Comput Visual Sci (2011) 14:51–65
DOI 10.1007/s00791-011-0163-7

Iterant recombination with one-norm minimization for multilevel Markov chain algorithms via the ellipsoid method

Hans De Sterck · Killian Miller · Geoffrey Sanders

Received: 9 January 2011 / Published online: 20 November 2011

Abstract Recently, it was shown how the convergence of a class of multigrid methods for computing the stationary distribution of sparse, irreducible Markov chains can be accelerated by the addition of an outer iteration based on iterant recombination. The acceleration was performed by selecting a linear combination of previous fine-level iterates with probability constraints to minimize the two-norm of the residual using a quadratic programming method. In this paper we investigate the alternative of minimizing the one-norm of the residual. This gives rise to a nonlinear convex program which must be solved at each acceleration step. To solve this minimization problem we propose to use a deep-cuts ellipsoid method for nonlinear convex programs. The main purpose of this paper is to investigate whether an iterant recombination approach can be obtained in this way that is competitive in terms of execution time and robustness. We derive formulas for subgradients of the one-norm objective function and the constraint functions, and show how an initial ellipsoid can be constructed that is guaranteed to contain the exact solution and give conditions for its existence. We also investigate using the ellipsoid method to minimize the two-norm. Numerical tests show that the one-norm and two-norm acceleration procedures yield a similar reduction in the number of multigrid cycles. The tests also indicate that one-norm ellipsoid acceleration is competitive with two-norm quadratic programming acceleration in terms of running time with improved robustness.

Keywords Markov chain · Iterant recombination · Ellipsoid algorithm · Multigrid · Convex programming

Communicated by C. W. Oosterlee and A. Borzi Pl.

H. De Sterck · K. Miller (✉)
Department of Applied Mathematics, University of Waterloo, Waterloo, ON N2L 3G1, Canada
e-mail: k7miller@uwaterloo.ca

H. De Sterck
e-mail: hdesterck@uwaterloo.ca

G. Sanders
Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, P.O. Box 808, Livermore, CA 94551, USA
e-mail: sanders29@llnl.gov

1 Introduction

This paper deals with the numerical computation of the stationary distribution of large, sparse and irreducible Markov chains. In previous work [14], we showed how algebraic multigrid (AMG) methods for Markov chains can be accelerated by constrained minimization of the two-norm of linear combinations of iterants on the top level, using a quadratic programming solver. In this paper we explore the alternative of minimizing the one-norm with positivity constraints for iterant recombination acceleration. Our main motivation is to investigate whether a one-norm minimization method can be obtained that is competitive with the constrained two-norm minimization method of [14] in terms of compute time and robustness with regard to sign constraints. To do so, we develop a constrained one-norm iterant recombination minimization approach for Markov chains based on the ellipsoid method. We also show how the ellipsoid method can be used to solve the two-norm minimization problem. We consider the ellipsoid algorithm because it is easy to understand and implement efficiently, it is a robust solver for nonlinear programming problems [18], and at some levels of solution error it is competitive with other general-purpose solvers [18]. There are two additional but smaller motivations for considering one-norm minimization via the ellipsoid method in this paper. First, in probability theory, the one-norm is normally used to measure distances between probability vectors (for example, in the context of the theory of convergence speeds of Markov chains), which is natural since probability vectors are unit vectors in the one-norm. It is then only natural to also consider minimization in the one-norm in our algorithm. And second, one-norm minimization methods have recently raised significant interest in emerging fields such as compressive sensing, sparse representation, and sparse factorization. Consequently the question of whether one-norm minimization can be done efficiently as compared to two-norm minimization is receiving greater attention, and the current paper shows how this can be done for the Markov chain multigrid application at hand.

For a Markov chain with a finite state space, the problem of finding the stationary distribution may be stated as follows. We seek the vector $\mathbf{x} \in \mathbb{R}^n$ that satisfies

$$\mathbf{B}\,\mathbf{x} = \mathbf{x}, \quad \mathbf{1}^T\mathbf{x} = 1, \quad x_i \geq 0\ \forall i, \tag{1}$$

where $\mathbf{B} \in \mathbb{R}^{n\times n}$ is *column-stochastic* (i.e., each column is a probability distribution) and $\mathbf{1}$ is the vector of all ones. The matrix $\mathbf{B}$ corresponds to the transition probability matrix of the Markov chain, and the constraints ensure that $\mathbf{x}$ is a probability distribution. If the Markov chain is irreducible, then there exists a unique solution to (1), which has strictly positive components. This is a result of the Perron-Frobenius theorem for irreducible nonnegative matrices [3,22]. Recall that a square matrix $\mathbf{A}$ is *irreducible* if for any two distinct nodes i and j in the directed graph of $\mathbf{A}$, there exists a directed path from i to j [22]. In our work we find it beneficial to work in terms of the equivalent formulation of (1)

$$\mathbf{A}\,\mathbf{x} = \mathbf{0}, \quad \mathbf{1}^T\mathbf{x} = 1, \quad x_i \geq 0\ \forall i, \tag{2}$$

where $\mathbf{A} = \mathbf{I} - \mathbf{B}$ is an irreducible singular M-matrix.

Horton and Leutenegger were among the first to consider numerical methods for Markov chains with more than two levels [23,29], see also [27]. Their multilevel aggregation method, which we call AGG in this paper, is a direct extension of the two-level iterative aggregation/disaggregation (IAD) method for Markov chains due to Takahashi [40], which makes use of the aggregated equations proposed by Simon and Ando [38]. Since the pioneering work of Takahashi, two-level methods for Markov chains have been considered widely in the Markov chain literature [9,10,21,26,28,31–33]. Two-level IAD methods have been shown to be particularly effective at solving nearly completely decomposable (NCD) Markov chains, where the known structure of the problem can be exploited by the aggregation process, which typically results in fast convergence. Historically, multilevel methods were largely disregarded for Markov chain problems; however, recent work has resulted in new multilevel methods with significantly improved convergence behavior that is often nearly linear in the number of unknowns, even for difficult, *slowly mixing* Markov chains [8,11–15,41,42]. The methods in [11,12,14,15,41,42] are basically extensions of Horton and Leutenegger's AGG method from [23,29], with convergence properties that are often significantly improved, especially for the case of so-called slowly mixing Markov chains, in which the subdominant eigenvalues approach one as the size of the Markov chain increases. This improvement can be obtained by techniques from algebraic multigrid or smoothed aggregation, see, for example, [11,12] for detailed explanations and numerical illustrations. While theoretical convergence results are very difficult to obtain for any of these methods, especially for general nonsymmetric problems, in many cases empirical evidence has demonstrated good convergence properties and robustness.

The rest of this paper is organized as follows. In Sect. 2 we briefly discuss the multilevel algorithms we will consider in this paper, and outline the constrained iterant recombination acceleration algorithm from [14]. In Sect. 3 we give a brief description of the ellipsoid method for nonlinear convex programs. In Sect. 4 we develop a constrained one-norm iterant recombination minimization approach for Markov chains using the ellipsoid method. In particular, we derive formulas for subgradients of the objective function and the constraint functions, and as our main technical contribution we show how an initial ellipsoid can be constructed that is guaranteed to contain the exact solution, and give conditions for its existence. In Sect. 5 numerical tests are used to investigate whether the resulting one-norm minimization method is competitive with the two-norm minimization method from [14]. Section 6 contains concluding remarks.

2 Background

2.1 Standalone multigrid solver

Iterative solvers tend to perform poorly for slowly mixing Markov chains, which are characterized by their *subdominant eigenvalue* approaching unity in modulus as the size of their state space increases. A subdominant eigenvalue of a stochastic matrix $\mathbf{B}$ is any eigenvalue μ of $\mathbf{B}$ for which

$$|\mu| = \max\{|\lambda| : \lambda \in \Lambda(\mathbf{B}),\ |\lambda| \neq 1\},$$

where $\Lambda(\mathbf{B})$ is the spectrum of $\mathbf{B}$. Traditional iterative methods for computing the stationary probability vector such as the power method, may be unacceptably slow to converge for slowly mixing problems due to poor damping of the error component associated with the subdominant eigenvalue [35,39]. Multilevel methods aim to accelerate convergence for this type of problem by reducing error components with different scales on progressively coarser levels.

However, Horton and Leutenegger's original AGG method from [23,29] may still perform poorly for these types of Markov chains (see [11,12]), and large gains can often be made by the improvements considered in [11,12,14,15,41,42].

While iterant recombination acceleration can be applied to various classes of multilevel Markov chain algorithms [14], we limit the scope of this paper to two specific multigrid methods. First, we apply iterant recombination to the original multilevel aggregation algorithm (AGG) of [23], in which aggregates do not overlap. We use the neighborhood-based aggregation from [14] in this algorithm. We primarily consider the AGG algorithm since we want to illustrate that its performance can be significantly improved by iterant recombination, as an alternative to the more sophisticated improvements of [11,12,14,15,41,42]. For a few of our test problems, we also apply iterant recombination to one of those more sophisticated algorithms, namely, the MCAMG algorithm developed in [12]. MCAMG tends to perform in a scalable way for many test problems, but for those problems where it does not, we will show that iterant recombination can be useful. The framework of the MCAMG algorithm is similar to that of the multilevel method proposed by Horton and Leutenegger [23]; however, MCAMG uses an algebraic multigrid approach [7,43] to compute the interpolation and restriction operators and to define the coarse-level problem, see [12] for details.

2.2 Constrained iterant recombination acceleration

This paper considers acceleration of the convergence of multigrid methods for Markov chains by iterant recombination [6,43,44]. This process constructs an improved approximation as a linear combination of successive approximations from previous multigrid cycles, where the linear combination is chosen in such a way that the residual is minimized with respect to some norm. In this respect multigrid acceleration by iterant recombination is closely related to multigrid-preconditioned Krylov subspace iterations. For example, restarted GMRES with multigrid preconditioning is theoretically equivalent to multigrid acceleration by iterant recombination with a fixed number of previous iterates and two-norm residual minimization [43]. As such, the distinction between multigrid as a preconditioner and multigrid accelerated by iterant recombination depends largely on one's perspective. In this paper we find it natural to adopt the viewpoint of multigrid accelerated by iterant recombination.

Suppose we have a sequence of successive fine-level approximations $\{\mathbf{x}_i\}_{i\geq 1}$ from previous multigrid cycles. In order to find an improved approximation $\mathbf{x}^\star$, we consider

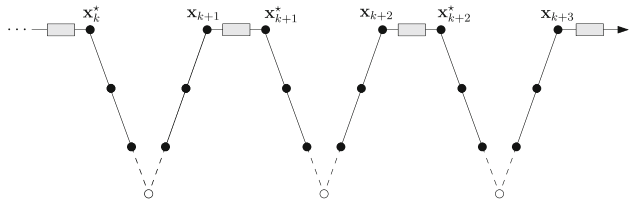

Fig. 1 Accelerated multigrid V-cycles. The *black dots* (●) represent relaxation operations on their respective levels and the *open dots* (○) represent coarse-level solves. An acceleration step, represented by a *grey* box, occurs after each V-cycle

a linear combination of the m most recent approximations $\mathbf{x}_k, \mathbf{x}_{k-1}, \ldots, \mathbf{x}_{k-m+1}$, where m is the *window size*. Let $\mathbf{X}$ be the $n \times m$ matrix

$$\mathbf{X} = [\mathbf{x}_{k-m+1}, \ldots, \mathbf{x}_{k-1}, \mathbf{x}_k],$$

where $\mathbf{x}_k$ is the most recent approximation, and assume that each column of $\mathbf{X}$ is a probability distribution with strictly positive entries (this is a property of the standalone multigrid solvers considered here). Then the improved approximation is given by $\mathbf{x}^\star = \mathbf{X}\mathbf{z}^\star$ for some $\mathbf{z}^\star \in \mathbb{R}^m$. This is repeated after each multilevel cycle where $\mathbf{x}^\star$ serves as the initial guess for the next cycle. This process is illustrated in Fig. 1 for multigrid V-cycles. Now we need some criteria on which to base our choice of $\mathbf{z}^\star$, and hence $\mathbf{x}^\star$. In [14] we computed $\mathbf{z}^\star$ by solving the following constrained minimization problem

$$\begin{aligned} \text{minimize} \quad & \mathcal{F}(\mathbf{X}\mathbf{z}) \\ \text{subject to} \quad & \mathbf{X}\mathbf{z} \geq 0 \\ & \mathbf{1}^T\mathbf{z} = 1 \end{aligned} \tag{3}$$

with $\mathcal{F}(\mathbf{x}) = \|\mathbf{A}\,\mathbf{x}\|_2$. We note that the constraints in (3) ensure that $\mathbf{x}^\star = \mathbf{X}\mathbf{z}^\star$ is a probability vector. The constraint $\mathbf{X}\mathbf{z} \geq 0$ is necessary to maintain nonnegative signs throughout the computations, not only because this is desired for probability vectors, but also because our multilevel cycles may become ill-posed if iterates with negative signs occur (see [11,12]). This two-norm minimization leads to a quadratic programming problem that can be solved with standard techniques [14]. In this paper we explore solving (3) with $\mathcal{F}(\mathbf{x}) = \|\mathbf{A}\,\mathbf{x}\|_1$. We minimize the one-norm using the ellipsoid method, and investigate whether the resulting one-norm minimization scheme is competitive with the two-norm minimization approach from [14]. The iterant recombination procedure is given by Algorithm 1 below. We note that in line 7 of Algorithm 1 the improved approximation is rejected if it does not yield a smaller residual then the current approximation.

Algorithm 1: Iterant recombination (window size m)

```
1. Set $k \leftarrow 1$ and choose an initial guess $\mathbf{x}_0^\star$
2. Set $\tau \leftarrow \tau_{rel}\,\|\mathbf{A}\,\mathbf{x}_0^\star\|_1$
3. Obtain the next multigrid iterate $\mathbf{x}_k$, with $\mathbf{x}_{k-1}^\star$ as the
   initial guess
4. Set $j \leftarrow \min\{k, m\}$
5. Set $\mathbf{X} \leftarrow [\mathbf{x}_{k-j+1}, \ldots, \mathbf{x}_{k-1}, \mathbf{x}_k]$
6. Solve (3) for $\mathbf{z}^\star$, and set $\mathbf{x}_k^\star \leftarrow \mathbf{X}\mathbf{z}^\star$
7. if $\|\mathbf{A}\,\mathbf{x}_k^\star\|_1 > \|\mathbf{A}\,\mathbf{x}_k\|_1$ then
       $\mathbf{x}_k^\star \leftarrow \mathbf{x}_k$
   end
8. Check convergence, $\|\mathbf{A}\,\mathbf{x}_k^\star\|_1 < \tau$, otherwise set
   $k \leftarrow k+1$ and go to 3
```

3 The ellipsoid method

In this section we give a brief description of the ellipsoid method for nonlinear convex programs. For an excellent overview of the development of the ellipsoid method we recommend the survey paper [5]. The book on linear optimization by Bertsimas and Tsitsiklis [4], and the papers [16–18] contain further useful information. The ellipsoid method was first described by Iudin and Nemirovskii in 1976 [24], and was later explicitly stated as we know it today by Shor in 1977 [37]. It gained notoriety in the early 1980s when Khachiyan showed that an ellipsoid method for linear optimization could be implemented with polynomial time complexity [25]. Although the ellipsoid method was not competitive in practice for linear optimization, it has shown itself to be a robust solver for nonlinear convex programs, which at some levels of solution error is competitive with other more mainstream solvers [18].

The ellipsoid method was originally intended as a solver for nonlinear convex optimization problems of the form

$$\begin{aligned}&\text{minimize} && f_0(\mathbf{x})\\ &\text{subject to} && \mathbf{x} \in S = \{\mathbf{y} \in \mathbb{R}^m : f_i(\mathbf{y}) \le 0,\ i = 1, \ldots, n\},\end{aligned} \tag{4}$$

where each $f_i : \mathbb{R}^m \to \mathbb{R},\ i = 0, \ldots, n$ is a finite convex function on $\mathbb{R}^m$. Here, the function f_0 is referred to as the *objective function* and the functions $f_1, \ldots, f_n$ are referred to as *constraint functions*. The set S of all points that satisfy the n inequality constraints is called the *feasible set*. Hence, any point belonging to S is called *feasible*, and any point not belonging to S is called *infeasible*. We assume that the feasible set is nonempty and that there exists an optimal solution $\mathbf{x}^\star$ to (4).

An m-dimensional ellipsoid is a subset of $\mathbb{R}^m$ defined as follows.

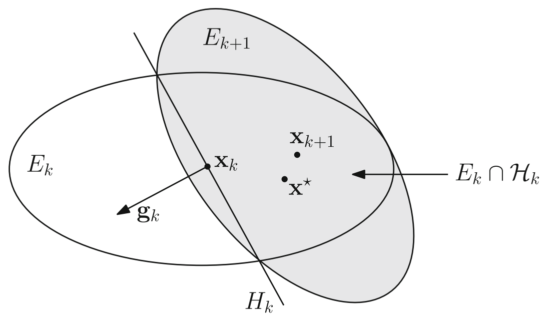

Fig. 2 Construction of the minimum volume ellipsoid E_{k+1} (in *grey*) from the ellipsoid E_k and hyperplane H_k. Here $\mathbf{g}_k$ is the normal vector to H_k

Definition 1 (*Ellipsoid*) Let $\mathbf{D}$ be an $m \times m$ symmetric positive definite (SPD) matrix and let $\mathbf{z}$ be any point in $\mathbb{R}^m$. Then the set

$$E(\mathbf{z}, \mathbf{D}) = \{\mathbf{x} \in \mathbb{R}^n : (\mathbf{x} - \mathbf{z})^T \mathbf{D}^{-1} (\mathbf{x} - \mathbf{z}) \le 1\}$$

is an ellipsoid with center $\mathbf{z}$.

Suppose that we have an initial ellipsoid E_0 that contains $\mathbf{x}^\star$. The ellipsoid method iteratively constructs a sequence of successively "smaller" ellipsoids each of which contains $\mathbf{x}^\star$. By smaller, we mean that the volume of the next ellipsoid is strictly smaller than the volume of the previous ellipsoid. Suppose now that $E_k = E(\mathbf{x}_k, \mathbf{D}_k)$ is the kth ellipsoid in this sequence. Then we can build E_{k+1} as follows. Construct a hyperplane H_k that passes through $\mathbf{x}_k$. Then $\mathbf{x}^\star$ is contained in one of the halfspaces generated by H_k, call it $\mathcal{H}_k$. Now define $E_{k+1} = E(\mathbf{x}_{k+1}, \mathbf{D}_{k+1})$ as the *minimum volume ellipsoid* that contains the intersection of E_k with $\mathcal{H}_k$. Since $\mathbf{x}^\star \in (E_k \cap \mathcal{H}_k)$, it follows that $\mathbf{x}^\star \in E_{k+1}$. Furthermore, if the center point $\mathbf{x}_{k+1}$ is feasible, then it is the $(k+1)$th approximation of $\mathbf{x}^\star$. This procedure is illustrated in Fig. 2. By choosing H_k to pass through $\mathbf{x}_k$ and then building E_{k+1}, we have constructed a *center-cut* ellipsoid. If instead the hyperplane H_k passes between $\mathbf{x}_k$ and $\mathbf{x}^\star$, then E_{k+1} is a *deep-cut* ellipsoid [17]. Intuitively, it is clear that E_{k+1} constructed by deep cuts contains $\mathbf{x}^\star$, but less than half of E_k. For the remainder of this section our focus is on the center-cut algorithm; we give only a few details regarding the deep-cuts variant.

We next describe how to construct the hyperplane $H_k = \{\mathbf{y} \in \mathbb{R}^m : \mathbf{g}_k^T(\mathbf{y} - \mathbf{x}_k) = 0\}$, i.e., how to choose a normal vector $\mathbf{g}_k$. The normal vector is chosen in such a way that it is easy to decide on which side of the hyperplane $\mathbf{x}^\star$ is located. First, we require the definition of a *subgradient* and *subdifferential*.

Definition 2 (*Subgradient, Subdifferential*) Let $f : C \to \mathbb{R}$ be a convex function whose domain is a convex set $C \subset \mathbb{R}^m$, and let $\mathbf{z} \in C$. Then the vector $\mathbf{g}$ is a subgradient of f at $\mathbf{z}$ if

$$f(\mathbf{z}) + \mathbf{g}^T(\mathbf{x} - \mathbf{z}) \le f(\mathbf{x}), \quad \forall \mathbf{x} \in C.$$

The set of all subgradients of f at $\mathbf{z}$ is denoted by $\partial f(\mathbf{z})$ and is called the subdifferential of f at $\mathbf{z}$. If f convex and differentiable at $\mathbf{z}$ then $\partial f(\mathbf{z}) = \{\nabla f(\mathbf{z})\}$.

There are two possibilities to consider for the center-cut algorithm. If the current approximation $\mathbf{x}_k$ is infeasible, i.e., there exists an index $j > 0$ such that $f_j(\mathbf{x}_k) > 0$, then we choose $\mathbf{g}_k \in \partial f_j(\mathbf{x}_k)$. Otherwise, if $\mathbf{x}_k$ is feasible, then we choose $\mathbf{g}_k \in \partial f_0(\mathbf{x}_k)$. In either case it is easy to verify that $\mathbf{x}^\star \in \mathcal{H}_k = \{\mathbf{y} \in \mathbb{R}^m : \mathbf{g}_k^T(\mathbf{y} - \mathbf{x}_k) \leq 0\}$. In order to define a hyperplane it is necessary to find a nonzero subgradient vector. To show that the subgradient exists and is nonzero, we start with the fact that since each f_i is convex on $\mathbb{R}^m$, it has a nonempty subdifferential at any point in $\mathbb{R}^m$ [2]. Now it is still possible that the subdifferential of f_i at some point in $\mathbb{R}^m$ contains only the zero vector. By the definition of the subgradient, for any iterate $\mathbf{x}_k$ we have that

$$\mathbf{g}_k^T(\mathbf{x} - \mathbf{x}_k) \leq f_i(\mathbf{x}) - f_i(\mathbf{x}_k)\ \forall \mathbf{x} \in \mathbb{R}^m, \quad \mathbf{g}_k \in \partial f_i(\mathbf{x}_k). \tag{5}$$

In the infeasible case, $f_i(\mathbf{x}_k) > 0$ for some $i \in \{1, \dots, n\}$, and it follows from (5) that a nonzero subgradient exists. In the feasible case, if $\partial f_0(\mathbf{x}_k) = \{\mathbf{0}\}$, then it follows from (5) that $\mathbf{x}_k$ is optimal. Therefore, if $\mathbf{x}_k$ is feasible but not optimal a nonzero subgradient must exist.

It remains to describe the update equations for E_{k+1}. Given $E_k = E(\mathbf{x}_k, \mathbf{D}_k) \in \mathbb{R}^m (m > 1)$ and the subgradient vector $\mathbf{g}_k$ defining the halfspace $\mathcal{H}_k$, the center-cut minimum volume ellipsoid $E_{k+1} = E(\mathbf{x}_{k+1}, \mathbf{D}_{k+1})$ that contains the region $E_k \cap \mathcal{H}_k$ is constructed according to

$$\begin{aligned}
\mathbf{x}_{k+1} &= \mathbf{x}_k - \frac{1}{m+1}\frac{\mathbf{D}_k\mathbf{g}_k}{\sqrt{\mathbf{g}_k^T\mathbf{D}_k\mathbf{g}_k}},\\
\mathbf{D}_{k+1} &= \frac{m^2}{m^2-1}\left(\mathbf{D}_k - \frac{2}{m+1}\frac{\mathbf{D}_k\mathbf{g}_k\,\mathbf{g}_k^T\mathbf{D}_k}{\mathbf{g}_k^T\mathbf{D}_k\mathbf{g}_k}\right).
\end{aligned} \tag{6}$$

Indeed, it can be verified that $\mathbf{D}_{k+1}$ is SPD and the volume of E_{k+1} is strictly less than the volume of E_k. For a derivation of these results we refer to [4] (see also [5]). In the one-dimensional case ($m = 1$) the update formulas are given by

$$x_{k+1} = x_k - \frac{1}{2}\operatorname{sgn}(g_k)\sqrt{D_k}, \quad D_{k+1} = \frac{D_k}{4}, \tag{7}$$

where $\operatorname{sgn}(\cdot)$ is the signum function and all quantities are scalar. In this case one can show that the ellipsoid method reduces to the bisection method.

The update formulas for an ellipsoid method constructed using deep cuts are very similar to those given in (6). Since we make use of deep cuts in our numerical tests, we briefly describe them below. Let $E(\mathbf{x}, \mathbf{D})$ be an ellipsoid in $\mathbb{R}^m$. It was shown in [18] that any hyperplane $H = \{\mathbf{y} \in \mathbb{R}^m : \mathbf{g}^T\mathbf{y} = \beta\}$ with $\beta = \mathbf{g}^T\mathbf{x} - \alpha\sqrt{\mathbf{g}^T\mathbf{D}\mathbf{g}}$ and $-1 \leq \alpha \leq 1$ has a nonempty intersection with $E(\mathbf{x}, \mathbf{D})$. Furthermore, for $-1/m \leq \alpha \leq 1$ it is possible to construct a minimum volume ellipsoid that contains $E(\mathbf{x}, \mathbf{D}) \cap \mathcal{H}$, where $\mathcal{H}$ is the halfspace $\mathcal{H} = \left\{\mathbf{y} \in \mathbb{R}^m : \mathbf{g}^T(\mathbf{y}-\mathbf{x}) \leq -\alpha\sqrt{\mathbf{g}^T\mathbf{D}\mathbf{g}}\right\}$. For $m > 1$ define the parameters:

$$\begin{aligned}
\tau &:= \frac{1+\alpha m}{m+1}, \quad \sigma := \frac{2(1+\alpha m)}{(m+1)(1+\alpha)},\\
\delta &:= \frac{m^2(1-\alpha^2)}{m^2-1}.
\end{aligned} \tag{8}$$

Then according to [5] the deep-cut ellipsoid $E_{k+1} = E(\mathbf{x}_{k+1}, \mathbf{D}_{k+1})$ with volume strictly less than $E_k = E(\mathbf{x}_k, \mathbf{D}_k)$ is given by the formulas:

$$\begin{aligned}
\mathbf{x}_{k+1} &= \mathbf{x}_k - \tau\frac{\mathbf{D}_k\mathbf{g}_k}{\sqrt{\mathbf{g}_k^T\mathbf{D}_k\mathbf{g}_k}},\\
\mathbf{D}_{k+1} &= \delta\left(\mathbf{D}_k - \sigma\frac{\mathbf{D}_k\mathbf{g}_k\,\mathbf{g}_k^T\mathbf{D}_k}{\mathbf{g}_k^T\mathbf{D}_k\,\mathbf{g}_k}\right).
\end{aligned} \tag{9}$$

In the one-dimensional case these formulas simplify to

$$\begin{aligned}
x_{k+1} &= x_k - \frac{(1+\alpha)}{2}\operatorname{sgn}(g_k)\sqrt{D_k},\\
D_{k+1} &= \frac{(1-\alpha)^2}{4}D_k.
\end{aligned} \tag{10}$$

The parameter α in the equations above determines the depth of the cut. For $-1/m \leq \alpha < 0$ we refer to E_{k+1} as a *shallow-cut* ellipsoid [5], and for $\alpha = 0$ we recover the formulas for the center-cut ellipsoid in (6) and (7). If $0 < \alpha \leq 1$, then E_{k+1} is a deep-cut ellipsoid. In our implementation we compute α_k (the depth of cut on the kth iteration) as follows. If $\mathbf{x}_k$ is feasible, then

$$\begin{aligned}
\alpha_k &= (f_0(\mathbf{x}_k) - u_k)\Big/\sqrt{\mathbf{g}_k^T\mathbf{D}_k\mathbf{g}_k},\\
u_k &= \min\{f_0(\mathbf{x}_i) : i \leq k,\ \mathbf{x}_i \in S\}.
\end{aligned} \tag{11}$$

Otherwise, if $f_j(\mathbf{x}_k) > 0$ for some index $j > 0$, then

$$\alpha_k = f_j(\mathbf{x}_k)\Big/\sqrt{\mathbf{g}_k^T\mathbf{D}_k\mathbf{g}_k}. \tag{12}$$

It was shown in [18] that computing α_k according to formulas (11) and (12) always yields a valid cut. For further details regarding deep cuts as well as examples of their use in deep-cut ellipsoid methods we refer to [16,18].

A high-level description of the center-cut ellipsoid method is given below in Algorithm 2. For further details regarding the numerical stability and the computer implementation of the ellipsoid method we refer to [5,19] and references therein.

Algorithm 2: Center-cut ellipsoid method

```
1. Let $E_0 = E(\mathbf{x}_0, \mathbf{D}_0)$ be an initial ellipsoid such that
   $\mathbf{x}^\star \in E_0$
2. Set $k \leftarrow 0$
   while $k < K$ do
3.     Depending on the feasibility of $\mathbf{x}_k$, find a
       subgradient vector $\mathbf{g}_k$ such that
       $\mathbf{x}^\star \in \mathcal{H}_k = \{\mathbf{y} \in \mathbb{R}^m : \mathbf{g}_k^T(\mathbf{y} - \mathbf{x}_k) \leq 0\}$
4.     If $\mathbf{x}_k$ is feasible, check the stopping criterion with
       tolerance $\varepsilon > 0$
5.     Construct a new ellipsoid $E_{k+1} = E(\mathbf{x}_{k+1}, \mathbf{D}_{k+1})$
       according to (6)–(7).
6.     Set $k \leftarrow k + 1$
   end
```

In this paper we make use of the following stopping criterion. We keep track of the current best upper and lower bounds for the optimal objective value, $l_k \leq f_0(\mathbf{x}^\star) \leq u_k$, and iterate until $u_k - l_k < \varepsilon$ for some tolerance $\varepsilon > 0$. The bounds u_k and l_k are defined as

$$u_k := \min\{f_0(\mathbf{x}_i) : i \leq k,\ \mathbf{x}_i \in S\},$$
$$l_k := \max\left\{f_0(\mathbf{x}_i) - \sqrt{\mathbf{g}_i^T \mathbf{D}_i\, \mathbf{g}_i} : i \leq k,\ \mathbf{x}_i \in S\right\}.$$

The lower bound l_k can be derived as follows. For any feasible iterate $\mathbf{x}_k$, it follows by the subgradient inequality that

$$\begin{aligned} f_0(\mathbf{x}^\star) &\geq f_0(\mathbf{x}_k) + \mathbf{g}_k^T(\mathbf{x}^\star - \mathbf{x}_k) \\ &\geq f_0(\mathbf{x}_k) + \inf_{\mathbf{z} \in E_k} \mathbf{g}_k^T(\mathbf{z} - \mathbf{x}_k). \end{aligned} \tag{13}$$

Since E_k is a compact subset of $\mathbb{R}^m$, the continuity of $\mathbf{g}_k^T(\mathbf{z} - \mathbf{x}_k)$ implies that the infimum is attained on the boundary of E_k. The minimizer can then be obtained through a straightforward application of Lagrange multipliers. At convergence we have that $0 \leq u_k - f_0(\mathbf{x}^\star) \leq u_k - l_k < \varepsilon$, and we return the feasible iterate $\mathbf{x}_{best}$ that satisfies $f_0(\mathbf{x}_{best}) = u_k$. We note that since the ellipsoid method is not a descent method it is necessary to keep track of the best feasible iterate found so far.

We conclude this section with a brief discussion regarding convergence of the ellipsoid algorithm. Convergence of Algorithm 2 for convex programming problems of the form in (4) was proved in [30] using an approach based on variational inequalities. It was shown in Theorem 2.3.2 of [18] that a center-cut and deep-cut version of the ellipsoid method are guaranteed to converge when applied to the convex program (4). Here, convergence is understood in the following sense,

$$\lim_{k \to \infty} u_k = f_0(\mathbf{x}^\star).$$

Furthermore, convergence was shown to be geometric with a rate that asymptotically approaches 1 as $m \to \infty$ [18]. In the next section we provide detailed per iteration complexity estimates for our implementation and outline the overhead costs of using iterant recombination in conjunction with the ellipsoid method.

4 Ellipsoid method for Markov acceleration

In this section we discuss how the ellipsoid algorithm can be applied as a solver for the iterant recombination problem. Specifically, we give the formulas for the subgradients of the objective function and the constraint functions, we present an equivalent formulation of (3) with the equality constraint removed, and we show how an initial ellipsoid E_0 can be constructed that is guaranteed to contain the exact solution.

We are interested in solving the following convex optimization problem:

$$\begin{aligned} &\text{minimize} && \|\mathbf{AXz}\|_1 \\ &\text{subject to} && \mathbf{Xz} \geq 0 \\ &&& \mathbf{1}^T\mathbf{z} = 1, \end{aligned} \tag{14}$$

where $\mathbf{X} \in \mathbb{R}^{n \times m}$ is the matrix of previous fine-level iterates and m is the window size. Due to the single equality constraint, there are only $m - 1$ degrees of freedom. Thus, we can obtain an equivalent inequality-form problem with $m - 1$ unknowns by removing one of the variables. Eliminating the variable z_1 and letting $\hat{\mathbf{z}} = (z_2, \ldots, z_m)^T$, the equality constraint implies that $\mathbf{z} = (1 - \mathbf{1}^T\hat{\mathbf{z}},\ \hat{\mathbf{z}})^T$. Using Matlab notation we define $\hat{\mathbf{X}} = -\mathbf{X}(:, 2{:}m) + \mathbf{x}_1\mathbf{1}^T$ and $\hat{\mathbf{A}} = -\mathbf{A}\hat{\mathbf{X}}$, where $\mathbf{x}_1$ is the first column of $\mathbf{X}$ and $\mathbf{a}_1 = \mathbf{A}\,\mathbf{x}_1$. The equivalent problem can now be stated as:

$$\begin{aligned} &\text{minimize} && \|\hat{\mathbf{A}}\,\hat{\mathbf{z}} + \mathbf{a}_1\|_1 \\ &\text{subject to} && \hat{\mathbf{X}}\hat{\mathbf{z}} - \mathbf{x}_1 \leq 0. \end{aligned} \tag{15}$$

This formulation is obtained by substituting $\mathbf{z} = (1 - \mathbf{1}^T\hat{\mathbf{z}},\ \hat{\mathbf{z}})^T$ into (14) and then simplifying. Note that once the solution to (15) has been computed (call it $\hat{\mathbf{z}}^\star$), the solution to the original problem (14) is given by $\mathbf{z}^\star = (1 - \mathbf{1}^T\hat{\mathbf{z}}^\star,\ \hat{\mathbf{z}}^\star)^T$.

It is obvious from (15) that the objective function is

$$f_0(\hat{\mathbf{z}}) = \|\hat{\mathbf{A}}\,\hat{\mathbf{z}} + \mathbf{a}_1\|_1$$

and the constraint functions are

$$f_i(\hat{\mathbf{z}}) = \hat{\mathbf{x}}_i^T\hat{\mathbf{z}} - (\mathbf{x}_1)_i, \quad i = 1, \ldots, n$$

where $\hat{\mathbf{x}}_i$ is the ith row of $\hat{\mathbf{X}}$ in column format. Since the constraint functions are convex and differentiable with respect to $\mathbf{z}$, the subdifferential $\partial f_i(\hat{\mathbf{z}}) = \{\nabla f_i(\hat{\mathbf{z}})\} = \{\hat{\mathbf{x}}_i\}$ for all $\hat{\mathbf{z}} \in \mathbb{R}^{m-1}$. The objective function is not differentiable, however it can be written as a composition of functions, $f_0(\hat{\mathbf{z}}) = h(\hat{\mathbf{A}}\,\hat{\mathbf{z}} + \mathbf{a}_1)$, where $h(\cdot) = \|\cdot\|_1$. Application of the chain rule [36] gives $\partial f_0(\hat{\mathbf{z}}) = \hat{\mathbf{A}}^T \partial h(\hat{\mathbf{A}}\,\hat{\mathbf{z}} + \mathbf{a}_1)$. Since the vector function $\mathbf{q} \in \mathbb{R}^n$ defined by

$$q_i(\mathbf{x}) = \begin{cases} 1 & \text{if } x_i \geq 0 \\ -1 & \text{if } x_i < 0 \end{cases} \tag{16}$$

is a subgradient for $h(\mathbf{x})$, it follows that $\hat{\mathbf{A}}^T \mathbf{q}(\hat{\mathbf{A}}\,\hat{\mathbf{z}} + \mathbf{a}_1)$ is a subgradient for $f_0(\hat{\mathbf{z}})$.

The main computational costs of Algorithm 2 are the subgradient vector construction (line 3), and the ellipsoid update (line 5). Construction of the subgradient vector consists of four steps: (1) Perform a feasibility check, (2) compute $\hat{\mathbf{A}}\,\hat{\mathbf{z}} + \mathbf{a}_1$, (3) build $\mathbf{q}$, (4) compute $\hat{\mathbf{A}}^T \mathbf{q}$. The feasibility check consists of evaluating $\hat{\mathbf{X}}\hat{\mathbf{z}} - \mathbf{x}_1$ and then searching for a positive entry, which requires $\mathcal{O}(mn)$ flops. The order in which the feasibility constraints are examined is discussed in [16], where the authors advocate a cyclical order since it yields slightly better efficiency. In this paper, however, we use a straightforward top-down sequential search $f_1, \ldots, f_n$, which for the test problems considered yields an efficient and robust method. Steps (2) and (4) each require $\mathcal{O}(mn)$ flops, and step (3) requires only $\mathcal{O}(n)$ flops. A slightly more detailed analysis reveals that construction of the subgradient vector requires $2mn$ flops when the current iterate is infeasible, and $(6m + 2)n$ flops when it is feasible. Here we have assumed that a sequential search of a length n array requires n flops. Referring to the equations in (6) the ellipsoid update requires approximately $5m^2$ flops. Given that $m \ll n$ this is negligible compared to the subgradient vector construction. Therefore, we can conclude that each iteration requires $\mathcal{O}(n)$ flops. We note that these estimates apply to both the center-cut and deep-cut algorithms.

We now proceed with our main technical contribution, namely, the construction of an initial ellipsoid $E_0 = E(\hat{\mathbf{z}}_0, \mathbf{D}_0)$ that is guaranteed to contain the exact solution. Additionally, we state necessary and sufficient conditions for its existence. We begin by deriving a formula for an initial ellipsoid $E_0 = E(\hat{\mathbf{z}}_0, \mathbf{D}_0)$ that is guaranteed to contain the exact solution $\mathbf{z}^\star$. Intuitively, we expect that most of the weight in the optimal linear combination $\mathbf{x}^\star = \mathbf{X}\mathbf{z}^\star$ will be associated with the most recent fine-level approximation $\mathbf{x}_k$, which is the rightmost column of $\mathbf{X}$. This assumption was confirmed by numerical tests. Therefore, we use $\hat{\mathbf{z}}_0 = (0, \ldots, 0, 1)^T$ as the center point for the initial ellipsoid. We now derive the matrix $\mathbf{D}_0$.

For any feasible point $\mathbf{z}$ of (14), the corresponding point $\hat{\mathbf{z}}$ is feasible for (15) and $\mathbf{AXz} = \hat{\mathbf{A}}\,\hat{\mathbf{z}} + \mathbf{a}_1$. Therefore, given some feasible point $\mathbf{z}$ it follows that

$$\hat{\mathbf{z}}^\star \in \{\mathbf{y} \in \mathbb{R}^{m-1} : \|\hat{\mathbf{A}}\,\mathbf{y} + \mathbf{a}_1\|_1 \leq \alpha\},$$
$$\alpha = \|\mathbf{AXz}\|_1 = \|\hat{\mathbf{A}}\,\hat{\mathbf{z}} + \mathbf{a}_1\|_1.$$

For example, $\mathbf{z} = \mathbf{e}_i$, the ith canonical basis vector in $\mathbb{R}^m$ is a feasible point for (15). In practice we choose α according to

$$\alpha = \min_{i=1,\ldots,m} \|\mathbf{AXe}_i\|_1,$$

which is equal to the minimum absolute column sum of $\mathbf{AX}$. Since $\hat{\mathbf{A}}\,\mathbf{y} + \mathbf{a}_1 = \hat{\mathbf{A}}(\mathbf{y} - \hat{\mathbf{z}}_0) + (\hat{\mathbf{A}}\,\hat{\mathbf{z}}_0 + \mathbf{a}_1)$ for any $\mathbf{y} \in \mathbb{R}^{m-1}$, it is clear that

$$\hat{\mathbf{z}}^\star \in \{\mathbf{y} \in \mathbb{R}^{m-1} : \|\hat{\mathbf{A}}(\mathbf{y} - \hat{\mathbf{z}}_0) + (\hat{\mathbf{A}}\,\hat{\mathbf{z}}_0 + \mathbf{a}_1)\|_1 \leq \alpha\}.$$

Applying the reverse triangle inequality, $|\|\mathbf{u}\|_1 - \|\mathbf{v}\|_1| \leq \|\mathbf{u} \pm \mathbf{v}\|_1$, we obtain

$$\hat{\mathbf{z}}^\star \in \{\mathbf{y} \in \mathbb{R}^{m-1} : \|\hat{\mathbf{A}}(\mathbf{y} - \hat{\mathbf{z}}_0)\|_1 \leq r\},$$
$$r = \alpha + \|\hat{\mathbf{A}}\,\hat{\mathbf{z}}_0 + \mathbf{a}_1\|_1.$$

Using the vector norm inequality $\|\cdot\|_2 \leq \|\cdot\|_1$, we arrive at the desired result that

$$\|\hat{\mathbf{A}}(\hat{\mathbf{z}}^\star - \hat{\mathbf{z}}_0)\|_2 \leq r \;\Leftrightarrow\; (\hat{\mathbf{z}}^\star - \hat{\mathbf{z}}_0)^T \mathbf{D}_0^{-1} (\hat{\mathbf{z}}^\star - \hat{\mathbf{z}}_0) \leq 1, \tag{17}$$

where $\mathbf{D}_0 = r^2(\hat{\mathbf{A}}^T \hat{\mathbf{A}})^{-1}$. Therefore, it follows by (17) that the optimal solution $\hat{\mathbf{z}}^\star$ belongs to the ellipsoid $E_0 = E(\hat{\mathbf{z}}_0, \mathbf{D}_0)$.

The ellipsoid method can also be used to solve the two-norm minimization problem, which is equivalent to (15) with the objective function given by

$$f_0(\hat{\mathbf{z}}) = \langle \hat{\mathbf{A}}\,\hat{\mathbf{z}} + \mathbf{a}_1, \hat{\mathbf{A}}\,\hat{\mathbf{z}} + \mathbf{a}_1 \rangle.$$

Since in this case the objective function is convex and differentiable, its subdifferential is equal to its gradient $\nabla f_0(\hat{\mathbf{z}}) = 2\hat{\mathbf{A}}^T(\hat{\mathbf{A}}\,\hat{\mathbf{z}} + \mathbf{a}_1)$. An initial ellipsoid $E_0 = E(\hat{\mathbf{z}}_0, \mathbf{D}_0)$ that contains the optimal solution can be similarly defined with

$$r = \|\hat{\mathbf{A}}\,\hat{\mathbf{z}}_0 + \mathbf{a}_1\|_2 + \min_{i=1,\ldots,m} \|\mathbf{AXe}_i\|_2.$$

Given the form of the subgradient it is clear that the cost per iteration of the ellipsoid method for two-norm minimization is approximately equal to the cost per iteration for one-norm minimization.

It is necessary to consider whether $\mathbf{D}_0$ exists, i.e., if $\hat{\mathbf{A}}^T \hat{\mathbf{A}}$ is invertible. It is clear that $\hat{\mathbf{A}}^T \hat{\mathbf{A}}$ is an $(m-1) \times (m-1)$ symmetric positive-semidefinite matrix, and provided that $\hat{\mathbf{A}}$ is of full rank, $\mathbf{D}_0$ exists and is symmetric positive-definite. The following proposition establishes necessary and sufficient conditions for the existence of $\mathbf{D}_0$.

Proposition 1 *The matrix $\hat{\mathbf{A}}$ has full rank if and only if $\hat{\mathbf{X}}$ has full rank and the exact solution of* (2) *is not in the range of* $\hat{\mathbf{X}}$.

Proof We prove the contrapositive statement: $\hat{\mathbf{A}}$ is rank deficient if and only if $\hat{\mathbf{X}}$ is rank deficient or $\mathbf{x} \in \text{range}(\hat{\mathbf{X}})$. Recall that $\hat{\mathbf{A}} = -\mathbf{A}\hat{\mathbf{X}}$. If $\hat{\mathbf{X}}$ is rank deficient, then $\hat{\mathbf{X}}$ has a nontrivial nullspace, and hence $\hat{\mathbf{A}}$ must also have a nontrivial nullspace. If $\mathbf{x} \in \text{range}(\hat{\mathbf{X}})$, then there exists a nonzero vector $\mathbf{y}$ such that $\mathbf{x} = \hat{\mathbf{X}}\mathbf{y}$. Since $\mathbf{A}\mathbf{x} = \mathbf{0}$, it follows that $\mathbf{y} \in \text{null}(\hat{\mathbf{A}})$. In either case it is clear that $\hat{\mathbf{A}}$ must be rank deficient. Now suppose that $\hat{\mathbf{A}}$ is rank deficient. Then there exists a nonzero vector $\mathbf{y}$ such that $\hat{\mathbf{A}}\mathbf{y} = \mathbf{0}$, which implies that $\hat{\mathbf{X}}\mathbf{y} \in \text{null}(\mathbf{A})$. Since $\text{null}(\mathbf{A}) = \text{span}(\mathbf{x})$, this implies that either $\hat{\mathbf{X}}\mathbf{y} = \mathbf{x}$ or $\hat{\mathbf{X}}\mathbf{y} = \mathbf{0}$. This is equivalent to the statement that either $\mathbf{x} \in \text{range}(\hat{\mathbf{X}})$ or $\hat{\mathbf{X}}$ is rank deficient. □

We can now use Proposition 1 to identify two possible but exclusive cases under which $\hat{\mathbf{A}}$ is rank deficient and our proposed initial ellipsoid does not exist:

Proposition 2 *Let $\mathbf{X}_{k-1}$ be the matrix of fine-level iterates after $k-1$ multigrid cycles with iterant recombination, and suppose that $\mathbf{x} \notin \text{range}(\mathbf{X}_{k-1})$ and $\mathbf{X}_{k-1}$ has full rank. Moreover, let $\mathbf{X}_k$ be the corresponding matrix after the kth multigrid cycle with iterant recombination. If $\hat{\mathbf{A}}_k$ is rank deficient, then either $\mathbf{x} \in \text{range}(\hat{\mathbf{X}}_k)$ or $\hat{\mathbf{X}}_k$ is rank deficient, but not both.*

Proof Since $\hat{\mathbf{A}}_k$ is rank deficient, it follows by Proposition 1 that either $\hat{\mathbf{X}}_k$ is rank deficient or $\mathbf{x} \in \text{range}(\hat{\mathbf{X}}_k)$. Without restricting generality suppose that $\hat{\mathbf{X}}_k$ is rank deficient. This implies that $\mathbf{X}_k$ is also rank deficient, and by the full rank assumption on $\mathbf{X}_{k-1}$ it follows that $\mathbf{x}_k \in \text{range}(\mathbf{X}_{k-1})$. Thus, $\text{range}(\mathbf{X}_k) \subset \text{range}(\mathbf{X}_{k-1})$, and hence $\mathbf{x} \notin \text{range}(\mathbf{X}_k)$. The desired result now follows by the fact that $\text{range}(\hat{\mathbf{X}}_k) \subset \text{range}(\mathbf{X}_k)$. □

Now suppose we run the iterant recombination algorithm until at some point $\hat{\mathbf{A}}$ is rank deficient and our proposed initial ellipsoid does not exist (note that in practice this may never occur, see below). Then Proposition 2 implies that only one of the following two cases is possible, and we give a strategy for handling each case in our iterant recombination algorithm:

Case 1: If $\hat{\mathbf{X}}$ is rank deficient, then $\mathbf{X}$ is also rank deficient. Here, the most obvious approach is to drop all the columns of $\mathbf{X}$ except for $\mathbf{x}_k$, and to skip the kth iterant recombination step. Doing so implies that $\mathbf{X} = [\mathbf{x}_k]$ has full rank (since $\mathbf{x}_k \neq \mathbf{0}$) and $\mathbf{x} \notin \text{range}(\mathbf{X})$. Therefore, the results of Proposition 2 may be applied to future iterant recombination iterations.

Case 2: If $\hat{\mathbf{X}}$ has full rank, then it must be true that $\mathbf{x} \in \text{range}(\hat{\mathbf{X}})$. Thus, one can solve the following full rank overdetermined system of equations for $\mathbf{y}$

$$(\mathbf{A}\hat{\mathbf{X}})\mathbf{y} = \mathbf{0}$$

and then compute the exact solution according to $\mathbf{x} = \hat{\mathbf{X}}\mathbf{y}/\|\hat{\mathbf{X}}\mathbf{y}\|_1$.

In practice, rank deficiency of $\hat{\mathbf{A}}$ is typically not an issue. In fact, rank deficiency of $\hat{\mathbf{A}}$ was not observed for any of the iterant recombination numerical tests. This is most likely due to the nonlinear nature of the underlying multilevel algorithm, i.e., the range of the multilevel operator changes with each iteration.

As one final note we mention some overhead costs of using iterant recombination in conjunction with one-norm or two-norm minimization. These costs are in addition to those described above, however they represent only a small part of the overall ellipsoid method cost per multigrid cycle. After each multigrid cycle it is necessary to update $\mathbf{AX}$ with the new approximation, build $\hat{\mathbf{A}}$ and build $\hat{\mathbf{A}}^T\hat{\mathbf{A}}$. To leading order this requires $2\,\text{nnz}(\mathbf{A})+m^2n$ flops, where $\text{nnz}(\mathbf{A})$ is the number of nonzero entries in the sparse matrix $\mathbf{A}$. To initialize the ellipsoid method it is necessary to compute $\mathbf{D}_0 = r^2(\hat{\mathbf{A}}^T\hat{\mathbf{A}})^{-1}$. For window sizes $m \leq 4$ there exist analytic formulas for the inverse that require 32 flops when $m = 4$ and 7 flops when $m = 3$ (in general for $m > 4$ computing the inverse requires $(8/3)m^3$ flops). The computation of r requires $(m+1)n$ flops.

5 Numerical results

In this section we perform numerical tests of the iterant recombination acceleration algorithm for a variety of test problems that appear in [12,14]. Two of these test problems are standard Markov chain tests from [39] and [23]. Each of the test problems is slowly mixing (see Sect. 2.1) and has a complex spectrum. The code was implemented and executed in Matlab 7.5.0 on a 2.50 GHz Intel Core 2 Duo CPU with 4 GB of RAM. Tests are conducted for window sizes $m = 1, 2, 3, 4$. We consider both the one-norm and two-norm minimization problems, where the former is solved by the ellipsoid method, and the latter is solved by the ellipsoid method and also by Matlab's built-in quadratic programming solver quadprog, as in [14]. In particular, the *medium-scale* quadprog algorithm which employs an active set method is used, since only a few of the inequality constraints may be relevant. Also, instead of using quadprog to solve the two-norm minimization problem with window size 2 we use the efficient algorithm employed in [14,15]. We note that in some cases the solution generated by quadprog may have negative entries even though positivity constraints were enforced. If this robustness problem occurs, then at most ten additional relaxations are performed (see also Theorem 5.1 in [14]). If after ten relaxations the improved approximation still does not have strictly positive entries, it is discarded, and the most recent fine-level iterate is used. We note that the time to perform extra relaxations is added to the overall solve time. Entries in the tables for which extra relaxations were necessary are indicated by a superscript asterisk.

As the primary standalone solver to be accelerated we consider the unsmoothed multilevel aggregation algorithm (AGG) of [23] with neighborhood-based aggregation [14]. We also compare AGG with the more sophisticated MCAMG algorithm from [12]. Weighted Jacobi relaxation is used on all levels, except on the coarsest level where a direct solve is performed via the GTH algorithm [20]. For the MCAMG algorithm we use V-cycles, and for the AGG algorithm we experiment with F-cycles and W-cycles, since this gave the most favorable results. We note that for both AGG and MCAMG the transfer operators are frozen after 10 multigrid cycles, however, the coarse-level operators are always rebuilt on each level. Furthermore, in order to obtain competitive operator complexities for MCAMG we use the first pass only of the classical Ruge-Stüben AMG coarsening algorithm (see [12]). As our stopping criterion we use

$$\text{stop if} \quad k > maxit \quad \text{or} \quad \|\mathbf{A}\,\mathbf{x}_k\|_1 < 10^{-8}\|\mathbf{A}\,\mathbf{x}_0\|_1,$$

where k is the iteration count and $maxit$ is the maximum number of iterations the algorithm will be allowed to perform. The initial guess $\mathbf{x}_0$ is uniformly randomly generated; it has strictly positive entries and unit one-norm. We note that all iterates have strictly positive entries and are normalized to have unit one-norm. The parameters for the AGG and MCAMG algorithms are given in Table 1.

The ellipsoid method with deep cuts is given by Algorithm 2 with E_{k+1} updated according to (8)–(12). The stopping criterion parameters are $maxit = 300$ with convergence tolerances $\varepsilon_1 = 10^{-8}$ for one-norm minimization and $\varepsilon_2 = 10^{-21}$ for two-norm minimization. These values were determined experimentally to give good results for a wide range of test problems.

Table 1 MCAMG and AGG parameters

Parameter	Value
Number of pre-relaxations ν_1	1
Number of post-relaxations ν_2	1
Strength of connection parameter θ	0.25
Lumping parameter η (MCAMG only)	0.01
Weighted Jacobi relaxation parameter ω	0.7
Max number of points on coarsest level n_{coarse}	12
Max number of iterations $maxit$	700

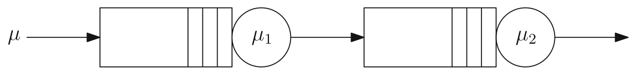

Fig. 3 Tandem queueing network

In the tables below we report the total number of iterations required by the iterant recombination accelerated MCAMG and AGG algorithms to converge for window size m. For tests without iterant recombination acceleration we report the number of levels, $lvls$, the number of iterations, it, and the operator complexity on the last cycle, C_{op}. The operator complexity is defined as the sum of the number of nonzero elements in all operators, $\mathbf{A}$, on all levels divided by the number of nonzero elements in the fine-level operator. This number gives a good indication of the amount of work required for a cycle and, for a scalable (or optimal) method, it should be bounded by a constant not too much larger than one as the problem size increases. In what follows we refer to the iterant recombination acceleration with two-norm minimization as “two-norm acceleration” and in the one-norm case we say “one-norm acceleration”.

5.1 Tandem queue

The first test problem we consider is the tandem queueing network from [39], where two finite queues with single servers are placed in tandem. Customers arrive according to a Poisson distribution with rate μ, and the service time distribution at the two single-server stations is Poisson with rates μ_1 and μ_2. This is illustrated in Fig. 3. The states of the system can be represented by tuples (n_1, n_2), where n_i is the number of customers waiting in the ith queue. We choose $(\mu,\ \mu_1,\ \mu_2) = (10, 11, 10)$ for the weights, which leads to a case of slow mixing.

The results of applying the acceleration schemes to AGG F-cycles and W-cycles are given in Table 2. We note that MCAMG was not considered for this test problem since it produces scalable results without acceleration. From the table it is clear that W-cycles outperform F-cycles, and that iterant recombination acceleration is able to significantly reduce the iteration counts and make the method more scalable. For $n = 262{,}144$ there is an 83% reduction in the iteration count for F-cycles and a 74% reduction for W-cycles. We also observe a roughly equivalent decrease in iterations between the one-norm and two-norm acceleration routines for each window size, where it is evident that window sizes 2 and 3 give the most significant reduction. We note that in the window size 4 column for quadprog with W-cycles the iteration count is significantly higher for $n = 262{,}144$ than for the other methods. This is due to the fact that quadprog failed to converge a large number of times for this problem. In fact over all the numerical tests the ellipsoid method demonstrated robustness in terms

Table 2 Tandem queue

n	$lvls$	C_{op}	it	Ellipsoid (one-norm)			Ellipsoid (two-norm)			Quadprog (two-norm)		
				Window size			Window size			Window size		
				2	3	4	2	3	4	2	3	4
AGG F-cycles												
1,024	4	1.45	136	50	46	42	52	56	44	52	56	44
4,096	4	1.47	173	57	50	51	64	60	59	63	61	59
16,384	5	1.47	328	83	72	66	77	68	71	77	67	68
65,536	6	1.47	395	100	79	92	106	77	89	106	80	85
262,144	6	1.46	629	135	112	146	131	116	132	130	116	129
AGG W-cycles												
1,024	4	1.47	121	47	44	42	49	53	43	49	53	43
4,096	4	1.50	143	55	46	45	54	53	52	54	53	52
16,384	5	1.51	210	66	61	55	66	56	52	67	57	54
65,536	6	1.50	231	71	61	65	66	63	64	66	63	63
262,144	6	1.50	315	94	81	86	86	80	85	86	77	117

Iteration counts for various window sizes for one-norm and two-norm minimization strategies applied to the AGG algorithm. The number of levels $lvls$, the operator complexities C_{op}, and the iteration counts it are given for the unaccelerated AGG algorithm

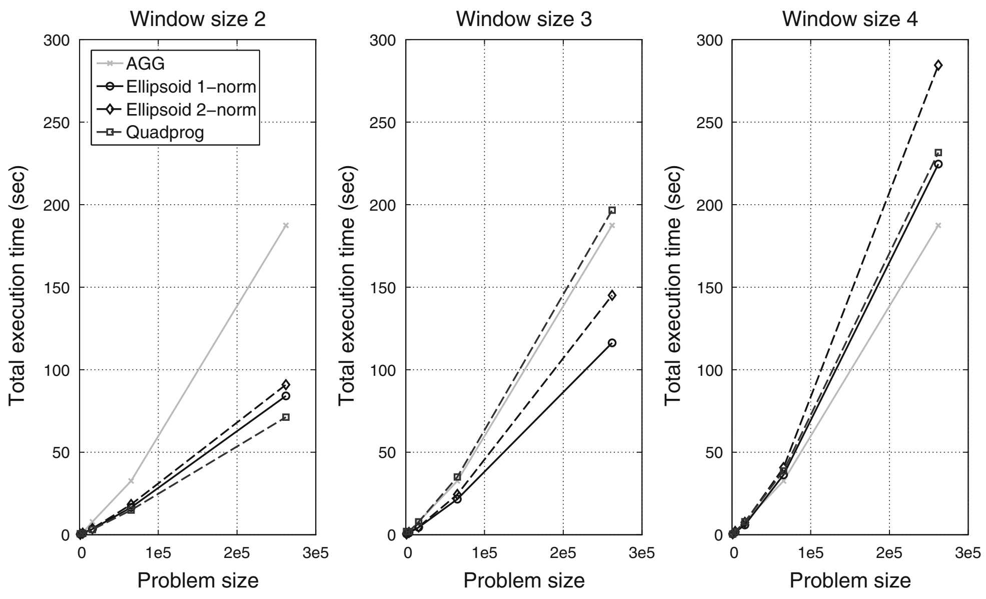

Fig. 4 Total execution time for accelerated AGG F-cycles applied to the tandem queue problem. The *solid gray line* with cross '×' markings is for unaccelerated AGG F-cycles

of iteration counts that is comparable or superior to quadprog.

Figures 4 and 5 show the total execution times for accelerated AGG F-cycles and W-cycles as well as for unaccelerated AGG cycles. We note that total execution time refers to the multigrid solve time plus the iterant recombination time. The figures clearly demonstrate that AGG with window size 2 acceleration gives the best overall solve times, while window size 4 acceleration results in slower overall performance. The best results are achieved by the efficient algorithm for two-norm minimization with window size 2 from [14,15], which is very hard to compete with. In fact, for $n = 262,144$ we obtain a 62% iterant recombination speedup for F-cycles and a 48% speedup for W-cycles. The figure also shows that the ellipsoid method for one-norm minimization tends to be a bit faster than the ellipsoid method for two-norm minimization. An examination of the data shows that in general the ellipsoid method for two-norm minimization requires more iterations per multigrid cycle to converge, which explains its performance given

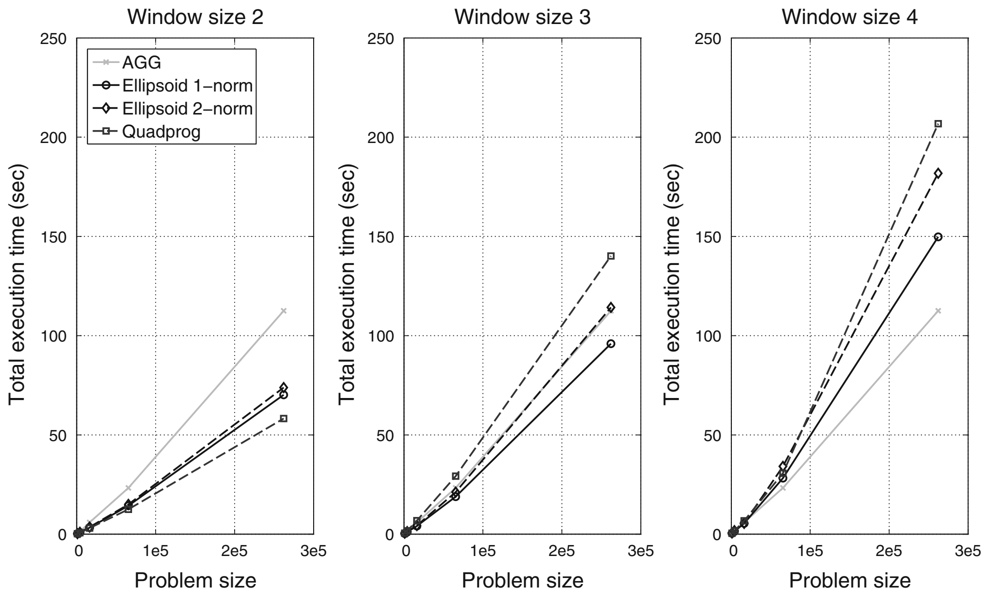

Fig. 5 Total execution time for accelerated AGG W-cycles applied to the tandem queue problem. The *solid gray line* with cross '×' markings is for unaccelerated AGG W-cycles

that the per iteration cost of the two methods is roughly equivalent.

5.2 Directed random planar graph (DRPG)

In this test problem we consider a random walk on an unstructured, directed planar graph. To construct the directed planar graph $\mathcal{D}$ we begin by randomly distributing n points in the unit square $(0, 1) \times (0, 1)$. These points are then connected via Delaunay triangulation, which yields an undirected planar graph $\mathcal{G}$. To obtain a directed graph $\mathcal{D}$, we randomly select a set of edges from $\mathcal{G}$ and make them uni-directional. This is done in such a way that irreducibility is preserved (see [14] for further details regarding the construction of $\mathcal{D}$). A Markov chain is then obtained by performing a random walk on $\mathcal{D}$, where the probability of transitioning from node i to node j is given by the reciprocal of the number of outward arcs from node i. It is clear by the construction of $\mathcal{D}$ that the resulting Markov chain has a nonsymmetric sparsity structure. Furthermore, numerical computations of the transition matrices' spectra confirms that this is a slowly mixing problem.

The results of applying the acceleration schemes to MCAMG V-cycles and AGG W-cycles are given in Table 3. We note that although MCAMG with two-pass coarsening performs quite well for this problem in terms of iteration counts, its operator complexity is high (see [12,14]). For this reason, we consider one-pass coarsening here, which reduces the operator complexity (and thus memory usage) but increases the iteration counts. The results in Table 3 show that by applying iterant recombination acceleration with one-pass coarsening we can improve the iteration counts with good operator complexity. We observe a significant reduction in iteration counts with comparable performance between one-norm and two-norm acceleration. For $n = 262,144$ there is a 63% reduction for MCAMG V-cycles and a 90% reduction for AGG W-cycles.

Figures 6 and 7 show the execution time results for MCAMG V-cycles and AGG W-cycles, respectively. It is evident that the fastest solve times are again given by window size 2 acceleration. For MCAMG we observe similar execution times for the ellipsoid method one-norm and two-norm minimization approaches with a 36% speedup for $n = 262,144$ and window size 2. In the case of AGG W-cycles ellipsoid method two-norm minimization slightly outperforms the one-norm minimization. For $n = 262,144$ we obtain an impressive 56% speedup with window size 2 acceleration. It is also interesting to note that in tandem queue problem window size 4 acceleration lead to a much slower solver, whereas here we obtained some improvement. However, the numerical results still imply that window size 4 acceleration is unnecessary in the context of AGG F-cycles and W-cycles.

5.3 Stochastic Petri net

The final test problem we consider was previously considered in [23], and is derived from a stochastic Petri net (SPN). Petri nets are a formalism for the description of concurrency and synchronization in distributed systems. They consist of: *places*, which model conditions or objects; *tokens*, which represent the specific value of the condition or object; *tran-*

Table 3 Directed random planar graph

n	$lvls$	C_{op}	it	Ellipsoid (one-norm)			Ellipsoid (two-norm)			Quadprog (two-norm)		
				Window size			Window size			Window size		
				2	3	4	2	3	4	2	3	4
MCAMG V-cycles												
1,024	5	1.56	37	19	16	16	23	18	17	23	18	17
4,096	6	1.56	37	19	18	17	21	18	17	21	18	17
16,384	7	1.59	46	23	22	22	25	21	20	25	21	20
65,536	8	1.59	62	26	24	24	29	25	23	29	25	23
262,144	9	1.59	65	32	27	24	34	28	26	34	28	26
AGG W-cycles												
1,024	4	1.32	113	41	58	64	49	38	37	35	45	50
4,096	4	1.34	166	62	28	37	42	37	39	49	42	57
16,384	5	1.36	231	62	71	38	57	42	39	58	37	39
65,536	6	1.37	316	39	45	47	62	54	45	61	69	38
262,144	6	1.37	373	36	42	51	71	58	47	39	46	48

Iteration counts for various window sizes for one-norm and two-norm minimization strategies applied to the AGG and MCAMG algorithms. The number of levels $lvls$, the operator complexities C_{op}, and the iteration counts it are given for the unaccelerated AGG and MCAMG algorithms

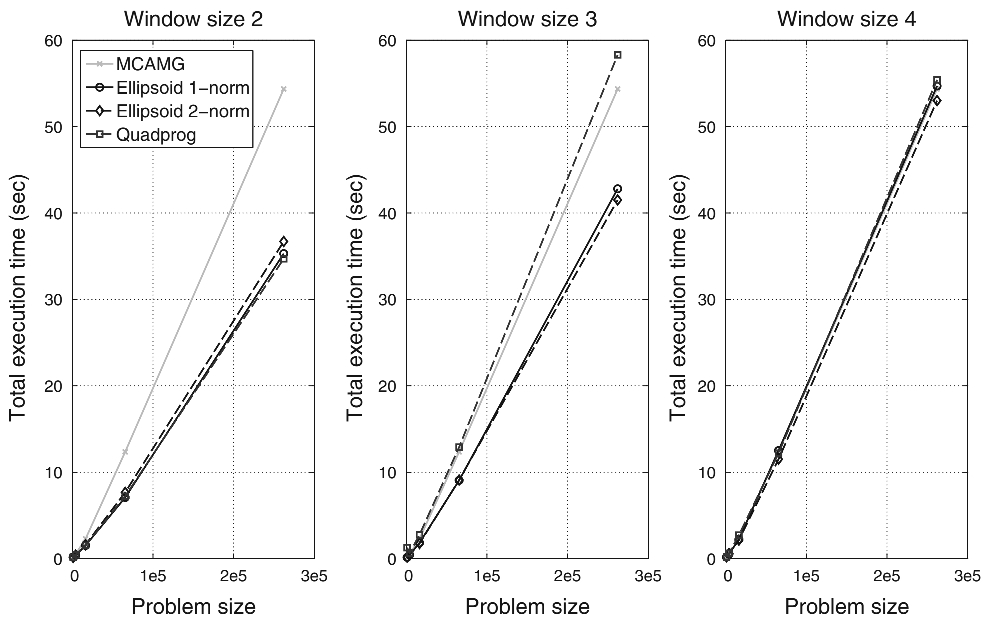

Fig. 6 Total execution time for accelerated MCAMG V-cycles applied to the directed random planar graph problem. The *solid gray line* with cross '×' markings is for unaccelerated MCAMG V-cycles

sitions, which model activities that change the value of conditions or objects; and *arcs*, which specify interconnection between places and transitions. A stochastic Petri net is a standard Petri net together with a tuple $\boldsymbol{\lambda} = (r_1, \ldots, r_n)$ of exponentially distributed transition firing rates. We know from [34] that a finite place, finite transition, marked stochastic Petri net is isomorphic to a one-dimensional discrete-space Markov process. For an in-depth discussion of Petri Nets we refer to [1,34].

The results of applying the acceleration schemes to AGG F-cycles and W-cycles are given in Table 4. We note that MCAMG was not considered for this test problem since it produces scalable results without acceleration. In this case we observe scalable performance with AGG W-cycles. As before there is little difference between one-norm and two-norm acceleration except in the case of quadprog where extra relaxations were necessary to obtain a strictly positive solution. The asterisks in the table show that, for this test problem,

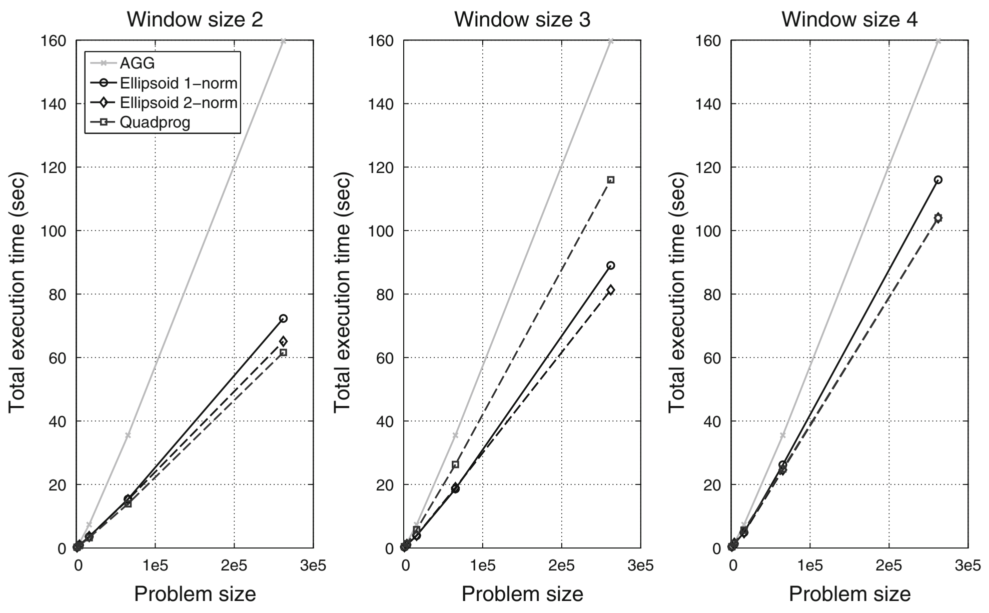

Fig. 7 Total execution time for accelerated AGG W-cycles applied to the directed random planar graph problem. The *solid gray line* with cross '×' markings is for unaccelerated AGG W-cycles

Table 4 Stochastic Petri net

n	$lvls$	C_{op}	it	Ellipsoid (one-norm)			Ellipsoid (two-norm)			Quadprog (two-norm)		
				Window size			Window size			Window size		
				2	3	4	2	3	4	2	3	4
MCAMG V-cycles												
2,470	5	1.75	78	33	33	35	41	49	35	33	47*	41*
10,416	5	1.67	87	36	30	30	37	33	51	31	34*	34*
23,821	5	1.56	75	45	46	31	38	41	38	46	28*	31*
45,526	6	1.51	86	39	32	35	31	34	34	45	46*	28*
121,836	6	1.46	106	30	31	36	36	27	31	33	55*	52*
AGG W-cycles												
2,470	5	1.83	65	29	30	30	34	36	29	33	37*	33*
10,416	6	1.88	66	30	26	36	35	31	34	30	30*	32*
23,821	6	1.77	65	35	39	29	34	34	35	34	25*	28*
45,526	6	1.67	66	33	34	33	30	30	32	35	39*	27*
121,836	6	1.58	66	29	28	29	32	26	29	31	38*	29*

Iteration counts for various window sizes for one-norm and two-norm minimization strategies applied to the AGG algorithm. The number of levels $lvls$, the operator complexities C_{op}, and the iteration counts it are given for the unaccelerated AGG algorithm. The superscript asterisks (*) indicate that extra relaxations were necessary to obtain a strictly positive solution

quadprog is less robust than the ellipsoid methods in terms of maintaining the sign constraints. For $n = 121{,}836$ we obtain a 75% reduction in iterations for F-cycles and a 61% reduction for W-cycles.

Figure 8 shows the execution times for AGG W-cycles. For $n = 121{,}836$ the fastest solve time is given by window size 2 acceleration with a modest but still relevant 16% speedup over the unaccelerated cycles. We observe that in this case the ellipsoid method with one-norm minimization outperforms the two-norm method. We also observe that the quadprog execution times are quite high for window sizes 3 and 4, which is due to the extra relaxations.

6 Concluding remarks

In this paper we proposed a one-norm minimization method for the constrained iterant recombination acceleration app-

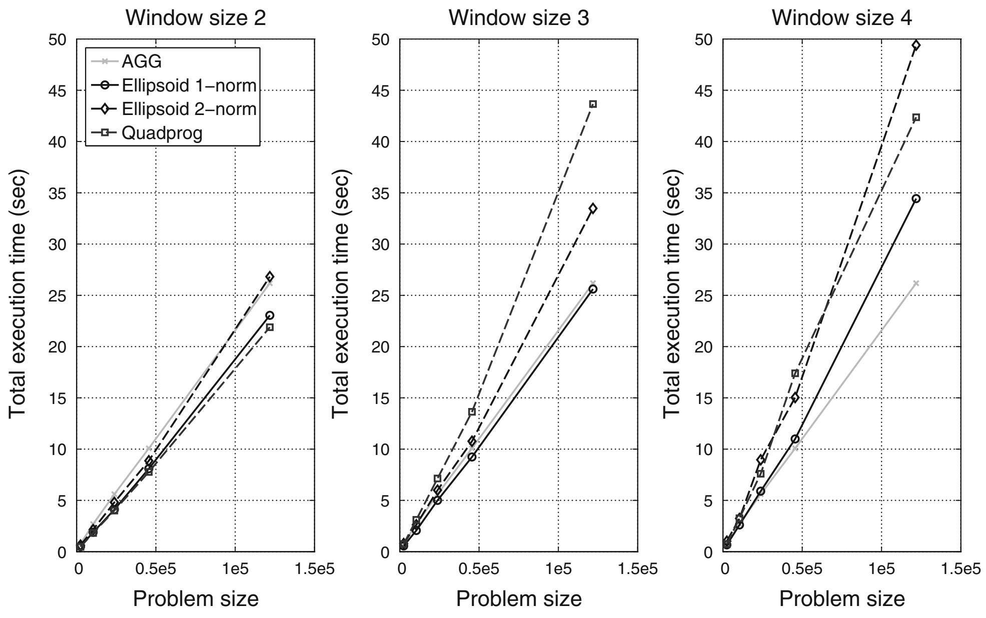

Fig. 8 Total execution time for accelerated AGG W-cycles applied to the stochastic Petri net problem. The *solid gray line* with cross '×' markings is for unaccelerated AGG W-cycles

roach for Markov chains developed in [14]. We formulated a nonlinear convex optimization problem and proposed a solution via the ellipsoid algorithm. We showed how an initial ellipsoid can be constructed that is guaranteed to contain the exact solution, and how rank deficiencies can be addressed. We also showed how the ellipsoid method could be used to minimize the constrained two-norm problem.

The numerical tests indicated that in terms of the reduction of iterations, there is little difference between the one-norm or two-norm variants. Furthermore, as was demonstrated in [14], it was confirmed that window sizes of 2 or 3 are sufficient to reduce the iteration count, and that the number of iterations is not further significantly reduced for larger window sizes. In fact, for window sizes larger than 3, the added overhead of solving a larger minimization problem lead to increasing execution times. The execution time data also showed that for window size 2, the one-norm and two-norm acceleration procedures were comparable, while for window size 3, the one-norm acceleration was typically faster. Our numerical tests illustrated that iterant recombination can make the simple aggregation method from [23] competitive with the more advanced methods from, for example, [11,12]. It was also shown how iterant recombination acceleration could provide significant speedup when MCAMG does not scale well (i.e., iterations increase with the problem size). Overall, the tests indicated that one-norm acceleration with the ellipsoid method is competitive with two-norm acceleration in terms of running time. It is interesting that we see a similar reduction in multigrid iterations for the one-norm and two-norm minimization. Also, the ellipsoid method is more robust than quadprog in terms of maintaining the sign constraints.

There is another approach to iterant recombination acceleration with one-norm minimization that we did not consider. Instead of solving a nonlinear convex optimization problem, we could have instead formulated the equivalent linear programming problem from (14) and used the simplex method to obtain the solution. It is possible that this approach could result in a faster accelerated solver, and we will consider it in the future.

Acknowledgments The authors would like to thank Steve Vavasis for his insightful comments and suggestions.

References

1. Bause, F., Kritzinger, P.: Stochastic Petri Nets. Springer, Germany (1996)
2. Bazaraa, M.S., Sherali, H.D., Shetty, C.M.: Nonlinear Programming: Theory and Algorithms, 3rd edn. Wiley, New Jersey (2006)
3. Berman, A., Plemmons, R.J.: Nonnegative Matrices in the Mathematical Sciences. SIAM, Philadelphia, PA (1987)
4. Bertsimas, D., Tsitisklis, J.N.: Introduction to Linear Optimization. Athena Scientific, Belmont, MA (1997)
5. Bland, R.G., Goldfarb, D., Todd, M.J.: The ellipsoid method: a survey. Oper. Res. **29**(6), 1039–1091 (1981)
6. Brandt, A., Mikulinsky, V.: On recombining iterants in multigrid algorithms and problems with small islands. SIAM J. Sci. Comput. **16**, 20–28 (1995)
7. Briggs, W.L., Henson, V.E., McCormick, S.F.: A Multigrid Tutorial, 2nd edn. SIAM, Philadelphia, PA (2000)
8. Buchholz, P.: Multilevel solutions for structured Markov chains. SIAM J. Matrix Anal. Appl. **22**(2), 342–357 (2000)

9. Cao, W.L., Stewart, W.J.: Iterative aggregation/disaggregation techniques for nearly uncoupled Markov chains. JACM **32**(3), 702–719 (1985)
10. Chatelin, F., Miranker, W.L.: Acceleration by aggregation of successive approximation methods. Linear Algebra Appl. **43**, 17–47 (1982)
11. De Sterck, H., Manteuffel, T., McCormick, S.F., Miller, K., Pearson, J., Ruge, J., Sanders, G.: Smoothed aggregation multigrid for Markov chains. SIAM J. Sci. Comput. **32**, 40–61 (2010)
12. De Sterck, H., Manteuffel, T., McCormick, S.F., Miller, K., Ruge, J., Sanders, G.: Algebraic multigrid for Markov chains. SIAM J. Sci. Comput. **32**, 544–562 (2010)
13. De Sterck, H., Manteuffel, T., McCormick, S.F., Nguyen, Q., Ruge, J.: Multilevel adaptive aggregation for Markov chains, with application to web ranking. SIAM J. Sci. Comput. **30**, 2235–2262 (2008)
14. De Sterck, H., Manteuffel, T., Miller, K., Sanders, G.: Top-level acceleration of adaptive algebraic multilevel methods for steady-state solution to Markov chains. Adv. Comput. Math. **35**, 375–403 (2010)
15. De Sterck, H., Miller, K., Sanders, G., Winlaw, M.: Recursively accelerated multilevel aggregation for Markov chains. SIAM J. Sci. Comput. **32**(3), 1652–1671 (2010)
16. Dziuban, S.T., Ecker, J.G., Kupferschmid, M.: Using deep cuts in an ellipsoid algorithm for nonlinear programming. Math. Program. Stud. **25**, 93–107 (1985)
17. Ecker, J.G., Kupferschmid, M.: An ellipsoid algorithm for nonlinear programming. Math. Program. **27**, 83–106 (1983)
18. Frenk, J.B.G., Gromicho, J., Zhang, S.: A deep cut ellipsoid algorithm for convex programming: theory and applications. Math. Program. **63**, 83–108 (1994)
19. Goldfarb, D., Todd, M.J.: Modifications and implementation of the ellipsoid algorithm for linear programming. Math. Program. **23**, 1–19 (1982)
20. Grassmann, W., Taksar, M., Heyman, D.: Regenerative analysis and steady-state distributions for Markov chains. Oper. Res. **33**(5), 1107–1116 (1985)
21. Haviv, M.: Aggregation/disaggregation methods for computing the stationary distribution of Markov chains. SIAM J. Numer. Anal. **24**(4), 952–966 (1987)
22. Horn, R.A., Johnson, C.R.: Matrix Analysis. Cambridge University Press, New York, NY (1985)
23. Horton, G., Leutenegger, S.T.: A multi-level solution algorithm for steady-state Markov chains. In: Proceedings of the 1994 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems, pp. 191–200 (1994)
24. Iudin, D.B., Nemirovskii, A.S.: Informational complexity and effective methods of solution for convex extremal problems. Matekon Transl. Russ. East Eur. Math. Econ. **13**, 3–25 (1976)
25. Khachiyan, L.G.: A polynomial algorithm in linear programming. Sov. Math. Doklady **20**, 191–194 (1976)
26. Koury, J.R., McAllister, D.F., Stewart, W.J.: Iterative methods for computing stationary distributions of nearly completely decomposable Markov chains. SIAM J. Alg. Disc. Meth. **5**(2), 164–186 (1984)
27. Krieger, U.R.: Numerical solution of large finite Markov chains by algebraic multigrid techniques. In: Stewart, W. (ed.) Numerical Solution of Markov Chains, pp. 403–424. Kluwer, Dordrecht (1995)
28. Krieger, U.R.: On a two-level multigrid solution method for Markov chains. Linear Algebra Appl. **223–224**, 415–438 (1995)
29. Leutenegger, S.T., Horton, G.: On the utility of the multi-level algorithm for the solution of nearly completely decomposable Markov chains. Tech. Rep. 94-44, ICASE (1994)
30. Lüthi, H.-J.: On the solution of variational inequalities by the ellipsoid method. Math. Oper. Res. **10**(3), 515–522 (1985)
31. Mandel, J., Sekerka, B.: A local convergence proof for the iterative aggregation method. Linear Algebra Appl. **51**, 163–172 (1983)
32. Marek, I., Mayer, P.: Convergence analysis of an iterative aggregation/disaggregation method for computing stationary probability vectors of stochastic matrices. Numer. Linear Algebra Appl. **5**, 253–274 (1998)
33. Marek, I., Mayer, P.: Convergence theory of some classes of iterative aggregation/disaggregation methods for computing stationary probability vectors of stochastic matrices. Linear Algebra Appl. **363**, 177–200 (2003)
34. Molloy, M.K.articletitlePerformance analysis using stochastic Petri nets : IEEE Trans. Comput. **C-31**, 913–917 (1982)
35. Philippe, B., Saad, Y., Stewart, W.J.: Numerical methods in Markov chain modeling. Oper. Res. **40**(6), 1156–1179 (1992)
36. Rockafellar, R.T.: Convex Analysis. Princeton University Press, New Jersey (1970)
37. Shor, N.Z.: Cut-off method with space extension in convex programming problems. Cybernetics **13**, 94–96 (1977)
38. Simon, H.A., Ando, A.: Aggregation of variables in dynamic systems. Econometrica **29**, 111–138 (1961)
39. Stewart, W.J.: An Introduction to the Numerical Solution of Markov Chains. Princeton University Press, Princeton, NJ (1994)
40. Takahashi, Y.: A lumping method for numerical calculations of stationary distributions of Markov chains. Tech. Rep. B-18, Department of Information Sciences, Tokyo Institute of Technology (1975)
41. Treister, E., Yavneh, I.: On-the-fly adaptive smoothed aggregation for Markov chains. SISC **33**, 2927—2949 (2011)
42. Treister, E., Yavneh, I.: Square and stretch multigrid for stochastic matrix eigenproblems. Numer. Linear Algebr. **17**, 229–251 (2010)
43. Trottenberg, U., Oosterlee, C.W., Schüller, A.: Multigrid. Elsevier Academic Press, San Diego, California (2001)
44. Washio, T., Oosterlee, C.W.: Krylov subspace acceleration for nonlinear multigrid schemes. Electron. Trans. Numer. Anal. **6**, 271–290 (1997)