Visibility Based Preconditioners for Bundle Adjustment
Supplementary Material

Avanish Kushal
University of Washington, Seattle
kushalav@cs.washington.edu

Sameer Agarwal
Google Inc.
sameeragarwal@google.com

In this document, we report in more detail, the comparative performance of the preconditioners introduced in the main paper. Also proofs are provided for the claim that band diagonals of positive semidefinite matrices can be indefinite and that they can be made positive semi-definite via a simple scaling.

1 Linear Problems

1.1 Comparison by Iterations

Table 1 lists the problems used for this experiment, which compared the six preconditioners (including gsp-3) by the number of iterations it took for them to converge.

Ladybug	Venice
problem-162-22824	problem-52-64053
problem-282-37322	problem-89-110973
problem-339-44056	problem-245-198739
problem-384-49181	problem-427-310384
problem-412-52215	

Table 1: Problems for Experiment 1

We also reproduce here in Figure 1, the performance profiles of the six preconditioners on these problems. Performance is measured here in terms of number of iterations it took the solver to achieve convergence. As can be seen from the figure, the Schur-complement based preconditioners dominate the preconditioners operating over the Hessian. Also observe, that cluster-tridiagonal performs the best.

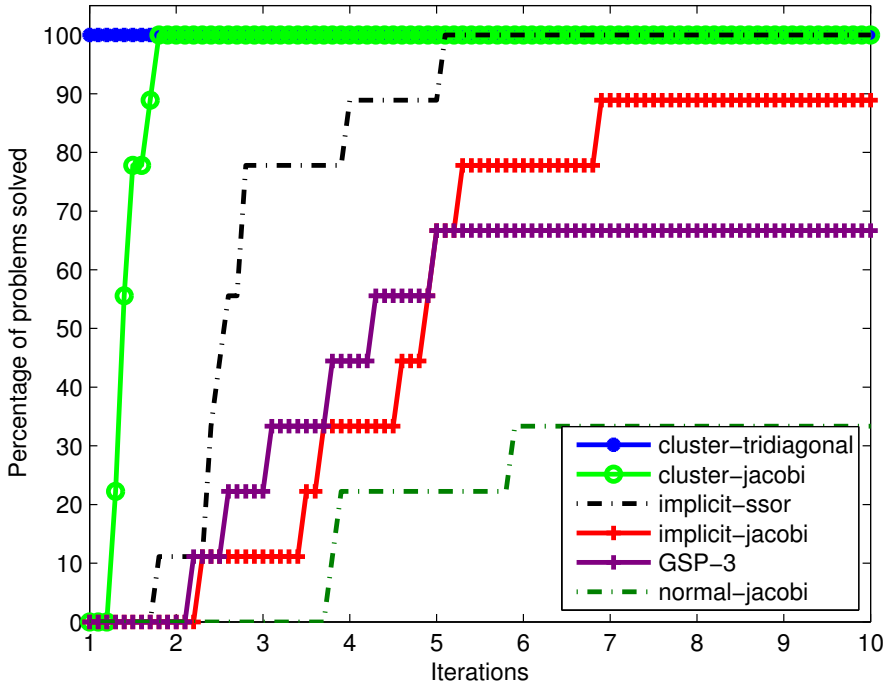

Figure 1: Iteration based performance profile for the 6 preconditioners with $\tau = 10^{-5}$

Table 2 shows the condition numbers for the six preconditioners cluster-tridiagonal, cluster-jacobi, implicit-jacobi, implicit-ssor, normal-jacobi, gsp-3 we considered for the problems in Experiment 1. We also include the condition number of the Schur complement S, and the Hessian H, to show how the various preconditioners improve them. The smallest condition number for each problem is indicated in bold. Observe that in every instance cluster-tridiagonal

problem	Hessian	normal-jacobi	gsp-3	S	implicit-jacobi	implicit-ssor	cluster-jacobi	cluster-tridiagonal
problem-162-22824	4.58e+08	3.43e+08	8.01e+07	1.04e+07	4.47e+06	1.14e+06	1.35e+06	**1.05e+06**
problem-282-37322	1.15e+09	7.40e+08	2.75e+07	3.29e+07	1.34e+06	1.27e+06	1.24e+05	**8.63e+05**
problem-339-44056	1.49e+09	1.87e+09	5.63e+07	6.00e+07	3.00e+07	2.18e+06	8.67e+05	**6.71e+05**
problem-384-49181	1.15e+09	1.18e+09	3.31e+07	3.39e+07	1.98e+07	8.81e+05	3.82e+05	**3.31e+05**
problem-412-52215	1.20e+09	1.00e+09	8.12e+07	3.36e+07	1.53e+07	2.04e+06	1.24e+06	**5.07e+05**
problem-52-64053	5.73e+09	8.37e+09	2.69e+09	1.99e+07	7.44e+06	8.08e+06	3.54e+06	**1.00e+00**
problem-89-110973	7.40e+09	1.08e+10	2.69e+09	2.55e+07	9.89e+06	9.08e+06	4.66e+06	**1.67e+06**
problem-245-198739	8.90e+09	1.23e+10	2.72e+09	4.32e+07	1.46e+07	1.15e+07	5.94e+06	**2.55e+06**
problem-427-310384	1.09e+10	1.91e+10	2.37e+09	5.29e+07	2.10e+07	1.68e+07	7.00e+06	**2.62e+06**

Table 2: Condition Numbers

has the best condition number. Observe also that gsp-3, has a worse condition number than the Schur complement S, thus at least in this small test it seems that eliminating the points from the linear problem provides as much benefit or more as constructing a low stretch spanning tree and using it to precondition H.

Like Agarwal et al., (Bundle Adjustment in the Large), the Hessian matrix ($J^\top J$) has been scaled by the inverse of its diagonal (the scalar Jacobi preconditioner).

1.2 Comparison by Time

Table 3 lists the problems used for this experiment, which compares the four Schur-based preconditioners on the time it takes for them to solve large linear least squares problems to convergence. It consists of 26 problems each from the Venice and Ladybug datasets.

Figure 2 reproduces at full scale the performance profiles of the four Schur based preconditioners for varying values of $\tau = 10^{-2}, 10^{-3}$ and 10^{-5}. The first row shows the performance profiles for Ladybug, the next for Venice, and the third for the combined dataset(all 52 problems together). Note that cluster-tridiagonal outperforms the others in all the charts.

In Figures 3–8 we show the detailed convergence behavior of the four preconditioners. For each preconditioner, we plot the log relative residual $log(\frac{||Ax_k - b||}{||Ax_0 - b||})$ as a function of time. To be fair to each preconditioner, the time needed to compute the preconditioner is accounted for. Note that Conjugate Gradients algorithm does not reduce the residual monotonically, thus the relative residual plots are oscillatory. As can be seen from the plots for most of the problems, the cluster-tridiagonal performs the best, followed by cluster-jacobi, implicit-ssor and implicit-jacobi in that order.

LADYBUG	VENICE
problem-412-52215	problem-427-310384
problem-460-56811	problem-744-543562
problem-539-65220	problem-951 -708276
problem-598-69218	problem-1102-780462
problem-646-73548	problem-1158-802917
problem-707-78455	problem-1184-816583
problem-783-84444	problem-1238-843534
problem-810-88814	problem-1288-866452
problem-856-93344	problem-1350-894716
problem-885-97473	problem-1408-912229
problem-931-102699	problem-1425-916895
problem-969-105826	problem-1473-930345
problem-1031-110968	problem-1490-935273
problem-1064-113655	problem-1521-939551
problem-1118-118384	problem-1544-942409
problem-1152-122269	problem-1638-976803
problem-1197-126327	problem-1666-983911
problem-1235-129634	problem-1672-986962
problem-1266-132593	problem-1681-983415
problem-1340-137079	problem-1682-983268
problem-1469-145199	problem-1684-983269
problem-1514-147317	problem-1695-984689
problem-1587-150845	problem-1696-984816
problem-1642-153820	problem-1706-985529
problem-1695-155710	problem-1776-993909
problem-1723-156502	problem-1778-993923

Table 3: List of Problems for experiment 2

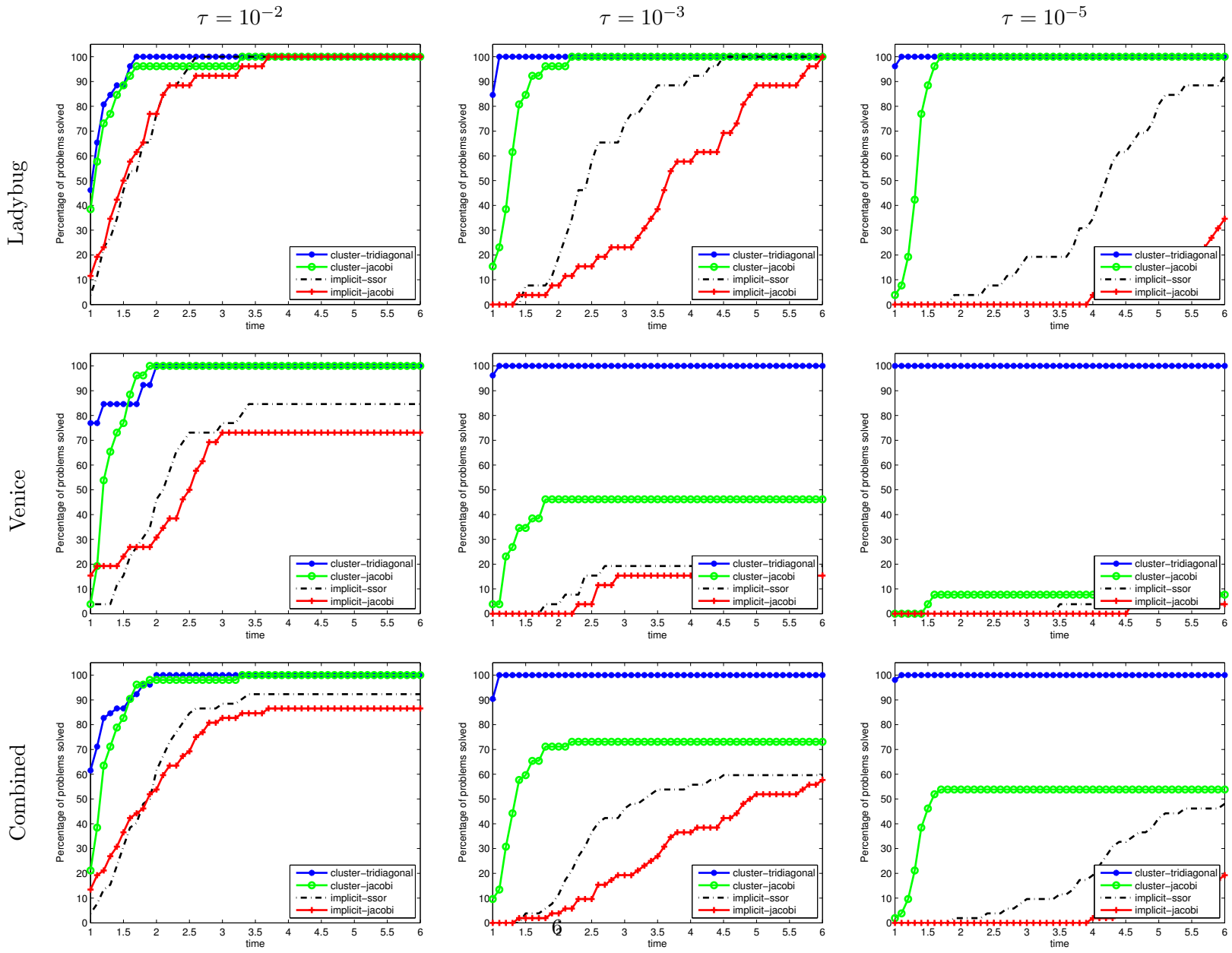

Figure 2: Time based performance profiles.

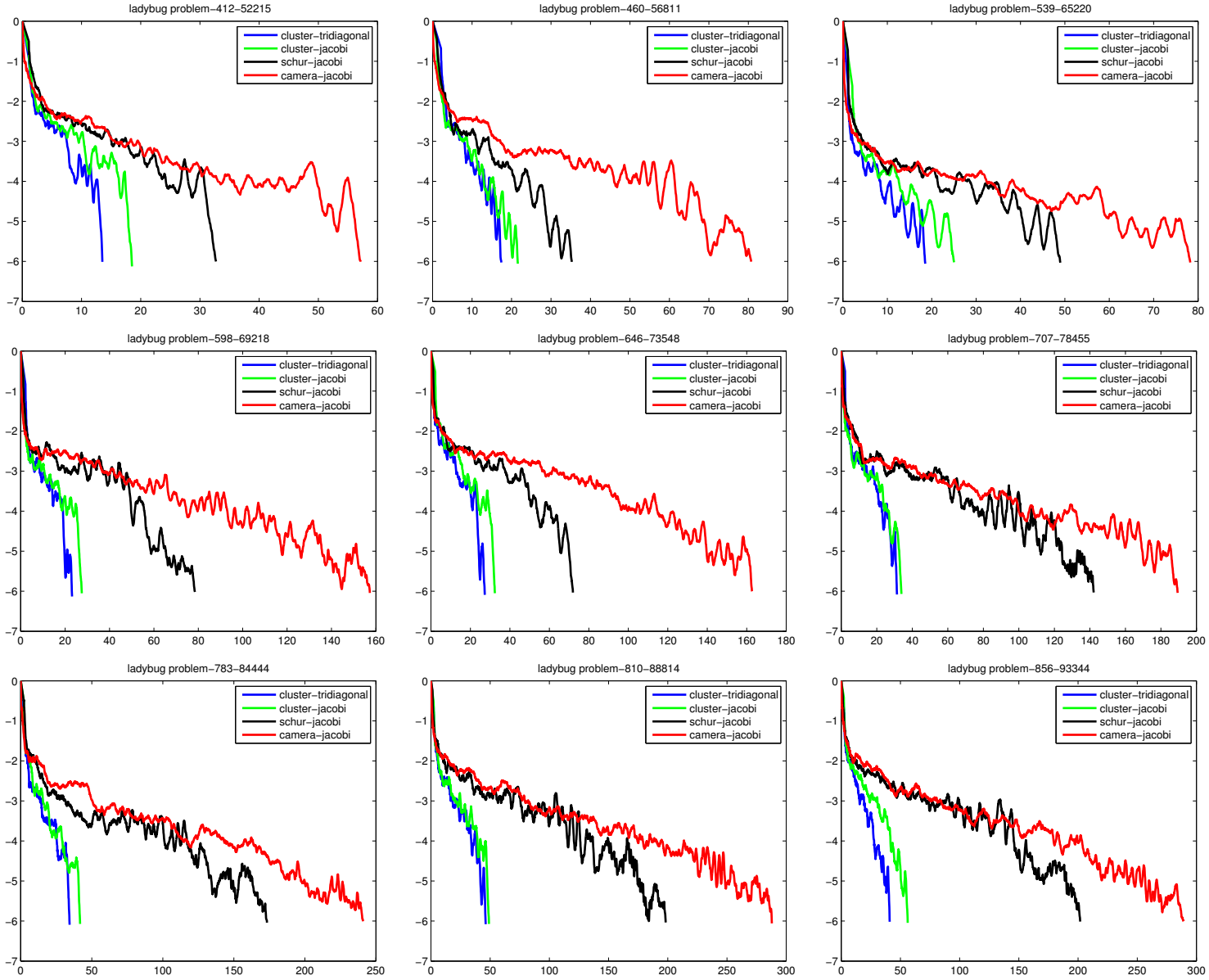

Figure 3: Convergence Plots for Problems 1-9 of the Ladybug dataset

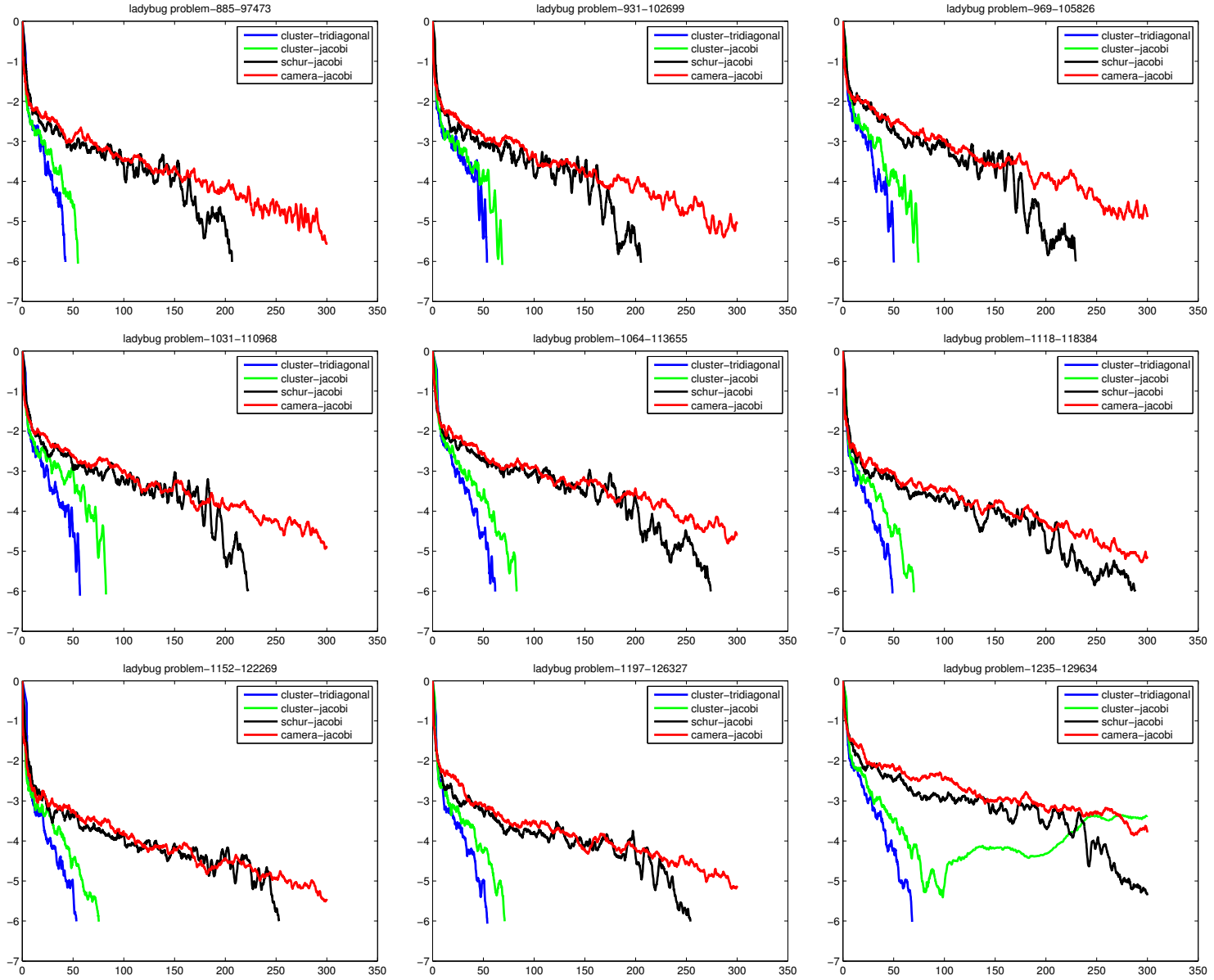

Figure 4: Convergence Plots for Problems 10-18 of the Ladybug dataset

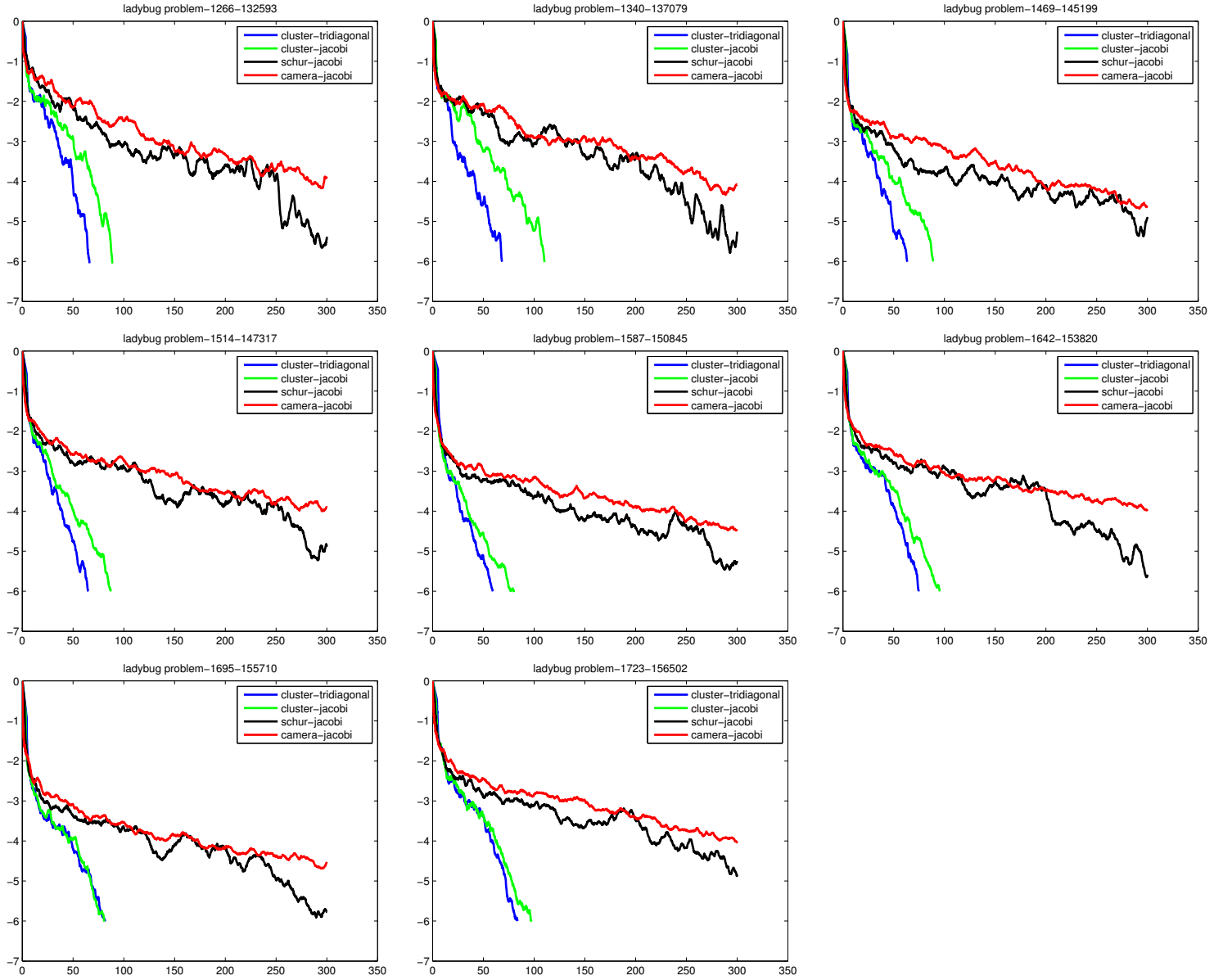

Figure 5: Convergence Plots for Problems 19-26 of the Ladybug dataset

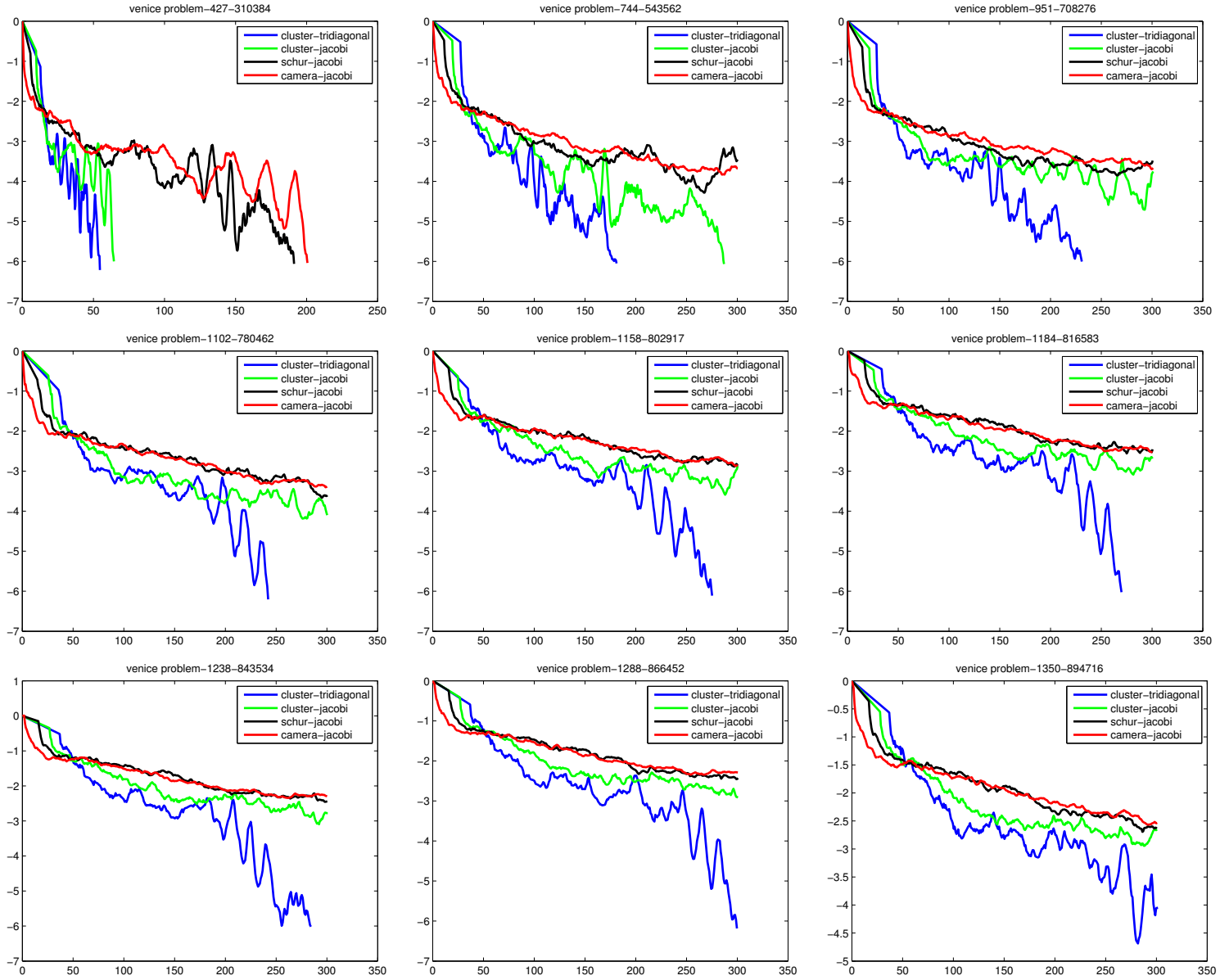

Figure 6: Convergence Plots for Problems 1-9 of the Venice dataset

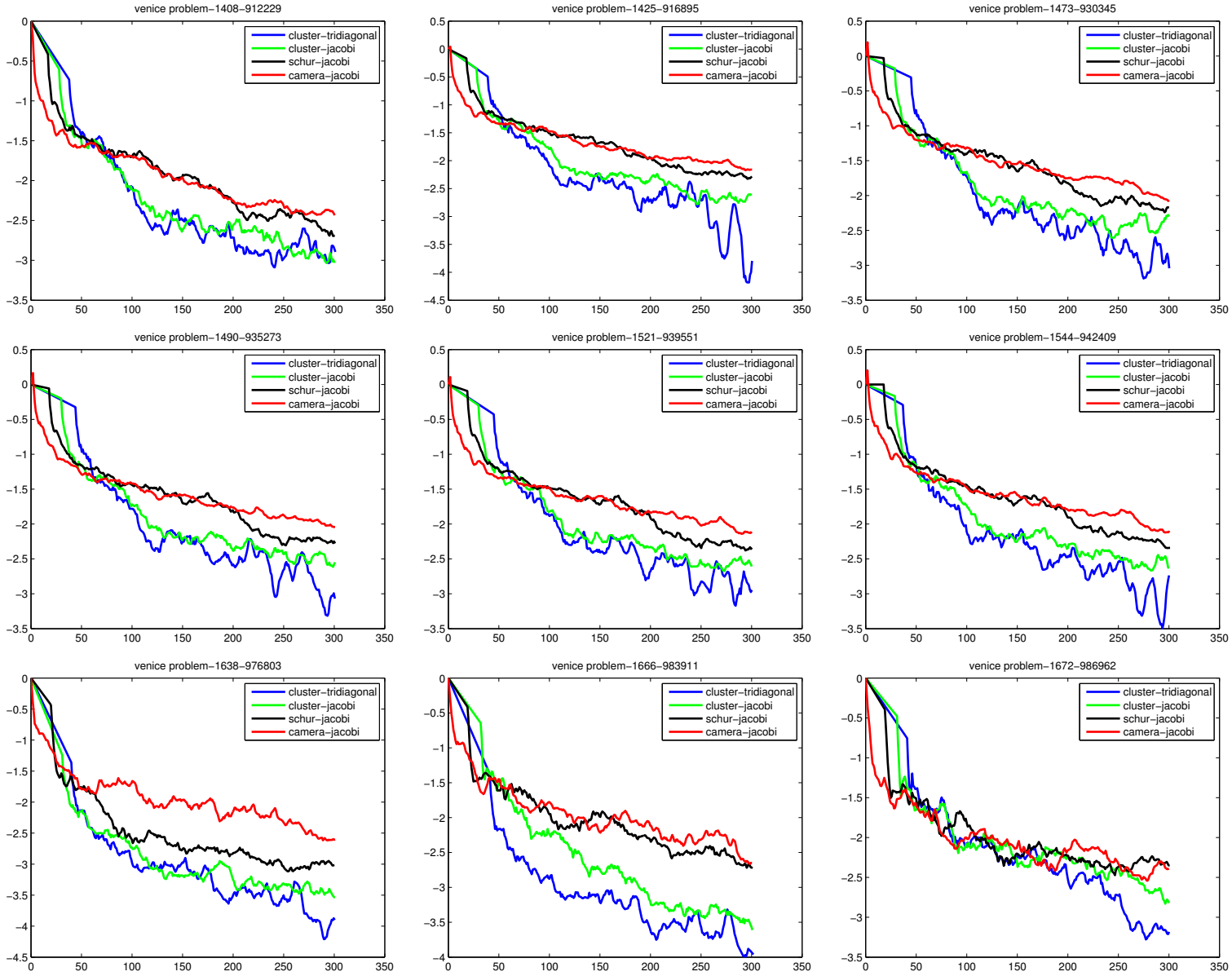

Figure 7: Convergence Plots for Problems 10-18 of the Venice dataset

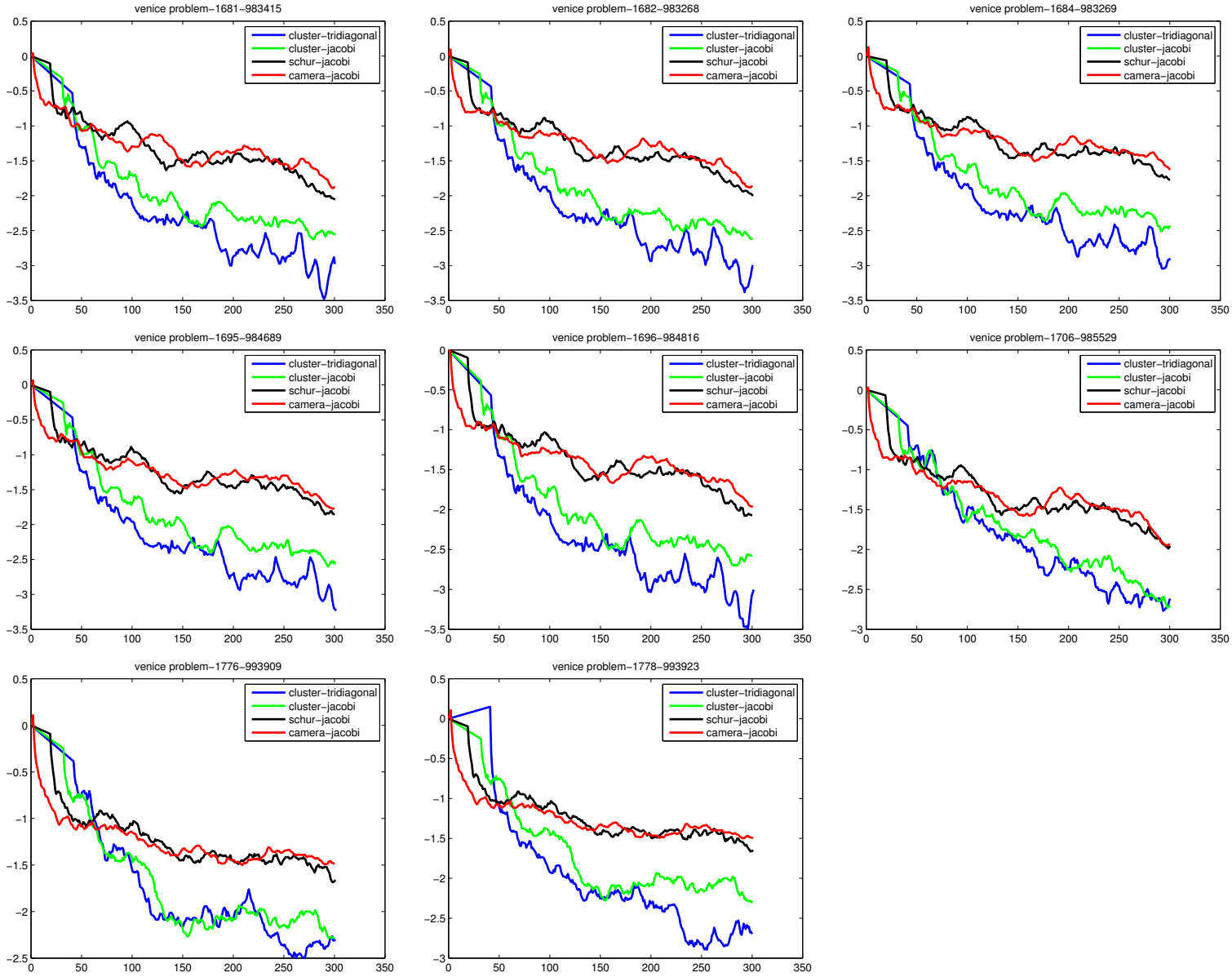

Figure 8: Convergence Plots for Problems 19-26 of the Venice dataset

2 Non-linear Problems

We use the problems listed in Table 3(same as that used for the experiment on the linear problems based on time) to report the performance of the non-linear solvers.

Figure 9 shows the performance profiles for the four preconditioners and the explicit-sparse solver for $\tau = 10^{-1}$, $\tau = 10^{-2}$, $\tau = 10^{-3}$. Here, the initial setup time to compute the clustering as well as the *degree 2* forest is also taken into account. As can be seen from the plots, for tighter values of τ, cluster-tridiagonal outperforms the rest. Even for $\tau = 10^{-1}$, where the initial setup cost becomes a factor, the preconditioners cluster-tridiagonal, cluster-jacobi catch up quickly with the others as the value of α is increased.

Figures10–15 show the detailed convergence behavior of Levenberg-Marquardt as a function of the linear solvers used. Again, we plot the log relative error for each problem.

$$e_{k,s} = \log \frac{f_{k,s} - f^*}{f_0 - f^*}, \tag{1}$$

Where, f_0 is the initial error, f^* is the lowest error across all solvers and $f_{k,s}$ is the error for solver s at iteration k. The log relative error $e_{k,s}$ is plotted against time. As can be seen, for most of the plots, cluster-tridiagonal and cluster-jacobi compete for the best preconditioner.

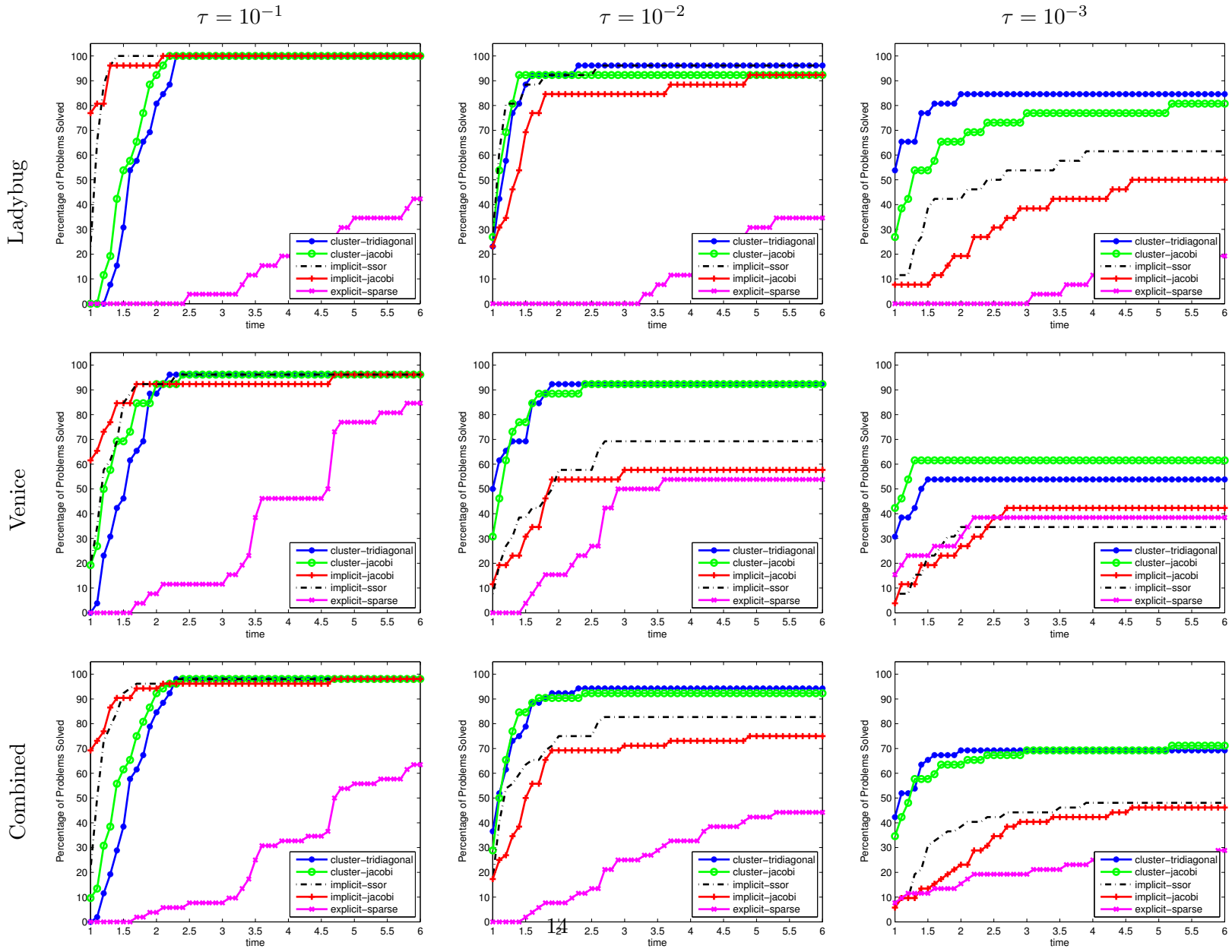

Figure 9: Performance profiles for the non-linear solver as the linear solvers are varied.

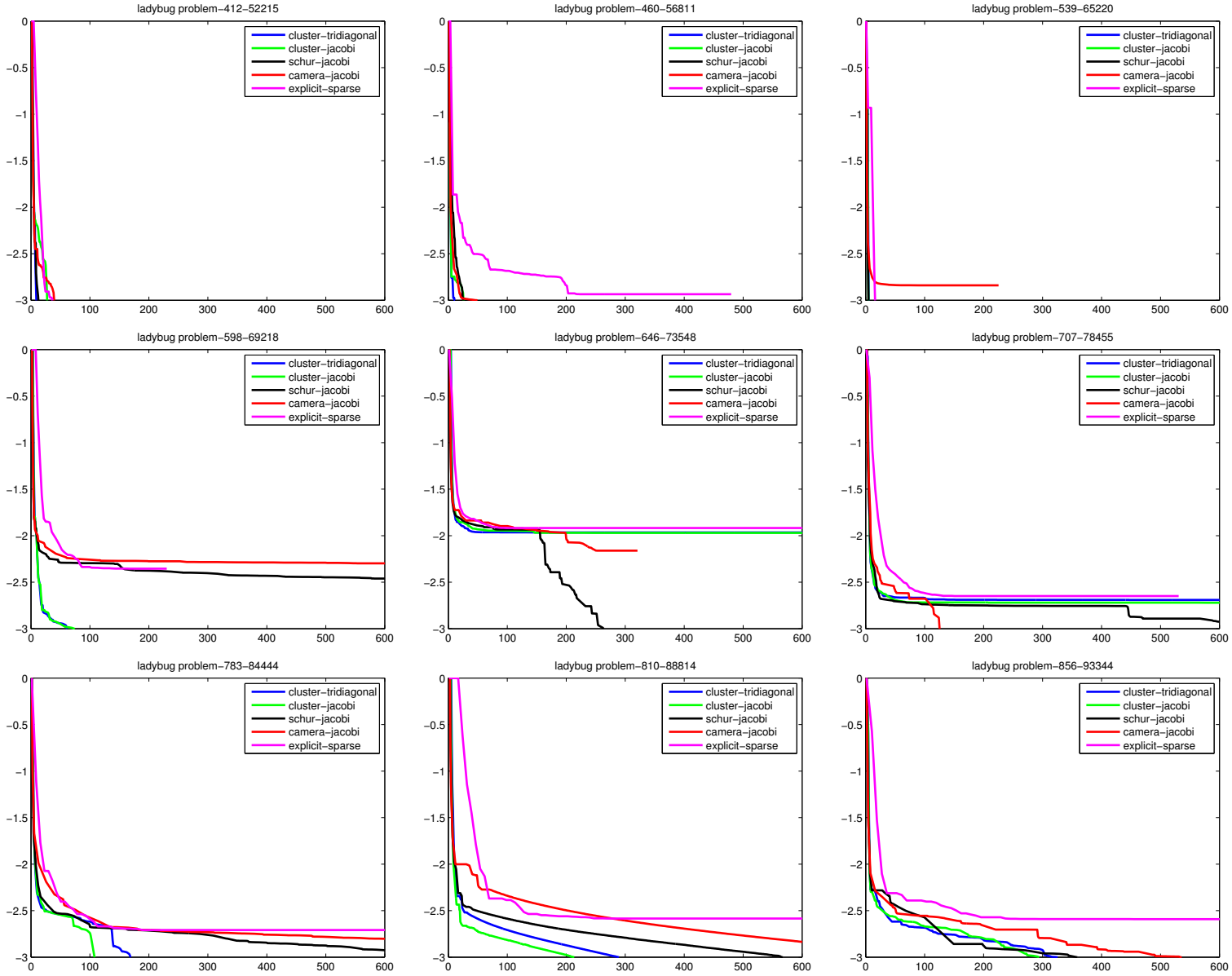

Figure 10: Convergence Plots for Problems 1-9 of the Ladybug dataset

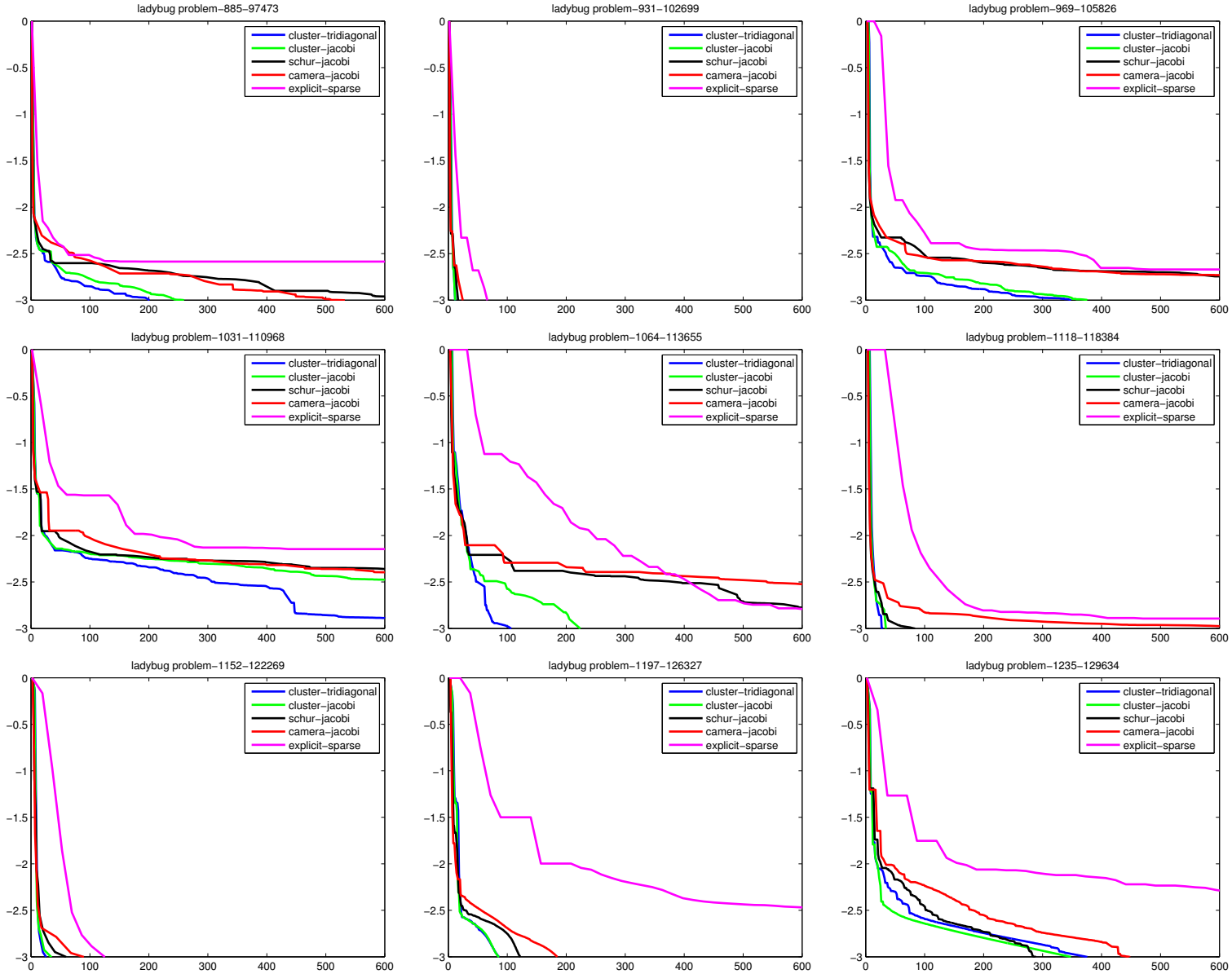

Figure 11: Convergence Plots for Problems 10-18 of the Ladybug dataset

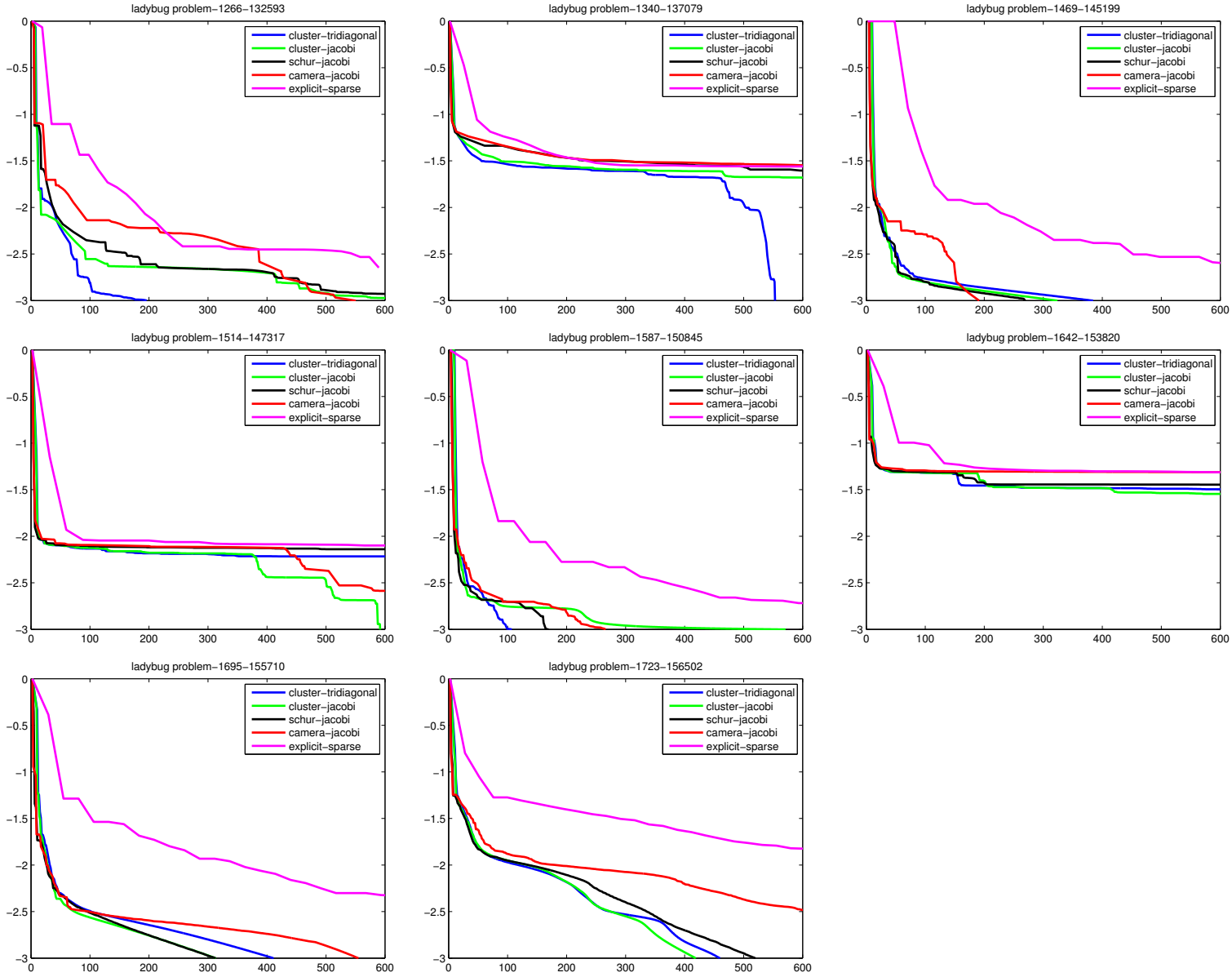

Figure 12: Convergence Plots for Problems 19-26 of the Ladybug dataset

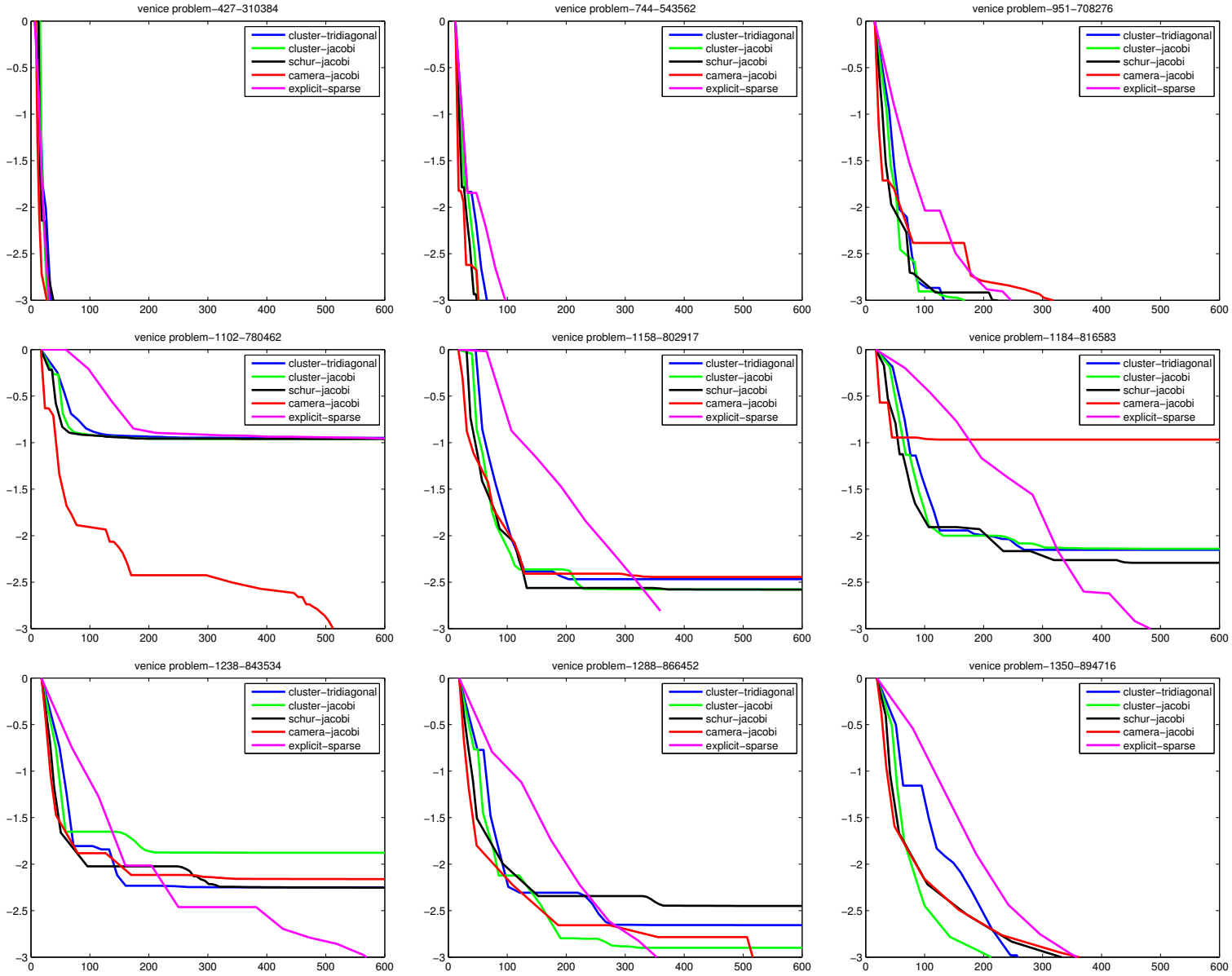

Figure 13: Convergence Plots for Problems 1-9 of the Venice dataset

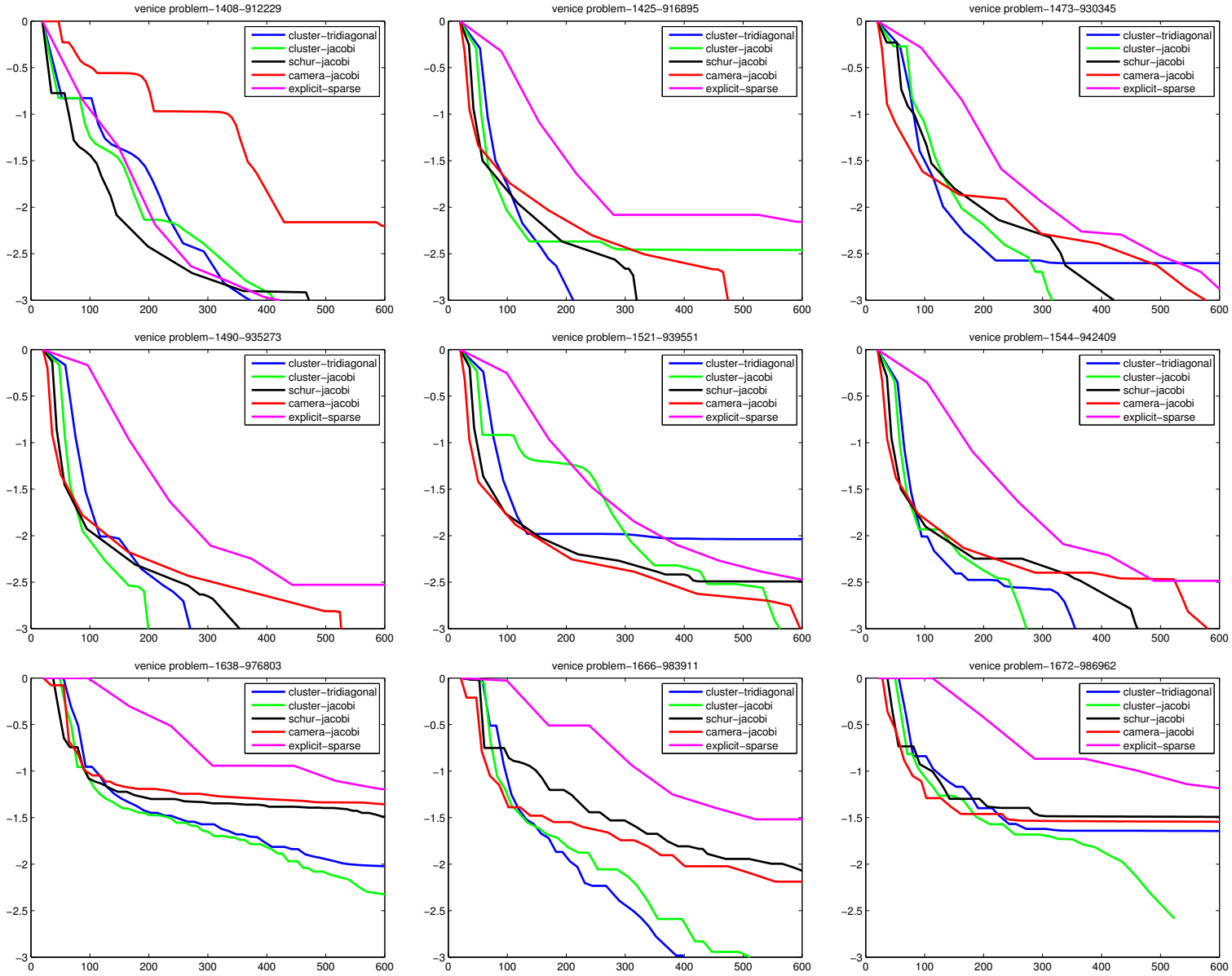

Figure 14: Convergence Plots for Problems 10-18 of the Venice dataset

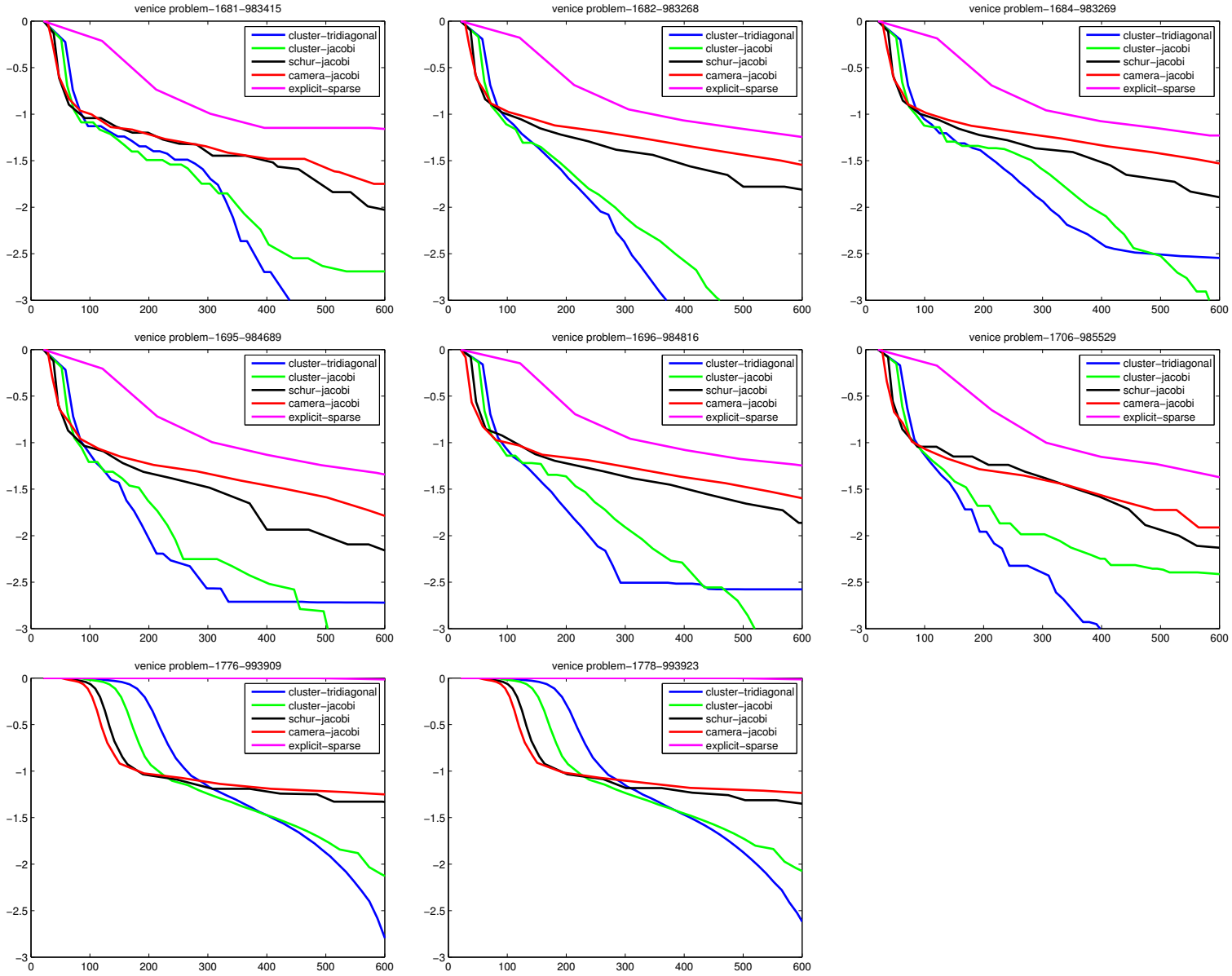

Figure 15: Convergence Plots for Problems 19-26 of the Venice dataset

3 Proof of Lemma 1

Lemma 1. *Let A be a symmetric positive semidefinite(PSD) matrix, then the tridiagonal matrix $M(\nu)$*

$$m_{ij}(\nu) = \begin{cases} a_{ij} & i = j \\ \nu a_{ij} & |i-j| = 1 \\ 0 & otherwise \end{cases} \tag{2}$$

is positive semidefinite for $\nu = 0.5$ and for every $\epsilon > 0$, there exists a positive semidefinite matrix A, such that $M(0.5+\epsilon)$ is indefinite.

Proof. Since A is symmetric PSD, there exists a matrix U such that $A = U^\top U$. Let the column vectors of $U = [u_1, u_2 ... \cdots u_n]$. Consider $M(0.5)$ .Then for any x,

$$\begin{aligned} x^\top M x =& \sum_{i=1}^{n} x_i M_{i,i} x_i + 2\sum_{i=1}^{n-1} x_i M_{i,i+1} x_{i+1} \\ =& \sum_{i=1}^{n} x_i A_{i,i} x_i + \sum_{i=1}^{n-1} x_i A_{i,i+1} x_{i+1} \\ =& \sum_{i=1}^{n} x_i^\top u_i^\top u_i x_i + \sum_{i=1}^{n-1} x_i^\top u_i^\top u_{i+1} x_{i+1} \\ =& \frac{1}{2}\sum_{i=1}^{n-1} \left(x_i u_i + x_{i+1} u_{i+1}\right)^\top \left(x_i u_i + x_{i+1} u_{i+1}\right) + \frac{1}{2} x_1^\top u_1^\top u_1 x_1 + \frac{1}{2} x_n^\top u_n^\top u_n x_n \\ \geq& 0 \end{aligned}$$

Next we show that this is the best static scaling strategy, i.e. for $\nu > 0.5$, we can find a PSD matrix A, s.t. $M(\nu)$ is indefinite. Choose an integer $n > \frac{1}{2\epsilon} + 1$, and set A to be the square matrix of size n with all entries set to 1. Then clearly $A = \frac{1}{n}(11^\top)$ is PSD. Now consider the matrix $M(0.5+\epsilon)$ that has 1 on the diagonals and $0.5+\epsilon$ on the super

and sub diagonal and the vector $v = [1, -1, 1, -1....]^\top$, then observe that

$$\begin{aligned} v^\top M v &= \sum_{i=1}^{n} v_i^2 + 2\sum_{i=1}^{n-1}(0.5+\epsilon)v_i v_{i+1} \\ &= \sum_{i=1}^{n} 1 - 2\sum_{i=1}^{n-1}(0.5+\epsilon) \\ &= 1 - 2(n-1)\epsilon \\ &< 0 \end{aligned}$$

Thus $\nu = 0.5$ is the largest number that can be used in any static scaling strategy for scaling the sub and super diagonals, and ensuring that the matrix M(ν) still remains PSD. □