2053	5.4005e-07	405	51.3268
	30	-6.2053	1.6456e-05	1000	112.2343
	50	-6.2053	2.0270e-07	1000	123.9923
$n = 3000$					
	0.05	-6.2498	1.9656e-06	156	47.5674
	0.1	-6.2498	2.8187e-06	81	28.8213
	0.5	-6.2498	0	55	13.5878
	1	-6.2498	0	98	17.1145
	1.5	-6.2498	1.3914e-06	127	21.6821
	10	-6.2498	1.4628e-06	605	88.4328
	30	-6.2498	3.2489e-07	764	123.7520
	50	-6.2498	0	1000	164.5448
$n = 6000$					
	0.05	-6.3118	0	155	359.1312
	0.1	-6.3118	2.7176e-06	136	314.3021
	0.5	-6.3118	1.8761e-06	60	116.45527
	1	-6.3118	0	93	128.0554
	1.5	-6.3118	0	104	147.2203
	10	-6.3118	2.4342e-07	340	432.7178
	30	-6.3118	4.9753e-08	806	978.0022
	50	-6.3118	0	1000	1252.4976

positive. In our tests, we found out that EPDM is so slow and for most of the problems tested in the paper, at larger iterations, it stopped with an error. Indeed, the sequence w^k or u_i^k may go to zero due to some rounding errors and then the steps of EPDM become not well-defined. Hence, in our experiments, we allowed EPDM proceed at most 4 times the CPU Time of ESALA. We

TABLE 4. Pb-3—— ESALA vs IDFGPM

n	Ac(ESALA)	Ac(IDFGPM)	Time(ESALA)	Time(IDFGPM)
3000	0.0000e+00	5.5536e-10	0.5241	17.6972
5000	0.0000e+00	3.7101e-10	2.2748	25.3192
7500	0.0000e+00	2.6631e-10	3.8373	35.2849
10000	0.0000e+00	2.1477e-10	2.8164	131.5980
15000	0.0000e+00	1.5427e-10	2.5294	164.5731
30000	0.0000e+00	8.8866e-11	3.4581	292.7930
50000	0.0000e+00	5.9385e-11	31.7933	1598.3232
100000	0.0000e+00	3.4198e-11	16.6264	2289.0331

reported the accuracy and the CPU Time of EPDM averaged over cases among 5 instances for which EPDM returns a feasible solution in the expected time. Otherwise, we did not report the CPU Time of EPDM and used "failed" to denote that EPDM did not return a feasible solution for all 5 instances. We gathered the results for problems Pb-1, Pb-2, and Pb-3 with diagonal matrices in Tables 5, 6, and 7. As we can see in Table 5, ESALA is much faster and more accurate than HDA and EPDM fails for all cases. We observe in Tables 6 and 7 that ESALA is always successful and EPDM for about 57% could not return a feasible solution in the expected time. We have summarized the results for problems Pb-1, Pb-2, and Pb-3 with non-diagonal matrices in Tables 8 , 9 and 10. As we can see, ESALA is still much more accurate and faster than HDA and EPDM. Moreover, for none of the generated problems, EPDM returned a feasible solution in the time less than 4 times the CPU Time of ESALA. We also adjusted the results of ESALA, EPDM, and HDA for problem Pb-4 in Table 11. ESALA is much faster and more accurate than HDA and EPDM.

TABLE 5. Pb-2–$m = 1$——ESALA vs EPDM vs HDA

n	Ac(ESALA)	Ac(HDA)	Ac(EPDM)	Tine(ESALA)	Time(HDA)	Time(EPDM)
3000	0.0000e+00	3.8812e-02	failed	4.4469	14.2990	–
5000	0.0000e+00	4.9671e-02	failed	5.3936	15.0785	–
7500	0.0000e+00	5.2978e-02	failed	5.7055	16.3941	–
10000	0.0000e+00	5.8560e-02	failed	12.5185	30.3567	–
15000	0.0000e+00	5.6241e-02	failed	17.0581	33.7876	–
30000	0.0000e+00	2.8153e-02	failed	36.3829	125.7315	–
50000	0.0000e+00	3.1078e-02	failed	41.2171	137.0350	–
100000	0.0000e+00	2.2931e-02	failed	90.5268	263.5478	–

4.6. **Comparison of ESALA with CVX.** In this subsection, we compared ESALA to the CVX. The comparison was done on problems Pb-1, and Pb-2 with $m = 3$. We reported the accuracy and the CPU Time averaged over 5 randomly generated examples of problems Pb-1 and Pb-2 with diagonal matrices in Tables 12 and 13, respectively. As wee can see, ESALA is as accurate as CVX and it is much faster than CVX, especially for larger dimensions.

TABLE 6. Pb-1–$m = 3$——ESALA vs EPDM

n	Ac(ESALA)	Ac(EPDM)	Time(ESALA)	Time(EPDM)
3000	0.0000e+00	failed	31.1247	–
5000	0.0000e+00	failed	23.3445	–
7500	0.0000e+00	failed	19.1277	–
10000	0.0000e+00	failed	37.8387	–
15000	0.0000e+00	failed	30.0383	–
30000	0.0000e+00	3.6751e-01	93.0276	374.7136
50000	0.0000e+00	failed	536.6294	–
100000	0.0000e+00	failed	827.4386	–

TABLE 7. Pb-2–$m = 4$——ESALA vs EPDM

n	Ac(ESALA)	Ac(EPDM)	Time(ESALA)	Time(EPDM)
3000	0.0000e+00	3.5624e+00	8.5693	10.4394
5000	0.0000e+00	3.0346e+00	9.3850	38.0293
7500	0.0000e+00	3.4869e+00	12.0222	48.9500
10000	0.0000e+00	3.0641e+00	22.0021	82.1584
15000	0.0000e+00	failed	28.0383	–
30000	0.0000e+00	failed	216.8607	–
50000	0.0000e+00	2.2352e+00	298.549	486.3085
100000	0.0000e+00	5.4263e+00	311.0990	1256.8133

TABLE 8. Pb-1–$m = 3$——ESALA vs EPDM

n	Ac(ESALA)	Ac(EPDM)	Time(ESALA)	Time(EPDM)
3000	0.0000e+00	failed	6.7469	–
5000	0.0000e+00	8.3287e-01	9.5498	38.6162
7500	0.0000e+00	8.4462e-01	12.1916	49.7288
10000	0.0000e+00	8.4746e-01	24.6134	101.8769
15000	0.0000e+00	8.5017e-01	32.8956	144.9485
30000	0.0000e+00	8.4469e-01	133.2530	535.6955
50000	0.0000e+00	8.5705e-01	131.7358	533.6992
100000	0.0000e+00	8.6751e-01	265.8168	1101.2206

5. CONCLUSION

In this paper, we investigated the numerical study of the performances of the exponential separable augmented Lagrangian algorithm (ESALA), which is favorable to parallel computations. The ESALA can be seen as a special case of a class of algorithms developed and studied theoretically in Hamdi and Mukheimer [5]. This family of the algorithms can be seen as a separable version of the nonlinear rescaling principle method and is closely related to the separable augmented Lagrangian Algorithm developed in Hamdi et al. [6] and Hamdi [7]. In this paper we discussed the behaviour of the ESALA and its scalar factor's influence. In addition,

TABLE 9. Pb-2– $m = 1$——ESALA vs HDA vs EPDM

n	Ac(ESALA)	Ac(HDA)	Ac(EPDM)	Time(ESALA)	Time(HDA)	Time(EPDM)
3000	0.0000e+00	9.9770e-02	failed	3.6184	6.8409	–
5000	0.0000e+00	5.5058e-02	failed	25.4648	51.6043	–
7500	0.0000e+00	5.1578e-02	failed	59.2928	109.5256	–
10000	0.0000e+00	3.2144e-02	failed	97.5230	177.4020	–
15000	0.0000e+00	3.9773e-02	failed	211.5374	394.4116	–
30000	0.0000e+00	3.8420e-02	failed	46.4010	85.1572	–
50000	0.0000e+00	3.2581e-02	failed	305.6814	565.1728	–
100000	0.0000e+00	1.4131e-02	failed	335.7976	527.9453	–

TABLE 10. Pb-2–$m = 3$——ESALA vs EPDM

n	Ac(ESALA)	Ac(EPDM)	Time(ESALA)	Time(EPDM)
3000	0.0000e+00	8.6952e-01	12.9169	47.2504
5000	0.0000e+00	9.5622e-01	13.3285	53.7632
7500	0.0000e+00	9.7202e-01	16.1445	65.5938
10000	0.0000e+00	9.7673e-01	35.2868	142.5933
15000	0.0000e+00	9.8593e-01	45.1383	190.6988
30000	0.0000e+00	9.6216e-01	179.3735	720.3878
50000	0.0000e+00	1.0012e+00	154.6512	626.0942
100000	0.0000e+00	1.0128e+00	429.9544	1740.8870

TABLE 11. Pb-4——ESALA vs HDA vs EPDM

n	Ac(ESALA)	Ac(HDA)	Ac(EPDM)	Time(ESALA)	Time(HDA)	Time(EPDM)
3000	0.0000e+00	5.8694e-06	failed	19.5644	8.5002	–
5000	0.0000e+00	7.6883e-06	failed	12.6300	12.6255	–
7500	0.0000e+00	1.3786e-05	failed	18.9453	19.4656	–
10000	0.0000e+00	1.2957e-05	1.4395e-02	27.6293	28.1020	111.2833
15000	0.0000e+00	8.0035e-06	6.2854e-02	50.0008	58.0602	203.0111
30000	0.0000e+00	1.2168e-05	1.5808e-01	64.2507	75.9674	262.0433
50000	0.0000e+00	7.8245e-06	2.5728e-01	169.1854	192.6791	688.4140
75000	0.0000e+00	1.8025e-05	3.9499e-01	415.2343	456.9790	1671.9234
100000	0.0000e+00	1.2625e-05	4.1997e-01	732.4990	787.1099	2988.2697

TABLE 12. Pb-1-m=3——ESALA vs CVX

n	Ac(ESALA)	Ac(CVX)	Time(ESALA)	Time(CVX)
500	7.0082e-10	4.9468e-10	0.707	12.8081
1000	0	1.7456e-09	2.3151	34.779
3000	1.4561e-10	2.6259e-09	8.7873	393.6231

TABLE 13. Pb-2-m=3——ESALA vs CVX

n	Ac(ESALA)	Ac(CVX)	Time(ESALA)	Time(CVX)
500	8.9338e-09	0	1.3302	15.025
1000	1.5231e-09	4.9303e-09	3.7274e	57.596
3000	5.4700e-07	7.8579e-08	26.4142	678.4716

TABLE 14. Pb-1—$m = 2$——ESALA vs MBSALA

n	Time(ESALA)	Time(MBSALA)	Acc(ESALA)	Acc(MBSALA)
1000	5.5588	5.0078	8.85e-09	6.25e-09
3000	4.2774	4.5100	1.73e-08	8.48e-09
5000	13.4779	13.5650	4.56e-08	5.23e-08
10000	28.0762	33.1203	1.23e-09	8.47e-09

TABLE 15. Pb-2 ——ESALA vs MBSALA

n	Time(ESALA)	Time(MBSALA)	Acc(ESALA)	Acc(MBSALA)
1000	1.0870	1.0015	1.51e-08	5.03e-08
3000	1.3876	1.3614	2.23e-08	1.07e-08
5000	5.3791	5.3690	1.79e-08	8.25e-09
10000	22.5522	23.7408	2.18e-09	9.00e-09

we gave a computational comparison of ESALA with other decomposition algorithms: The entropic decomposition proximal method (EPDM) of Auslender and Teboulle [9], the hyperbolic decomposition algorithm (HDA) (Hamdi et al. [15]), and with the inexact dual fast gradient-projection method (IDFGPM) for linearly coupled constraints (Li et al. [16]). The ESALA can be made faster if we could program it on parallel processors machines. According to our experience with these type of algorithms, we think that the proposed update of the scaling factor sequence λ_k (4.5), which is based on the Lagrange multipliers sequence, improves a lot the optimality and gives better results. As we said it earlier, the IDFGPM needs `many parameters`, where some of them are theoretical and related to the problems data. IDFGPM is based particularly on many parameters. The EPDM was particularly unstable because of the rounding errors and needs more time to reach comparable efficiency as ESALA. We may observe in Tables 5, 9 and 11, that HDA shows less efficiency than the ESALA. It should be noticed that in terms of CPU Time, HDA is a promising algorithm. This hyperbolic penalization algorithm heavily influenced by its penalty parameters τ and λ could be re-shaped as a primal-dual method by incorporating Lagrange multipliers and we believe that better efficiency will be reached.

It will be worth for future work to study and compare the performance of many candidate $\varphi SALA$. In the Tables 14-16, we implemented the Modified Barrier Separable Augmented Lagrangian Algorithm (MBSALA), when $\varphi_2(t) = \ln(1+t)$, $\tau = -0.5$ and

$$\widehat{\varphi_2}(t) = \begin{cases} \ln(1+t), & \text{if } t \geq -0.5, \\ at^2 + bt + c, & \text{if } t \leq -0.5, \end{cases} \qquad a = -2, \ b = 0, \ \text{and } c = \frac{1}{2} - \ln(2).$$

The computational results (Table 14 - Table 16) show that these two schemes are efficient and are promising. We believe that (MBSALA) motivates further investigations for larger scale

TABLE 16. Pb-3–$m = 3$ ——ESALA vs MBSALA

n	Time(ESALA)	Time(MBSALA)	Acc(ESALA)	Acc(MBSALA)
1000	2.4263	3.1380	9.21e-02	4.38e-08
3000	2.0046	2.1237	2.76e-08	1.96e-08
5000	9.9922	9.7551	1.42e-08	1.71e-09
10000	40.3633	36.8102	6.64e-09	3.77e-09

minimization problems. Moreover, one of the future research direction would be to extend the ideas in the paper to handle non-convex problems.

Acknowledgments
The authors would like to thank the referees for the constructive comments which led to significant improvement of the paper. The authors would like to express their thanks to Qatar University for supporting their project under Grant NCBP-QUCP-CAS-2020-1.

REFERENCES

[1] B.W. Kort, D.P. Bertsekas, Combined primal-dual and penalty methods for convex programming, SIAM J. Control Optim. 14 (1976), 268-294.
[2] R. Polyak, Modified barrier function (Theory and Methods), Math. Program. 54 (1992), 177-222.
[3] A.E. Xavier, Hyperbolic penalty: a new method for nonlinear programming with inequalities, Int. Trans. Oper. Res. 8 (2001), 659-671.
[4] R. Polyak, Log-sigmoid multipliers method in constrained optimization, Ann. Oper. Res. 101 (2001), 427-460.
[5] A. Hamdi, A.A. Mukheimer, Modified Lagrangian methods for separable optimization problems, Abst. Appl. Anal. 2012 (2012), 471854.
[6] A. Hamdi, P. Mahey, J.P. Dussault, A new decomposition method in non convex programming via separable augmented Lagrangians, in Recent Advances in Optimization, Gritzmann, Horst, Sachs and Tichatsch, 1997.
[7] A. Hamdi, Decomposition for structured convex programs with smooth multiplier methods, Appl. Math. Comput. 169 (2006), 218-241.
[8] A. Hamdi, S.K. Mishra, Decomposition Methods Based on Augmented Lagrangians: A Survey, Topics in Nonconvex Optimization: Theory and Applications, pp. 175-204, Springer, 2011.
[9] A. Auslender, M. Teboulle, Entropic proximal decomposition methods for convex programs and variational inequalities, Math. Program. 91 (2001), 33–47.
[10] M. Kyono, M. Fukushima, Nonlinear proximal decomposition method for convex programming, J. Optim. Theory Appl. 106 (2000), 357-372.
[11] A. Bnouhachem, A. Hamdi, Parallel LQP alternating direction method for solving variational inequality problems with separable structure, J. Inequal. Appl. 2014 (2014), 392.
[12] A. Bnouhachem, A. Hamdi, M.H. Xu, A new LQP alternating direction method for solving variational inequality problems with separable structure, Optimization 65 (2016), 2251-2267.
[13] Y. Jung, N. Kang, I. Lee, Modified augmented Lagrangian coordination and alternating direction method of multipliers with parallelization in non-hierarchical analytical target cascading, Structural Multidisciplinary Optim. 58 (2018), 555–573.
[14] K.M. Palanduz, A.A. Groenwold, A separable augmented Lagrangian algorithm for optimal structural design, Structural Multidisciplinary Optim. 61 (2020), 343–352.
[15] A. Hamdi, T.A. Al-Maadeed, A. Taati, A hyperbolic penalty method to solve structured convex minimization problems, Int. J. Adv. Appl. Sci. 7 (2020), 87-97.
[16] J. Li, Z. Wu, C. Wu, Q. Long, X. Wang, An inexact dual fast gradient-projection method for separable convex optimization with linear coupled constraints, J. Optim. Theory Appl. 168 (2016), 153–171.

[17] R. Polyak, Nonlinear rescaling vs. smoothing techniques in convex optimization, Math. Program. 101 (2002), 427-460.
[18] D. Gabay, B. Mercier, A dual algorithm for the solution of nonlinear variational inequalities via finite element approximation, Comput. Math. App. 2 (1976), 17-40.
[19] D.P. Bertsekas, J.N. Tsitsiklis, Parallel and Distributed Computation: Numerical Methods (Prentice-Hall, New Jersey, 1989.
[20] J. Eckstein, M. Fukushima, Some reformulations and applications of the alternating direction method of multipliers, in Large Scale Optimization: State of the Art, W. Hager, D. Hearn and P. Pardalos (eds), pp. 119-138, Kluwer Academic Pub., 1993.
[21] M. Fukushima, Application of the alternating direction of multipliers to separable convex programming problems, Comput. Optim. Appli. 1 (1992), 93-112.
[22] E.G. Gol'stein, The block method of convex programming, Soviet Math. Dokl. 33 (1986), 584–587.
[23] C.D. Ha, Decomposition Methods for Structured Convex Programming, Ph.D. Thesis, Univ. Wisconsin-Madison, 1980
[24] M. Grant, S. Boyd, CVX: Matlab software for disciplined convex programming, version 2.0 beta, http://cvxr.com/cvx.
[25] M.G. Breifeld, D.F. Shanno, Computational experience with penalty-barrier methods for nonlinear programming, RUTCOR Research Report RRR 17-93, Rutgers University, New Jersey, 1994.