Iteration and evaluation complexity for the minimization of functions whose computation is intrinsically inexact [*]

E. G. Birgin[†] N. Krejić[‡] J. M. Martínez[§]

September 25, 2017[¶]

Abstract

In many cases in which one wishes to minimize a complicated or expensive function, it is convenient to employ cheap approximations, at least when the current approximation to the solution is far from the solution. Adequate strategies for deciding the accuracy desired at each stage of optimization are crucial for the global convergence and overall efficiency of the process. A recently introduced procedure [E. G. Birgin, N. Krejić, and J. M. Martínez, On the employment of Inexact Restoration for the minimization of functions whose evaluation is subject to errors, *Mathematics of Computation* 87, pp. 1307-1326, 2018] based on Inexact Restoration is revisited, modified, and analyzed from the point of view of worst-case evaluation complexity in this work.

Key words: Inexact Restoration, global convergence, worst-case evaluation complexity.

2010 Mathematics Subject Classification: 65K05, 65K10, 90C30, 90C90.

1 Introduction

During more than 50 years, proving global convergence for continuous optimization algorithms involved to show that, independently of the initial approximation, limit points of the the generated sequence satisfy some optimality condition. After 2006, starting with the Nesterov-Polyak paper [37], it became consensual that complexity results, which state the worst-case computer work being necessary to achieve a given precision, are relevant. In unconstrained optimization, Newton-like algorithms that use regularized or adapted trust-regions supbproblems with proven

[*]This work has been partially supported by the Brazilian agencies FAPESP (grants 2013/03447-6, 2013/05475-7, 2013/07375-0, 2014/18711-3, and 2016/01860-1) and CNPq (grants 309517/2014-1 and 303750/2014-6) and by the Serbian Ministry of Education, Science, and Technological Development (grant 174030).

[†]Department of Computer Science, Institute of Mathematics and Statistics, University of São Paulo, Rua do Matão, 1010, Cidade Universitária, 05508-090, São Paulo, SP, Brazil. e-mail: egbirgin@ime.usp.br

[‡]Department of Mathematics and Informatics, Faculty of Sciences, University of Novi Sad, Trg Dositeja Obradovića 4, 21000 Novi Sad, Serbia. e-mail: natasak@uns.ac.rs

[§]Department of Applied Mathematics, Institute of Mathematics, Statistics, and Scientific Computing (IMECC), University of Campinas, 13083-859 Campinas SP, Brazil. e-mail: martinez@ime.unicamp.br

[¶]Revision made on August 16, 2018. The second revision on January 31, 2019.

complexity $O(\varepsilon^{-3/2})$ were given in [8, 14, 15, 16, 19, 21, 36, 37]. The sharpness of the complexity $O(\varepsilon^{-3/2})$ was proved in [16]. All these papers employ, in an explicit or implicit form, cubic regularization ideas [27], and assume Lipschitz-continuity of second derivatives.

In [6], a generalization of cubic regularization to arbitrary p-th regularization for unconstrained optimization was given. At each iteration, the method introduced in [6] approximately minimizes a p-th Taylor polynomial around the current point plus a regularization term of order $p+1$. Using Lipschitz conditions on the derivatives of order p, it was proved that the method achieves a gradient norm smaller than ε in at most $O(\varepsilon^{-(p+1)/p})$ iterations and functional evaluations. In [25], the Lipschitz-continuity assumption on second-derivatives was relaxed to Hölder-continuity. In [17] and [34], this approach was generalized in order to consider approximations of order p of the objective function and Hölder-continuity of p-th derivatives.

This state of facts motivates the complexity analysis of existing algorithms for solving different continuous optimization problems as well as the introduction of new algorithms with promising worst-case complexity results. See, for example, [7]. The objective of the present paper is to analyze, from the point of view of complexity, suitable modifications of the algorithms introduced in [30] and [11] for the minimization of functions whose evaluation is intrinsically affected by errors.

The algorithms introduced in [30] and [11] are based on the Inexact Restoration framework [1, 3, 5, 12, 13, 23, 29, 33, 35]. Inexact Restoration (IR) is a family of techniques for solving continuously constrained optimization problems where feasibility and optimality are handled separately in order to better exploit problem structure. The employment of IR in the cases of uncertain evaluation of the objective relies on the analogy between *restoring feasibility* and *improving accuracy*. IR techniques put some rationality in the process of evaluating the objective function with reasonable accuracy, according to an implicit estimation of the distance of the current approximation to a solution. In [30], the problem of minimizing an unconstrained function with a finite number of inaccuracy levels was considered. For the minimization of $f(x)$, the equivalence with the minimization of z subject to $z = f(x)$ was employed; and the IR scheme was applied to this problem. Following the traditional IR approach, the optimization phase of the algorithm begins with the determination of a guaranteed descent direction and a line-search procedure is used. It was proved that the maximal accuracy is reached and that the size of the search direction converges to zero, implying optimality for some directional choices. An application concerning an electronic calculation problem in which the objective function comes from an iterative process was given. In [11], the problem of minimizing a continuous function whose evaluation is subject to errors onto a closed and convex set was considered. In contrast to [30], an infinite number of inaccuracy levels was considered. As in [30], an IR approach was considered and a descent direction that guarantees termination of the IR optimization phase was employed together with a simple line-search procedure. The convergence theory guarantees that full accuracy of the objective function is achieved in the limit and that the projected gradient of an approximate objective function onto the convex domain tends to zero. Applications were given to classification and portfolio optimization problems. In the present work, a suitable modification of the algorithms introduced in [30] and [11] is considered. On the one hand, the main algorithm introduced in the present work generalizes the ones introduced in [30] and [11] by defining feasibility with respect to an abstract set Y. On the other hand, by employing regularization techniques in the optimization phase of the IR approach, in substitution of the line-search procedures considered in [30] and [11], complexity results for the introduced algorithm will be obtained.

Let us start by firstly introducing a new formulation of the problem of minimizing a function whose evaluation is intrinsically affected by errors. In the proposed formulation, minimization occurs with respect to a continuous variable x and an abstract variable y that may represent the variations of the problem that lead to solutions with different accuracies. Assume that Y is a set, $h : Y \to \mathbb{R}_+$, $f : \mathbb{R}^n \times Y \to \mathbb{R}$, and $\Omega \subset \mathbb{R}^n$ is defined by the set of equations $c_E(x) = 0$ and inequations and $c_I(x) \leq 0$, where $c_E : \mathbb{R}^n \to \mathbb{R}^m$ and $c_I : \mathbb{R}^n \to \mathbb{R}^p$. The problem considered in this work is given by

$$\text{Minimize (with respect to } x\text{)} \;\; f(x, y) \text{ subject to } h(y) = 0 \text{ and } x \in \Omega, \tag{1}$$

where

$$\Omega = \{x \in \mathbb{R}^n \mid c_E(x) = 0 \text{ and } c_I(x) \leq 0\}. \tag{2}$$

In other words, we will try to find $x \in \Omega$ and $y \in Y$ such that $h(y) = 0$ and $f(x, y) \leq f(z, y)$ for all $z \in \Omega$. An obvious algorithm for solving (1) could proceed by means of two global phases. In the first phase, one should find $y \in Y$ such that $h(y) = 0$. In the second phase, with y fixed, one should solve the optimization problem that consists of minimizing $f(x, y)$ subject to Ω. In this work, we are interested in problems of the form (1) in which this "obvious algorithm" is not affordable. The main reason for this is that the evaluation of $f(x, y)$ may become very expensive when $h(y)$ is close to zero. Assuming that it makes sense to think about a true function $f_{\text{true}}(x)$, we may think that $f(x, y)$ approaches $f_{\text{true}}(x)$ when $h(y) \to 0$, uniformly with respect to x, i.e.

$$\lim_{h(y)\to 0} |f(x, y) - f_{\text{true}}(x)| = 0$$

uniformly with respect to x. The non-uniform case, in which δ depends on x cannot be addressed by the present approach.

Assume that Y is a collection of model schemes that aim to represent a physical phenomenon. As a trivial example, an element of Y could be "Try linear regression". The vector x represents parameters, probably with physical meaning, that are present in all the models that compound the collection Y. To each $y \in Y$, we associate a quantity $h(y) \geq 0$ that represents its "simplicity". Therefore, big values of $h(y)$ correspond to very simple model schemes and $h(y) = 0$ represents maximal model complexity. Simple models are easier to fit than complex models and cheaper to run in the context of simulation. In this type of applications, the function $f(x, y)$ is the error in the reproduction of data obtained after fitting the model y using the physical parameters x.

In [18], a computer model for particle-like simulation in broiler houses is presented, the fitting process of which corresponds to the description above. In this model, a 45-days period of the life of approximately 35,000 broilers in a broiler house 40 meters wide and 200 meters long is simulated. Interactions of broilers are considered emulating attraction and repulsion rules in a similar way as done in Molecular Dynamics. Running this model is expensive. The simulation of a 45-days period with 35,000 chickens takes almost a day in a computer environment compatible with the industrial employment of the model. The simulation model has approximately 15 parameters that need to be fitted using available data. In the fitting process, each evaluation of the error involves to run the model at least once; and obtaining the optimal parameters may need many evaluations of the error (functional evaluations). Therefore, it is crucial to approximate the optimal parameters by means of suitable model simplifications. The obvious simplification consists in reducing the size of the broiler house and the number of broilers, preserving the relation between number of broilers and area of the house.

As a second example, consider now that we wish to optimize a process that depends on the control variables x and the environmental variables y, which are hard, or even impossible, to measure accurately. The precision on the measurement y is given by $h(y)$. One possible reason for the lack of precision may be the fact that the true y is its value at some future period of time. Therefore the true value of the cost $f(x, y)$ may be obtained only when the expiration time arrives. The problem has the form (1) and the obvious algorithm is impracticable because one cannot wait until having the true y before taking decisions about x.

Another example related to the development of algorithms follows. The efficiency of an algorithm for solving a mathematical problem depends on the judicious choice of parameters x. This efficiency may be measured taking into account the behavior of the algorithm for solving a possibly large number of problems. Assume that each $y \in Y$ is a subset of problems, $h(y)$ is inversely related to the number of problems in y, and $f(x, y)$ is a measure of the efficiency of the algorithm when running the problems in y with the parameters x. Again, this is an example of a problem of the form (1).

Many additional practical applications come from the minimization of a function $F(x)$ whose exact evaluation is not known or is meaningless in different senses. As a consequence, each time that we need to compute $F(x)$ we obtain a value $f(x, y)$, where y lies in some unspecified set Y. Some examples follows:

- If $F(x)$ is computed in multiple precision floating point arithmetic, y may be the number of digits used for the computations.
- If $F(x)$ comes from taking the average of some quantity over a sample, y may be the sample. Note that here Y is not necessarily a numerical set.
- If $F(x)$ is a quantity computed only approximately by means of an iterative process, y may be a limit on the number of iterations or the computer time, or a combination of both.
- Consider the case in which $F(x)$ is in fact a function $f(x, y)$ but y is not known. In other words, we do not know if the value of y that we are taking for the evaluation of F is the correct one or not. This happens, in Economic models, when one considers that the variable represented by $F(x)$ depends on a vector of variables x although being aware that other variables y intervene on its determination. In some economic theories [2], it is considered that the level of aggregated production and employment is determined (in the sense of "causa causans") by investment (x); although it also depends on other variables as the propensions to saving and spending, the monetary policy, and the confidence with respect to rentability of capital assets. In the fitting process of economic models that consider that (say) the gross national product is a function of investment, the variables y involved in data are known with considerable uncertainty and obtaining them with good accuracy involves computer time and human effort. Similar considerations may be done with respect to models that use fitted production functions [28].

Finally, it is interesting to consider also the case in which $F(x)$ is $f(x, y)$ where x is a real vector but y has a completely different structure which requires an algorithmic approach that handles it separability.

The rest of this paper is organized as follows. In Section 2, we define a basic algorithm, we prove that, for given $\varepsilon_{feas}, \varepsilon_{opt} \geq 0$, after at most $O(1/\varepsilon_{feas})$ iterations we obtain that $h(y) \leq \varepsilon_{feas}$, and we obtain a bound for the number iterations at which the increment in the variable x is bigger than ε_{opt}. In Section 3, we described two algorithms for the optimization phase, which differ in the order of regularization. Complexity results $O(\varepsilon_{opt}^{-2})$ and $O(\varepsilon_{opt}^{-3/2})$ are obtained for the gradient-like and the Newton-like versions of the algorithm, respectively. Final remarks are given in Section 4.

Notation. $\mathbb{N}_+$ denotes the non-negative integer numbers; while $\mathbb{R}_+$ denotes the non-negative real numbers. The symbol $\|\cdot\|$ denotes the Euclidean norm of vector and matrices.

2 Main algorithm

We begin introducing a merit function related to problem (1). The merit function $\Phi : \mathbb{R}^n \times Y \times (0,1) \to \mathbb{R}$ is defined by

$$\Phi(x, y, \theta) = \theta f(x, y) + (1 - \theta) h(y).$$

Algorithm 2.1 below is the main model algorithm that generalizes the ones introduced in [30] and [11] by defining feasibility with respect to an abstract set Y. Moreover, as it will be described in Section 3, in contrast to the line-search procedures employed in the optimization phase of the algorithms in [30] and [11], the optimization phase of Algorithm 2.1 will employ regularization techniques, which enables the obtention of complexity results.

Algorithm 2.1 will be stated without stopping criterion. This means that, in principle, the algorithm produces an infinite sequence of iterates. The stopping tolerances that are used in practice *are not* algorithmic parameters. Theoretical arguments related to this algorithm may refer to infinite sequences and meaningful results for the case in which a stopping criterion is included are obtained without contradictions. Given an arbitrary $\varepsilon > 0$, the typical complexity result will report the total number of iterations (and evaluations) that may occur along the computation of the sequence of iterates until the satisfaction of the stopping criterion based on ε. It is easy to see that, when such a result takes place, the consequence for the infinite sequence is that *every* accumulation point is stationary (which means satisfying the stopping criterion with tolerance $\varepsilon = 0$). In other optimization algorithms, the tolerance ε is an algorithmic parameter and the typical convergence result says that, after a proven number of iterations, the algorithm stops. In this case, we can also think about the infinite sequence generated by the algorithm, but the property of this sequence will be only that *some* accumulation point is stationary. In our case, the analysis of the potentially infinite sequence simplifies the proofs.

Algorithm 2.1. Let $x_0 \in \Omega$, $y_0 \in Y$, $\theta_0 \in (0,1)$, $\nu > 0$, $r \in (0,1)$, $\alpha > 0$, and $\beta > 0$ be given. Set $k \leftarrow 0$.

Step 1. *Restoration phase*

Define $y_k^{\rm re} \in Y$ in such a way that

$$h(y_k^{\rm re}) \leq r h(y_k) \tag{3}$$

and

$$f(x_k, y_k^{\mathrm{re}}) \le f(x_k, y_k) + \beta h(y_k). \tag{4}$$

Step 2. *Updating the penalty parameter*

If

$$\Phi(x_k, y_k^{\mathrm{re}}, \theta_k) \le \Phi(x_k, y_k, \theta_k) + \frac{1-r}{2}\left(h(y_k^{\mathrm{re}}) - h(y_k)\right), \tag{5}$$

set $\theta_{k+1} = \theta_k$. Otherwise, set

$$\theta_{k+1} = \frac{(1+r)\left(h(y_k) - h(y_k^{\mathrm{re}})\right)}{2\left(f(x_k, y_k^{\mathrm{re}}) - f(x_k, y_k) + h(y_k) - h(y_k^{\mathrm{re}})\right)}. \tag{6}$$

Step 3. *Optimization phase*

Compute $y_{k+1} \in Y$ and $s_k \in \mathbb{R}^n$ such that $x_k + s_k \in \Omega$,

$$f(x_k + s_k, y_{k+1}) \le f(x_k, y_k^{\mathrm{re}}) - \alpha \|s_k\|^{\nu} \tag{7}$$

and

$$\Phi(x_k + s_k, y_{k+1}, \theta_{k+1}) \le \Phi(x_k, y_k, \theta_{k+1}) + \frac{1-r}{2}\left(h(y_k^{\mathrm{re}}) - h(y_k)\right). \tag{8}$$

Define $x_{k+1} = x_k + s_k$, update $k \leftarrow k+1$, and go to Step 1.

Assume that x_k is the current set of parameters that aim to fit a given phenomenon and y_k is the model with which the error is $f(x_k, y_k)$. At Step 1, we choose a more complex model given by y_k^{re}. The condition (4) states that the current parameters x_k with the new model y_k^{re} can not provide an error much worse than the one defined by x_k and the previous model y_k. At Step 2, we define a penalty term that will help to decide acceptance or rejection of a new pair model-parameters. At Step 3, the choice of a new y_{k+1} that may be different from y_k^{re} is crucial because, in practice, y_{k+1} may represent a simpler model than y_k^{re} (and even simpler than y_k) thus avoiding computing work. For example, y_k and y_k^{re} could involve simulations with $\approx$ 10,000 chickens in the Broiler House problem, for which a good error has not been obtained yet. Therefore, it may be interesting to improve the error employing less computational effort, represented by a simulation with $\approx$ 1,000 chickens. However, the new model y_{k+1} needs to satisfy the requirements (7) and (8). In this section, we will show that (7) and (8) can be fulfilled and we will deduce complexity results coming from this fact.

Note that $y_{k+1} \in Y$ chosen at the optimization phase (Step 3) does not need to satisfy $h(y_{k+1}) \le h(y_k)$. Instead, y_{k+1} must only fulfill conditions (7) and (8), which impose descent to the objective function and to the merit function, respectively. Therefore, many different heuristics may be admissible for the choice of the first trial of y_{k+1} (called $y_{k,0}$ in the forthcoming Algorithm 3.1). For example, since the cost of the objective function, or the subproblem solution, generally increases when $h(y)$ tends to zero, we may try to maintain $y_{k+1} = y_0$ as far as possible. Algorithm 2.1 allows us such a choice while conditions (7) and (8) are satisfied (taking, say, $h(y_k^{\mathrm{re}}) \approx \frac{1}{2}h(y_k)$) for a suitable x-increment s_k. Therefore, we may solve many (possibly easy) optimization subproblems with a low-precision y_{k+1}, employing (possibly easy) function evaluations. However, our results in this section will show that this strategy cannot continue for ever. It will be shown that, although conditions (7) and (8) are tolerant, they ultimately force, for a given tolerance $\varepsilon_{feas} > 0$, that $h(y_k) \leq \varepsilon_{feas}$, preventing more than $O(1/\varepsilon_{feas})$ iterations such that $h(y_k) > \varepsilon_{feas}$.

Recall that Algorithm 2.1 has been defined without a stopping criterion. This means that, as defined, the algorithm does not stop when sufficient feasibility and sufficiently optimality, according to some given tolerances, are obtained. From the pure mathematical point of view, the algorithm continues iterating so that the generated sequence involves infinitely many iterates. In this context, given arbitrary tolerancies ε_{feas} and ε_{opt}, in Corollary 2.2, it will be proved that the number of iterations such that $h(y^k) > \varepsilon_{feas}$ is smaller than $O(1/\varepsilon_{feas})$; while, in Theorem 3.1, it will prove that, in the case $\nu = 2$, the number of iterations such that $\|\nabla_x f(x_k, y_k) + \nabla c_E(x_k)\lambda_k + \nabla c_I(x_k)\mu_k\| > \varepsilon_{opt}$ is smaller than $O(1/\varepsilon_{opt}^2)$. As a consequence, the number of iterations at which

$$h(y^k) > \varepsilon_{feas} \;\text{ or }\; \|\nabla_x f(x_k, y_k) + \nabla c_E(x_k)\lambda_k + \nabla c_I(x_k)\mu_k\| > \varepsilon_{opt}$$

will be proved to be smaller than $O(1/\varepsilon_{feas}) + O(1/\varepsilon_{opt}^2)$. This implies that a possible stopping criterion given by

$$h(y^k) \leq \varepsilon_{feas} \;\text{ and }\; \|\nabla_x f(x_k, y_k) + \nabla c_E(x_k)\lambda_k + \nabla c_I(x_k)\mu_k\| \leq \varepsilon_{opt}$$

would be satisfied in less than $O(1/\varepsilon_{feas}) + O(1/\varepsilon_{opt}^2)$ iterations. Similar considerations may be done for the case $\nu = 3$, with $O(\varepsilon_{opt}^{-3/2})$ replacing $O(1/\varepsilon_{opt}^2)$. One should notice here that we are discussing two tolerance levels, ε_{feas} and ε_{opt}. In general, $\varepsilon_{feas} = \varepsilon_{opt}$ is possible but we will explain later the benefits of allowing different tolerances for the feasibility and optimality.

Lemma 2.1 is a technical result related to the definition of the penalty parameter.

Lemma 2.1 *For all $k \in \mathbb{N}_+$, the choice of the penalty parameter θ_{k+1} at Step 2 of Algorithm 2.1 guarantees that*

$$\Phi(x_k, y_k^{\rm re}, \theta_{k+1}) \leq \Phi(x_k, y_k, \theta_{k+1}) + \frac{1-r}{2}\left(h(y_k^{\rm re}) - h(y_k)\right). \tag{9}$$

Proof: For all $k \in \mathbb{N}_+$, if (5) takes place then we have that $\theta_{k+1} = \theta_k$ and, therefore, (9) follows from (5). For all $k \in \mathbb{N}_+$, if (5) does not hold, it turns out that

$$\Phi(x_k, y_k^{\rm re}, \theta) > \Phi(x_k, y_k, \theta) + \frac{1-r}{2}\left(h(y_k^{\rm re}) - h(y_k)\right) \tag{10}$$

for $\theta = \theta_k$. On the other hand, since, by (3), $h(y_k^{\rm re}) < h(y_k)$ and, thus,

$$h(y_k^{\rm re}) - h(y_k) < \frac{1-r}{2}(h(y_k^{\rm re}) - h(y_k)),$$

we have that the negation of (10) holds strictly for $\theta = 0$. Since Φ is a linear function on θ, this means that there exists $\theta \in (0, \theta_k)$ that verifies

$$\Phi(x_k, y_k^{\rm re}, \theta) = \Phi(x_k, y_k, \theta) + \frac{1-r}{2}\left(h(y_k^{\rm re}) - h(y_k)\right). \tag{11}$$

This θ is the quantity θ_{k+1} given by (6), computed at Step 2 when (5) does not hold, meaning that in this case (9) holds by equality. This completes the proof. $\square$

Lemma 2.2 guarantees that, taking the (almost surely inefficient) choice $y_{k+1} = y_k^{\rm re}$ and $s_k = 0$ the conditions (7) and (8) are satisfied, which guarantees that Step 3 is well defined.

Lemma 2.2 *For all $k \in \mathbb{N}_+$, the choice $y_{k+1} = y_k^{\rm re}$ and $s_k = 0$ at Step 3 of Algorithm 2.1 satisfies (7) and (8).*

Proof: For all $k \in \mathbb{N}_+$, the fact that $y_{k+1} = y_k^{\rm re}$ and $s_k = 0$ satisfy (7) is trivial and the fact that $y_{k+1} = y_k^{\rm re}$ and $s_k = 0$ satisfy (8) follows from Lemma 2.1. $\square$

In Assumption A1, we state that a model $y_k^{\rm re}$ satisfying (3) and (4) can always be found. Note that (3) states that a model better than y_k can be found and (4) states that, at such a model, the fitting error must not grow excessively.

Assumption A1 *At Step 1 of Algorithm 2.1, for all $k \in \mathbb{N}_+$ it is possible to compute, in finite time, $y_k^{\rm re}$ satisfying (3) and (4).*

In many situations, satisfying condition (3) is trivial. This is the case of the examples considered in [30], [11], and the present contribution, where one controls a precision criterion represented by $h(y)$. The fulfillment of (4), on the other hand, is highly problem-dependent. When y is a continuous variable and f satisfies a Lipschitz condition with respect to y, a sufficient condition for (4) is given by $\|y_k - y_k^{\rm re}\| \leq \bar{\beta} h(y_k)$, where $\bar{\beta}$ is an algorithmic parameter that replaces β. In turn, this condition is satisfied, under suitable assumptions, when one uses some iterations of a Newton scheme for solving $h(y) = 0$. In the abstract case $y \in Y$, common sense leads to computing the sequence

$$\max_{k=0,1,2,\dots} \{[\beta_k]_+\},$$

where

$$\beta_k \equiv \frac{f(x_k, y_k^{\rm re}) - f(x_k, y_k)}{h(y_k)}$$

and $[\cdot]_+ = \max\{0, \cdot\}$ and verifying whether this sequence appears to be bounded above (meaning that Assumption A1 holds) or not.

Lemma 2.3 *Suppose that Assumption A1 holds. Then, Algorithm 2.1 is well defined.*

Proof: For all $k \in \mathbb{N}_+$, $y_k^{\rm re}$ satisfying (3) can be computed in finite time at Step 1 by Assumption A1. Lemma 2.2 shows that $y_{k+1} = y_k^{\rm re}$ and $s_k = 0$ is a possible choice at Step 3. This completes the proof. $\square$

In the following lemma, we prove that the parameters θ_k are bounded away from zero. Therefore, the merit function $\Phi(x, y, \theta)$ is always meaningfully affected by the functional value $f(x, y)$.

Lemma 2.4 *Suppose that Assumption A1 holds. Then, the sequence $\{\theta_k\}$ generated by Algorithm 2.1 is non-increasing, remains in $(0,1)$, and is bounded below by $\theta_{\min} > 0$ defined by*

$$\theta_{\min} = \min\left\{\theta_0, \frac{1-r^2}{2\,(1+\beta-r)}\right\}. \tag{12}$$

Proof: For all $k \in \mathbb{N}_+$, if (5) does not hold then we have that

$$\Phi(x_k, y_k^{\rm re}, \theta_k) > \Phi(x_k, y_k, \theta_k) + \frac{1-r}{2}\left(h(y_k^{\rm re}) - h(y_k)\right).$$

Substituting Φ by its definition and isolating θ_k, we obtain

$$\theta_k > \frac{(1+r)\left(h(y_k) - h(y_k^{\rm re})\right)}{2\left(f(x_k, y_k^{\rm re}) - f(x_k, y_k) + h(y_k) - h(y_k^{\rm re})\right)}. \tag{13}$$

The right-hand-side of (13) coincides with the definition (6) of θ_{k+1} that is used in the case in which (5) does not hold. Thus, in this case, we have that $\theta_{k+1} < \theta_k$. Since, when (5) holds, $\theta_{k+1} = \theta_k$, we have that $\theta_{k+1} \le \theta_k$ for all $k \ge 0$ as we wanted to prove.

We now need to show that θ_{k+1} defined by (6) is bounded away from zero. By the definition (6) of θ_{k+1}, (3), and (4), we have that

$$\frac{1}{\theta_{k+1}} = \frac{2}{1+r}\left[\frac{f(x_k, y_k^{\rm re}) - f(x_k, y_k)}{h(y_k) - h(y_k^{\rm re})} + 1\right] \le \frac{2}{1+r}\left[\frac{\beta h(y_k)}{(1-r)h(y_k)} + 1\right] = \frac{2}{1+r}\left[\frac{\beta}{(1-r)} + 1\right].$$

This means that, for all $k \ge 0$, if (5) does not hold and, in consequence, θ_{k+1} is defined by (6) then we have that

$$\theta_{k+1} \ge \frac{1-r^2}{2\,(1+\beta-r)}; \tag{14}$$

while $\theta_{k+1} = \theta_k$ if (5) hold. Therefore, (12) follows from (14) and the fact that $\theta_0 \in (0,1)$ is arbitrary and it may be below the computed lower bound (in this case we have $\theta_k = \theta_0$ for all k). The fact that $\theta_{\min} > 0$ follows directly from $\theta_0 > 0$, $\beta > 0$, and $r \in (0,1)$. Finally, the fact that $\{\theta_k\}$ remains in $(0,1)$ follows from $\theta_k \in [\theta_{\min}, \theta_0] \subset (0,1)$. □

Corollary 2.1 *Suppose that Assumption A1 holds. Let $\{\theta_k\}$ be the sequence generated by Algorithm 2.1 and define*

$$\rho_k = \frac{1}{\theta_k} - 1. \tag{15}$$

Then, $\{\rho_k\}$ is non-decreasing and $\rho_k \le \rho_{\max}$ for all $k \in \mathbb{N}_+$, where $\rho_{\max} = 1/\theta_{\min} - 1$ is a quantity that only depends on β, r, and θ_0.

Proof: This is a direct consequence of Lemma 2.4. □

In the following lemma, we prove that, in the case that $y_{k+1} = y_k^{\rm re}$, the fulfillment of (7) implies the fulfillment of (8). Clearly, this is not the case for an arbitrary choice of $y_{k+1} \in Y$.

Lemma 2.5 *For all $k \in \mathbb{N}_+$, if the choice $y_{k+1} = y_k^{\rm re}$ and $s_k \in \mathbb{R}^n$ at Step 3 of Algorithm 2.1 is such that (7) holds then (8) holds as well.*

Proof: Since $y_{k+1} = y_k^{\text{re}}$ and by the definition of Φ, we have that

$$\begin{aligned}\Phi(x_k + s_k, y_{k+1}, \theta_{k+1}) &= \Phi(x_k + s_k, y_k^{\text{re}}, \theta_{k+1}) \\ &= \theta_{k+1} f(x_k + s_k, y_k^{\text{re}}) + (1 - \theta_{k+1}) h(y_k^{\text{re}}) \\ &\leq \theta_{k+1} \left(f(x_k, y_k^{\text{re}}) - \alpha \|s_k\|^\nu\right) + (1 - \theta_{k+1}) h(y_k^{\text{re}}) \\ &\leq \theta_{k+1} f(x_k, y_k^{\text{re}}) + (1 - \theta_{k+1}) h(y_k^{\text{re}}) \\ &= \Phi(x_k, y_k^{\text{re}}, \theta_{k+1}),\end{aligned}$$

where the first inequality follows from (7) and the fact that, by Lemma 2.4, $\theta_{k+1} > 0$. Thus, (8) follows from Lemma 2.1. □

Assumption A2 *There exist $h_{\max} > 0$ and $f_{\min} \in \mathbb{R}$ such that, for all $y \in Y$ and $x \in \Omega$ we have that $h(y) \leq h_{\max}$ and $f(x, y) \geq f_{\min}$.*

The following lemma states that the series $\sum h(y_k)$ is convergent.

Theorem 2.1 *Suppose that Assumptions A1 and A2 hold. The series $\sum_{k=0}^{\infty} h(y_k)$ and $\sum_{k=0}^{\infty} h(y_k^{\text{re}})$ are convergent. Moreover,*

$$\sum_{k=0}^{\infty} h(y_k) \leq \frac{2}{(1-r)^2} \big((\rho_{\max} - \rho_0) h_{\max} + f(x_0, y_0) + \rho_0 h(y_0) - f_{\min}\big). \tag{16}$$

Proof: By (3), for proving that $\sum_{k=0}^{\infty} h(y_k^{\text{re}})$ is convergent, we only need to prove that $\sum_{k=0}^{\infty} h(y_k)$ is convergent. By Corollary 2.1,

$$\sum_{k=0}^{\infty} (\rho_{k+1} - \rho_k) = \lim_{k \to \infty} \rho_{k+1} - \rho_0 \leq \rho_{\max} - \rho_0 < \infty. \tag{17}$$

Now, since $\{h(y_k)\}$ is bounded, (17) implies that

$$\sum_{k=0}^{\infty} (\rho_{k+1} - \rho_k) h(y_k) \leq (\rho_{\max} - \rho_0) h_{\max} < \infty. \tag{18}$$

By the definition of the algorithm and (3), we have that

$$\begin{aligned}\Phi(x_{k+1}, y_{k+1}, \theta_{k+1}) &\leq \Phi(x_k, y_k, \theta_{k+1}) + \frac{1-r}{2} \left(h(y_k^{\text{re}}) - h(y_k)\right) \\ &\leq \Phi(x_k, y_k, \theta_{k+1}) - \frac{(1-r)^2}{2} h(y_k),\end{aligned} \tag{19}$$

that is equivalent to

$$f(x_{k+1}, y_{k+1}) + \rho_{k+1} h(y_{k+1}) \leq f(x_k, y_k) + \rho_{k+1} h(y_k) - \frac{(1-r)^2}{2\theta_{k+1}} h(y_k).$$

Adding and subtracting $\rho_k h(y_k)$ and rearranging, since $\theta_{k+1} \in (0,1)$, we get

$$\begin{aligned}\frac{(1-r)^2}{2} h(y_k) \;\; &\le \;\; (\rho_{k+1}-\rho_k) h(y_k) + \\ &\Big(\big(f(x_k,y_k)+\rho_k h(y_k)\big) - \big(f(x_{k+1},y_{k+1})+\rho_{k+1} h(y_{k+1})\big) \Big).\end{aligned}$$

Summing for $k = 0, 1, \ldots, m$, we obtain

$$\begin{aligned}\sum_{k=0}^{m} \frac{(1-r)^2}{2} h(y_k) \;\; &\le \;\; \sum_{k=0}^{m} (\rho_{k+1}-\rho_k) h(y_k) + \\ &\Big(\big(f(x_0,y_0)+\rho_0 h(y_0)\big) - \big(f(x_{m+1},y_{m+1})+\rho_{m+1} h(y_{m+1})\big) \Big).\end{aligned}$$

Therefore, since $\rho_{m+1} h(y_{m+1}) \ge 0$ and $f(x_{m+1},y_{m+1}) \ge f_{\min}$, we have that

$$\sum_{k=0}^{m} \frac{(1-r)^2}{2} h(y_k) \le \sum_{k=0}^{m} (\rho_{k+1}-\rho_k) h(y_k) + \Big(\big(f(x_0,y_0)+\rho_0 h(y_0)\big) - f_{\min} \Big).$$

Thus, by (18), taking limits for $m \to \infty$,

$$\sum_{k=0}^{\infty} \frac{(1-r)^2}{2} h(y_k) \le (\rho_{\max}-\rho_0) h_{\max} + f(x_0,y_0) + \rho_0 h(y_0) - f_{\min}$$

from which (16) follows. □

Given an arbitrary $\varepsilon_{feas} > 0$, the following result establishes a bound for the number of iterations k such that $h(y_k) > \varepsilon_{fdeas}$. It is shown that, after $O(1/\varepsilon_{feas})$ iterations, all the iterates are such that $h(y_k) \le \varepsilon_{feas}$.

Corollary 2.2 *Suppose that Assumptions A1 and A2 hold. Given $\varepsilon_{feas} > 0$, the number of indices k such that $h(y_k) > \varepsilon_{feas}$ is bounded above by*

$$\frac{1}{\varepsilon_{feas}} \left[\frac{2}{(1-r)^2} \Big((\rho_{\max}-\rho_0) h_{\max} + f(x_0,y_0) + \rho_0 h(y_0) - f_{\min} \Big) \right]. \tag{20}$$

Proof: Let $K \subset \mathbb{N}_+$ be the subset of indices such that $h(y_k) > \varepsilon_{feas}$. By Theorem 2.1, we have that

$$\sum_{k \in K} \varepsilon_{feas} \le \frac{2}{(1-r)^2} \Big((\rho_{\max}-\rho_0) h_{\max} + f(x_0,y_0) + \rho_0 h(y_0) - f_{\min} \Big),$$

from which (20) follows. □

Of course, the mere obtention of y_k such that $h(y_k) < \varepsilon_{feas}$ in short time is not an achievement at all. In Corollary 2.2 it was proved that this can be achieved by means of a process that admits to increase $h(y)$ when a condition concerning the objective function is satisfied. In other words, the question is "Can we accept y_{k+1}, perhaps not satisfying $h(y_{k+1}) < h(y_k)$?"; and the answer to this question is positive, provided that the descent conditions (7) and (8) are satisfied. In other words, a precision degradation is admissible if it can be compensated with sufficient progress in the objective function.

Theorem 2.2 *Suppose that Assumptions A1 and A2 hold. The series* $\sum_{k=0}^{\infty} \|s_k\|^\nu$ *is convergent. Moreover,*

$$\sum_{k=0}^{\infty} \|s_k\|^\nu \le \frac{1}{\alpha}\left(\frac{2\beta}{(1-r)^2}\Big((\rho_{\max}-\rho_0)h_{\max}+f(x_0,y_0)+\rho_0 h(y_0)-f_{\min}\Big)-f_{\min}+f(x_0,y_0)\right). \tag{21}$$

Proof: By (4) and (7),

$$\begin{aligned} f(x_{k+1},y_{k+1})-f(x_k,y_k) &= f(x_{k+1},y_{k+1})-f(x_k,y_k^{\text{re}})+f(x_k,y_k^{\text{re}})-f(x_k,y_k) \\ &\le -\alpha\|s_k\|^\nu+\beta h(y_k). \end{aligned}$$

Summing for $k=0,1,\ldots,m$, we obtain

$$f(x_{m+1},y_{m+1})-f(x_0,y_0) \le -\alpha\sum_{k=0}^{m}\|s_k\|^\nu+\beta\sum_{k=0}^{m}h(y_k).$$

Therefore, since $f(x_{m+1},y_{m+1}) \ge f_{\min}$,

$$\sum_{k=0}^{m}\|s_k\|^\nu \le \frac{1}{\alpha}\left(\beta\sum_{k=0}^{m}h(y_k)-f_{\min}+f(x_0,y_0)\right).$$

Taking limits for $m\to\infty$, by Theorem 2.1, we obtain (21). This completes the proof. $\square$

The following result establishes that the number of iterations at which $\|s_k\|>\varepsilon$ is $O(1/\varepsilon^\nu)$. This will be the final result of this section.

Corollary 2.3 *Suppose that Assumptions A1 and A2 hold. Given* $\varepsilon_{opt}>0$*, the number of iterates* k *at which* $\|s_k\|>\varepsilon_{opt}$ *is not bigger than*

$$\frac{1}{\varepsilon_{opt}^\nu}\left[\frac{1}{\alpha}\left(\frac{2\beta}{(1-r)^2}\Big((\rho_{\max}-\rho_0)h_{\max}+f(x_0,y_0)+\rho_0 h(y_0)-f_{\min}\Big)-f_{\min}+f(x_0,y_0)\right)\right]. \tag{22}$$

Proof: Let $K\subset\mathbb{N}_+$ be the subset of indices such that $\|s_k\|>\varepsilon_{opt}$. By Theorem 2.2, we have that

$$\sum_{k\in K}\varepsilon_{opt}^\nu \le \frac{1}{\alpha}\left(\frac{2\beta}{(1-r)^2}\Big((\rho_{\max}-\rho_0)h_{\max}+f(x_0,y_0)+\rho_0 h(y_0)-f_{\min}\Big)-f_{\min}+f(x_0,y_0)\right)$$

from which (22) follows. $\square$

3 Algorithms for the optimization phase

In Section 2, we showed that after a well determined finite number of iterations Algorithm 2.1 finds $y_k \in Y$ such that $h(y_k) \le \varepsilon_{feas}$. Moreover, eventually all the iterates satisfy this inequality and the number of iterations that are necessary for its fulfillment is $O(1/\varepsilon_{feas})$. Obviously, this result cannot be considered satisfactory because it tells nothing about the optimality with respect to x. In this section we will show that, with adequate choices of y_{k+1} and s_k at Step 3 of Algorithm 3.1, and under suitable assumptions, we obtain that a reasonable optimality condition is achieved after a finite number of iterations and employing a finite number of evaluations of f. Moreover, we will prove that the computational effort in terms of iterations and evaluations of f is $O(\varepsilon_{opt}^{-2})$ or $O(\varepsilon_{opt}^{-3/2})$, depending on the chosen value of $\nu \in \{2, 3\}$.

From now on, we assume that the functions $c_E : \mathbb{R}^n \to \mathbb{R}^m$ and $c_I : \mathbb{R}^n \to \mathbb{R}^p$ that define the feasible set Ω in (2) are continuously differentiable for all $x \in \mathbb{R}^n$. Their Jacobians will be denoted $c_E'(x)$ and $c_I'(x)$, respectively. Consequently, we denote $\nabla c_E(x) = [c_E'(x)]^T$ and $\nabla c_I(x) = [c_I'(x)]^T$. Moreover, for all $y \in Y$ we assume that $f(x, y)$ is continuously differentiable with respect to x.

Algorithm 3.1 describes how to compute y_{k+1} and s_k at Step 3 of Algorithm 2.1.

Algorithm 3.1. Assume that $\delta > 0$, $\kappa > 0$, $\sigma_{\min} > 0$, and $\tau > 1$ are given independently of k.

Step 1. Compute $H_k \in \mathbb{R}^{n\times n}$ symmetric and $y_{k,0} \in Y$ and set $\sigma_{k,0} = 0$ and $j \leftarrow 0$.

Step 2. If possible, find an approximate solution $s_{k,j}$ to

$$\begin{array}{ll} \underset{s\in\mathbb{R}^n}{\text{Minimize}} & \nabla_x f(x_k, y_{k,j})^T s + \frac{1}{2} s^T H_k s + \sigma_{k,j}\|s\|^\nu \\ \text{subject to} & c_E(x_k + s) = 0 \text{ and } c_I(x_k + s) \le 0 \end{array} \tag{23}$$

in the sense that

$$\nabla_x f(x_k, y_{k,j})^T s_{k,j} + \frac{1}{2} s_{k,j}^T H_k s_{k,j} + \sigma_{k,j}\|s_{k,j}\|^\nu \le 0, \tag{24}$$

$$c_E(x_k + s_{k,j}) = 0,\ c_I(x_k + s_{k,j}) \le 0, \tag{25}$$

and there exist vectors of multipliers $\lambda_{k,j} \in \mathbb{R}^m$, $\mu_{k,j} \in \mathbb{R}^p_+$ such that

$$\left\|\nabla_x f(x_k, y_{k,j}) + H_k s_{k,j} + \nabla[\sigma_{k,j}\|s\|^\nu]|_{\bar{s}_{k,j}} + \nabla c_E(x_k + s_{k,j})\lambda_{k,j} + \nabla c_I(x_k + s_{k,j})\mu_{k,j}\right\| \le \kappa\|s_{k,j}\|^{\nu-1} \tag{26}$$

and

$$\|\min\{-c_I(x_k + s_{k,j}), \mu_{k,j}\}\| \le \delta. \tag{27}$$

Step 3. If $s_{k,j}$ satisfying (24,25,26,27) was not found, define $y_{k,j+1} = y_k^{\rm re}$ and $\sigma_{k,j+1} = \max\{\sigma_{\min}, \tau\sigma_{k,j}\}$, set $j \leftarrow j + 1$, and go to Step 2.

Step 4. Consider the conditions

$$f(x_k + s_{k,j}, y_{k,j}) \le f(x_k, y_k^{\rm re}) - \alpha\|s_{k,j}\|^\nu \tag{28}$$

and

$$\Phi(x_k + s_{k,j}, y_{k,j}, \theta_{k+1}) \le \Phi(x_k, y_k, \theta_{k+1}) + \frac{1-r}{2}\left(h(y_k^{\rm re}) - h(y_k)\right). \tag{29}$$

If (28) does not hold or (29) does not hold, define $y_{k,j+1} = y_k^{\rm re}$, $\sigma_{k,j+1} = \max\{\sigma_{\min}, \tau\sigma_{k,j}\}$, set $j \leftarrow j+1$, and go to Step 2. Otherwise, define $y_{k+1} = y_{k,j}$, $s_k = s_{k,j}$, $\sigma_k = \sigma_{k,j}$, $\lambda_{k+1} = \lambda_{k,j}$, and $\mu_{k+1} = \mu_{k,j}$ and return.

Note that it is assumed that x_k is feasible and the algorithm tries to find an approximate solution $x_k + s_{k,j}$ to (23) satisfying (24) that (i) preserves feasibility (25), (ii) satisfies complementarity (27) with tolerance δ, and (iii) annihilates the gradient of the Lagrangian with tolerance $\kappa\|s_{k,j}\|^{\nu-1}$ (26). At a global solution to (23), that obviously satisfies (25), since $x_k + 0$ is feasible, we have that (24) holds. Moreover, assuming that the constraints satisfy some constraint qualification, (26) and (27) hold at a global solution to (23) for any $\delta \ge 0$. Therefore, the practicality of this algorithm depends on the simplicity of the constraints. In general, the approximate minimization of the objective function of the subproblem (23) preserving feasibility of $x_k + s$ is not easy to achieve. Only in special cases (that include boxes, polytopes, balls, spheres, intersections of sets of these types [32], and, in the matricial world, under orthonormality, idempotency, or positive semidefiniteness), there are algorithms that find feasible minimizers of the subproblem (23), at least when $\nu = 2$. If the constraints are arbitrary and one uses a standard nonlinear programming algorithm for minimizing the subproblem, the fulfillment of (24) and (25), as well as the complementary condition (27), can be expected under reasonable assumptions.

On the other hand, exact feasibility may fail to be satisfied. A remedy could be to project the output of the nonlinear programming algorithm onto the feasible region, if this is possible, with the hope that (24), (26), and (27) will be maintained. However, there are no guarantees that such procedure will be successful. Among the algorithms that could be employed with that purpose, the ones that guarantee optimal complexity (involving the constraints in this case) should be preferable [7]. If the projections to the feasible region are not exact the problem of preserving feasibility would persist as approximate feasibility with respect to the constraints on x ($x \in \Omega$) is not enough. Let us explain why. Assume, for a moment, that $x^k \in \Omega$ and, when solving the subproblem (23), we tolerate some infeasibility, say, $\|c_E(x_k + s_{k,j})\| \le \eta$, $\|c_I(x_k + s_{k,j})_+\| \le \eta$, with $\eta > 0$ small. As a consequence of this, it is possible that x^{k+1} violates the constraints with an error η. This means that in the next iteration ($k \leftarrow k+1$), $x^k \notin \Omega$ but belongs to an expanded set, say, $\Omega(\eta)$. But now, at the new iteration, we must find $x_k + s_k \in \Omega$ satisfying (7) and (8). But, since $x_k \notin \Omega$ we have no guarantee that such point exists. This would lead to relax the requirement $x_k + s_k \in \Omega$ to $x_k + s_k \in \Omega(\eta)$. But these are new constraints that could not be exactly satisfied again, so that we should need to relax them to $\|c_E(x_k + s_{k,j})\| \le 2\eta$, $\|c_I(x_k + s_{k,j})_+\| \le 2\eta$ and so on. The solution to this inconvenient could be to relax with $\eta/2, \eta/2 + \eta/4, \eta/2 + \eta/4 + \eta/8, \dots$ so that, ultimately, all the iterates belong to $\Omega(\eta)$. However, we found this procedure cumbersome and rather impractical to deserve its inclusion in the main algorithm of this paper.

Algorithm 3.1 may be viewed as an independent algorithm that seeks the fulfillment of the sufficient descent conditions (28) and (29) for the objective function and the merit function, respectively. For $\nu = 2$, the evaluation complexity of this algorithm under a Lipschitz condition on ∇f (Assumption A4) will be given in Lemma 3.1. It will be shown that the algorithm returns (i.e. stops its execution at Step 4) after a finite number of evaluations of f. Analogously, for $\nu = 3$, the evaluation complexity of Algorithm 3.1 under a Lipschitz condition on $\nabla^2 f$ (Assumption A7) will be given in Lemma 3.3.

3.1 Case $\nu = 2$

When $\nu = 2$ and $\sigma_{k,j} > \|H_k\|$ the objective function of (23) is a strictly convex quadratic. Therefore, problem (23) admits a global minimizer. If the constraints are simple enough, in particular, if they satisfy some constraint qualification, the KKT conditions are satisfied at any minimizer. Therefore, it is reasonable to state Assumption A3, which concerns the solvability of (24)–(27).

Assumption A3 *For all x_k and $y_{k,j}$ computed by Algorithm 2.1 and Algorithm 3.1 with $\nu = 2$, respectively, an approximate solution to (23) satisfying (24)–(27) can be computed whenever $\sigma_{k,j} > \|H_k\|$.*

Assumption A4 below state the Lipschitz-like assumptions that must be satisfied by f and ∇f in order to prove worst-case complexity of Algorithm 2.1 when it uses Algorithm 3.1 with $\nu = 2$ for computing s_k at Step 3.

Assumption A4 *There exists $\gamma_1 > 0$ such that for all x_k computed by Algorithm 2.1 and all $y_{k,j}$, $s_{k,j}$, and s_k computed by Algorithm 3.1 with $\nu = 2$,*

$$\|\nabla_x f(x_k + s_k, y_{k,j}) - \nabla_x f(x_k, y_{k,j})\| \leq \gamma_1 \|s_k\| \tag{30}$$

and

$$f(x_k + s_{k,j}, y_{k,j}) \leq f(x_k, y_{k,j}) + \nabla_x f(x_k, y_{k,j})^T s_{k,j} + \gamma_1 \|s_{k,j}\|^2. \tag{31}$$

In Lemma 3.1, we prove that after a given finite number of evaluations of f, Algorithm 3.1 finds s_k and y_{k+1} satisfying the conditions (28) and (29). This lemma also provides a bound for the regularization parameter. In the proof of this lemma we use (31) but we do not use (30).

Lemma 3.1 *Suppose that Assumptions A1–A4 hold. Then, for all $k \in \mathbb{N}_+$, Algorithm 3.1 with $\nu = 2$ is well defined and finishes after at most $\lceil \log_\tau (\|H_k\|/2 + \gamma_1 + \alpha) \rceil + 1$ evaluations of f with*

$$\sigma_k \leq \tau \left(\|H_k\|/2 + \gamma_1 + \alpha \right).$$

Proof: By (31) and (24),

$$\begin{aligned} f(x_k + s_{k,j}, y_{k,j}) &\leq f(x_k, y_{k,j}) + \nabla_x f(x_k, y_{k,j})^T s_{k,j} + \gamma_1 \|s_{k,j}\|^2 \\ &\leq f(x_k, y_{k,j}) - \frac{1}{2} s_{k,j}^T H_k s_{k,j} - \sigma_{k,j} \|s_{k,j}\|^2 + \gamma_1 \|s_{k,j}\|^2 \\ &\leq f(x_k, y_{k,j}) + \left(\|H_k\|/2 - \sigma_{k,j} + \gamma_1 \right) \|s_{k,j}\|^2. \end{aligned}$$

Therefore, if

$$\sigma_{k,j} \geq \|H_k\|/2 + \gamma_1 + \alpha,$$

we have that

$$f(x_k + s_{k,j}, y_{k,j}) \le f(x_k, y_{k,j}) - \alpha \|s_{k,j}\|^2.$$

Thus, since, by the definition of the algorithm, $\sigma_{k,j} = \tau^{j-1}\sigma_{\min}$ and $y_{k,j} = y_k^{\mathrm{re}}$ for all $j \ge 1$, we have that (28) holds for all j satisfying

$$j \ge \left\lceil \log_\tau \left(\frac{\|H_k\|/2 + \gamma_1 + \alpha}{\sigma_{\min}} \right) \right\rceil + 1.$$

Finally, by the definition of the algorithm, $y_{k,j} = y_k^{\mathrm{re}}$ for all $j \ge 1$. Thus, by Lemma 2.5, for all $j \ge 1$ such that (28) holds, we have that (29) holds as well. This completes the proof. $\square$

In Lemma 3.2, we prove that the gradient of the Lagrangian is bounded by a multiple of the increment s_k.

Lemma 3.2 *Suppose that Assumptions A1–A4 hold. Then, for all $k \in \mathbb{N}_+$, y_{k+1} and s_k computed by Algorithm 3.1 with $\nu = 2$ satisfy*

$$\begin{aligned} &\|\nabla_x f(x_k + s_k, y_{k+1}) + \nabla c_E(x_k + s_k)\lambda_{k+1} + \nabla c_I(x_k + s_k)\mu_{k+1}\| \\ &\qquad \le [\gamma_1 + \|H_k\| + 2\tau\left(\|H_k\|/2 + \gamma_1 + \alpha\right) + \kappa]\, \|s_k\|. \end{aligned} \tag{32}$$

Proof: For all k and j, by (26) for the case $\nu = 2$ and since $\nabla[\sigma_{k,j}\|s\|^2]|_{s_{k,j}} = 2\sigma_{k,j}s_{k,j}$, we have that

$$\begin{aligned} & \|\nabla_x f(x_k, y_{k,j}) + \nabla c_E(x_k + s_{k,j})\lambda_{k,j} + \nabla c_I(x_k + s_{k,j})\mu_{k,j}\| - \|H_k s_{k,j} + 2\sigma_{k,j}s_{k,j}\| \\ \le\; & \|\nabla_x f(x_k, y_{k,j}) + H_k s_{k,j} + 2\sigma_{k,j}s_{k,j} + \nabla c_E(x_k + s_{k,j})\lambda_{k,j} + \nabla c_I(x_k + s_{k,j})\mu_{k,j}\| \le \kappa \|s_{k,j}\|, \end{aligned}$$

or, equivalently,

$$\|\nabla_x f(x_k, y_{k,j}) + \nabla c_E(x_k + s_{k,j})\lambda_{k,j} + \nabla c_I(x_k + s_{k,j})\mu_{k,j}\| \le \left(\|H_k\| + 2\sigma_{k,j} + \kappa\right) \|s_{k,j}\|. \tag{33}$$

On the other hand,

$$\begin{aligned} &\|\nabla_x f(x_k + s_k, y_{k,j}) + \nabla c_E(x_k + s_{k,j})\lambda_{k,j} + \nabla c_I(x_k + s_{k,j})\mu_{k,j}\| \\ \le\; & \|\nabla_x f(x_k + s_k, y_{k,j}) - \nabla_x f(x_k, y_{k,j})\| + \left\|\nabla_x f(x_k, y_{k,j}) + \nabla c_E(x_k + s_{k,j})\lambda_{k,j} + \nabla c_I(x_k + s_{k,j})\mu_{k,j}\right\|. \end{aligned} \tag{34}$$

Therefore, since, by Lemma 3.1,

$$\sigma_k \le \tau\left(\|H_k\|/2 + \gamma_1 + \alpha\right),$$

we obtain that (32) follows from (33), (34), (30), and the definition of the algorithm. $\square$

Assumption A5 *There exists $H_{\max} > 0$ such that for all $k \in \mathbb{N}_+$, $\|H_k\| \le H_{\max}$.*

In Theorem 3.1, we prove that a possible stopping criterion given by $h(y_k) \le \varepsilon_{feas}$ and

$$\|\nabla_x f(x_k, y_k) + \nabla c_E(x_k)\lambda_k + \nabla c_I(x_k)\mu_k\| \le \varepsilon_{opt}$$

with $c_E(x_k) = 0$, $c_I(x_k) \le 0$, and $\|\min\{-c_I(x_k), \mu_k\}\| \le \delta$ is necessarily fulfilled in $O(\varepsilon_{feas}^{-1}) + O(\varepsilon_{opt}^{-2})$ iterations and evaluations of f.

Theorem 3.1 *Suppose that Assumptions A1–A5 hold and that Algorithm 3.1 with $\nu = 2$ is used to compute s_k and y_{k+1} at Step 3 of Algorithm 2.1. Then, Algorithm 2.1 generates an infinite sequence $\{(x_k, y_k, \lambda_k, \mu_k)\}$ and, given arbitrary $\varepsilon_{feas}, \varepsilon_{opt} > 0$:*

1. *The number of iterations such that $h(y_k) > \varepsilon_{feas}$ is not bigger than*

$$\frac{1}{\varepsilon_{feas}} \left[\frac{2}{(1-r)^2} \Big((\rho_{\max} - \rho_0) h_{\max} + f(x_0, y_0) + \rho_0 h(y_0) - f_{\min} \Big) \right]. \tag{35}$$

2. *For all $k \geq 1$, we have that*

$$c_E(x_k) = 0, \; c_I(x_k) \leq 0, \tag{36}$$

and

$$\| \min\{-c_I(x_k), \mu_k\} \| \leq \delta. \tag{37}$$

3. *The number of iterations such that*

$$\|\nabla_x f(x_k, y_k) + \nabla c_E(x_k)\lambda_k + \nabla c_I(x_k)\mu_k\| > \varepsilon_{opt}$$

is not bigger than

$$\frac{1}{\varepsilon_{opt}^2} \left[\frac{\frac{1}{\alpha}\left(\frac{2\beta}{(1-r)^2} \Big((\rho_{\max} - \rho_0) h_{\max} + f(x_0, y_0) + \rho_0 h(y_0) - f_{\min} \Big) - f_{\min} + f(x_0, y_0) \right)}{\left[\gamma_1 + H_{\max} + 2\tau \left(H_{\max}/2 + \gamma_1 + \alpha \right) + \kappa \right]^{-2}} \right]. \tag{38}$$

Moreover, at each iteration, no more than

$$\left\lceil \log_\tau \left(\frac{H_{\max}/2 + \gamma_1 + \alpha}{\sigma_{\min}} \right) \right\rceil + 1 \tag{39}$$

evaluations of f are performed.

Proof: The upper bound (35) on the number of iterations required to achieve the desired value of h is given by Corollary 2.2. The feasibility and δ-complementarity of all the iterates x_k (for $k \geq 1$), given by (36) and (37), are a direct consequence of the definition of Algorithm 3.1. The upper bound (38) on the number of iterations required to achieve the ε_{opt}-optimality is a direct consequence of Corollary 2.3, Lemma 3.2, and Assumption A4. Finally, the upper bound on the number of evaluations of f per iteration is a direct consequence of Lemma 3.1 and Assumption A4. □

3.2 Case $\nu = 3$

At Step 2 of Algorithm 3.1 we find an approximate minimizer of a quadratic approximation of $f(x, y_{k,j})$ plus a regularization term $\sigma_{k,j}\|s\|^\nu$. The case $\nu = 2$, studied in the previous subsection, corresponds to the situation in which the quadratic $\nabla_x f(x_k, y_{k,j})^T s + \frac{1}{2} s^T H_k s$ guaranteedly approximates $f(x_k + s)$ up to first order. So, the matrix H_k does not need to be a good approximation of the Hessian of f in the case $\nu = 2$. On the other hand, in the case $\nu = 3$ we will essentially assume that first and second derivatives of the quadratic coincide with first and second derivatives of the objective function, respectively, thus providing second-order approximation. Roughly speaking $\nu = 2$ corresponds to gradient-like and quasi-Newton schemes; whereas $\nu = 3$ resembles Newton-like approaches.

In the case $\nu = 3$, the global minimizer of (23) necessarily exists if $\sigma_{k,j} > 0$. Therefore, the following Assumption A6 is plausible.

Assumption A6 *For all x_k and $y_{k,j}$ computed by Algorithm 2.1 and Algorithm 3.1 with $\nu = 3$, respectively, an approximate solution to (23) satisfying (24)–(27) can be computed except, perhaps, when $j = 0$ (in which case $\sigma_{k,j} = 0$). In this case, the impossibility of computing an approximate solution to (23) satisfying (24)–(27) can be detected in finite time, independently of k.*

Assumption A7 states a weak version of the fact that the quadratic in (23) approximates $f(x_k + s, y_{k,j})$ in a similar way as the second-order Taylor approximation does.

Assumption A7 *There exists $\gamma_2 > 0$ such that for all x_k computed by Algorithm 2.1 and all $y_{k,j}$, $s_{k,j}$, and s_k computed by Algorithm 3.1 with $\nu = 3$,*

$$\|\nabla_x f(x_k + s_k, y_{k,j}) - [\nabla_x f(x_k, y_{k,j}) + H_k s_k]\,\| \le \gamma_2 \|s_k\|^2 \tag{40}$$

and

$$f(x_k + s_{k,j}, y_{k,j}) \le f(x_k, y_{k,j}) + \nabla_x f(x_k, y_{k,j})^T s_{k,j} + \frac{1}{2} s_{k,j}^T H_k s_{k,j} + \gamma_2 \|s_{k,j}\|^3. \tag{41}$$

In Lemma 3.3 we prove that, when the regularization parameter is bigger than a constant that only depends on γ_2 and α, the sufficient descent conditions (28) and (29) take place.

Lemma 3.3 *Suppose that Assumptions A1, A2, A6, and A7 hold. Then, for all $k \in \mathbb{N}_+$, Algorithm 3.1 with $\nu = 3$ is well defined and it finishes after at most $\lceil \log_\tau((\gamma_2 + \alpha)/\sigma_{\min}) \rceil + 1$ evaluations of f with $\sigma_k \le \tau(\gamma_2 + \alpha)$.*

Proof: By (41) and (24),

$$\begin{array}{rcl} f(x_k + s_{k,j}, y_{k,j}) & \le & f(x_k, y_{k,j}) + \nabla_x f(x_k, y_{k,j})^T s_{k,j} + \frac{1}{2} s_{k,j}^T H_k s_{k,j} + \gamma_2 \|s_{k,j}\|^3 \\ & \le & f(x_k, y_{k,j}) - (\sigma_{k,j} - \gamma_2)\, \|s_{k,j}\|^3. \end{array}$$

Therefore, if $\sigma_{k,j} \ge \gamma_2 + \alpha$, we have that

$$f(x_k + s_{k,j}, y_{k,j}) \le f(x_k, y_{k,j}) - \alpha \|s_{k,j}\|^3.$$

Thus, since, by the definition of the algorithm, $\sigma_{k,j} = \tau^{j-1}\sigma_{\min}$ and $y_{k,j} = y_k^{\rm re}$ for all $j \ge 1$, we have that (28) holds for all j satisfying

$$j \ge \left\lceil \log_\tau \left(\frac{\gamma_2 + \alpha}{\sigma_{\min}} \right) \right\rceil + 1.$$

Finally, by the definition of the algorithm, $y_{k,j} = y_k^{\rm re}$ for all $j \geq 1$. Thus, by Lemma 2.5, for all $j \geq 1$ such that (28) holds, we have that (29) holds as well. This completes the proof. $\square$

In Lemma 3.4, we prove that the gradient of the Lagrangian computed at every iteration is smaller than a multiple of $\|s_k\|^2$. This fact will be essential to prove that the complexity result in terms of $\|s_k\|$ given in Corollary 2.3 implies complexity $O(\varepsilon^{-3/2})$ in terms of the gradient of the Lagrangian.

Lemma 3.4 *Suppose that Assumptions A1, A2, A6, and A7 hold. Then, for all $k \in \mathbb{N}_+$, y_{k+1} and s_k computed by Algorithm 3.1 with $\nu = 3$ satisfy*

$$\|\nabla_x f(x_k + s_k, y_{k+1}) + \nabla c_E(x_k + s_k)\lambda_{k+1} + \nabla c_I(x_k + s_k)\mu_{k+1}\| \leq (\kappa + 3\tau(\gamma_2 + \alpha) + \gamma_2)\|s_k\|^2 \tag{42}$$

Proof: On the one hand, for all k and j, by the triangle inequality, we have that

$$\begin{aligned} &\|\nabla_x f(x_k + s_{k,j}, y_{k,j}) + \nabla c_E(x_k + s_{k,j})\lambda_{k,j} + \nabla c_I(x_k + s_{k,j})\mu_{k,j}\| \\ &\qquad\qquad - \|\nabla_x f(x_k, y_{k,j}) + H_k s_{k,j} - \nabla_x f(x_k + s_k, y_{k,j})\| - 3\sigma_{k,j}\|s_{k,j}\|^2 \\ &\leq \|\nabla_x f(x_k, y_{k,j}) + H_k s_{k,j} + (3\sigma_{k,j}\|s_{k,j}\|s_{k,j}) + \nabla c_E(x_k + s_{k,j})\lambda_{k,j} + \nabla c_I(x_k + s_{k,j})\mu_{k,j}\|. \end{aligned} \tag{43}$$

On the other hand, for all k and j, by (26) for the case $\nu = 3$ and since $\nabla[\sigma_{k,j}\|s\|^3]|_{s_{k,j}} = 3\sigma_{k,j}\|s_{k,j}\|s_{k,j}$, we have that

$$\|\nabla_x f(x_k, y_{k,j}) + H_k s_{k,j} + (3\sigma_{k,j}\|s_{k,j}\|s_{k,j}) + \nabla c_E(x_k + s_{k,j})\lambda_{k,j} + \nabla c_I(x_k + s_{k,j})\mu_{k,j}\| \leq \kappa\|s_{k,j}\|^2. \tag{44}$$

Therefore, by (43), (44), and (40), we have that

$$\|\nabla_x f(x_k + s_{k,j}, y_{k,j}) + \nabla c_E(x_k + s_{k,j})\lambda_{k,j} + \nabla c_I(x_k + s_{k,j})\mu_{k,j}\| \leq (\kappa + 3\sigma_{k,j} + \gamma_2)\|s_{k,j}\|^2.$$

Thus, (42) follows from the definition of the algorithm and the fact that, by Lemma 3.3, for all k and j, $\sigma_k \leq \tau(\gamma_2 + \alpha)$. $\square$

Theorem 3.2 is the final result of this section. It will be proved that the number of iterations and evaluations of f that are necessary to obtain that the gradient of the Lagrangian is smaller than ε_{opt} is $O(\varepsilon_{opt}^{-3/2})$. This result, combined with the complexity $O(\varepsilon_{feas}^{-1})$ with respect to $h(y)$, completes the complexity analysis of the algorithm for $\nu = 3$.

Theorem 3.2 *Suppose that Assumptions A1, A2, A6, and A7 hold and that Algorithm 3.1 with $\nu = 3$ is used to compute s_k and y_{k+1} at Step 3 of Algorithm 2.1. Then, Algorithm 2.1 generates an infinite sequence $\{(x_k, y_k, \lambda_k, \mu_k)\}$ and, given arbitrary $\varepsilon_{feas}, \varepsilon opt > 0$:*

1. *The number of iterations such that $h(y_k) > \varepsilon_{feas}$ is not bigger than*

$$\frac{1}{\varepsilon_{feas}}\left[\frac{2}{(1-r)^2}\Big((\rho_{\max} - \rho_0)h_{\max} + f(x_0, y_0) + \rho_0 h(y_0) - f_{\min}\Big)\right]. \tag{45}$$

2. For all $k \geq 1$ we have that

$$c_E(x_k) = 0,\ c_I(x_k) \leq 0, \tag{46}$$

and

$$\|\min\{-c_I(x_k), \mu_k\}\| \leq \delta. \tag{47}$$

3. The number of iterations such that

$$\|\nabla_x f(x_k, y_k) + \nabla c_E(x_k)\lambda_k + \nabla c_I(x_k)\mu_k\| > \varepsilon_{opt}$$

is not bigger than

$$\frac{1}{\varepsilon_{opt}^{3/2}} \left[\frac{\frac{1}{\alpha}\left(\frac{2\beta}{(1-r)^2}\Big((\rho_{\max} - \rho_0)h_{\max} + f(x_0, y_0) + \rho_0 h(y_0) - f_{\min}\Big) - f_{\min} + f(x_0, y_0)\right)}{[\kappa + 3\tau(\gamma_2 + \alpha) + \gamma_2]^{-3/2}} \right]. \tag{48}$$

Moreover, at each iteration, no more than

$$\left\lceil \log_\tau \left(\frac{\gamma_2 + \alpha}{\sigma_{\min}} \right) \right\rceil + 1 \tag{49}$$

evaluations of f are performed.

Proof: The upper bound (45) on the number of iterations required to achieve the desired value of h is given by Corollary 2.2. The feasibility and δ-complementarity of all the iterates x_k (for $k \geq 1$), given by (46) and (47), are a direct consequence of the definition of Algorithm 3.1. The upper bound (48) on the number of iterations required to achieve the ε_{opt}-optimality is a direct consequence of Corollary 2.3 and Lemma 3.4. Finally, the upper bound on the number of evaluations of f per iteration is a direct consequence of Lemma 3.3. □

Let us now explain the reasoning behind considering feasibility and optimality tolerance, ε_{feas} and ε_{opt}, separately. Naturally one should try to approach as much as possible the solution x using low precision in y, in such a way that few expensive iterations with high precision y should be performed and taking $\varepsilon_{feas} = \varepsilon_{opt}$ might cause high precision prematuraly. Our algorithm tries to keep low precision in two ways: On the one hand, at Step 1 of Algorithm 3.1 the choice of $y_{k,0}$ is arbitrary, which means that the precision $h(y_{k,0})$ could be not very strict. However, the corresponding trial point is accepted if the conditions (28) and (29) are fulfilled. Therefore, in specific implementations we can define a coarse value for $y_{k,0}$ (for example, $y_{k,0} = y_k$). The criterion for such a decision comes from our estimation of how far we are from the solution. On the other hand, if the value of the parameter r in Algorithm 2.1 is close to 1, even the precision of y_k^{re} may be only slightly better than the one of y_k, which, in practice, will delay the speed in which $h(y)$ tends to zero.

The estimation of how far we are from the solution may come from the actual value of the norm of the gradient of the Lagrangian. Recall that the complexity result reports the maximal number of iterations at which that quantity is bigger than ε_{opt}. Putting this observation together with the one above, it is natural to implement the method in such a way that, when the gradient of the Lagrangian is big, we choose both $h(y_k^{re})$ close to $y(k)$ and $y_{k,0} = y_k$.

Our method may be interpreted as a minimization method for $f(x, y)$, with respect to x, where y changes from one iteration to another. As a consequence, the complexity results are related to the complexity results enjoyed by algorithms for minimizing ordinary functions that only depend on x. Roughly speaking, the worst-case complexity results inherited by our method are $O(\varepsilon_{opt}^{-2})$ for gradient-like methods with quadratic regularization and $O(\varepsilon_{opt}^{-3/2})$ for Newton-like methods with cubic regularization. In practice, these estimatives are very pessimistic. Gradient-like methods under reasonable assumptions usually enjoy linear convergence (complexity $O(|\log \varepsilon_{opt}|)$ and Newton-like methods exhibit quadratic convergence (complexity $O(\log |\log \varepsilon_{opt}|))$. The complexity results that we prove in the paper do not assume the conditions that would produce more optimistic complexity results. This is the reason why it seems that the computer time used to obtain optimality is much bigger than the one used for getting feasibility if we consider $\varepsilon_{feas} = \varepsilon_{opt}$. Roughly speaking, in our case the speed of convergence to feasibility with respect to $h(y)$ in completely monitored by the algorithm, in such a way that the worst case is the that effectively occurs. On the other hand, the worst-case in terms of optimality is a lot worse than the one that occurs in most cases.

However, one can interpret the results in term of real complexity using the following reasoning. In general, we cannot give a general rule for estimating the cost of a function evaluation in terms of $h(y^k)$. But in many cases we can re-paramenterize $h(y)$ in such a way that the cost of one function evaluation $f(x, y)$ is $1/y$. If, in Algorithm 3.1 we have that, for all k, $y_{k+1} = y_k^{re}$, we have that $h(y_{k+1}) \le rh(y_k)$ for all k and, so, the number of iterations such that $h(y_k) > \varepsilon_{feas}$ is bounded above by $\log(\varepsilon_{feas}/h(y_0))/\log(r)$. This could represent an undesirable quick decrease, leading very soon to $h(y) < \varepsilon_{feas}$, where the cost of every function evaluation should be smaller than $1/\varepsilon_{feas}$. Fortunately, the possibility of choosing y_{k+1} different from y_k^{re} implies that the number of iterations such that $h(y_k) > \varepsilon_{feas}$ could reach $O(1/\varepsilon_{feas})$. This is proved in Theorem 3.1, formula (35). Now, at the optimization phase of every iteration k we call Algorithm 3.1 for performing one iteration of a (quadratic or cubic) regularization method with the objective of decreasing f and the merit function. The maximal number of function evaluations computed at these iterations is given by Theorem 3.1 (38) or Theorem 3.2 (48). Both estimations are very pessimistic because when we use gradient-like methods in the optimization phase we expect to achieve the desired precision in $O(|\log \varepsilon_{opt}|)$ iterations and when we use Newton-like methods we expect $O(|\log |\log \varepsilon_{opt}||)$ iterations with, ultimately, only one function evaluation per iteration. In general,

$$O(|\log |\log \varepsilon_{opt}||) < O(|\log \varepsilon_{opt}|) < O(1/\varepsilon_{feas}),$$

which means that, in general, optimality will be reached before reaching $h(y) \le \varepsilon_{feas}$. (In practice this will be reflected in a null or almost null increment $s_{k,0}$.) Since, at each optimization phase, the number of evaluations of f is $O(1)$, it turns out that the expected computer effort in terms of function evaluations is going to be less than

$$\sum_{k \le O(\log \varepsilon_{opt})} \frac{1}{y_k}.$$

Now, assuming that $h(y_k^{re}) = rh(y^k)$,

$$1/y_k = h(y_k) \ge h(y_{k-1}^{re}) = h(y^0)r^{k-1}.$$

Therefore, the expected computed number of function evaluations to achieve precisions ε_{feas} and ε_{opt} is

$$\sum_{k \leq O(\log \varepsilon_{opt})} r^{k-1} h(y^0) = h(y^0) \frac{1 - r^{O(\log \varepsilon_{opt})}}{1 - r}.$$

3.3 Sample Average Approximation

In this section, we consider the stochastic optimization problem given by

$$\text{Minimize } \mathbb{E}[g(x, \xi)] \text{ subject to } x \in \Omega, \tag{50}$$

where Ω is given by (2), $\xi \in \mathbb{R}^q$ is a random vector with probability distribution P, $g : \mathbb{R}^n \times \mathbb{R}^q \to \mathbb{R}$, and $\mathbb{E}[g(x, \xi)] = \int g(x, \xi) P(d\xi)$. The case of interest is the one in which $\mathbb{E}[g(x, \xi)]$ can not be analytically computed and, under reasonable assumptions, there exists $\underline{N} > 0$ such that a solution to problem (50) can be approximated by a solution to the deterministic optimization problem given by

$$\text{Minimize } \frac{1}{\underline{N}} \sum_{i=1}^{\underline{N}} g(x, \xi^i) \text{ subject to } x \in \Omega, \tag{51}$$

where $\xi^1, \dots, \xi^{\underline{N}}$ is an identically distributed (i.i.d.) random sample of $\underline{N}$ realizations of the random vector ξ. Problem (51), that is a deterministic problem, is known as Sample Average Approximation (SAA) problem; and solving it would provide an approximation to a solution to (50). However, $\underline{N}$ is in general large, making a method that at every iteration performs one or more evaluations of the objective function of (51) unaffordable.

The Sample Average Approximation (SAA) method (see, for example, [42]) is a Monte Carlo simulation-based approach that applies to problem (51). At iteration k, a subsample $\xi^1, \dots, \xi^{N_k}$ with, hopefully, $N_k \ll \underline{N}$ is considered; and the sample average function

$$\hat{f}_{N_k}(x) = \frac{1}{N_k} \sum_{i=1}^{N_k} g(x, \xi^i)$$

is approximately minimized subject to $x \in \Omega$ by a deterministic optimization method. Iterations proceed until $N_k = \underline{N}$ is satisfied. The sequence of natural numbers $N_0, N_1, N_2, \dots$ is known as schedule. By letting Y be a set of subsamples y such that if $|y| = \ell$ then $y = \{\xi^1, \dots, \xi^\ell\}$ and defining $h(y) = \underline{N} - |y|$ and

$$f(x, y) = \hat{f}_{|y|}(x), \tag{52}$$

we have that problem (51) can be seen as a particular case of problem (1). Therefore, Algorithms 2.1–3.1 can be used to find a solution to problem (51). This means that Algorithm 2.1–3.1 can be seen as an SAA method that possesses the theoretical properties given in Sections 2 and 3. These properties are summarized in the two theorems below.

The key feature of Algorithms 2.1–3.1 when interpreted as an SAA method is that its framework allows the reduction of the sample size, providing a theoretical framework for opportunistic heuristics that may reduce the computational cost of the objective function being evaluated.

Theorems 3.3 and 3.4 correspond to the cases in which Algorithm 3.1 is used with $\nu = 2$ and $\nu = 3$, respectively. On the one hand, both theorems say that after a finite number of iterations the sample size is equal to the desired size $\underline{N}$. On the other hand, Theorem 3.3 says that finding an optimal to problem (51) with precision ε requires $O(\varepsilon_{opt}^{-2})$ functional evaluations when $\nu = 2$; while Theorem 3.4 says that this task takes $O(\varepsilon_{opt}^{-3/2})$ functional evaluations when $\nu = 3$.

Theorem 3.3 *Suppose that Assumptions A1–A5 hold and that Algorithm 3.1 with $\nu = 2$ is used to compute s_k and y_{k+1} at Step 3 of Algorithm 2.1. Then, Algorithm 2.1 generates an infinite sequence $\{(x_k, y_k, \lambda_k, \mu_k)\}$ such that*

1. *For all $k \geq k_0$, we have that $|y_k| = \underline{N}$, where*

$$k_0 = \left\lceil \frac{2}{1-r^2}\Big((\rho_{\max} - \rho_0)h_{\max} + f(x_0, y_0) + \rho_0 h(y_0) - f_{\min}\Big)\right\rceil.$$

2. *For all $k \geq 1$, we have that*

$$c_E(x_k) = 0,\ c_I(x_k) \leq 0,\ \text{and}\ \|\min\{-c_I(x_k), \mu_k\}\| \leq \delta.$$

3. *Given an arbitrary $\varepsilon_{opt} > 0$, the number of iterations such that*

$$\|\nabla_x f(x_k, y_k) + \nabla c_E(x_k)\lambda_k + \nabla c_I(x_k)\mu_k\| > \varepsilon$$

is not bigger than

$$\frac{1}{\varepsilon_{opt}^2}\left\lceil \frac{\frac{1}{\alpha}\left(\frac{2\beta}{(1-r)^2}\Big((\rho_{\max} - \rho_0)h_{\max} + f(x_0, y_0) + \rho_0 h(y_0) - f_{\min}\Big) - f_{\min} + f(x_0, y_0)\right)}{\left[\gamma_1 + H_{\max} + 2\tau\left(H_{\max}/2 + \gamma_1 + \alpha\right) + \kappa\right]^{-2}}\right\rceil. \tag{53}$$

Moreover, at each iteration, no more than

$$\left\lceil \log_\tau\left(\frac{H_{\max}/2 + \gamma_1 + \alpha}{\sigma_{\min}}\right)\right\rceil + 1 \tag{54}$$

evaluations of f are performed.

Theorem 3.4 *Suppose that Assumptions A1, A2, A6, and A7 hold and that Algorithm 3.1 with $\nu = 3$ is used to compute s_k and y_{k+1} at Step 3 of Algorithm 2.1. Then, Algorithm 2.1 generates an infinite sequence $\{(x_k, y_k, \lambda_k, \mu_k)\}$ and we have that items 1 and 2 of Theorem 3.3 follow; while, given an arbitray $\varepsilon > 0$, item 3 of Theorem 3.3 follows substituting (53) and (54) with*

$$\frac{1}{\varepsilon_{opt}^{3/2}}\left\lceil \frac{\frac{1}{\alpha}\left(\frac{2\beta}{(1-r)^2}\Big((\rho_{\max} - \rho_0)h_{\max} + f(x_0, y_0) + \rho_0 h(y_0) - f_{\min}\Big) - f_{\min} + f(x_0, y_0)\right)}{\left[\kappa + 3\tau(\gamma_2 + \alpha) + \gamma_2\right]^{-3/2}}\right\rceil$$

and

$$\left\lceil \log_\tau\left(\frac{\gamma_2 + \alpha}{\sigma_{\min}}\right)\right\rceil + 1,$$

respectively.

In Theorems 3.3 and 3.4, $f(x, y)$ corresponds to (52), i.e. to $\hat{f}_{|y|}$. This means that the theorems are counting all together evaluations with different computational costs. However, as a consequence of the fact that inexact evaluations are being done a finite number of times, the SAA method described by Algorithms 2.1–3.1 possesses the same worst-case evaluation complexities already described in, for example, [6, 7, 34], for methods that perform exact evaluations of the objective function only. No need to say that, in practice, most of the evaluations are inexact, i.e. evaluations of a cheap objective funcion (see, for example, the numerical experiments in [11]); while the number of iterations remains the same. Several assumptions can be made about the cost of evaluating $f(x, y)$ in terms of the size of the sample ($h(y)$). Combined with suitable assumptions on the variability of f with respect to the sample size, optimal choices can be given for the size of the sample and the precision required at each subproblem, by means of which even one-shot choices of the sample are efficient (see [38], [41]). We do not have that type of information in the general case but, again, we believe that we can take advantage for its availability in particular situations in order to optimize the choice of parameters of our method.

4 Final remarks

In this paper we proved worst-case iteration and evaluation complexity results for algorithms inspired in our previous contributions [30, 11]. The new algorithms share the Inexact-Restoration flavor of [30] and [11], but may be studied and analyzed disregarding classical IR theories. In any case, our analysis here is self-contained so that the theoretical results may be followed by readers not aware of typical IR papers. The modifications of the new algorithms with respect to the previous ones rely, as usually in complexity theories, on the employment of regularization. We studied separately the case $\nu = 2$, which is suitable to analyze gradient-like and quasi-Newton approaches [26] and $\nu = 3$, which corresponds to the celebrated cubic regularization. Algorithms with $\nu > 3$ may be considered along the lines of [6, 17, 34] although they do not seem to be promising for improving this particular approach in practice.

Extensive numerical examples of applications of the methods revisited in the present work were given in [30] and [11]. In particular, the popular family of problems in which $f(x, y)$ is the average of a random variable over a given sample denoted by y was considered in [30, 11]. In this case, Y is the set of samples and $h(y)^{-1}$ is the sample size. In the so called diagonalization approach [39, 40], one solves a sequence of Sample Average Approximation (SAA) subproblems corresponding to samples of sizes N_k, taking the solution of the k-th subproblem as an initial approximation for solving the $(k+1)$-th subproblem and employing variable precisions ε_k. The sample sizes N_k are determined either heuristically or solving an additional nonlinear programming subproblem [39]. The additional nonlinear subproblem is defined assuming that the convergence rate of the solver used for each SAA subproblem is known. The approach presented in this paper allows us to take the decisions that are inherent to the SAA strategy, i.e. to determine the sample sizes of the SAA subproblems, with proven worst-case complexity. Notice that the proposed framework allows us to deal with the case of arbitrary $\underline{N}$, i.e. the presented theory stands valid for the case of unbounded samples as well. Covering both cases, bounded and unbounded sample sizes, is, to best of our knowledge, a unique property of the IR scheduling.

When y is a continuous variable and $h(y)$ is a continuous function, problem (1) admits several different interpretations. On the one hand, it may be seen as a particular Nash Equilibrium problem [31] with two players, in which the first player minimizes $\|h(y)\|$ and the second one minimizes $f(x, y)$ with respect to x. On the other hand, problem (1) can also be seen as a Bilevel Programming problem [20] in which the master minimizes $\|h(y)\|$ subject to the fact that $f(x, y)$ is minimized, with respect to x, in the lower level. The fact that complexity results can be obtained for an IR algorithm that tackles simplified versions of Nash Equilibrium and Bilevel Optimization problems suggests that similar IR techniques may be useful to prove complexity of general Bilevel Programming and Equilibrium algorithms. The application to Stochastic Optimization is immediate, specially in the case in which one minimizes risk measures as VaR and CVaR, which are recommendable in the case of catastrophic events and consequent determination of insurance policies [22]. IR techniques allow one to avoid "oversampling", providing a rational balance between accuracy and optimization, which is crucial in this type of problems. (Note that we are using the term "oversampling" according to its intuitive meaning, not in the algorithmic sense of image processing problems.) Robust Optimization [4] also provides a good number of potential applications of IR, perhaps with suitable modifications in order to cope uncertainty in the constraints, with an economic computation of scenarios.

From the technical point of view, considering that filter methods were used for globalization of Inexact Restoration algorithms in [24], it may be conjectured that filter techniques could be also employed for solving (1), replacing our regularization strategies and, perhaps, also providing complexity bounds. The techniques of [7] for constrained optimization, for which iteration and evaluation complexity can be proved, are connected in spirit to Inexact Restoration but, according to our knowledge, were not yet implemented in order to provide practical algorithms.

We believe that the main field of challenging applications of the techniques introduced in this paper corresponds to the problem of fitting parameters of expensive simulations, that cannot be solved without the help of clever reduced versions of the models. Of course, opportunistic heuristics may be employed that can be useful in specific applications, but the theoretically oriented algorithm developed here should probably be the first approach with applicability to a broad scope of situations.

References

[1] R. Andreani, S. L. Castro, J. L. Chela, A. Friedlander, and S. A. Santos, An inexact-restoration method for nonlinear bilevel programming problems, *Computational Optimization and Applications* 43, pp. 307–328, 2009.

[2] A. Asimakopulos, *Keynes's General Theory and Accumulation*, Cambridge University Press, 1991.

[3] N. Banihashemi and C. Y. Kaya, Inexact Restoration for Euler discretization of box-constrained optimal control problems, *Journal of Optimization Theory and Applications* 156, pp. 726–760, 2013.

[4] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski, *Robust Optimization*, Princeton University Press, Princeton, 2009.

[5] E. G. Birgin, L. F. Bueno, and J. M. Martínez, Assessing the reliability of general-purpose Inexact Restoration methods, *Journal of Computational and Applied Mathematics* 282, pp. 1–16, 2015.

[6] E. G. Birgin, J. L. Gardenghi, J. M. Martínez, S. A. Santos, and Ph. L. Toint, Worst-case evaluation complexity for unconstrained nonlinear optimization using high-order regularized models, *Mathematical Programming* 163, pp. 359–368, 2017.

[7] E. G. Birgin, J. L. Gardenghi, J. M. Martínez, S. A. Santos, and Ph. L. Toint, Evaluation complexity for nonlinear constrained optimization using unscaled KKT conditions and high-order models, *SIAM Journal on Optimization* 26, pp. 951–967, 2016.

[8] E. G. Birgin and J. M. Martínez, The use of quadratic regularization with cubic descent for unconstrained optimization, *SIAM Journal on Optimization* 27, pp. 1049–1074, 2017.

[9] E. G. Birgin, J. M. Martínez, and M. Raydan. Nonmonotone spectral projected gradient methods on convex sets. *SIAM Journal on Optimization* 10, pp. 1196–1211, 2000.

[10] E. G. Birgin, J. M. Martínez, and M. Raydan, Spectral Projected Gradient Methods: Review and Perspectives, *Journal of Statistical Software* 60, issue 3, 2014.

[11] E. G. Birgin, N. Krejić, and J. M. Martínez, On the employment of Inexact Restoration for the minimization of functions whose evaluation is subject to errors, *Mathematics of Computation* 87, pp. 1307–1326, 2018.

[12] E. G. Birgin and J. M. Martínez, Local convergence of an Inexact Restoration method and numerical experiments, *Journal of Optimization Theory and Applications* 127, pp. 229–247, 2005

[13] L. F. Bueno, A. Friedlander, J. M. Martínez, and F. N. C. Sobral, Inexact Restoration method for derivative-free optimization with smooth constraints, *SIAM Journal on Optimization* 23, pp. 1189–1231, 2013.

[14] C. Cartis, N. I. M. Gould, and Ph. L. Toint, On the complexity of steepest descent, Newton's and regularized Newton's methods for nonconvex unconstrained optimization, *SIAM Journal on Optimization* 20, pp. 2833–2852, 2010.

[15] C. Cartis, N. I. M. Gould, and Ph. L. Toint, Adaptive cubic regularization methods for unconstrained optimization. Part I: motivation motivation, convergence and numerical results, *Mathematical Programming* 127, pp. 245–295, 2011.

[16] C. Cartis, N. I. M. Gould, and Ph. L. Toint, Adaptive cubic regularization methods for unconstrained optimization. Part II: worst-case function and derivative complexity, *Mathematical Programming* 130, pp. 295–319, 2011.

[17] C. Cartis, N. I. M. Gould, and Ph. L. Toint, Universal regularization methods - varying the power, the smoothness and the accuracy, Preprint RAL-P-2016-010, Rutherford Appleton Laboratory, Chilton, England, 2016.

[18] B. H. Cervelin, D. Conti, M. A. Diniz-Ehrhardt, and J. M. Martínez, A computer model for particle-like simulation in broiler houses, *Computers and Electronics in Agriculture* 141, pp. 1–14, 2017.

[19] F. E. Curtis, D. P. Robinson, and M. Samadi, A trust-region algorithm with a worst-case iteration complexity of $O(\varepsilon^{-3/2})$, *Mathematical Programming* 162, pp. 1–32, 2017.

[20] S. Dempe, *Foundations of Bilevel Programming*, Kluwer Academic Publishers, Dordrecht, 2002.

[21] J. P. Dussault, ARCq: a new adaptive regularization by cubics, *Optimization Methods and Software* 33, pp. 322–335, 2018.

[22] Y. Ermoliev, *Stochastic Programming Methods*, Academy of Sciences of the USSR, Kiev, 1967.

[23] A. Fischer and A. Friedlander, A new line search Inexact Restoration approach for nonlinear programming, *Computational Optimization and Applications* 46, pp. 333–346, 2010.

[24] C. C. Gonzaga, E. W. Karas, and M. Vanti, A globally convergent filter method for nonlinear programming, *SIAM Journal on Optimization* 14, pp. 646–669, 2004.

[25] G. N. Grapiglia and Yu. Nesterov, Globally convergent second-order schemes for minimizing twice differentiable functions, CORE Discussion Paper 2016/28, Université Catholique de Louvain, Louvain, Belgium, 2016.

[26] G. N. Grapiglia, J-Y Yuan, and Y-X Yuan, On the convergence and worst-case complexity of trust-region and regularization methods for unconstrained optimization, *Mathematical Programming* 152, pp. 491–520, 2015.

[27] A. Griewank, *The modification of Newton's method for unconstrained optimization by bounding cubic terms*, Technical Report NA/12, Department of Applied Mathematics and Theoretical Physics, University of Cambridge, 1981.

[28] A. O. Herrera, H. D. Scolnik, G. Chichilnisky, G. C. Gallopin, J. E. Hardoy, *Catastrophe or New Society? A Latin America world model*, IDRC, Ottawa, ON, CA, 1976.

[29] C. Y. Kaya and J. M. Martínez, Euler discretization and Inexact Restoration for optimal control, *Journal of Optimization Theory and Applications* 134, pp. 191–206, 2007.

[30] N. Krejić and J. M. Martínez, Inexact Restoration approach for minimization with inexact evaluation of the objective function, *Mathematics of Computation* 85, pp. 1775–1791, 2016.

[31] Z. Luo, J. Pang, and D. Ralph, *Mathematical Programs with equilibrium constraints*, Cambridge University Press, Cambridge, 1996.

[32] J. M. Martínez, Local minimizers of quadratic functions under spherical and ball constraints, *SIAM Journal on Optimization* 4, pp. 159–176, 1994.

[33] J. M. Martínez, Inexact Restoration method with Lagrangian tangent decrease and new merit function for nonlinear programming, *Journal of Optimization Theory and Applications* 111, pp. 39–58, 2001.

[34] J. M. Martínez, On high-order model regularization for constrained optimization, *SIAM Journal on Optimization* 27, pp. 2447–2458, 2017.

[35] J. M. Martínez and E. A. Pilotta, Inexact restoration algorithms for constrained optimization, *Journal of Optimization Theory and Applications* 104, pp. 135–163, 2000.

[36] J. M. Martínez and M. Raydan, Cubic-regularization counterpart of a variable-norm trust-region method for unconstrained minimization, *Journal of Global Optimization* 68, pp. 367–385, 2017.

[37] Y. Nesterov and B. T. Polyak, Cubic regularization of Newton's method and its global performance, *Mathematical Programming* 108, pp. 177–205, 2006.

[38] R. Pasupathy, On Choosing Parameters in Retrospective-Approximation Algorithms for Stochastic Root Finding and Simulation Optimization, *Operations Research* Vol. 58, No. 4 (2010), pp. 889-901.

[39] E. Polak, J. O. Royset, Eficient sample sizes in stochastic nonlinear programing, *Journal of Computational and Applied Mathematics* 217, pp. 301–310, 2008.

[40] J. O. Royset, Optimality functions in stochastic programming, *Mathematical Programming* 135, pp. 293–321, 2012.

[41] J.O. Royset, Szechtman, R., Optimal Budget Allocation for Sample Average Approximation *Operations Research*, 61, 3, pp. 762-776.

[42] R. Y. Rubinstein and A. Shapiro, *Discrete Event Systems: Sensitivity Analysis and Stochastic Optimization by the Score Function Method*, Wiley, New York, 1993.