American Journal of Operations Research, 2013, 3, 431-438
http://dx.doi.org/10.4236/ajor.2013.35041 Published Online September 2013 (http://www.scirp.org/journal/ajor)

On Cost Based Algorithm Selection for Problem Solving

Edilson F. Arruda[1], Fabrício Ourique[2], Anthony Almudevar[3], Ricardo C. Silva[4]
[1]Federal University of Rio de Janeiro, Alberto Luiz Coimbra Institute-Graduate School and Research in Engineering, Industrial Engineering Program, Rio de Janeiro, RJ, Brazil
[2]Federal University of Santa Catarina, Campus Araranguá, Araranguá, SC, Brazil
[3]Department of Biostatistics, University of Rochester Medical Center, Rochester, NY, USA
[4]Institute of Science and Tehcnology, Federal University of Sao Paulo, So Jos dos Campos, SP, Brazil
Email: efarruda@po.coppe.ufrj.br, Almudevar@urmc.rochester.edu, ricardo.coelho@unifesp.br, fabricio.ourique@ufsc.br

Received July 9, 2013; revised August 9, 2013; accepted August 16, 2013

ABSTRACT

This work proposes a novel framework that enables one to compare distinct iterative procedures with known rates of convergence, in terms of the computational effort to be employed to reach some prescribed vicinity of the optimal solution to a given problem of interest. An algorithm is introduced that decides between two competing algorithms, which algorithm makes the best use of the computational resources for some prescribed error. Several examples are presented that illustrate the trade-offs involved in such a choice and demonstrate that choosing an algorithm over another with a higher rate of convergence can be perfectly justifiable in terms of the overall computational effort.

Keywords: Decision Analysis; Computational Effort; Numerical Analysis Optimization

1. Introduction

It is not automatic that the runner with the longest stride will win a marathon, although it can be ascertained that she will complete the distance with the fewest number of strides. The same analysis can be applied to numerical algorithms: it is certain that the algorithm with the faster convergence rate will take fewer steps to converge, but it does not necessarily follow that its convergence is the fastest. This happens because convergence rate is often defined in terms of the iteration counter. Hence, an analysis based solely on such a rate is equivalent to picking as the marathon winner the runner with the longest stride.

This paper is concerned with proposing new measures for algorithm efficiency based on the computational effort. Such a measure can be more appropriate than the usual convergence rate with respect to the iteration counter. It permits us to assess how efficiently an algorithm employs the computational resources at hand, while searching for a solution to a given problem. That, in turn, allows one to compare algorithms not in terms of how many steps they employ to reach a solution, but based on how much computational effort (time) they apply to reach the solution. Alternatively, one can compare algorithms based on their usage of limited available computational effort, *i.e.* given a fixed amount of computational effort, one would wish to identify which algorithms get closer to the solution while applying at most the prescribed effort. In terms of our initial analogy, that would be equivalent to letting an athlete run for a fixed amount of time, declaring the winner as the athlete who covers more ground in that prescribed time.

We stress that this paper strives to classify algorithms for a given specific application. Hence, the rationale is to compare competing algorithms that can be used to solve a given problem up to some prescribed error tolerance, from a given starting point, with a view to identifying the alternative that makes the best use of the computational resources available. One could equivalently state that the proposed approach is instance-based, *i.e.* it focuses on identifying the best algorithm for a specific instance of a given problem. That can be viewed as a complement to the theory of computational complexity [1-3], which attempts to establish bounds—typically worst case scenario bounds—for the convergence time for general classes of algorithms, typically based on the dimension of the solution domain. Such an approach can be described as algorithm based, since it strives to classify algorithms based on their performance bounds, and individual problem instances may have very little to do with these bounds, see for example [4,8], and [1,27]. It is worth pointing out that, under the proposed approach, the computational effort of an algorithm is no longer determined

by its convergence rate. Rather, it is determined by the total number of elementary operations (Floating point operations—FLOP, for example) iteration times per the total number of iterations [5-7].

The idea of identifying iterative procedures that enhance the efficiency of the search for the solution of a given problem is hardly new. Multigrid methods [8,9], in which a system is first discretized on a grid, following which the grid resolution is systematically refined, are commonly used in the numerical solution of partial differential equations [8,9]. The objective is to save computational resources in the process of reaching a vicinity of the optimal solution. Heuristic procedures, such as genetic algorithms [10], also rely on an efficient use of the computational resources, aiming at getting within a vicinity of the optimal solution in reduced time. However, when advocating for a given method over another one, one has often to rely on extensive empirical data or domain specific mathematics, e.g. [11]. The novel feature in this paper is that it proposes a general framework to compare distinct iterative procedures with known rates of convergence based on the overall computational effort prior to convergence.

The main contributions of this paper are twofold. Firstly, considering that the convergence time of an algorithm is not based on its iteration count, but rather on the overall computational effort it employs, we propose a framework for algorithm comparison based on how much of the (limited) computational resources each algorithm applies to reach a desired precision. Secondly, given two algorithms starting from the same initial point, we propose a routine that identifies which algorithm requires less computational effort to converge, for any given error tolerance. The proposed routine requires a previous knowledge of the properties of both algorithms: order of convergence and rate of convergence; as well as the ratio of the computational efforts per iteration of the algorithms under consideration.

This approach addresses directly a trade-off commonly encountered in the design of iterative algorithms. A higher rate or order of convergence is usually achieved at the cost of an increased computation time per iteration. If we specify a desired tolerance β, we can expect the number of iterations required to achieve this tolerance to be smaller for a faster converging algorithm for all small enough β. If we instead adopt total computation time as a metric, the question becomes whether or not faster convergence with respect to iteration will always overcome the disadvantage of an increased computation time per iteration, promising greater efficiency for all small enough β. We show that this is not the case, that is, an algorithm may have both higher rate and higher order of convergence than an alternative and still require greater computation time to achieve tolerance β for all $\beta > 0$, provided the computation effort per iteration exceeds that of the alternative by a large enough factor.

This paper is organized as follows. Section 2 addresses the properties of iterative algorithms. Section 3 derives bounds on the number of iterations to reach a desired precision. Section 4 makes use of these bounds to derive a framework for algorithm comparison based on the overall computational effort, and shows that, under some circumstances, algorithms with lower order of convergence can always converge faster than higher converging ones, provided the computation time per iteration of the latter algorithm exceeds that of the alternative by a large enough factor. Numerical experiments that illustrate the proposed approach are presented in Section 5. Finally, Section 6 concludes the paper.

2. Numerical Formulation

This paper deals with numerical algorithms which take the form a convergent iterative sequence

$$V_k = T\left(V_{k-1}\right), k = 1, 2, \cdots \tag{1}$$

given a starting element $V_0 = v_0 \in \mathscr{V}$, where $T : \mathscr{V} \to \mathscr{V}$ is an operator on a normed linear space $\left(\mathscr{V}, \|\cdot\|\right)$, and $\mathscr{V}$ is the solution domain. The objective is to converge to a fixed point $V' = T\left(V'\right), V' \in \mathscr{V}$, which provides the solution to a given problem of interest.

When assessing how many evaluations of mapping T in Equation (1) are necessary until we found ourselves within a prescribed vicinity of the fixed point V', two important attributes stand out, namely the order of convergence and the convergence rate, which are defined below:

Definition 1 (Order of Convergence) The algorithm converges with order d if

$$\lim_{k\to\infty} \frac{\left\|V_{k+1} - V'\right\|}{\left\|V_k - V'\right\|^d} < M, \tag{2}$$

for some scalar $M < \infty$, e.g. [12].

Definition 2 (Convergence Rate) An algorithm is said to have convergence rate M, if M satisfies Equation (2).

If $d = 1$ and $M < 1$, the algorithm is said to be linearly convergent. Observe that both the order of convergence d and the convergence rate M are defined with respect to the iteration counter.

Remark 1 *Note that both order of convergence and convergence rate are indicative of how fast an algorithm converges to the solution in terms of the number of iterations. While they may be used to obtain an estimate of how many iterations are needed for the algorithm to converge, they are not sufficient to determine the convergence time, for the latter depends also on how much*

time each iteration takes up to be completed.

Typically, one lets Algorithm (1) run for a finite number of iterations, until a prescribed vicinity of the solution V' is reached. Let $\varepsilon \in \mathbb{R}, \varepsilon > 0$ be a prescribed error tolerance. We can define the total number of iterations needed for convergence as

$$k(\varepsilon) = \min_{k>0} : \|V_k - V'\| < \varepsilon. \tag{3}$$

Let $g_k, k \geq 0$ represent the computational effort of iteration k of Algorithm (1). Then, the overall computational effort of Algorithm (1) to attain a desired precision ε is defined as

$$G(\varepsilon) = \sum_{n=1}^{k(\varepsilon)} g_n. \tag{4}$$

In the present analysis, we assume that the overall computational time to attain precision ε is proportional to the overall computational effort $G(\varepsilon)$. Hence, the following analysis can be solely based on the overall computational effort. We also stress that computational time and computational cost are used interchangeably in this paper.

Let

$$\alpha_k \triangleq \|V_k - V'\|, k = 0,1,\cdots \tag{5}$$

be a sequence of iteration errors with respect to the solution. Without loss of generality, we employ a normalized error sequence

$$\beta_k \triangleq \frac{\alpha_k}{\alpha_0}, k = 0,1,\cdots \tag{6}$$

Note that, regardless of the value of V_0, $\beta_0 = 1$. That greatly simplifies our subsequent analysis. Moreover, β_k can be seen as the ratio of improvement at iteration k with respect to the initial solution. For the sake of simplicity, we assume that

$$\frac{\|V_{k+1} - V'\|}{\|V_k - V'\|^d} < M, \forall k \geq 0.$$

That can be accomplished by having an arbitrarily high index k relabeled as zero. Hence, we have

$$\alpha_{k+1} < M\alpha_k^d, k \geq 0,$$

which implies

$$\beta_{k+1} < \mathscr{M}\beta_k^d, \quad \mathscr{M} = M\alpha_0^{d-1} \tag{7}$$

Observe that a sufficient condition for the convergence of Algorithm (1) is $\mathscr{M} < 1$.

Remark 2 *We note that* $\mathscr{M}$, *defined in Equation* (7), *is a renormalization of the convergence rate that takes into account the initial error* α_0, *defined in* (5). *Such a renormalization is intended to simplify the subsequent analysis. Moreover, it enables us to assess the performance of the algorithm at iteration* k *by evaluating the attained relative improvement with respect to the initial solution* β_k, *as defined in* (7).

On the Definition of Computational Effort and Its Relation with the Computation Time

It must be acknowledged that the actual computation time of an algorithm does depend on the platform running the algorithm. Indeed, complexity theory acknowledges this issue and typically addresses it in two distinct ways:

- Assuming that the analysis is carried on for a single platform, see for example [4].
- By defining computational complexity (effort) in terms of elementary operations performed by the algorithm, e.g. [13].

In this text we assume that the computational effort is defined in terms of elementary operations. We also assume that the elementary operations are defined in such a way that does not depend on the platform. Additionally, the overall computational time (cost) is assumed to be proportional to the overall computational effort, with the platform determining only what the constant of proportionality is.

The definition of elementary operation is left to the user. Since our analysis is focused on the problem, the function $G(\varepsilon)$ could be tailor-made for the problem. Or it could, alternatively, be a general function, such as a counter of floating point operations.

3. An Upper Bound on the Iteration Counter Prior to Convergence

In this paper, we represent an iterative algorithm of the form in (1) by the pair (M,d). Hence, $A = (M,d)$ describes a convergence rate M with respect to the iteration count, with an iterative algorithm of order d.

We start this section with an upper bound on the error achieved by an iterative Algorithm $A = (M,d)$, of the form in (1), formalized in Theorem 1 below.

Theorem 1 Let $\beta_k, k \geq 1$, be the sequence in (6). Then

$$\beta_k \leq \mathscr{M}^{\sum_{i=1}^{k} d^{i-1}}, \tag{8}$$

where $\mathscr{M}$ is the quantity defined in (7).

Proof. It follows from (7) that Equation (8) holds for $k = 1$. Assume it also holds for $k = n$. Then, Equation (2) implies

$$\beta_{n+1} \leq \mathscr{M}\beta_n^d$$

$$\beta_{n+1} \leq \mathscr{M}\left(\mathscr{M}^{\sum_{i=1}^{n} d^{i-1}}\right)^d$$

$$\beta_{n+1} \leq \mathscr{M}^{\sum_{i=1}^{n} d^i + 1}$$

$$\beta_{n+1} \le \mathscr{M}^{\sum_{i=1}^{n+1} d^{i-1}}$$

Hence, Equation (8) also holds for $n+1$ and that completes the proof.

Let $\beta \in \mathbb{R}_+$ be a prescribed error and assume $d \ge 2$. Suppose also that after n of iterations we have

$$\beta_n < \mathscr{M}^{\sum_{i=1}^{n} d^{i-1}} < \beta.$$

The expression above yields:

$$\sum_{i=1}^{n} d^{i-1} \log_{\mathscr{M}} \mathscr{M} > \log_{\mathscr{M}}(\beta)$$

$$\frac{1-d^n}{1-d} > \log_{\mathscr{M}}(\beta)$$

$$d^n > (d-1)\log_{\mathscr{M}}\beta + 1$$

$$n > \log_d(d-1) + \log_d\left(\log_{\mathscr{M}}\beta\right)$$

If $d = 1$, Equation (8) implies:

$$\mathscr{M}^n \le \beta$$

$$n \ge \log_{\mathscr{M}}(\beta).$$

Consequently, an appropriate bound $k(\beta)$ on the number of iterations of Algorithm A to reach a tolerance β is

$$k(\beta) = \begin{cases} \log_{\mathscr{M}}\beta, & d = 1 \\ 1 + \log_d\left(\log_{\mathscr{M}}\beta\right), & d \ge 2, \end{cases} \tag{9}$$

which we refer to as the iteration cost.

4. Algorithm Comparison Based on the Overall Computational Effort

For the analysis in this section, we assume that the computational effort of Algorithm (1) does not change with the iteration counter, *i.e.* $g_k = g, \forall k \ge 1$. In the analysis that follows, we use $k(\beta)$, defined in (9), as an estimate for the quantity defined in (3), which indicates the number of iterations required for a given Algorithm $A = (M, d)$ to reach a prescribed normalized error β. The objective is to assess the efficiency of the algorithm based on the overall computational effort prior to reaching the prescribed error β. Such an effort is defined as

$$E_\beta(A) = G(\varepsilon) = gk(\beta) = g \cdot \begin{cases} \log_{\mathscr{M}}\beta, & d = 1 \\ 1 + \log_d\left(\log_{\mathscr{M}}\beta\right), & d \ge 2, \end{cases} \tag{10}$$

where g is the *per iteration effort* (*PIE*): the computational effort of a single iteration of Algorithm A, and d is the order of convergence of A, and function $G: \mathbb{R} \to \mathbb{R}$ is defined in (4).

Now, suppose we have an alternative algorithm $A' = (M', d')$, with PIE $\mathbb{R}$. Then, in order to choose the best algorithm for a given problem, one can seek an interval I for values of α that yield

$$E_\beta(A) < E_\beta(A'). \tag{11}$$

If both algorithms have convergence orders higher than 1 $(d, d' > 1)$, Equation (11) implies

$$g\left(1 + \log_d\left(\log_{\mathscr{M}}\beta\right)\right) < g'\left(1 + \log_{d'}\left(\log_{\mathscr{M}'}\beta\right)\right)$$

$$g\left(1 + \log_d\left(\log_{\mathscr{M}}\beta\right)\right) < \alpha g\left(1 + \log_{d'}\left(\log_{\mathscr{M}'}\beta\right)\right)$$

$$\alpha > \frac{\left(1 + \log_d\left(\log_{\mathscr{M}}\beta\right)\right)}{\left(1 + \log_{d'}\left(\log_{\mathscr{M}'}\beta\right)\right)}.$$

Hence,

$$\bar{\alpha}(\beta) = \frac{\left(1 + \log_d\left(\log_{\mathscr{M}}\beta\right)\right)}{\left(1 + \log_{d'}\left(\log_{\mathscr{M}'}\beta\right)\right)} \tag{12}$$

is a threshold that indicates a situation when both algorithms are equivalent in terms of computational cost for a prescribed error β. Whenever $g' > \bar{\alpha} g$, algorithm A is more economical in terms of computational effort. Otherwise, A' is the more efficient algorithm to reach the prescribed error.

When both $A = (M, 1)$ and $A' = (M', 1)$ are linearly convergent, the threshold $\bar{\alpha}$ becomes:

$$\bar{\alpha}(\beta) = \frac{\log_{\mathscr{M}}\beta}{\log_{\mathscr{M}'}\beta} = \frac{\log \mathscr{M}'}{\log \mathscr{M}}. \tag{13}$$

Observe that, in such case, the threshold is independent of the tolerance, but it does depend on the initial solution through $\mathscr{M}$, defined in (7). If only $A = (M, 1)$ is linear, the threshold becomes

$$\bar{\alpha}(\beta) = \frac{\log_{\mathscr{M}}\beta}{\left(1 + \log_{d'}\left(\log_{\mathscr{M}'}\beta\right)\right)}. \tag{14}$$

With the results above, one can define a general procedure for selecting between any two competing algorithms $A = (M, d)$ and $A' = (M', d)$, the one with the fastest convergence with respect to the overall computational effort. Such a procedure is centered on the per iteration effort ratio

$$\alpha = \frac{g'}{g}, \tag{15}$$

with g and g' denoting, respectively the PIE of Algorithms A and A'. The procedure is summarized in Algorithm 1 below.

Algorithm 1 (Algorithm Selection Procedure)

1) Initialize $A = (M, d)$ and $A' = (M', d')$, g and g'.

2) Choose an appropriate error $\beta > 0$.

3) Evaluate α, by using Equation (15).

4) Determine $\alpha(\beta)$ by choosing the appropriate expression from Equation (12)-(14).

5) Define $I=(\alpha(\beta),\infty)$.

6) If $\alpha \in I$, select Algorithm A and terminate.

7) Select Algorithm A' and terminate.

In the following sections, we present some experiments which illustrate the tradeoffs for choosing from two iterative algorithms for solving a given problem. The experiments illustrate the thresholds for per iteration ratios and demonstrate the existence of situations for which the choice of a lower convergence algorithm is more efficient in terms of computational effort.

4.1. A Perspective on the Proposed Approach

We argue that the proposed approach to selecting algorithms for problem solving is instance-based in the sense that, given a problem and an initial solution, and fixed a desired tolerance, it guides the choice of which algorithm to apply. It enables us to select a priori which of the two alternatives converges employing less computational effort. So far as we know, no equivalent formulation has been introduced in the literature, which would be directly comparable.

A related approach that could be contrasted with the present formulation is complexity theory. However, complexity theory is typically centered on the algorithms, often providing worst-case bounds on the convergence time of algorithms for classes of problems. Such bounds can have very little to do with the particular instance of interest [4,8]. Moreover, the responses obtained would be of different nature: complexity theory would identify, for a given problem, which of two algorithms has a better performance in a worst-case scenario. Hence, the response would be static and a single algorithm would be identified. The proposed approach, on the other hand, could identify different algorithms as the best alternative for different problem instances. In fact, an example is presented in the next section where two different problem instances yield two different responses. Such an answer would not be possible within a complexity theory framework.

4.2. Asymptotic Comparison of Algorithms

Suppose A and A' are two linearly convergent algorithms. Note that the ratio $\bar{\alpha}(\beta)$ in Equation (13) does not depend on β. Thus, even if A' has the theoretically faster convergence rate, if it also has a PIE g' sufficiently larger than that applicable to A, it will be the inferior choice for all tolerance values β. In this section we consider this type of comparison for arbitrary order of convergence d.

A common intuiton is that an algorithm with the better rate or order of convergence will eventually outperform an alternative, in the sense that the computation time will be smaller for all small enough β. Accordingly, we say that g' overtakes A if for any two PIEs g, g' there exists β_0 such that $g'k'(\beta) < gk(\beta)$ for all $\beta \le \beta_0$, where $k(\beta)$, $k'(\beta)$ are the respective iteration costs. If neither algorithm overtakes the other, we say they are computationally equivalent.

Interestingly, this relation can be resolved by considering the order of convergence d alone. To see this, we first note that the question reduces analytically to an evaluation of the the limit $\tau = \lim_{\beta\to 0} \bar{\alpha}(\beta)$, where $\bar{\alpha}(\beta) = k(\beta)/k'(\beta)$, since if $\tau = \infty$ then A' overtakes A, while if $0 < \tau < \infty$ then A and A' are computationally equivalent. We next state the main result.

Theorem 2 Let $A=(M,d)$ and $A'=(M',d')$ be two algorithms. Then if $d=d'=1$, or if $d>1$ and $d'>1$ then A and A' are computationally equivalent. Conversely, if $d=1$ and $d'>1$ then A' overtakes A.

Proof. As remarked above, we proceed by evaluating $\tau = \lim_{\beta\to 0} \bar{\alpha}(\beta)$. The case of $d=d'=1$ follows directly from Equation (13). Then suppose $d,d'>1$. The derivative of $k(\beta)$ may be evaluated as $\mathrm{d}k(\beta)/\mathrm{d}\beta = (\beta\ln(\beta)\ln(d))^{-1}$. Using L'Hôpital's rule we obtain

$$\tau = \lim_{\beta\to 0} \frac{\beta\ln(\beta)\ln(d')}{\beta\ln(\beta)\ln(d)} = \frac{\ln(d')}{\ln(d)},$$

which is a positive finite constant, thus the theorem holds for the case $d,d'>1$. Finally, suppose $d=1$ and $d'>1$. Then we similarly argue that

$$\tau = \lim_{\beta\to 0} \frac{\beta\ln(\beta)\ln(d')}{\beta\ln(\mathscr{M})} = \infty,$$

so that A' overtakes A.

5. Numerical Examples

In order to grasp the meaning of the proposed analysis, a series of numerical experiments are presented in this section, which make comparisons between an incumbent algorithm $A=(M,d)$ with PIE g and a challenging algorithm $A'=(M',d')$, with PIE g'. The experiments depict a curve of threshold $\bar{\alpha}(\beta)$ values, as defined in Equations (12)-(14), for a prescribed range o convergence rates M', for fixed values of M,d and d'. Such a curve is here called the effort ratio frontier. We recall that the $\bar{\alpha}(\beta)$ represents the per iteration effort ratio for which both A and A' are equivalent in terms of the overall computational effort. For our experiments, we apply an initial point with $\alpha_0 = 1$ in (6), which implies $M = \mathscr{M}$ in (7), for each considered algorithm.

Figure 1 comprises the effort ratio frontier for linearly

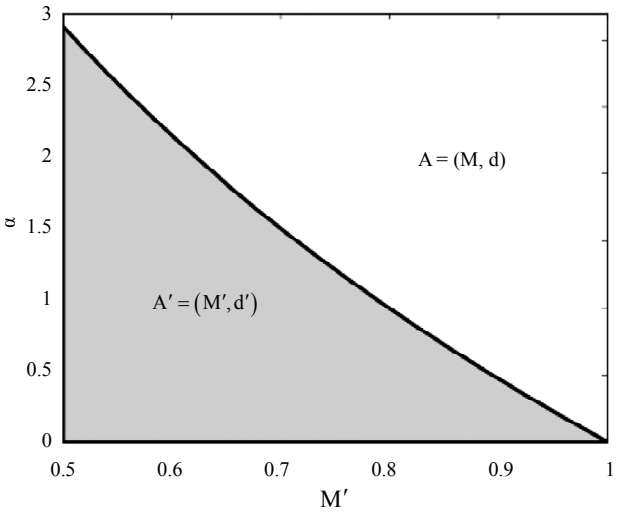

Figure 1. Effort ratio frontier, $A=(0.8,1)$, $d'=d=1$.

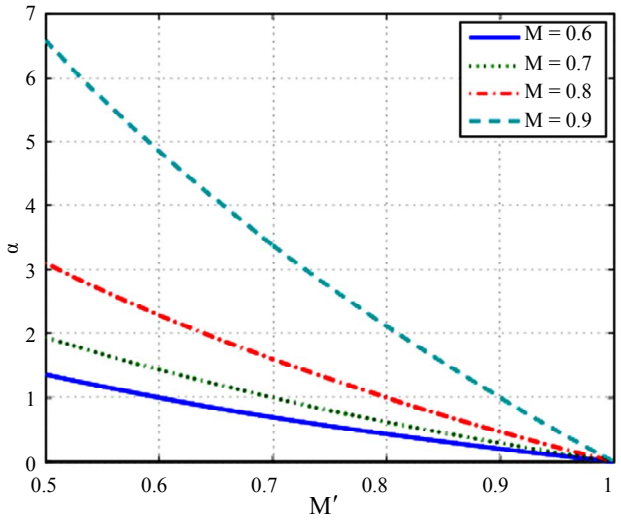

Figure 2. Behavior of effort ratio frontier, $d'=d=1$.

convergent algorithms $A=(0.8,1)$ and $A'=(M',1)$. It is worth mentioning that, since both algorithms are linear, Equation (14) implies that the threshold $\bar{\alpha}(\beta)$ is independent of the prescribed error β. As a result, the frontier in **Figure 1** is valid for all possible values of $\beta\in(0,1)$. As one can infer from Algorithm 1, the shadowed area below the frontier indicates the values of per iteration effort ratio (α) for which Algorithm A' is more efficient. For values of α outside of this area, A is a better choice. As an illustrative example, let us fix $M'=0.7$. For this value, we have $\alpha(\beta)=1.5$, which means that A' is a better choice whenever $g'<1.5g$ and A is a better choice whenever $g'>1.5g$.

In the second experiment, we wish to evaluate the effect of the convergence rate on the behavior of the effort ratio frontier. To this end, we compare two linearly convergent algorithms $A=(M,1)$ and $A'=(M',1)$, for varying M', while presenting a series of frontiers, for selected values of rate M. The results are depicted in **Figure 2**. One can notice that, as the convergence rate increases, *i.e.* Algorithm $A=(M,1)$ becomes slower, the value of the threshold $\alpha(\beta)$ increases. For example, $A'=(0.6,1)$ is preferable to $A=(M,1)$ if

$$\begin{cases} g'<g, & \text{if } M=0.6 \\ g'<\approx 1.5\cdot g, & \text{if } M=0.7 \\ g'<\approx 2.2\cdot g, & \text{if } M=0.8 \\ g'<\approx 4.9\cdot g, & \text{if } M=0.9. \end{cases}$$

The third experiment is aimed at providing some insight on the influence of the order of convergence on the effort ratio frontier. **Figure 3** conveys the frontier for an incumbent algorithm $A=(0.9,3)$ and a challenging algorithm $A'=(M',2)$, for a fixed $\beta=10^{-20}$. Note that $\alpha(\beta)=1$ for $M'=0.37$. That means that Algorithm

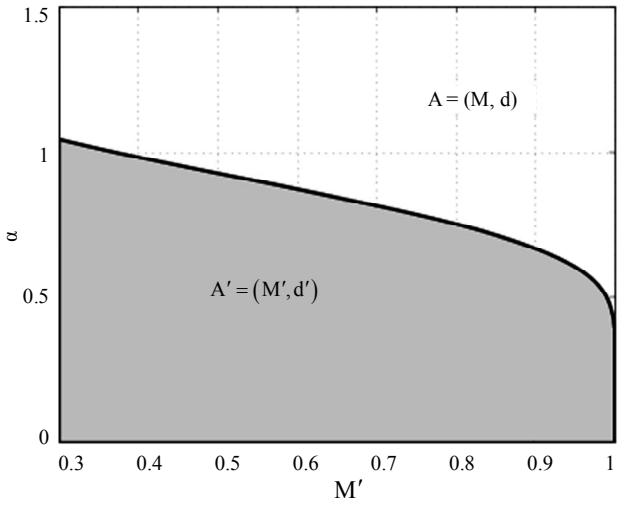

Figure 3. Effort ratio frontier, $A=(0.9,3)$, $d'=2$.

A is equivalent in terms of overall computation effort to a given $A'=(0.37,2)$, with the same effort per iteration. Moreover, any algorithm $A'=(M',2):M'<0.27$, with $g'=g$, is more efficient than A for the selected error β. This illustrates that the intuition that a higher order algorithm $(d>d')$ is always better than its lower order counterpart can be misleading. Furthermore, by Theorem 2, any two algorithms of order 2 and 3 are computationally equivalent, and it is therefore possible for the order 2 algorithm to have strictly smaller computation time for all $\beta>0$.

Our forth experiment generalizes the previous one and derives the effort ratio frontier for a challenging algorithm $A'=(M',2)$, with varying M', competing against incumbent algorithms $A=(0.9,d), d=2,3,4,5$, for $\beta=10^{-20}$. The results are depicted in **Figure 4**. The curve for a given order of convergence $d=2,\cdots,5$, illu-

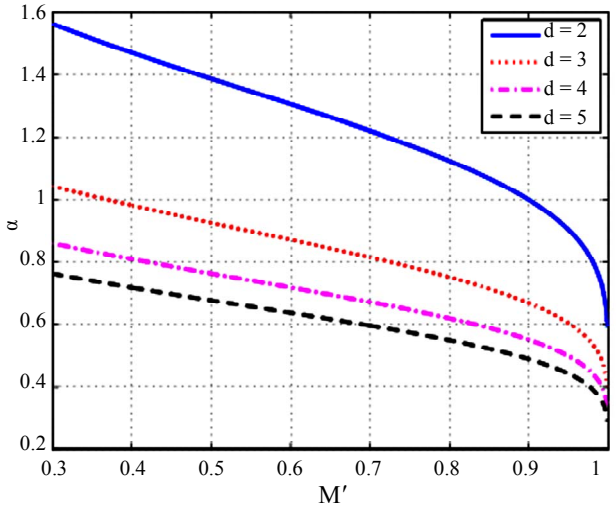

Figure 4. Order of convergence, $M = 0.9$, $d' = 2$.

strates the thresholds $\bar{\alpha}(\beta)$ below which the choice of A' is more advantageous. The collection of curves comprising **Figure 4** illustrate that a second order algorithm can outperform many higher order algorithms for appropriate values of per iteration effort. As an example, consider an algorithm $A' = (M', 2), M' \le 0.9$, with $g' = 0.4g$. Observe that an A' thus defined outperforms every other depicted algorithm, even the fifth-order algorithm $A = (0.9, 5)$.

The fifth experiment, depicted in **Figure 5**, replicates the previous experiment for varying values of error $\beta > 0$. The thicker-red line is the frontier for a precision $\beta = 10^{-50}$. The results show that lower order algorithms tend to be more attractive for higher values of β. However, even for very low values of β, lower order algorithms can remain appealing. Note in **Figure 5** that the region below the threshold $\alpha(\beta)$ does not vanish as $\beta \to 10^{-50}$, even when $A' = (M', 2)$ is competing against a fifth order algorithm $A = (0.95, 5)$. The set of curves for $A = (0.95, 5)$ show, for example, that any algorithm $A' = (0.7, 2)$ outperforms an incumbent algorithm $A = (0.95, 5)$, whenever $g' \le 0.6g$, for any precision β up to the order of 10^{-50}.

In the last experiment, we randomly generated a matrix $\boldsymbol{A}$ and a vector $\boldsymbol{b}$ to comprise a linear system of the form $\boldsymbol{Ax} = \boldsymbol{b}$, with 2395 equations and unknowns. Two well known algorithms were employed to solve this system: Gauss-Seidel and Conjugate Gradient. For both algorithms, the convergence rate was estimated as

$$\left(\frac{\beta_n}{\beta_0}\right)^{\frac{1}{n}}, \text{ where } \beta_i \approx \left\| x_{i+1} - x_i \right\|_\infty .$$

Here, n is the number of iterations up to convergence. The computational effort per iteration of each algorithm is the total number of sums and multiplications.

Figure 6 illustrates the results: the red-solid line and the blue-dashed line are the effort ratio frontiers for two distinct errors: $\beta = 10^{-5}$ and $\beta = 10^{-2}$, respectively. For each choice of error, the Conjugate Gradient Algorithm is the best choice for effort ratios α above the respective frontier, whereas the Gauss-Seidel Algorithm performs better for effort ratios below the frontier. The blue-round marker below the frontier indicates that the Gauss-Seidel Algorithm $A' = (0.51, 1)$ outperforms the Conjugate Gradient Algorithm $A = (0.31, 1)$, for $\beta = 10^{-2}$. The red-square marker indicates that for a higher precision, $\beta = 10^{-5}$, the Conjugate Gradient Algorithm $A = (0.27, 1)$ outdoes the Gauss-Seidel Algorithm $A' = (0.59, 1)$.

6. Concluding Remarks

This paper introduces a novel approach for comparing algorithms in terms of their overall computational effort,

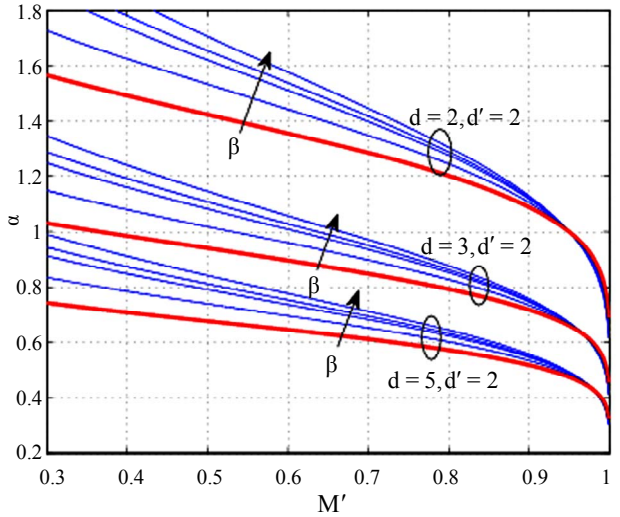

Figure 5. Influence of tolerance error on the boundary iteration effort, $M = 0.95$.

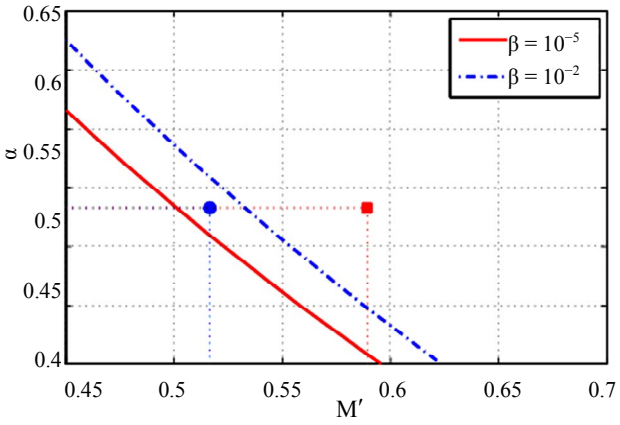

Figure 6. Algorithm comparison for a linear system example.

rather than in terms of the convergence-order, convergence-rate pair. A threshold is derived for the ratio between the per-iteration effort of two competing algorithms that indicate which of the competing algorithm makes the more efficient use of the computational resources available. In addition, an algorithm is proposed for choosing between two competing algorithms under the proposed setting, which makes use of this threshold.

The derived results are applied in a few examples that provide an insight on the compromises involved in the proposed approach. The experiments illustrate that a lower order algorithm can be more advantageous in terms of the overall computational effort to reach a prescribed error than a higher order counterpart. In particular, even as we let the prescribed error approach the order of 10^{-50}, using a lower order algorithm can be more advantageous under suitable conditions. This demonstrates that an analysis of algorithms based only on their order and rate of convergence can be very misleading. By applying an analysis based on the computational effort, on the other hand, one can identify the algorithm that makes the best use of the (limited) computational resources made available.

7. Acknowledgments

This work was partially supported by the Brazilian National Research Council-CNPq, under Grant No. 302716/2011-4.

REFERENCES

[1] T. H. Cormen, C. Stein, R. L. Rivest and C. E. Leiserson, "Introduction to Algorithms," 3rd Edition, MIT Press, Cambridge, 2009.

[2] J. Hartmanis and R. E. Stearns, "On the Computational Complexity of Algorithms," *Transactions of the American Mathematical Society*, Vol. 117, No. 5, 1965, pp. 285-306. doi:10.1090/S0002-9947-1965-0170805-7

[3] J. Belanger, A. Pavan and J. Wang, "Reductions Do Not Preserve Fast Convergence Rates in Average Time," *Algorithmica*, Vol. 23, No. 4, 1999, pp. 363-378.

[4] M. R. Garey and D. S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness," Freeman and Company, New York, 1979.

[5] D. B. Lloyd Trefethen, "Numerical Linear Algebra," SIAM, Philadelphia, 1997. doi:10.1137/1.9780898719574

[6] J. W. Demmel, "Applied Numerical Linear Algebra," SIAM, Philadelphia, 1997. doi:10.1137/1.9781611971446

[7] N. J. Higham, "Accuracy and Stability of Numerical Algorithms," SIAM, Philadelphia, 2002. doi:10.1137/1.9780898718027

[8] W. Hackbusch, "Multi-Grid Methods and Applications," 2nd Edition, Springer Series in Computational Mathematics, Springer-Verlag, Berlin, 2003.

[9] Y. Shapira, "Matrix-Based Multigrid: Theory and Applications," 2nd Edition, Springer Publishing Company, New York, 2008. doi:10.1007/978-0-387-49765-5

[10] M. Mitchell, "Introduction to Genetic Algoritms," MIT Press, Cambridge, 2008.

[11] C. Chow and J. N. Tsitsiklis, "An Optimal One-Way Multigrid Algorithm for Discrete-Time Stochastic Control," *IEEE Transactions on Automatic Control*, Vol. 36, No. 8, 1991, pp. 898-914. doi:10.1109/9.133184

[12] J. Nocedal and S. Wright, "Numerical Optimization," Springer, New York, 1999. doi:10.1007/b98874

[13] M. Hofri, "Analysis of Algorithms: Computational Methods and Mathematical Tools," Oxford University Press, New York, 1995.

