Department of Mathematics
TUM School of Computation, Information and Technology
Technical University of Munich

Learning Graphical Lyapunov Models using Best-subset Selection Methods

Rahul Radhakrishnan

Thesis for the attainment of the academic degree

Master of Science

at the TUM School of Computation, Information and Technology of the Technical University of Munich

Supervisor:

Prof. Dr. Mathias Drton

Advisors:

Msc. Philipp Dettling

Submitted:

Munich, 01 Aug. 2023

I hereby declare that this thesis is entirely the result of my own work except where otherwise indicated. I have only used the resources given in the list of references.

Munich, 01 Aug. 2023 Rahul Radhakrishnan

Acknowledgements

I would like to express my sincere gratitude to my thesis supervisor, Dr. Mathias Drton, for providing me with an opportunity to work under his guidance. He gradually guided me into the topics of this thesis by setting small milestones at every step along the way. This really helped me to stay motivated throughout the journey. In addition, I am deeply grateful to my advisor, Philipp Dettling, for helping me throughout this work with his ideas and suggestions. He was always available to address any doubts that I had or guide me in case I ran into some difficulties. His feedbacks helped me be on track and achieve my goals. I have learnt a lot working with both of them.

I would also like to thank Johannes Albert-von der Gönna, Dr. Th. Frank, Martin and the entire service desk team of the Leibniz-Rechenzentrum (LRZ) super computing facility. With absolutely no prior experience in parallel computing, I ran into so many difficulties in running simulations on the cluster. They were always there to help me overcome my problems. Without their support and help, this work would not have been possible.

In addition, I would like to thank Ms. Anja Hoffmann from the Examination committee of the Department of Mathematics. She was always there to clarify any doubts that I had throughout this journey. With her help, all the organizational aspects of my studies were carried out smoothly.

Finally, words cannot express my gratitute towards my family and friends. First and foremost, I would like to thank Mom, Dad and Kaushik for their constant love and support. They were always there to cheer me up when I was not feeling good. Kaushik was always there to motivate me and offer amazing advices. Many of his suggestions actually helped me with my thesis work. I always look up to him. I'm extremely grateful to Ivanna Godoy Muñoz, Joseph Saroufim, Kunal Rustagi, Nikhil Pingle, Kanika and Arjun Anish. They were always there for me when I needed them the most and offered me with unyielding emotional support. Each one of them helped me become a better person throughout this journey.

Zusammenfassung

Ursache-Wirkungs-Verhältnisse zwischen Variablen, die an einem dynamischen Prozess beteiligt sind, sind von großem Interesse. Bestehende grafische Modellansätze erlauben nur eine begrenzte Erforschung der Beziehungen. Insbesondere zyklische Interaktionen zwischen verschiedenen Komponenten, die Rückkopplungsschleifen darstellen, sind mit solchen Methoden im Allgemeinen schwieriger zu erfassen und zu interpretieren. Vor diesem Hintergrund bieten grafische Lyapunov-Modelle eine neue Perspektive für die Modellierung kausal-interpretierbarer Abhängigkeitsstrukturen in multivariaten Daten, indem sie jede unabhängige Beobachtung als eine einmalige Querschnittsschnappschuss eines multivariaten Ornstein-Uhlenbeck-Prozesses im Gleichgewicht behandeln. Das Gaußsche Gleichgewicht existiert unter einer Stabilitätsannahme für die Driftmatrix, und die Kovarianzmatrix des Gleichgewichts wird durch die kontinuierliche Lyapunov-Gleichung bestimmt. Die Modelle gehen davon aus, dass die Driftmatrix spärlich ist und ihre Unterstützung durch einen gerichteten Graphen bestimmt wird. Dieser gerichtete Graph kann insbesondere gerichtete Zyklen enthalten und ist einer der Hauptvorteile dieses Ansatzes. Frühere Arbeiten haben einen Ansatz zur Wiederherstellung dieser Unterstützung gezeigt, indem ein l_1- regularisierter Ansatz (Lasso) verwendet wurde, der darauf abzielt, eine spärliche Näherungslösung für die kontinuierliche Lyapunov-Gleichung zu finden, wenn eine Stichproben-Kovarianzmatrix gegeben ist, genannt *Direct Lyapunov Lasso*.

Diese Arbeit konzentriert sich auf die Wiederherstellung der Unterstützung der Driftmatrix aus der Lyapunov-Gleichung unter Verwendung der Best-Subset-Auswahl (BS). Die klassischen Methoden zur Lösung der Best-Subset-Auswahl, ein NP-hartes Problem, sind nicht auf große Problemgrößen skalierbar oder können keine globale Optimalität garantieren. Eine der jüngsten Arbeiten konzentriert sich auf die Umwandlung der Best-Subset-Auswahl in ein gemischt-ganzzahliges Optimierungsproblem (MIO). Mit den Fortschritten bei den Optimierungstechniken und dem parallelen Rechnen ist die gemischt-ganzzahlige Programmierung sehr leistungsfähig geworden. Ihre praktische Anwendbarkeit und Durchführbarkeit hängt von der Stärke der Formulierung, den Warmstarts und ihrer Qualität ab. Von den beiden vorgestellten Formulierungen für die Auswahl der besten Teilmenge wurde in dieser Studie die stärkere und rechnerisch effizientere Formulierung gewählt. Außerdem wurden die Formulierungen angepasst, um stabile Driftmatrizen wiederherzustellen. Die MIO-Formulierung wurde mit Gurobi auf seiner R-Schnittstelle gelöst.

Für die Warmstarts wurden vier verschiedene Initialisierungen untersucht: drei Varianten von projizierten Gradientenabstiegslösungen und direkte Lasso-Lösungen (BS_{lasso}). Die projizierten Gradientenlösungen erzielen eine $k-$sparse Lösung unter Verwendung von marginalen Regressionskoeffizienten als Initialisierungen. Ihre Varianten unterscheiden sich durch die Strategie, mit der diese marginalen Regressionskoeffizienten dem Algorithmus als Ausgangspunkt geliefert werden. Insbesondere kann man direkt die k-Kanten mit den größten Koeffizienten wählen (BS_{reg}), nur 1 Kante wählen (BS_{1edge}) und Kanten auf der Grundlage des Sparsamkeitsmusters aus der Schätzung der inversen Kovarianzmatrix wählen (BS_{glasso}).

Alle Initialisierungsstrategien werden miteinander und mit dem direkten Lyapunov-Lasso auf synthetischen und realen Datensätzen verglichen. Informationskriterien werden verwendet, um Driftmatrizen auszuwählen, die am besten zu den beobachteten Daten passen und gleichzeitig die Sparsamkeit ausgleichen. Die Ergebnisse zeigen, dass Lasso besser abschneidet als BS, sofern genügend Stichproben vorhanden sind. BS hat das Potenzial, bei geringer Stichprobengröße besser abzuschneiden als Lasso, insbesondere bei großen Graphenstrukturen. In dieser Studie wird auch festgestellt, dass Lasso-Lösungen im Vergleich zu projizierten Gradientenabstiegslösungen bessere Grenzen für die MIO-Formulierung liefern. BS_{lasso} schneidet also genauso gut oder besser ab als die anderen BS-Methoden. Alle auf projiziertem Gradientenabstieg basierenden Methoden haben eine ähnliche Leistung, wobei BS_{1edge} und BS_{glasso} geringfügig besser sind. Diese Vergleiche sind

relativ. Es zeigt sich, dass alle Methoden ab einer bestimmten Problemgröße und unterhalb eines bestimmten Wertes der Stichprobengröße sehr schlecht abschneiden. Wenn die Graphen weniger dünn sind (mehr Kanten), wird die Wiederherstellung der Unterstützung schwieriger.

Abstract

Cause and effect relationships between variables involved in a dynamical process are of great interest. Existing graphical model approaches allow limited exploration of relationships. In particular, cyclic interactions between different components that represent feedback loops are generally harder to capture and also to interpret using such methods. In this light, graphical Lyapunov models offer a new perspective on modeling causally interpretable dependence structure in multivariate data by treating each independent observation as a one-time cross-sectional snapshot of a multivariate Ornstein-Uhlenbeck processes in equilibrium. The Gaussian equilibrium exists under a stability assumption on the drift matrix, and the equilibrium covariance matrix is determined by the continuous Lyapunov equation. The models assume sparsity on the drift matrix and its support is determined by a directed graph. This directed graph, in particular, can include directed cycles and is one of the main advantage of this approach. Past works have shown one approach to this support recovery by using an l_1- regularized approach (lasso) that seeks to find sparse approximate solution to the continuous Lyapunov equation, given a sample covariance matrix, called the *Direct Lyapunov Lasso*.

This work focuses on recovering the support of the drift matrix from the Lyapunov equation using best-subset selection (BS). The classical methods of solving best-subset selection, a NP-hard problem, are not scalable to large problem sizes or cannot guarantee global optimality. One of the recent works focuses on converting best-subset selection to a mixed integer optimization (MIO) problem. With advancement in optimization techniques and parallel computing, mixed integer programming has become very powerful. Its practical applicability and feasibility relies on the strength of the formulation, warm starts and their quality. Out of the two formulations presented for best-subset selection, the stronger and computationally more efficient formulation was chosen in this study. Furthermore, the formulations were adapted to recover stable drift matrices. The MIO formulation was solved using Gurobi on its `R` interface.

For the warm starts, four different initializations were explored: three variants of projected gradient descent solutions and direct lasso solutions (BS_{lasso}). The projected gradient solutions achieve a $k-$sparse solution using marginal regression coefficients as initializations. Its variants differ in terms of the strategy used to supply these marginal regression coefficients to the algorithm as a starting point. In particular, one can directly choose the k edges with largest coefficients (BS_{reg}), choose only 1 edge (BS_{1edge}), and choose edges based on sparsity pattern of the estimate of inverse covariance matrix ($\text{BS}_{\text{glasso}}$).

All the initialization strategies are compared against each other and to the Direct Lyapunov Lasso on synthetic and real dataset. Information criterion are used to select drift matrices that are best fitting to the observed data, while balancing the sparsity. The results show that lasso outperforms BS, provided there are enough samples. BS has the potential to perform better than lasso for low sample size, especially for large graph structures. One also finds in this study that lasso solutions seem to provide better bounds to the MIO formulation compared to projected gradient descent solutions. So BS_{lasso} performs as good as or better than the other BS methods. All the projected gradient descent based methods have similar performance, with BS_{1edge} and $\text{BS}_{\text{glasso}}$ being marginally better. These comparisons are relative. It is shown that all the methods perform very poor beyond certain problem size and below a certain value of sample size. Furthermore, as the graphs become less sparse (more number of edges), one finds that the task of support recovery becomes harder.

Contents

1 Introduction

Natural processes are often too complex and cannot be fully understood without understanding completely how underlying systems interact with each other. With high-throughput technologies today, it is possible to get quality experimental data. To analyze these data and recover causal relations between measured quantities is a challenging task. Causal graphical models allow one to capture and explore dependencies induced by cause-effect relations. Causal relations among a set of variables, V, are represented by set of edges $E \subseteq (V \times V)$ connecting nodes in the graph. These relations are defined in terms of parametric functional relationships. If the underlying distribution is Gaussian, there is a well known result that the entries of the inverse covariance matrix (Σ^{-1}) are considered as parameters that encode the variable dependencies (Lemma 7.2 [Gir21]). In order to learn the dependencies, one aims to find Σ^{-1}. The non-zero entries of this matrix would correspond to edges of the graph. This approach is suitable for obtaining an undirected graph. If one requires information on interaction between variables, inverse covariance estimation is therefore not suitable.

Linear structural equation model is a statistical model that aims to determine underlying dependencies between variables [Drt18]. In particular, the variables of a random vector $X = \{X_i : i \in V\}$ are considered linearly dependent to other variables via

$$X = \Lambda^T X + \epsilon \tag{1.1}$$

where $\Lambda = (\lambda_{ij}) \in \mathbb{R}^{V \times V}$ is a matrix of unknown parameters that encode the relationships and ϵ is independent random noise. Assume ϵ has a covariance matrix Ω, positive definite and unknown. Then, provided $I - \Lambda$ is invertible, the standard parametrization of a linear causal model specifies the covariance matrix $\Sigma = \mathrm{Var}[X]$ as a solution to the equation

$$(I - \Lambda)^T \Sigma (I - \Lambda) = \Omega \tag{1.2}$$

where I is the identity matrix. The associated mixed graph has directed edges and bidirected edges determined by nonzero entries of Λ and Ω. If one fixes an acyclic graph, the framework provides a parametrization (covariance structure) of the observables from a directed acyclic model – potentially with latent variables. In the cyclic case, where one allow the graphs to have directed cycles, the covariance structure can, moreover, be interpreted as an equillibrium distribution for a deterministic process whenever the spectrum of Λ is inside the unit circle [HEH12]. But in general, multiple complications were reported associated with usage of structural models on graph with cycles ([Amé+20; DFW19; Ric13]).

Fitch [Fit19] and Varando and Hansen [VH20a] proposed an alternative approach on tackling this problem by using graphical continuous Lyapunov models. In particular, they show that for certain time stochastic processes, there exists an alternate parametrization for Σ based on the continuous Lyapunov equation,

$$M\Sigma + \Sigma M^T + C = 0 \tag{1.3}$$

where $M \in \mathbb{R}^{V\times V}$ and $C \in \mathbb{R}^{V\times V}$ are unknown parameters. Knowing the support or the sparsity pattern of M can reveal information about interactions between different variables involved in the process.

The objective of this thesis is to learn graphical structure that arise within such graphical continuous Lyapunov models. This, in particular, is attempted by trying to recover the support of M via mixed integer optimization techniques.

This thesis is organized as follows: In the subsequent sections of this chapter, some basic background concepts are introduced that are required for this work. In chapter 2, the methods implemented in this work are discussed in detail. In particular, contruction of initial guesses to solve optimization problem, mathematical formulation, model selection techniques and simulation settings are provided. Lastly in chapter 3, the results of this work are presented and discussed.

1.1 Notations

- Let $p \in \mathbb{N}$. $[p] = \{1, \ldots, p\}$.
- The l_b norm of $v \in \mathbb{R}^p$ is $\|v\|_b = (\sum_{i=1}^{p} |v_i|^b)^{1/b}$, with $\|v\|_{\infty=\max_{1\le i\le n}}|v_i|$. Applying this on to a matrix $A = (a_{ij}) \in \mathbb{R}^{p\times p}$ and obtain the norm $\|A\|_b = (\sum_{i=1}^{p}\sum_{j=1}^{n} |a_{ij}|^b)^{1/b}$. The operator norm is denoted by $|||A|||_b = \max\{\|Ax\|_b : \|x\|_b = 1\}$. $|||A|||_2$ is called a spectral norm (maximum singular value of A) and $|||A|||_\infty = \max_{1\le i\le p}\sum_{j=1}^{p} |a_{ij}|$.
- For an index set S, v_S is a subvector that is obtained by selecting the indices present in S. $v_{S,m}$, where $m \in \mathbb{N}$ is the number of iterations of an algorithm, represent the subvector at the mth iteration. When S is all the indices of v, it is usually ignored and understood as v_m. When there are nested iterations, there exist another $r \in \mathbb{N}$, the sequence would be denoted as v_m^r. Any other variation to this notation would either be obvious or explicitly stated when required.
- The Kroneckor product of A and another matrix $B = (b_{uv}) \in \mathbb{R}^{p\times p}$ is denoted by $A \otimes B$. It is a matrix in $\mathbb{R}^{p^2\times p^2}$ with the entries $[A \otimes B]_{(i-1)p+j,(k-1)p+l} = A_{ik}B_{jl}$.
- In this study p^2 and n^2 for $p, n \in \mathbb{N}$, are abbreviated as $P := p^2$ and $N := n^2$.
- In the performance metrics, TP - true positve, FP - false positive, FN - false negative, and TN - true negative.

1.2 Graphical Continuous Lyapunov Models

An i.i.d sample $X_1, \ldots, X_n \in \mathbb{R}^p$ is assumed to be drawn from a $p-$ dimensional Ornstein-Uhlenbeck process.

Theorem 1.2.1 (Arnold [L92])**.** *The solution of the equation*

$$d\mathrm{x} = (M(t) + a(t))\mathrm{x}dt + D(t)dW,\ \mathrm{x}_{t0} = c \tag{1.4}$$

is a stationary Gaussian process if $M(t) = M, a(t) = 0, D(t) = D$, the eigenvalues of M have negative real parts, and $W(t)$ is a standard brownian motion in $\mathbb{R}^p$ known as the Wiener process, where Σ, the steady state covariance matrix, is

$$\Sigma = \int_0^\infty e^{Mt} D D^T e^{M^T t} dt,$$

or equaivalently the solution of the Lyapunov equation

$$M\Sigma + \Sigma M^T + C = 0 \tag{1.5}$$

where $C := DD^T$.

Now one needs the following concepts from graphical models.

Definition 1.2.2 (Directed graph)**.** *Let $G = (V, E)$ be a graph, consisting of V is the set of nodes and $|E|$ is the set of edges, then a directed graph is an ordered pair of elements of V.*

Definition 1.2.3. *Let G be a directed graph with vertex set $V = [p]$ and an edge set E that includes all self-loops $i \to i$, $i \in [p]$. Given a choice of $C \in \mathrm{PD}_p$, the graphical continuous Lyapunov model of G is the set of covariance matrices*

$$\mathcal{M}_{G,C} = \{\Sigma \in \mathrm{PD}_p : M\Sigma + \Sigma M^T = -C \text{ with } M \in \mathbb{R}^E\},$$

where PD_p is the cone of positive definite matrices in $\mathbb{R}^{p\times p}$ and $\mathbb{R}^E$ is the space of matrices $M = (m_{ij}) \in \mathbb{R}^{p\times p}$ with $m_{ji} = 0$ whenever $i \to j \notin E$.

The *drift matrix* M, thus determines the relations between the coordinates of the Ornstein-Uhlenbeck process $X(t)$. By assumption of Theorem 1.2.1, M has negative eigenvalues and is a *stable matrix*. G will always contain self-loops $i \to i$ for every $i \in V$. Now, some concepts from matrix theory are introduced ahead.

Definition 1.2.4. *(Kronecker Product) If $\mathbf{A} \in \mathbb{R}^{m\times n}$ matrix and $\mathbf{B} \in \mathbb{R}^{p\times q}$ is a $p \times q$ matrix, then the Kronecker product $\mathbf{A} \otimes \mathbf{B}$ is the block matrix*

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & \cdots & a_{mn}\mathbf{B} \end{bmatrix} \in \mathbb{R}^{pm\times qn}$$

Definition 1.2.5. *The* vec*-operator turns a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ into a vector in $\mathbb{R}^{mn}$ by stacking its columns on each other such that*

$$\mathrm{vec}(A) = (a_{11}, a_{21}, \ldots, a_{m1}, \ldots, a_{1n}, \ldots, a_{mn})^T$$

Definition 1.2.6 (Commutation matrix)**.** *The Commutation matrix,* $K_{mn} \in \mathbb{R}^{mn \times mn}$ *transforms* $\text{vec}(A)$ *to* $\text{vec}(A^T)$ *for* $A \in \mathbb{R}^{m \times n}$*. It is given by,*

$$K_{(m,n)} = \sum_{i=1}^{m} \sum_{j=1}^{n} (a_i e_j^T) \otimes (e_j a_i^T),$$

where $a_i \in \mathbb{R}^m$ *and* $e_j \in \mathbb{R}^n$ *are the ith and jth column unit vector respectively. See [***magnus&Neudecker***].*

Lemma 1.2.7. *Let* $\mathbf{A} \in \mathbb{C}^{m \times n}$,$\mathbf{X} \in \mathbb{C}^{n \times o}$*, and* $\mathbf{B} \in \mathbb{C}^{o \times p}$*, then the following statement holds true*

$$\text{vec}(\mathbf{AXB}) = (\mathbf{B}^T \otimes \mathbf{A})\text{vec}(X) \tag{1.6}$$

Proof. Note that

$$(\mathbf{AXB})_{ij} = \sum_{l=1}^{o} \sum_{k=1}^{n} (\mathbf{A})_{ik} (\mathbf{X})_{kl} (\mathbf{B})_{lj}$$

Then the left hand side can be written as

$$\begin{aligned}
\text{vec}(\mathbf{AXB}) &= \begin{bmatrix} \sum_{l=1}^{o} \sum_{k=1}^{n} (\mathbf{A})_{1k} (\mathbf{X})_{kl} (\mathbf{B})_{l1} \\ \vdots \\ \sum_{l=1}^{o} \sum_{k=1}^{n} (\mathbf{A})_{mk} (\mathbf{X})_{kl} (\mathbf{B})_{l1} \\ \vdots \\ \sum_{l=1}^{o} \sum_{k=1}^{n} (\mathbf{A})_{1k} (\mathbf{X})_{kl} (\mathbf{B})_{lp} \\ \vdots \\ \sum_{l=1}^{o} \sum_{k=1}^{n} (\mathbf{A})_{mk} (\mathbf{X})_{kl} (\mathbf{B})_{lp} \end{bmatrix} \\
&= \begin{bmatrix} (\mathbf{A})_{11}(\mathbf{B})_{11} & \cdots & (\mathbf{A})_{1n}(\mathbf{B})_{l1} & \cdots & (\mathbf{A})_{11}(\mathbf{B})_{o1} & \cdots & (\mathbf{A})_{1n}(\mathbf{B})_{o1} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ (\mathbf{A})_{m1}(\mathbf{B})_{11} & \cdots & (\mathbf{A})_{mn}(\mathbf{B})_{l1} & \cdots & (\mathbf{A})_{m1}(\mathbf{B})_{o1} & \cdots & (\mathbf{A})_{mn}(\mathbf{B})_{o1} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ (\mathbf{A})_{11}(\mathbf{B})_{11} & \cdots & (\mathbf{A})_{1n}(\mathbf{B})_{l1} & \cdots & (\mathbf{A})_{11}(\mathbf{B})_{o1} & \cdots & (\mathbf{A})_{1n}(\mathbf{B})_{o1} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ (\mathbf{A})_{m1}(\mathbf{B})_{11} & \cdots & (\mathbf{A})_{mn}(\mathbf{B})_{l1} & \cdots & (\mathbf{A})_{m1}(\mathbf{B})_{o1} & \cdots & (\mathbf{A})_{mn}(\mathbf{B})_{o1} \end{bmatrix} \begin{bmatrix} (\mathbf{X})_{11} \\ \vdots \\ (\mathbf{X})_{n1} \\ \vdots \\ (\mathbf{X})_{1o} \\ \vdots \\ (\mathbf{X})_{no} \end{bmatrix} \\
&= (\mathbf{B}^T \otimes A)\text{vec}(X)
\end{aligned}$$

□

Theorem 1.2.8. *For a given* $M \in \text{Stab}(E)$ *and* $C \in \text{PD}_p$*, solving the Lyapunov equation (1.2.1) for* Σ *is equivalent to solving its vectorized form*

$$((I_p \otimes M) + (M \otimes I_p))\text{vec}(\Sigma) = -\text{vec}(C) \tag{1.7}$$

where $I_p \in \mathbb{R}^{p \times p}$ *is the identity matrix.*

Proof. One can vectorize and rewrite the Lyapunov equation as

$$\begin{aligned}
\text{vec}(M\Sigma I_p + I_p \Sigma M^T + \text{vec}(C)) &= \text{vec}(M\Sigma I_p) + \text{vec}(I_p \Sigma M^T) + \text{vec}(C) \\
&= ((I_p \otimes M) + (M \otimes I_p))\text{vec}(\Sigma) + \text{vec}(C) = 0
\end{aligned}$$

using Lemma 1.2.7 □

Remark 1.2.9. *One can similarly show that the Lyapunov equation could be rewritten alternatively in a vectorized form to solve for M.*

$$\begin{aligned}\text{vec}(I_p M\Sigma + \Sigma M^T I_p + \text{vec}(C)) &= \text{vec}(I_p M\Sigma) + \text{vec}(\Sigma M^T I_p) + \text{vec}(C)\\ &= (\Sigma^T \otimes I_p)\text{vec}(M) + (I_p \otimes \Sigma)\text{vec}(M^T) + \text{vec}(C)\end{aligned}$$

One can then use the commutation matrix defined in 1.2.6

$$A(\Sigma)\text{vec}(M) = -\text{vec}(C) \tag{1.8}$$

where $A(\Sigma) := \left((\Sigma^T \otimes I_p) + (I_p \otimes \Sigma)K_{(p^2,p^2)}\right) \in \mathbb{R}^{p^2\times p^2}$

Under the provided assumptions, once M and C are known, one can uniquely compute elements of Σ by solving the linear equation 1.7. In particular, one requires the condition that no two eigenvalues of M sum to zero. However, given a covariance matrix, Σ, a corresponding pair of (M, C) is not uniquely determined. In other words, there can multiple pairs of (M, C) that can give the same covariance matrix. Since M can be asymmetric, it cannot be recovered without any additional assumptions on its structure, even when C is assumed known. It was shown by Dettling et al. [Det+22] that M can be uniquely recovered, provided that the underlying graph does not have cycles of length two for a diagonal volatility matrix. The problem of recovery of M can be viewed based on the solvability of equation (1.8). M can be recoverable, if and only if $\det(A(\Sigma)) \neq 0$ is satisfied for all $\Sigma \in \mathcal{M}_{G,C}$ or if and only if there exists such a $\Sigma \in \mathcal{M}_{G,C}$ that satisfies it. However, when there are two-cycle, then this condition is not satisfied.

Provided this identifiability condition is satisfied, the following question arises then, how can one recover the support of M efficiently given covariance matrix, Σ and diagonal $C \in \text{PD}_p$. In the following sections, a brief review of few popular methods for support recovery are discussed.

1.3 Brief overview on best-subset selection methods

A linear regression model consists of a response vector $y \in \mathbb{R}^N$, model matrix $X \in \mathbb{R}^{N\times P}$, regression coefficients $\beta \in \mathbb{R}^P$ and errors $\epsilon \in \mathbb{R}^N$ such that $y = X\beta + \epsilon$. Throughout this study, it is assumed that the columns of X have zero mean and unit variance and if not, one standardizes them before. The goal of the best-subset selection problem is to identify the best k features $\beta_i, i \in [N]$ that fit the data to the response vector y.

Definition 1.3.1. *The best subset problem can be stated as*

$$\min_{\beta\in\mathbb{R}^P} \frac{1}{2}\|X\beta - y\|^2 \quad \textit{subject to} \quad \|\beta\|_0 \leq k \tag{1.9}$$

The (pseudo) norm on the vector β is referred to as l_0 and it counts the number of non-zeroes in the β. It is formally defined as $\|\beta\|_0 = \sum_{i=1}^P \mathbb{I}(\beta_i \neq 0)$, where $\mathbb{I}(\cdot)$ denotes the indicator function. This is a non-convex problem and it is one of the most significant NP-hard problems in Computational Statistics [Wel82]. Classical method of best subset selection [BKM67; HL67] involves exploring different possible combinations of regressions and comparing them using a suitable criterion to obtain the optimal solution. This approach is often computationally expensive and is not scalable to large problem size. Forward step selection [DS98; Efr66], on the other hand, uses a greedy approach. The algorithm starts with an empty set $A_0 = 0$, it iteratively selects a variable for $k = 1, \ldots, \min\{N, P\}$ by choosing the index j_k such that X_{j_k} achieves the maximum correlation with y after one removes the contributions from $X_{A_{k-1}}$. However, the solution obtained may not guaranteed to be globally optimum out of all possible subsets.

There have been numerous efforts in solving this problem by weakening the underlying assumptions and obtaining an approximate solution. One could use traditional Ordinary Least Squares (OLS) regression, for example, by not considering the constraint in (1.9). However, the solution obtained is not interpretable as it does not really select features with the strongest effect, but instead gives large number of predictors contributing small effects each. The more popular approach has been to use convex optimization methods such as lasso [Tib96]. The problem uses an l_1 penalty instead of l_0.

Definition 1.3.2. *The lasso problem can be stated as*

$$\min_{\beta \in \mathbb{R}^p} \frac{1}{2} \|X\beta - y\|^2 + \lambda \|\beta\|_1 \tag{1.10}$$

When the underlying model is reasonably sparse, the solutions obtained by lasso are usually sparse because it shrinks many predictor coefficients towards zero and thus are more interpretable. Furthermore, it can handle high-dimensional data considerably well, which is frequently encountered problem in statistical applications. Under certain regularity conditions on X, it is also shown to be model selection consistent [ZY06], meaning that the probability of the lasso solution being equal to the true predictors is 1. Since the problem is convex, it is computationally very efficient and problems of huge size can be solved considerably well by very efficient solvers with good predictive performance [Fri+07; Nes13]. Its performance seems to quickly degrades as a variable selector when such regularity conditions are not met or if there is high noise in the data. In such cases, the use of non-convex penalized regression have shown some potential [MFH11]. Nevertheless, even with its shortcomings, lasso and its derivatives are still considered powerful and are widely used for different variable selection applications [Guo+15; Moz+22; Pfe+22].

In the next section, attempts of support recovery of the drift matrices arising in graphical Lyapunov models via lasso are discussed.

1.4 Background on Direct Lyapunov Lasso

Fitch [Fit19], Varando and Hansen [VH20a] and Dettling et al. [DDK22] approached support recovery of the drift matrix via l_1-regularization method. Given an i.i.d sample consisting of centered observations $X_1, \ldots, X_n \in \mathbb{R}^p$, its sample covariance matrix is given by

$$S = \Sigma^{(n)} = \frac{1}{n} X^T X. \tag{1.11}$$

The *Direct Lyapunov Lasso* converts the Lyapunov equation (1.8) as a lasso problem to find a sparse estimate of M.

Definition 1.4.1. *The Direct Lyapunov Lasso aims to solve the following convex optimization problem*

$$\min_{\mathrm{vec}(M) \in \mathbb{R}^P} \frac{1}{2} \|A(S)\mathrm{vec}(M) + \mathrm{vec}(C)\|_2^2 + \lambda \|\mathrm{vec}(M)\|_1, \tag{1.12}$$

where $P := p^2$ for convenience and $\lambda > 0$ is a tuning parameter.

Definition 1.4.2. *The **Gram matrix** is defined as*

$$\Gamma(\Sigma) := A(\Sigma)^T A(\Sigma) \in \mathbb{R}^{N \times P} \tag{1.13}$$

and the linear vector,

$$g(\Sigma) := -A(\Sigma)^T \mathrm{vec}(C) \in \mathbb{R}^P \tag{1.14}$$

Remark 1.4.3. *The Direct Lyapunov Lasso problem (1.12) can be further reformulated as*

$$\min_{\text{vec}(M)\in\mathbb{R}^p} \frac{1}{2}\text{vec}(M)^T\Gamma(S)\text{vec}(M) - g(S)^T\text{vec}(M) + \lambda\|\text{vec}(M)\|_1. \tag{1.15}$$

Dettling et al. [DDK22] show that, under some regularity assumptions on the gram matrix $\Gamma(S)$, it is possible to recover the support of the true population drift matrix, M^*. In particular, the following theorem plays the central role in providing a deterministic condition for the support recovery from Direct Lyapunov Lasso.

Theorem 1.4.4 (Dettling et al. [DDK22])**.** *Let $M^* \in \text{Stab}(E)$ be the true drift matrix, and let S be its support. Assume that Γ^*_{SS} is invertible and that the irrepresentable condition*

$$\|\Gamma^*_{S^cS}(\Gamma^*_{SS})^{-1}\text{sign}(\text{vec}(M^*))_S\|_\infty \leq (1-\alpha) \tag{1.16}$$

holds with parameter $\alpha \in (0,1]$. Furthermore, assume that $\widehat{\Gamma}$ is a matrix such that

$$|||(\Delta_\Gamma)_{\cdot S}|||_\infty < \epsilon_1, \quad \|\Delta_g\|_\infty < \epsilon_2$$

with $\epsilon_1 \leq \alpha/(6c_{\Gamma^*}$. Then if

$$\lambda > \frac{3(2-\alpha)}{\alpha}\max\{c_M\epsilon_1, \epsilon_2\}$$

then $\widehat{M}$ obtained as a solution to the Direct Lyapunov Lasso, has its supported included in the true support ($\widehat{S} \subseteq S$), and satisfies

$$\|\widehat{M} - M^*\| < \frac{c_{\Gamma^*}}{2-\alpha}\lambda.$$

In addition, if

$$\min_{1\leq j<k<p,(j,k)\in S} |M^*_{jk}| > \frac{c_{\Gamma^*}}{2-\alpha}\lambda,$$

then $\widehat{S} = S$ and $\text{sign}(\widehat{M}_{jk}) = \text{sign}(M^*_{jk})\forall(j,k) \in S$.

Proof. See details in Appendix A.1 [DDK22]. □

Thus, the irrepresenatable condition becomes a necessary condition for support recovery. In practice, it is not satisfied so frequently in non-simple graphs (graph with cycles). Even in simple graphs (graph without cycles), the frequency of graphs actually satisfying this condition decreases as the number of edges increases. It then becomes very challenging to recover the true support via Direct Lyapunov Lasso. In the following sections, a new approach to support recovery via mixed integer optimization is discussed, which will be the main theme of this study.

1.5 Background on Mixed Integer Optimization (MIO)

The general form of Mixed Integer Quadratic Problem (MIQP) with binary variables is given by

$$\begin{aligned} &\min\ x^TQx + q^Tx \\ &\text{s.t } Ax \leq b, \\ &c_l \leq x \leq c_u, \\ &x_i \in \{0,1\} \quad i \in \mathcal{I} \end{aligned} \tag{1.17}$$

where $q \in \mathbb{R}^m, A \in \mathbb{R}^{k\times m}, b \in \mathbb{R}^k$ and $Q \in \mathbb{R}^{m\times m}$ are parameters of the given problem. The goal is to optimize the objective function over $x \in \mathbb{R}^m$, containing binary ($x_i, i \in \mathcal{I}$) and continuous ($x_i \notin \mathcal{I}$) variables, with $\mathcal{I} \subset \{1, \ldots, m\}$. Typically, such problems are approached via branch and bound method due to their remarkable performance [FL98]. It uses bounds on optimal cost to avoid exploring the entire feasible set to find an optimal solution.

Definition 1.5.1. *Given a mixed integer optimization problem like (1.17), its linear relaxation is defined as*

$$\begin{aligned} &\min\ x^T Q x + q^T x \\ &s.t\ Ax \leq b, \\ &c_l \leq x \leq c_u, \\ &0 \leq x_i \leq 1 \quad i \in \mathcal{I} \end{aligned} \tag{1.18}$$

where the requirement that x_i for $i \in \mathcal{I}$ to be binary variables is relaxed.

Let $\mathcal{F}$ be the set of feasible solutions to the problem (1.17). A linear relaxation of this problem, (1.18) is solved. If the optimal solution of this relaxation lies in $\mathcal{F}$, then the it is quite straightforward to see that it is also an optimal solution to problem (1.17). If not, $\mathcal{F}$ is divided into a finite collection of subsets $\mathcal{F}_1, \ldots, \mathcal{F}_k$ by partitioning on a variable that is *fractional*, x'_j for $j \in \mathcal{I}$ and not an integer. This is done for a single variable x_j by dividing the feasible set into two parts, one with a constraint $x_j \leq \lfloor x'_j \rfloor$ and one with $x \geq \lceil x'_j \rceil$. One then formulates k such subproblems (for $k/2$ fractional variables)

$$\begin{aligned} &\min\ x^T Q x + q^T x \\ &\text{s.t}\ x \in F_i, \quad i = 1, \ldots, k \end{aligned} \tag{1.19}$$

and solve each one separately following a similar procedure. Note that each subproblem can be splitted into further subproblems. This step is called *branching* and it leads to a tree of subproblems (see Figure 1.1) where each subproblem is represented as a node.

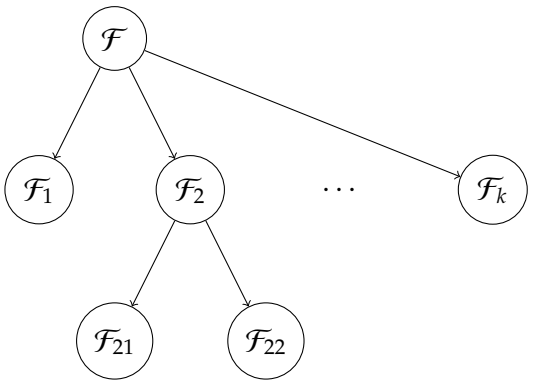


Figure 1.1 *Branching*: A tree of subproblems by partitioning the feasible set $\mathcal{F}$

If a particular node gives an infeasible solution or an integer feasible solution, it is designated as being *fathomed* and it is excluded from further branching. In the latter case, if it is the best integer solution obtained so far, it is further denoted as *incumbent*. The incumbent solution serves as an upper bound U to the optimal objective function of problem (1.17). At the beginning of the algorithm, U is taken as ∞ or a really large number and it is only updated when an incumbent solution is found at a given node. The nodes are also assigned as fathomed if the optimal solution found provides an objective greater than the current incumbent.

Furthermore, it is not always necessary to directly solve a subproblem at a given node to check if it is to be fathomed. One can also assign a node as fathomed by first checking if the lower

bound L_i (also called *best bound*) of that particular subproblem satisfies $L_i \geq U$. Then that node can be eliminated completely from further exploration. In many cases, computing a lower bound might actually be easier than computing the optimal solution to the subproblem. There are several ways of computing the lower bounds. The most straightforward method is to take L as the optimal objective of the linear relaxation of subproblem. Another more common approach is to use concepts of Duality theory in integer optimization [BW05]. The optimal solution to the dual problem provides a lower bound to the primal problem. If $L \leq U$ for a subproblem, an optimal solution is obtained or the subproblem is partitioned into further subproblems based on the linear relaxation solution. The difference between the current incumbent and the best bound serves as an optimality certificate and is called as *gap*. The optimal solution is found when the gap becomes zero. One of the biggest advantage of MIO solvers is that even though it stops before approaching the global optimum, the lower bounds gives one the certificate of suboptimality.

MIO solvers have become very powerful in recent years due to integration of powerful techniques. *Cutting planes* allows to use cuts while solving the subproblems. By introducing additional constraints (plane) that do not change the existing formulation, one can accelerate branch and bound considerably. Addition of cutting planes is one of the biggest factor contributing to the speed up of MIO solvers [Bix+04]. The second most biggest factor is *Presolving* [Ach+20; Bix+04]. Presolving is a preprocessing technique where the problem size is reduced before actually solving it. The reduction is done by identifying redundant information in the formulation, for example constraints that are only satisfied for certain values of x_j for $j \in \mathcal{F}$. There are several ways to perform the branching. Two most extreme cases are "depth-first search" and "breadth-first search". However, choosing a smart branching method has shown significant improvements in speed and solvability of MIO problems [AW13]. There are many more ingredients that have contributed to improvements in the solvers over the years and it is expected to get only more powerful in the future.

Bertismas et al. [BKM16] presented a novel approach of solving the best-subset selection problem (1.9) via mixed integer optimization. While the classical best-subset methods barely could manage to recover problems of size $p = 30$, they showed that highly optimized MIO solvers, like Gurobi [Gur23a], could handle problems of size in thousands. Furthermore, it was observed that MIO outperformed lasso with a good predictive performance. Tibshirani et al. [HTT17b] extended the results to more realistic signal to noise ratio regimes and performed a detail comparison of best-subset selection, lasso, Relaxed lasso and forward stepwise selection. It was observed that at low signal to noise ratio regimes, lasso outperformed best-subset selection and relaxed lasso consistently performed as well as, or better than, all the methods. This study aims to extend this comparison, in particular with lasso and best-subset selection for support recovery of drift matrices associated with graphical Lyapunov models.

1.6 Best Subset as a MIO Problem

Bertismas et al. [**Bertismas**] developed MIO formulations for the best-subset problem (1.9). It was shown, in particular, that the formulations could provide optimum solutions atleast as strong as lasso (1.3), if not better. The choice of a formulation is crucial in integer optimization.

Definition 1.6.1 (Convex hull)**.** *Consider a set $\mathcal{A} = \{x_1, \ldots, x_m\}$, where x_i represent points in $\mathbb{R}^n$ for $i = 1, \ldots, m$. The set $\mathcal{A}$ is convex if for every $x_i, x_j \in \mathcal{A}$, $(1-t)x_i + tx_j \in \mathcal{A}$ and $t \in [0, 1]$. Furthermore, the convex hull of the set $\mathcal{A}$ is defined as*

$$\operatorname{conv}(\mathcal{A}) := \left\{ \sum_{i=1}^{m} \lambda_i x_i \,\middle|\, \sum_{i=1}^{m} \lambda_i = 1, \lambda_i \geq 0 \right\}$$

and it is the smallest convex set that contains all the points in $\mathcal{A}$.

Theorem 1.6.2. *Let $\mathcal{F}$ be feasible set to problem (1.17), bounded and, therefore, finite. Then one could solve (1.17) by finding an extreme point solution to the following linear optimization problem*

$$\begin{aligned} &\min\ x^T Q x + q^T x \\ &s.t\ x \in \text{conv}(\mathcal{F}) \end{aligned}$$

Proof. The proof follows from the definition of convex set. The set $\text{conv}(\mathcal{F})$ is a polyhedron that contains all the integer feasible solutions in $\mathcal{F}$ as its extreme points. □

Remark 1.6.3. *So it is usually desirable to have a formulation that has its linear relaxation to be the convex hull* $\text{conv}(\mathcal{F})$*. However, this is often very difficult in practice. So one often tries to construct a polyhedron that closely approximates* $\text{conv}(\mathcal{F})$*. If P is the feasible set of linear relaxation, it satisfies* $\text{conv}(\mathcal{F}) \subset P$*. If Z_P and Z_{IP} are the optimum cost of the linear relaxation and the integer optimization problem respectively, then this would mean $Z_P \leq Z_{IP}$. Thus, the linear relaxation gives a lower bound to the optimum integer solution objective. Closer P is to the convex hull, better the lower bound.*

Here, in this study, the following two formulations for solving the best-subset selection (1.9) are considered from [BKM16].

Definition 1.6.4. ***Form-1*** *reformulation of the best-subset problem (1.9) is described as*

$$\begin{aligned} Z_{F1-IP} = \min_{\beta,z}\ & \frac{1}{2}\beta^T(X^TX)\beta - \langle X^T y, \beta\rangle \\ s.t\quad & -M_U z_i \leq \beta_i \leq M_U z_i, \qquad i = 1,\dots,P, \\ & z_i \in \{0,1\}, \qquad i = 1,\dots,P, \\ & \sum_{i=1}^{P} z_i \leq k, \end{aligned} \tag{1.20}$$

where $\beta \in \mathbb{R}^P, z \in \{0,1\}^P$ are optimization variables. M_U is a problem specific parameter that provides an upper bound on the absolute value of the elements of β : z is a binary variable that defines the sparsity of the solution β. So if $z_i = 1$, then $|\beta_i| \leq M_U$ and if $z_i = 0$, then $\beta_i = 0$.

Definition 1.6.5. ***Form-2*** *reformulation of the best-subset problem (1.9) is described as*

$$\begin{aligned} Z_{F2-IP} = \min_{\zeta,\beta,z}\ & \frac{1}{2}\zeta^T\zeta - \langle X^T y, \beta\rangle \\ s.t\quad & \zeta = X\beta, \\ & -M_U z_i \leq \beta_i \leq M_U z_i, \qquad i = 1,\dots,P \\ & z_i \in \{0,1\}, \qquad i = 1,\dots,P \\ & \sum_{i=1}^{p} z_i \leq k, \qquad i = 1,\dots,P \\ & -M_U^{\zeta} \leq \zeta_i \leq M_U^{\zeta} \qquad i = 1,\dots,P, \end{aligned} \tag{1.21}$$

where $\zeta \in \mathbb{R}^N, \beta \in \mathbb{R}^P, z \in \{0,1\}^P$ are optimization variables. M_U and M_U^{ζ} are problem specific parameters that provides an upper bound on the absolute value of the elements of β and $X\beta$ respectively : z is a binary variable that defines the sparsity of the solution β. So if $z_i = 1$, then $|\beta_i| \leq M_U$ and if $z_i = 0$, then $\beta_i = 0$.

Let $\mathcal{F}_{F1}$ and $\mathcal{F}_{F2}$ be the feasible sets, containing integer feasible solutions to Form-1 and Form-2 respectively. One can see that

$$\begin{aligned}\text{conv}(\mathcal{F}_{F2}) &= \{\beta : \|\beta\|_\infty \le M_U, \|\beta\|_1 \le M_U k, \|X\beta\|_\infty \le M_U^\zeta\} \\ &\subseteq \text{conv}(\mathcal{F}_{F1}) \\ &= \{\beta : \|\beta\|_\infty \le M_U, \|\beta\|_1 \le M_U k\} \subseteq \{\beta : \|\beta\|_1 \le M_U k\}.\end{aligned}$$

The formulation is constructed in such a way that the feasible sets of linear relaxation of both formulations are equal to their respective convex hulls. Thus both formulations are strong. However, the convex hulls satisfy $\text{conv}(\mathcal{F}_{F2}) \subseteq \text{conv}(\mathcal{F}_{F1})$, indicating that Form-2 is **atleast as strong as** Form-1. Let Z_{F1} and Z_{F2} be the optimum objective value of the linear relaxation of Form-1 and Form-2 respectively. Then, the optimum objective value of Form-1 is a lower bound to that of Form-2, i.e., $Z_{F1} \le Z_{F2}$. Furthermore, one can note that the minimum of both the problems are lower bounded by the optimum objective value of the following convex optimization problem:

$$Z_L := \min_\beta \frac{1}{2}\|X\beta - y\|_2^2 \quad \text{s.t} \quad \|\beta\|_1 \le M_u k \tag{1.22}$$

This is lasso in constrained form. Thus, we have $Z_L \le Z_{F1} \le Z_{F1-IP} \le Z_{F2} \le Z_{F2-IP}$. The MIO solver begins by first solving the relaxation problem and lasso formulation is weaker than this relaxation, irrespective of the formulation chosen.

Provided M_U and M_U^ζ are chosen appropriately, the solution to these formulations result in solution to the best-subset problem (1.9). So the choice of these parameters are crucial in determining good lower bounds in practice. Furthermore, it is often helpful if one provides the MIO solver an initial guess. This serves as an upper bound to the optimum objective value of the integer optimization problem and the more closer it is to the optimum solution, the faster MIO solver can find the global optimum. If one has an initial guess that is closer to the optimum solution, then they can also provide the MIO formulations with estimates for M_U and M_U^ζ. The discrete first-order methods, presented by Bertismas et al. [BKM16], provide good upper bounds to the best-subset problem.

Definition 1.6.6 (Convex function)**.** *A function $f(x)$ is convex on any interval $I = [a, b]$, if and only if*

$$f(tx + (1-t)y) \le tf(x) + (1-t)f(y) \tag{1.23}$$

$\forall\, x, y \in I$ and for any $t \in [0, 1]$

Consider the following optimization problem

$$\min_\beta g(\beta) \quad \text{s.t} \quad \|\beta\|_0 \le k \tag{1.24}$$

where, $g(\beta)$ is convex and is Lipschitz continuous, meaning

$$\|\nabla g(\beta) - \nabla g(\tilde{\beta})\| \le \mathcal{L}\|\beta - \tilde{\beta}\|. \tag{1.25}$$

with lipschitz constant $\mathcal{L}$.

Proposition 1.6.7 (Bertismas et al. [BKM16])**.** *If $\widehat{\beta}$ is an optimal solution to the problem*

$$\min_{\|\beta\|_0 \le k} \|\beta - c\|_2^2 \tag{1.26}$$

then,

$$\widehat{\beta}_i = \begin{cases} c_i, & i \in \mathcal{K}, \\ 0, & \text{otherwise}, \end{cases} \tag{1.27}$$

where, $\widehat{\beta}_i$ is the ith coordinate of $\widehat{\beta}$ and $\mathcal{K}$ is an index set containing the k largest (in absolute value) elements of c. The set of solutions can be represented by $\mathbf{H}_k(c)$, where $\mathbf{H}_k(\cdot)$ projects c such that $\widehat{\beta}$ retains the k largest elements (in absolute value) and assigns rest to zero (Hard thresholding [DJ94]).

Proposition 1.6.8 (Nesterov [Nes03])**.** *For a convex function $g(\beta)$ satisfying condition (1.25) and for any $L > \mathcal{L}$,*

$$g(\eta) \leq Q_L(\eta, \beta) := g(\beta) + \frac{L}{2}\|\eta - \beta\|_2^2 + \langle \nabla g(\beta), \eta - \beta \rangle \tag{1.28}$$

for all β, η with equality holding at $\beta = \eta$.

Definition 1.6.9. *A point η is called stationary point of the problem (1.24), if it satisfies a fixed point equation*

$$\eta \in \mathbf{H}_k\left(\eta - \frac{1}{L}\nabla g(\eta)\right) \tag{1.29}$$

The idea is to find a stationary point of the problem (1.24) when $g(\beta) = \|X\beta - y\|_2^2$ using discrete first order algorithm based on projected gradient descent.

Algorithm 1: Discrete first-order algorithm to find stationary point

Input: X, y, k, defaults: max iterations $N = 1000$, convergence tolerance ϵ = 1e-4
Output: β_{init}

Initialize $\beta_0 \in \mathbb{R}^P$ such that $\|\beta_0\| \leq k$;
Assign L as the largest eigenvalue of $X^T X$;
for $i = 1$ *to* N **do**
    $\beta_i = \mathbf{H_k}\left(\beta_{i-1} - \nabla g(\beta_{i-1})/L\right)$;
    Break if $\|\beta_i - \beta_{i-1}\|_2 / \max(\|\beta_i\|_2, 1) < \epsilon$
$\beta_{\text{init}} = \beta_i$

Once this algorithm finds a stationary point by selecting a support, then this solution is k-sparse and can be supplied as initial guess to the MIO solver for further improvement. The following theorem from Bertismas et al. [BKM16] guarantees that the algorithm is able to find a support that stabilizes as the algorithm progress and eventually converges to a stationary point.

Theorem 1.6.10. *Consider $g(\beta)$ and $\mathcal{L}$ as defined in (1.25). Let β_m, $m \geq 1$ be the sequence generated by Algorithm 1. Then*

a) For any $L \geq \mathcal{L}$, the sequence $g(\beta)$ is decreasing, converges and satisfies

$$g(\beta_m) - g(\beta_{m+1}) \geq \frac{L - \mathcal{L}}{2}\|\beta_{m+1} - \beta_m\|_2^2. \tag{1.30}$$

b) If $L > \mathcal{L}$, then $\beta_{m+1} - \beta_m \to 0$ as $m \to \infty$.

c) Let $\mathbf{1}_m$ represent the sparsity pattern of the support of β_m, $\alpha_{k,m} = \beta_{(k),m}$ and $\underline{\alpha}_k := \liminf_{m\to\infty} \alpha_{k,m}$. If $L > \mathcal{L}$ and $\alpha_k = 0$, then the sequence $\mathbf{1}_m$ converges after finitely many iterations. Furthermore, the sequence β_m is bounded and converges to a first-order stationary point.

d) If $L > \mathcal{L}$ and $\underline{\alpha}_k = 0$ then $\liminf_{m\to\infty} \|\nabla g(\beta_m)\| = 0$.

e) Let $\overline{\alpha}_k := \limsup_{m\to\infty} \alpha_{k,m}$ $L > \mathcal{L}, \overline{\alpha}_k = 0$ and suppose that the sequence β_m has a limit point. Then $g(\beta_m) \to \min_\beta g(\beta)$.

Proof. For clarity, only proofs of the statements a), b) and c) are shown here. For d) and e), please refer to [BKM15].

a) Let β be a vector satisfying $\|\beta\|_0 \leq k$. Let $\widehat{\eta} \in \mathbf{H}_k(\beta - \frac{1}{L}\nabla g(\beta))$,

$$\begin{aligned}
g(\beta) &= Q_L(\beta, \beta) \\
&\geq \inf_{\|\eta\|_0 \leq k} Q_L(\eta, \beta) \\
&= \inf_{\|\eta\|_0 \leq k} \left(\frac{L}{2}\|\eta - \beta\|_2^2 + \langle \nabla g(\beta), \eta - \beta \rangle + g(\beta) \right) \\
&= \inf_{\|\eta\|_0 \leq k} \left(\frac{L}{2}\|\eta - \left(\beta - \frac{1}{L}\nabla g(\beta)\right)\|_2^2 - \frac{1}{2L}\|\nabla g(\beta)\|_2^2 + g(\beta) \right) \\
&= \left(\frac{L}{2}\|\widehat{\eta} - \left(\beta - \frac{1}{L}\nabla g(\beta)\right)\|_2^2 - \frac{1}{2L}\|\nabla g(\beta)\|_2^2 + g(\beta) \right) \qquad \text{(From proposition 1.6.7)} \\
&= \left(\frac{L}{2}\|\widehat{\eta} - \beta\|_2^2 + \langle \nabla g(\beta), \widehat{\eta} - \beta \rangle + g(\beta) \right) \\
&= \left(\frac{L - \mathcal{L}}{2}\|\widehat{\eta} - \beta\|_2^2 + \frac{\mathcal{L}}{2}\|\widehat{\eta} - \beta\|_2^2 + \langle \nabla g(\beta), \widehat{\eta} - \beta \rangle + g(\beta) \right) \\
&= \frac{L - \mathcal{L}}{2}\|\widehat{\eta} - \beta\|_2^2 + Q_{\mathcal{L}}(\widehat{\eta}, \beta) \\
&\geq \frac{L - \mathcal{L}}{2}\|\widehat{\eta} - \beta\|_2^2 + g(\widehat{\eta}). \qquad \text{(From Proposition 1.6.8)}
\end{aligned}$$

This leads to

$$g(\beta) - g(\widehat{\eta}) \geq \frac{L - \mathcal{L}}{2}\|\eta - \beta\|_2^2 \tag{1.31}$$

$\beta_m = \beta$ and $\beta_{m+1} = \widehat{\eta}$ are two consecutive vectors of the sequence generated by Algorithm 1.Thus, for $L \geq \mathcal{L}$, $g(\beta_m)$ are decreasing and since $g(\beta)$ is bounded from below ($g(\beta) \geq 0$), it converges as $m \to \infty$.

b) Since $g(\beta_m)$ converges as $m \to \infty$, if $L > \mathcal{L}$ in (1.31), then $\|\beta_{m+1} - \beta_m\| \to 0$.

c) $\underline{\alpha}_k > 0$ means that $\|\beta_{(k),m}\| = k$ (all entries are bounded away from zero). Assume that the support, $\mathbf{1}_m$, does not converge as $m \to \infty$. Then there exists many values m' such that $\mathbf{1}_{m'} \neq \mathbf{1}_{m'+1}$. For i, j such that $\beta_{i,m'+1} = \beta_{j,m'} = 0$, one has

$$\|\beta_{m'} - \beta_{m'+1}\|_2 \geq \sqrt{\beta_{i,m'}^2 + \beta_{j,m'+1}^2} \geq \frac{|\beta_{i,m'}| + |\beta_{j,m'+1}|}{2}. \tag{1.32}$$

This contradicts the fact that $\|\beta_{m+1} - \beta_m\| \to 0$ as shown in b). Thus $\mathbf{1}_m$ converges after finitely many iterations M^*. Algorithm 1 is a standard gradient descent algorithm, restricted to the space $\mathbf{1}_m$ for $m \geq M^*$. A gradient descent algorithm for minimizing a convex function over a closed convex set leads to a sequence that converges [Nes13]. Thus β_m converges to β^*, a first-order stationary point and its boundedness follows from the convergence of β_m.

□

In practice, one runs Algorithm 1 multiple times by choosing different initial values and choose the solution with minimum objective. As mentioned before, this objective value will serve as a good upper bound to the MIO formulation and also gives a good estimate of the parameters M_U and M_U^ζ. If β_{init} denotes an estimate obtained by Algorithm 1, then $M_U := \tau\|\beta_{\text{init}}\|_\infty$ with τ being a multiplier greater than 1, for e.g., $\tau = 2$. For an estimate of M_U^ζ, the following result taken from Theorem 2.1 of [BKM16] is useful:

Theorem 1.6.11 (Bertismas et al. [BKM16]). *For any $k \geq 1$, any optimal solution, $\widehat{\beta}$, to the best-subset problem (1.9) satisfies*

$$\|X\widehat{\beta}\|_\infty \leq \left(\max_{i=1,\dots,N} \|\mathrm{x}_i\|_{1:k}\right) \|\widehat{\beta}\|_\infty, \tag{1.33}$$

where x_i, $i \in [N]$ denote the rows of X and $\|\mathrm{x}_i\|_{1:k}$ is the sum of the largest k values in x_i (in absolute value).

Proof.

$$\|X\widehat{\beta}\|_\infty \leq \max_{i\in[N]} |\langle \mathrm{x}_i, \widehat{\beta}\rangle| \leq \max_{i\in[N]} \|\mathrm{x}_i\|_{1:k} \|\widehat{\beta}\|_\infty$$

□

So a suitable estimate for the upper bound on $X\beta$ would be, $M_U^\zeta := (\max_{i=1,\dots,N} \|\mathrm{x}_i\|_{1:k})\, M_U$.

With this, one has all the ingredients to efficiently formulate and solve the best-subset problem. In this study, these techniques are used to convert the problem of recovery of drift matrices.

Definition 1.6.12. *The support recovery of the true drift matrix, M^* of a graphical continuous Lyapunov model of a directed graph, G can be written as the following best-subset selection problem:*

$$\min_{\mathrm{vec}(M)\in\mathbb{R}^p} \frac{1}{2}\|A(S)\mathrm{vec}(M) + \mathrm{vec}(C)\| \quad \text{s.t} \quad \|\mathrm{vec}(M)\|_0 \leq k \tag{1.34}$$

where S is the sample covariance matrix of a given set of n observations as described in (1.11), $A(S)$ is the matrix defined in (1.8), C is the volatility matrix (assumed known).

Remark 1.6.13. *The best-subset problem (1.34) can be further reformulated as*

$$\min_{\mathrm{vec}(M)\in\mathbb{R}^p} \frac{1}{2}\mathrm{vec}(M)^T\Gamma(S)\mathrm{vec}(M) - g(S)^T\mathrm{vec}(M) \quad \text{s.t} \quad \|\mathrm{vec}(M)\|_0 \leq k \tag{1.35}$$

In the next chapter, the details of this approach's implementation are elaborated. In particular, different initialization strategies were explored for the discrete first-order algorithm and modifications were made to the formulations to make it more suitable for recovery of true graph structures from graphical Lyapunov models. Furthermore, a detail comparison of this approach with Direct Lyapunov Lasso is done using numerical studies on synthetic and real datasets.

2 Structure Recovery

2.1 Initialization strategies

As shown in Bertismas et al. [BKM16], solutions obtained from discrete first-order method can serve as a good initial guess to MIO solver. Their combination with MIO result in better solutions when compared to the solutions obtained from either of the methods, when used independently. This approach was implemented by Hastie et al. [HTT17b] in the `bestsubset` R package. In this package, the discrete first-order method introduced in previous chapter is considered for the optimization problem (1.24). Algorithm 1 is initialized with fifty random values around least squares regression coefficients (when $P < N$) or marginal regression coefficients (when $P \geq N$). The idea is to pass each of them on to the algorithm to obtain a first-order stationary point. These initialiizations are not necessarily k-sparse as required by the algorithm, and one often needs a subset selection method as a pre-processing step. In the `bestsubset` package, the largest k values are chosen from each of these initializations. Out of the fifty converged solutions, the one with minimum objective value, β_{init}, was passed as an initial guess to the MIO solver.

To begin, this section will focus on the modifications made to the initialization strategy originally implemented by Hastie et al .[HTT17b]. Unlike their work, the aim is to not just recover the best-subset of size k of any vector. Here, one aims to recover drift matrices from graphical Lyapunov models. One considers, $g(\beta) = \|X\beta - y\|_2^2$, $X = A(\Sigma)$, $y = -\text{vec}(C)$ and $\beta = \text{vec}(M)$. In particular, these drift matrices are stable, meaning all the eigenvalues are negative and bounded away from zero. Let $\mathcal{D}$ represent an index set containing the indices of $\text{vec}(M)$ that correspond to the diagonal position in the drift matrix, M. While Algorithm 1 is running, the hard thresholding operator, $\mathbf{H}_k(\cdot)$ can assign $\beta_{i,m}$ to zero for $i \in \mathcal{D}$ and for some iteration m. This can happen if β_i for $i \in \mathcal{D}$ does not belong to the set of largest k elements (in absolute value). In order to handle this, the following optimization problem is considered

$$\min_{\beta} g(\beta) \quad \text{s.t} \quad \|\beta\|_0 \leq k, \|\beta_{\mathcal{D}}\|_0 \leq |\mathcal{D}| \tag{2.1}$$

where, $g(\beta)$ is convex and is Lipschitz continuous, meaning

$$\|\nabla g(\beta) - \nabla g(\tilde{\beta})\| \leq \mathcal{L}\|\beta - \tilde{\beta}\|. \tag{2.2}$$

with lipschitz constant $\mathcal{L}$. $\mathcal{D}$ is a fixed index set containing specific indices, $i \in [P]$, of the elements of $\beta \in \mathbb{R}^P$.

Definition 2.1.1. *The Selective Thresholding operator, $\mathbf{P}_k(\cdot) : \mathbb{R}^P \to \mathbb{R}^P$ is defined as*

$$\mathbf{P_k}(c) = \begin{cases} c_i, & i \in \mathcal{K} \\ c_j, & j \in \mathcal{D} \\ 0, & \text{otherwise} \end{cases} \tag{2.3}$$

where the set $\mathcal{K}$ and $\mathcal{D}$ are two disjoint index sets where $\mathcal{K}$ contains the largest $k - |\mathcal{D}|$ elements of the vector $c \in \mathbb{R}^P$ and $\mathcal{D}$ is fixed.

Lemma 2.1.2. *The Selective Thresholding operator (2.3) applied to $c \in \mathbb{R}^P$ gives the set of solutions to the following optimization problem:*

$$\min \|\beta - c\|_2^2 \qquad \text{s.t} \qquad \|\beta\|_0 \leq k, \|\beta_{\mathcal{D}}\|_0 \leq |\mathcal{D}| \tag{2.4}$$

Proof. It is sufficient to consider the case $|c_i| > 0$ for all $i \in [P]$. If $\widehat{\beta}$ is an optimal solution and $S = \{i : \widehat{\beta}_i \neq 0\}$. The objective function is given by, $\sum_{i \notin S} c_i^2 + \sum_{i \in S}(\beta_i - c_i)^2$. By choosing $\widehat{\beta}_i = c_i$ for $i \in S$, the second summation is zero. Since $\mathcal{D}$ is fixed, choosing the indices of the largest $k - |\mathcal{D}|$ elements of c in $\mathcal{K}$ is the only choice that will minimize $\sum_{i \notin S} c_i^2$. □

Definition 2.1.3. *A point η is called stationary point of problem (2.1), if it satisfies a fixed point equation*

$$\eta \in \mathbf{P}_k \left(\eta - \frac{1}{L} \nabla g(\eta) \right) \tag{2.5}$$

The goal is then to modify the algorithm such that it is able to find the stationary point of problem (2.1). In the first subsection, this algorithm is described. In the subsequent subsections, other initialization strategies that were explored in this study are described in detail. All of these strategies aim to provide the MIO solver with initial guesses that are as close as possible to the true solution.

2.1.1 Marginal Regression

The initialization approaches proposed by Bertismas et al. [BKM16] were poor for this problem because the initial values were drawn completely random. Hastie et al. [HTT17b] implemented a better initialization strategy. They draw all values around the least squares regression coefficients ($P < N$) or marginal regression coefficients ($P \geq N$). Thus the initial values contain some information about the true problem. This approach was slightly modified for the current problem. For convenience, this modified algorithm is stated below.

Algorithm 2: Projected Gradient with Marginal Regression

Input: X, y, k, defaults: number of initializations nruns = 50, max iterations $N = 1000$, convergence tolerance ϵ = 1e-4

Output: β_{init}

1 **if** $P < N$ **then**
2 $\quad \beta_0$ is least squares coefficients
3 **else**
4 $\quad$ Entries of β_0 obtained by marginal regression: $\beta_{0j} = \frac{\sum_{i=1}^n X_{ij} y_i}{\sum_{i=1}^n X_{ij}^2}$ for $j = 1, \ldots, P$;
5 $\beta_0^0 = \mathbf{P_k}(\beta_0)$;
6 Assign L as the largest eigenvalue of $X^T X$;
7 **for** $r = 1$ *to* nruns **do**
8 $\quad$ **for** $i = 1$ *to* N **do**
9 $\quad\quad \beta_i^r = \mathbf{P_k}\left(\beta_{i-1}^r - \nabla g(\beta_{i-1}^r)/L\right)$;
10 $\quad\quad$ Break if $\|\beta_i^r - \beta_{i-1}^r\|_2 / \max(\|\beta_i^r\|_2, 1) < \epsilon$
11 $\quad$ **if** β_i^r gives the best objective so far **then**
12 $\quad\quad \beta_{\text{init}} = \beta_i^r$;
13 $\quad$ Start the next run at a new location: $\beta_0^r = \beta_0^0 + 2 \cdot \texttt{runif}(p) \max(|\beta_0^0|, 1)$

Since here one mainly deals with the case where $P = N = p^2$, either of the methods in step 4 is an appropriate choice. In marginal regression, each covariate X_j is regressed individually on to Y to

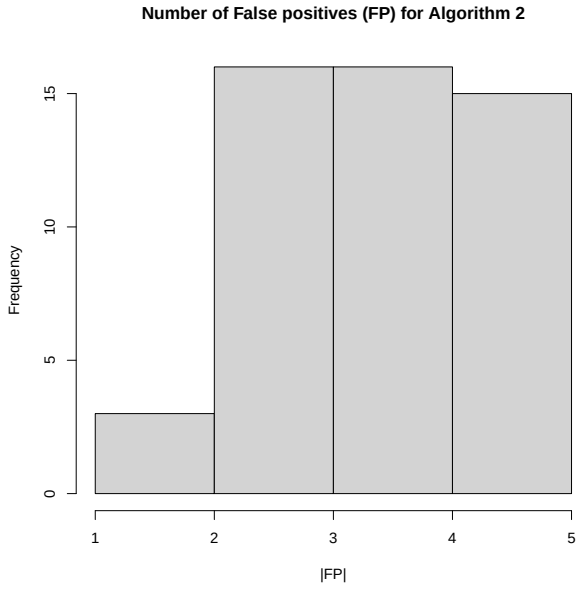


Figure 2.1 Number of false positives generated across all the initializations for a graph with $p = 5$ nodes and 10 true edges (including self-loops) using Algorithm 2

obtain β_j for $j \in [P]$. It was observed that the marginal regression performs better in estimating the strength of each predictor on the output variable without the influence of others and thus serves as a good starting point to identify the true edges of the graph. Furthermore, it is also simple and computationally efficient when compared to ordinary least squares regression.

Just like Algorithm 1, one can establish that Algorithm 2 finds a support that stabilizes and eventually converges to a stationary point as defined in (2.5).

Theorem 2.1.4. *Consider $g(\beta)$ and $\mathcal{L}$ as defined in (2.2). Let $\beta_m, m \geq 1$ be the sequence generated by Algorithm 2. Then all the statements of Theorem 1.6.10 also hold for Algorithm 2.*

Proof. Statements b), c), d) and e) are generic and follows if one proves that a) holds. For a), the result follows by taking infimum over the set $\|\eta\|_0 \leq k, \|\eta_{\mathcal{D}}\|_0 \leq |\mathcal{D}|$ (instead of just $\|\eta\|_0 \leq k$), and using Lemma 2.1.2 and Proposition 1.6.8 in a similar way as proof of Theorem 1.6.10. For more details see Bertismas et al. [BKM15] □

2.1.2 One Edge - Marginal Regression

Algorithm 2 consistently provides the MIO solver with an initial guess that is near $k-$sparse. However, it provides a solution with too many false positives (see Figure 2.1). This can be caused due to dense initializations to the algorithm. The primary objective of the algorithm is to eliminate any false positives while striving to identify the true positives. The proposed initialization strategy explores the algorithm's capability to reduce false positives by providing it with sparser initial guesses. It is similar to Algorithm 2, except that the initializations for the algorithm are always $p + 1$-sparse (p diagonal elements,i.e., self loops and 1 edge).

Definition 2.1.5. *Given a vector $c \in \mathbb{R}^P$, the one edge projector map $\mathbf{R}_k(\cdot) : \mathbb{R}^P \to \mathbb{R}^P$ is defined as*

$$\mathbf{R}_k(c) = \begin{cases} c_i, & i \in \mathcal{D} \\ c_i, & i = j \\ 0, & \text{otherwise} \end{cases} \tag{2.6}$$

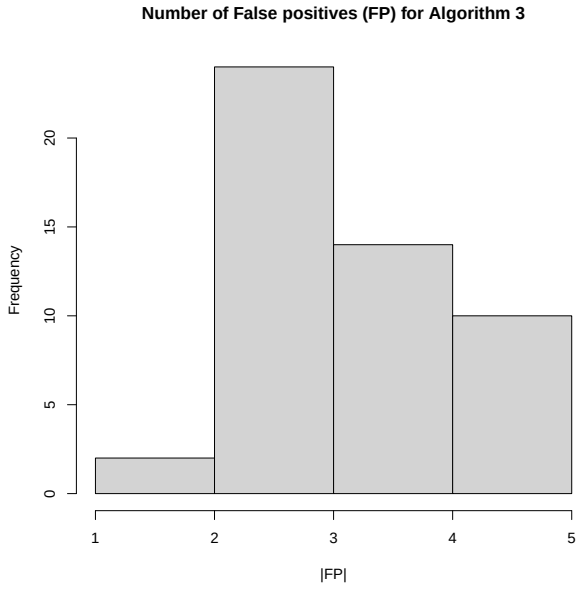


Figure 2.2 Number of false positives generated across all the initializations for a graph with $p = 5$ nodes and 10 true edges (including self-loops) using Algorithm 3

where j is a random number drawn from the set $\{j : j \in [P] \text{ and } j \notin \mathcal{D}\}$

Algorithm 3: Projected Gradient with Marginal Regression - One Edge Initialization

Input: X, y, k, defaults: Number of initializations nruns = 50, max iterations, $N = 1000$, convergence tolerance: ϵ = 1e-4

Output: β_{init}

if $P < N$ **then**
  β_0 is least squares coefficients
else
  Entries of β_0 obtained by marginal regression: $\beta_{0j} = \frac{\sum_{i=1}^{n} X_{ij} y_i}{\sum_{i=1}^{n} X_{ij}^2}$ for $j = 1, \ldots, P$;
$\beta_0^0 := \mathbf{R}_{p+1}(\beta_0)$;
Assign L as the largest eigenvalue of $X^T X$;
for $r = 1$ *to* nruns **do**
  for $i = 1$ *to* N **do**
    $\beta_i^r = \mathbf{P_k}\left(\beta_{i-1}^r - \nabla g(\beta_{i-1}^r)/L\right)$;
    Break if $\|\beta_i^r - \beta_{i-1}^r\|_2 / \max(\|\beta_i^r\|_2, 1) < \epsilon$
  if β_i^r gives the best objective so far **then**
    $\beta_{\text{init}} = \beta_i^r$;
  Start the next run at a new location by choosing another edge:
  $\beta_0^r = \mathbf{R_{p+1}}(\beta_0^0 + 2 \cdot \texttt{runif}(p) \max(|\beta_0^0|, 1))$

For the sake of completeness, this algorithm's performance in terms of TPs is shown in Figure 2.2.

2.1.3 Graphical Lasso informed Marginal Regression

The utilization of the graphical lasso model [FHT08] for inverse covariance estimation has gained significant popularity. The `glasso` R package gives a sparse estimate of the inverse covariance matrix by applying a lasso penalty on it. Given a data from a multivariate normal distribution of

dimension p, mean μ and covariance Σ, let $\Theta = \Sigma^{-1}$ and S be the empirical covariance matrix. The problem is to given by,

$$\max_{\Theta \in \mathbb{R}^{p \times p}} \log\det\Theta - \operatorname{tr}(S\Theta) - \rho\|\Theta\|_1; \tag{2.7}$$

where tr denotes the trace. Using a coordinate descent procedure for lasso, the inverse covariance matrix is estimated for a path of regularization parameter, ρ. The structure with a lowest Bayesian Information Criterion (BIC) value (further details provided in a subsequent section) is selected. The proposed initialization strategy uses the information obtained from this structure to construct an initial guess for the MIO solver. As mentioned before, this can be useful as the inverse covariance encodes relationships between variables.

Definition 2.1.6. *Let $\widehat{\Sigma}^{-1}$ represent an estimate of the inverse covariance matrix and $\mathcal{G}$ as the index set that contains the indices of the elements that are non-zero in* $\operatorname{vec}(\widehat{\Sigma}^{-1})$. *Given a vector $c \in \mathbb{R}^P$, the graphical lasso projector map $\mathbf{G}_k(\cdot) : \mathbb{R}^P \to \mathbb{R}^P$ is defined as*

$$\mathbf{G}(c) = \begin{cases} c_i, & i \in \mathcal{G} \\ 0, & \text{otherwise} \end{cases} \tag{2.8}$$

Algorithm 4: Projected Gradient with Marginal Regression - Graphical Lasso Informed

Input: X, y, k, sample covariance S, number of observations n, defaults: Number of initializations nruns = 50, max iterations, $N = 1000$, convergence tolerance: ϵ = 1e-4

Output: β_{init}

if $P < N$ **then**
  β_0 is least squares coefficients
else
  Entries of β_0 obtained by marginal regression: $\beta_{0j} = \frac{\sum_{i=1}^n X_{ij} y_i}{\sum_{i=1}^n X_{ij}^2}$ for $j = 1, \ldots, P$;
With `glassopath`(S), Choose minimum $\rho_{\min} = \rho_j, j = 1, \ldots, 10$ such that $\widehat{\Sigma}_j^{-1}$ is diagonal. ;
Compute $\widehat{\Sigma}^{-1}$ along a path containing 100 values uniformly spaced within $[\rho_{\min}/10, \rho_{\min}]$;
Choose $\widehat{\Sigma}_j^{-1}$ with best BIC score;
$\beta_0^0 = \mathbf{P}_k(\mathbf{G}(\beta_0))$ Assign L as the largest eigenvalue of $X^T X$;
for *r = 1 to* nruns **do**
  for *i = 1 to N* **do**
    $\beta_i^r = \mathbf{P_k}\left(\beta_{i-1}^r - \nabla g(\beta_{i-1}^r)/L\right)$;
    Break if $\|\beta_i^r - \beta_{i-1}^r\|_2 / \max(\|\beta_i^r\|_2, 1) < \epsilon$
  if β_i^r gives the best objective so far **then**
    $\beta_{\text{init}} = \beta_i^r$;
  Start the next run at a new location: $\beta_0^r = \mathbf{P}_k(\mathbf{G}(\beta_0^0 + 2 \cdot \texttt{runif}(p) \max(|\beta_0^0|, 1)))$.

2.1.4 Direct Lasso Initialization

The Direct Lyapunov lasso has demonstrated considerable potential for support recovery. Consequently, it becomes intriguing to examine whether the solutions obtained through this method can be enhanced when used as initial guesses for the Mixed-Integer Optimization (MIO) solver. Note that solutions from lasso, despite weak, form still valid lower bounds to the best-subset problem. The focus lies on minimizing the objective of Direct Lyapunov Lasso (1.12).

To compute solutions, `glmnet` package was employed along a path of values for the regularization parameter, λ. The regularization parameters are chosen similarly as proposed by Varando and Hansen [VH20a]. The selection of λ values ensures that the diagonal entries of the recovered drift matrices are non-zero. Initially, a coarser grid of λ values is employed to identify a suitable range where this condition holds. Subsequently, the lasso method is applied over a finer grid of λ values within this determined region. In summary, the algorithm can be outlined as follows:

Algorithm 5: Direct Lasso initialization

Input: X, y, k

Output: β_{init}

1 Construct grid_i by taking 100 uniformly spaced values between 10 and 10^{-5}.
2 Compute $\beta_{\text{lasso}-j}$ for $\lambda_j \in \text{grid}_i$ and $j = 1, \ldots, 100$, with no penalty on diagonal using `glmnet`.
3 Choose $\lambda_j \in \text{grid}_i$ such that $\beta_{\text{lasso}-j}$ is diagonal.
4 Construct grid_f by taking 100 uniformly spaced values between λ_j and $\lambda_j/10^4$.
5 Compute $\beta_{\text{lasso}-l}$ for $\lambda_l \in \text{grid}_f$ and $l = 1, \ldots, 100$ similar to step 3.
6 Choose l such that $g(\mathbf{P_k}(\beta_{\text{lasso}-l}))$ is minimum for $l = 1, \ldots, 100$. $\beta_{\text{init}} = \mathbf{P_k}(\beta_{\text{lasso}-l})$

2.2 Optimization Problem

The mathematical framework proposed by Bertsimas et al. [BKM16] serves as a foundation for all the modeling in this study. However, it was slightly adapted to address the problem at hand. In particular, as described before, the recovery of the drift matrix was converted to a best-subset problem (1.34). For convenience, it is rewritten below:

$$\min_{\text{vec}(M)\in\mathbb{R}^P} \frac{1}{2}\|A(S)\text{vec}(M) + \text{vec}(C)\| \quad \text{s.t} \quad \|\text{vec}(M)\|_0 \leq k \tag{2.9}$$

This problem is equivalent to (1.9) and can be solved by the MIO techniques introduced in section 1.6. However, it is important to address some challenges when using the formulations (1.20, 1.21) in this work directly. In practical scenarios, it is common to lack knowledge of the true graph structure and, therefore, one resorts to estimating the covariance structure for evaluating model performance. This estimation is useful in computing likelihood or information criterion or other similar model selection procedures. For unique estimation of Σ through (1.7), the sum of any two eigenvalues of M should be non-zero. Since all the problems to be considered in this study have stable drift matrix , therefore the true edge set E will always contain all self-loops $i \to i$ and all the diagonal entries will be negative. The assumption of stable drift matrix makes this estimation possible. However, the formulations do not guarantee that the obtained solution correspond to a stable drift matrix. There are two problems with the current formulations that could be of hindrance to the estimation of Σ:

1. The constraint $-M_U z_i \leq \beta_i \leq M_U z_i$ ensures that at most one of the variables (β_i or z_i) is non-zero. So it does not exclude the case where both the variables can be zero simultaneously. This case can be undesirable especially when $\beta_i = z_i = 0$ for $i \in \mathcal{D}$. In this scenario, unique estimation of Σ from the MIO solution would no longer be possible.

2. Apart from the constraints, it is always a good practice to provide the bounds for each optimization variable separately as it provides the MIO solver with search space. For β, the bounds were defined as:

   $$-M_U \leq \beta_i \leq M_U \qquad i \in [P].$$

   Note that the bounds on β is inclusive of zero. This would mean that β_i for $i \in \mathcal{D}$ are allowed to possibly be zero (and $z_i = 1$). Furthermore, even if such an element is non-zero, there is a

possibility that the solution is such that the eigenvalues of the solution are zero or close to zero. This again leads to numerical issues while computing Σ.

To address the aforementioned issues, it was necessary to modify and make the formulations problem-specific. The core reason to both the problems discussed above is that the upper bound of the diagonal entries of the solution is too loose for this particular problem. One needs a tighter upper bound for the diagonal elements to restrict their solution space.

$$-M_U z_i \leq \beta_i \leq -M_{U-\text{upper}} z_i \qquad i \in \mathcal{D} \tag{2.10}$$

where M_U is as defined in section 1.6 and $M_{U-\text{upper}}$ is defined as

$$M_{U-\text{upper}} := \max\left(\min(|\beta_{\text{init},\mathcal{D}}|), \epsilon\right) \tag{2.11}$$

ϵ is a small positive but bounded away from zero and $\beta_{\text{init},\mathcal{D}} \in \mathbb{R}^p$ is a vector excluding the elements present in diagonal position. Although all the initialization algorithms were designed to preserve the diagonal elements as much as possible, this condition further ensures that the solution to the diagonal elements does not include zero and are strictly away from zero along $\mathbb{R}_{<0}$.

Furthermore, as discussed in Bertismas et al. [BKM16], the bounds on β and ζ are not necessary, but if provided, they improve the strength of the MIO formulation. This was also observed in this study. The first formulation (1.20) has a quadratic form in P variables. On the other hand, the second formulation (1.21) has a quadratic form in N variables. However, in this specific problem, $N = P$ indicating that both formulations can be potentially equally relevant in terms of computational efficiency. But it was observed that the second formulation provided better lower bounds to the problem within a given amount of time, thus reducing the overall computational time.

In an effort to further reduce the computational effort, further possibilities to increase the strength of the formulation were explored. It was observed that the magnitude of the absolute value of diagonal elements and the non-diagonal elements of stable matrices in this study were different. The reason for this is quite evident if one looks at how these stable matrices were generated [VH20a]. In this scenario, one can tighten the bounds further by giving separate bounds for diagonal and non-diagonal elements.

$$-M_{U-d} \leq \beta_i \leq -M_{U-\text{upper}} \qquad i \in \mathcal{D} \tag{2.12}$$

$$-M_{U-nd} \leq \beta_i \leq M_{U-nd} \qquad i \notin \mathcal{D} \tag{2.13}$$

Here, $M_{U-d} := \tau \|\beta_{\text{init},\mathcal{D}}\|_\infty$ and $M_{U-nd} := \tau \|\beta_{\text{init},\mathcal{D}^c}\|_\infty . \beta_{\text{init},\mathcal{D}^c} \in \mathbb{R}^{(P-p)}$ contains all those elements of β_{init} whose positions correspond to the non-diagonal position and τ is a multiplier greater than 1. It is important to note that these estimates, M_{U-d} and M_{U-nd}, provide lower bounds for M_U. They offer tighter estimates than M_U and can be considered as good as, if not better than, the existing formulation.

By making these modifications, one can further strengthen formulation (1.21).

$$
\begin{aligned}
&\min_{\zeta,\beta,z} \frac{1}{2}\zeta^T\zeta - \langle X^T y, \beta\rangle \\
\text{such that}\quad &\zeta = X\beta, \\
&\quad -M_U z_i \le \beta_i \le M_{U-\text{upper}} z_i, \qquad i = 1,\dots,P \\
&\quad z_i \in \{0,1\}, \qquad i = 1,\dots,P \\
&\quad \sum_{i=1}^{p} z_i \le k, \qquad i = 1,\dots,P \\
&\quad -M_{U-d} \le \beta_i \le -M_{U-\text{upper}} \qquad i \in \mathcal{D} \\
&\quad -M_{U-nd} \le \beta_i \le M_{U-nd} \qquad i \notin \mathcal{D} \\
&\quad -M_U^{\zeta} \le \zeta_i \le M_U^{\zeta} \qquad i = 1,\dots,P.
\end{aligned}
\tag{2.14}
$$

where $X = A(S) \in \mathbb{R}^{N\times P}$, $y = -\text{vec}(C) \in \mathbb{R}^N$, $\beta = \text{vec}(M) \in \mathbb{R}^P$, and $P = N = p^2$.

2.3 Model Selection

The size of the best subset, k, is a hyper-parameter that is used to determine the edge set E of the graph. In practice one may not know the true support of the graph. To choose an optimum graph size, often a validation step needs to be employed using a grid with different values of k. By systematically evaluating the performance of the model using various values of k, one can aim to find the optimal graph from the space of recovered graphs. The choice and the number of values for k depends on the problem at hand. For a directed graph with p nodes, the number of possible directed edges including self-loops are $2 \cdot \binom{p}{2} + p = p^2$. So one can choose each value of k to be from the grid $[0, p^2]$. However, with increasing graph size, this is computationally very expensive. For a graph with $p = 10$ and time limit of 180 seconds per k, this is already 5 hours of computation time. For this reason, in this study, a fixed grid approach was followed. More details on the choice of time limit for this study is elaborated in section 2.5.

In the fixed grid approach, 10 uniformly spaced values were taken from the interval $[p, p^2]$. The lower limit on the choice of k was set to neglect the case of a graph with no self loops. After this, one then needs a systematic procedure to find the optimal graph from this space of recovered graphs. Information criterion are very useful tools in this scenario. They provide an objective to be minimized and help in choosing a suitable model. They are also shown to be asymptotically equivalent to different variants of cross-validation [Sha97; Sto77]. Suppose the estimate for the drift matrix, M, from the MIO solver is given by $\widehat{M}$. Now, one can compute the resulting estimate of the covariance matrixe, $\widehat{\Sigma}$, by solving eq. (1.7). The idea is to find out which one of the graphs are likely to be the true graph. The Gaussian likelihood function can provide useful information in such scenarios. However, this methodology does not impose any restrictions on the complexity of the model. So with this method, one may get a very complex model. This implies overfitting to the observed data. The true models, however, may lie in a lower dimensional space than the selected model [Blu+87]. In order to tackle this, complexity of model is usually penalized. In this study, three most well known information criterion were considered and will be briefly introduced in the subsequent subsections.

2.3.1 Akaike's Information Criterion (AIC)

Akaike [Aka73] proposed an extension of the maximum likelihood principle to choose models that are closer to the true sparse models. Let $X \in \mathbb{R}^{n\times p}$ be i.i.d observations, $\mathcal{M}_i, i = 1,\dots,k$ denote

the models (here, drift matrices) and $\Theta_{\mathcal{M}_i}$, the parameters of model $\mathcal{M}_i$. It measures the proximity of the estimated models with the observed model. The proximity is usually defined in terms of *Kullback-Leibler Divergence*. If the true density of the data is given by $q(X)$, the Kullback-Leibler divergence between $q(X)$ and $P(X|\mathcal{M})$ is given by

$$\begin{aligned} KL(\mathcal{M}) &= \int q(X) \log\left(\frac{q(X)}{P(X|\mathcal{M})}\right) \\ &= \underbrace{\int q(X) \log\left(q(X)dX\right)}_{\text{Unknown, but constant}} - \underbrace{\int q(X) \log\left(P(X|\mathcal{M})dX\right)}_{\text{Relative KL information}} \end{aligned} \tag{2.15}$$

The relative KL information depends on the unknown true distribution, $q(X)$. But, Akaike shows that, under appropriate conditions, the expected value of

$$AIC = -2\log\left(P(X|\mathcal{M}_i)\right) + 2P$$

approaches asymptotically to the expected value of $\int q(X) \log\left(P(X|\mathcal{M})dX\right)$. Here $P = \dim(\Theta_{\mathcal{M}})$. Thus, under suitable settings, AIC provides an unbiased estimator of the expectation of the relative KL information. So one can minimize the KL divergence between the estimated model and the true model by minimizing AIC. For Gaussian graphical models, this could be written as

$$AIC(\widehat{\Sigma}^{-1}) = n(-\log|\widehat{\Sigma}^{-1}| + \text{tr}(\widehat{\Sigma}^{-1}S)) + 2p_{\widehat{\Sigma}^{-1}} \tag{2.16}$$

where the first two terms arise from the Gaussian log-likelihood function, and

$$p_{\widehat{\Sigma}^{-1}} = |\{\text{unique non- zeroes in } \Sigma^{-1}\}|$$

2.3.2 Bayesian Information Criterion (BIC)

The BIC, introduced by Schwarz [Sch78], is an approximation to bayesian approach of model selection by estimating posterior model probability [BK10]. It has a stronger penalty on the number of parameters in the estimate of inverse covariance matrix in comparison to AIC, and thus chooses sparser models. The criterion is derived by choosing an uninformative, flat prior on $\Theta_{\mathcal{M}}$ and estimating the posterior probability of model $\mathcal{M}$. Applying Bayes theorem to calculate this,

$$\begin{aligned} P(\mathcal{M}|X) &\propto \frac{P(X|\mathcal{M})}{p(X)} \\ &\propto \int_{\Theta_{\mathcal{M}}} e^{n\bar{l}(\theta)}\partial\theta \\ &\approx e^{l(\widehat{\theta})} \int_{\Theta_{\mathcal{M}}} e^{-\frac{1}{2}(\theta-\widehat{\theta})n\frac{\partial^2}{\partial k^2}\bar{l}(\widehat{\theta})(\theta-\widehat{\theta})}\partial\theta \end{aligned}$$

where $\bar{l}(\theta)$ is the likelihood for each observation, and $\bar{l}(\widehat{\theta}) = l(\widehat{\theta})/n$. Ignoring terms independent of n, we get

$$P(\mathcal{M}|X) \approx e^{l(\widehat{\theta})} n^{-P/2}$$

for large n.

By applying − log and ignoring constants,

$$BIC(\mathcal{M}) = -2I(\widehat{\theta}) + P\log(n)$$

For a Gaussian graphical model,

$$BIC(\widehat{\Sigma}^{-1}) = n(-\log|\widehat{\Sigma}^{-1}| + \mathrm{tr}(\widehat{\Sigma}^{-1}S)) + p_{\widehat{\Sigma}^{-1}}\log n \tag{2.17}$$

where the symbols take the usual definition.

2.3.3 Extended Bayesian Information Criterion (eBIC)

Since BIC assigns a uniform prior, it typically works well when the space of models under consideration have uniform distribution of models of all size. Let $\mathcal{M}_{(\cdot,j)}$ represents all the models with j parameters, $j \in [P]$. BIC assigns equal probability to all of them irrespective of the number of models $\mathcal{M}_i$ having j parameters. Suppose there exist a $j' \in [P]$ for which $|\mathcal{M}_{(\cdot,j')}| >> |\mathcal{M}_{(\cdot,j)}|$, then all the models, $\mathcal{M}_{(i,j')}$, $i \in [|\mathcal{M}_{(\cdot,j')}|]$ receive higher probabilities than the rest. J. Chen and Z. Chen [CC08] showed that by choosing a different prior distribution, one can extend the BIC to work well in such cases. This prior assigns equal probability for any model $m \in \mathcal{M}_{(\cdot,j)}$, $P(m|\mathcal{M}_{(\cdot,j)}) = 1/|\mathcal{M}_{(\cdot,j)}|$. But for each $\mathcal{M}_{(\cdot,j)}$, $j \in [P]$, the probability taken to be proportional to $|\mathcal{M}_{(\cdot,j)}|^{\xi}$ instead of $|\mathcal{M}_{(\cdot,j)}|$, where $\xi \in [0,1]$. Then the prior probability is given by

$$P(m) = P(m|\mathcal{M}_{(\cdot,j)})P(\mathcal{M}_{(\cdot,j)}) \propto |\mathcal{M}_{(\cdot,j)}|^{\xi-1} = |\mathcal{M}_{(\cdot,j)}|^{-\gamma},$$

where $\gamma \in [0,1]$. In case of graphical models, this situation would be considered alike. As the graph size increases, the number of models with same cardinality of edge set grows. Foygel and Drton extended the above concept for Gaussian graphical models [FD10]

$$eBIC_{\gamma}(\widehat{\Sigma}^{-1}) = n(-\log|\widehat{\Sigma}^{-1}| + \mathrm{tr}(\widehat{\Sigma}^{-1}S)) + p_{\widehat{\Sigma}^{-1}}\log n + 4\gamma p_{\widehat{\Sigma}^{-1}}\log(p) \tag{2.18}$$

where the symbols take the usual definition, p is the number of nodes, and γ is a tuning parameter.$\gamma = 0$ implies the ordinary BIC, whereas $\gamma = 1$ demands additional sparsity. $\gamma = 0.5$ is a good trade-off.

The scores, s, obtained from these information criterion are evaluated from the results of each value of subset size, k. Then the scores are converted to posterior model probability. Minimizing the information criterion score is equivalent to maximizing the posterior model probability. The model weights are defined as,

$$W(\mathcal{M}) = e^{-s/2}$$

and the rescaled posterior model probability is given as

$$p(\mathcal{M}_i|X) = \frac{W(\mathcal{M}_i)}{\sum_i W(\mathcal{M}_i)}$$

The k corresponding to the graph with the maximum posterior model probability is then chosen as the optimal hyperparameter.

2.4 Metrics

The graph recovery problem was approached as a classification problem involving the off-diagonal elements of matrix $\widehat{M}$. Specifically, if M^* denotes the true drift matrix, the elements from both $\widehat{M}$ and M^* were compared to determine the presence of edges, which corresponded to non-zero entries in the off-diagonal. In order to measure the performance of selected model on unseen data, six different metrics were evaluated on the test dataset. They are described below along with a brief summary of how each one of them contributes towards the model performance evaluation.

1. **Accuracy** measures how well the binary classification correctly identifies or excludes an edge. It is given by

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

   Use of this metric can sometimes be quite misleading, especially in case of sparse graphs where the total number of true edges are much smaller than the total number of non-edges. A classifier can easily learn to produce a graph with no edges and still have good accuracy even though the there are true edges in the problem. This is referred to as the class imbalance and one might need other metrics to get a multifaceted viewpoint.

2. **True Positive Rate (TPR or Recall)** measures the classifier's ability to detect true edges among all edges actually present in the graph.

$$\text{TPR or Recall} = \frac{TP}{TP + FN}$$

   While this metric does give importance to the detection of true edges, it is completely blind to the detection of false edges. It is important to note that the TPR is not defined when TP = FN = 0. This scenario can occur if $\widehat{M}$ is a diagonal matrix. In such cases, it is common practice to assign TPR the value of 1, indicating perfect detection of true positive edges, as there are no false negatives to be considered [VH20b].

3. **False Positive Rate (FPR)** is the ratio of falsely detected edges to the total number of non-edges.

$$\text{FPR} = \frac{FP}{FP + TN}$$

   This metric, together with TPR, gives great insight on the classification performance.

4. **F1 Score** is the harmonic mean of precision and recall, both of which evaluates the performance of the classifier in detecting true edges. While precision takes into account the false positives, recall takes into account the the false negatives, thus giving a complete picture of classification.

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{F1 Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

   Similar to TPR, the cases where TP = FP = 0 are handled by assigning Precision to the value of 1. Similarly, when Precision and Recall are simultaneously zero, the F1 Score is assigned a value of 1.

5. **Area under the Receiver Operating Characteristics Curve (AUCROC).** The ROC curve, in general, is the plot between variation of TPR with FPR at different threshold probability of a classifier that is used for labelling the class positive or negative. It conveys information on the ability of a classifier in discriminating between positive and negative instances. However, the MIO solver merely returns the entries of the drift matrix which do not correspond to probabilities. So the absolute value of the diagonal entries of $\widehat{M}$ were converted to probabilities using a sigmoid function, $\sigma(x) = 1/(1 + \exp(-|x|))$. The sigmoid function maps all the entries to a value between $[0, 1]$. The entry with the lowest non-zero value determines the threshold probability of edge detection for that particular solution, as all the values above it are detected as positive by the MIO solver. 10 values of uniformly spaced threshold probabilities were taken from the interval $[0, 1]$ and for each threshold, the entries of the solution were labelled as positive or negative based on the selected threshold value. Then the TPR and FPR were computed for each value of selected threshold and the area under the resulting curve was computed. A perfect classifier would always detect all the true edges irrespective of the chosen threshold resulting in an Area under the Curve (AUC) of 1. An AUC value of 0.5 is also of interest. This means that, the discriminating potential (differentiating between TP and TN) of classifier is no better than a random guessing classifier until one has enough samples.

6. **Area under the Precision-Recall Curve (AUCPR).** As mentioned before, there is always the problem of class imbalance when dealing with sparse graphs. A large increase in FP will only result in a small increase in FPR due to the presence of large number of TN. So, the ROC curve will always be over-optimistic when compared across all threshold values. In such cases, PR curves can provide a different viewpoint. It conveys information on the ability of a classifier in correctly detecting true edges. In general, by plotting Precision and Recall values, one can view the classifier's performance without the influence of large number negative class in a dataset. Here, the PR curve is constructed following a similar procedure as ROC curve. A perfect classifier should be precise, meaning it should detect, if not all, only the true edges at different threshold values resulting in an Area under the Curve of 1. An AUC value of 0.5 means that that the classifier is no better than a random guessing classifier in choosing a TP.

To study the graph recovery performance across different hyper-parameter values, the following metrics were considered:

- The **maximum accuracy** of edge recovery along a path of hyperparameter values.
- The **maximum F1 score** along the hyper-parameter path.
- The **ROC curve along the hyper-parameter path** can be useful in determining the diagnostic capability of the graph recovery methods. Here, the variation in TPR with FPR as one varies the hyper-parameter value, k, is plotted. For each value of k, the TPR and FPR values were plotted. Then the area under this curve was computed using trapezoidal integration.
- The **PR curve along the hyperparameter path**, where the variation in Precision with Recall as one varies the hyper-parameter value, k, is plotted. For each value of k, the Precision and Recall values were computed. Then the area under the curve resulting from these values was estimated using trapezoidal integration.

It may be possible that one might have insufficient data points to compute the area under both the performance curves. In cases where the FPR axis in the ROC curve and the Recall axis in the PR curve do not span the full range from 0 to 1, the curves were extrapolated by adding boundary points. The True Positive Rate (TPR) and Precision values at these points were set to the maximum values observed among the existing data points. If there were fewer than 5 data points between 0 and 1 on the FPR axis in the ROC curve or the Recall axis in the PR curve, nearest neighbour

interpolation was performed. This involved adding additional points between the existing data points to ensure a more finely grained representation of the curve.

2.5 Simulation Setup

The MIO was setup in Gurobi [Gur23b] and it was run using its R interface. The formulation was taken and adapted from the works of Tibshirani et al. [HTT17a; HTT17b] for this application. For more details on implementation, see Radhakrishnan [Rad23]. Since performance of lasso on such problems has been investigated before [DDK22; VH20a], it was considered as a suitable benchmark for comparing the MIO results. The problem was solved using lasso following a similar strategy as proposed in [VH20a] and also as described in Algorithm 5.

The MIO problems were initialized using the four discrete first order algorithms discussed in section 2.1 and their results were compared to the solution obtained from lasso and to each other. To facilitate this comparison, large sets of datasets were generated from custom-generated graphs of varying sizes. These synthetic datasets allowed for a comprehensive evaluation of the performance of both methods and different initialization strategies. A fixed grid containing 10 uniformly spaced values within the interval $[p, p^2]$ was considered for k for model selection.

It was observed that standardizing the columns of the matrix $A(\Sigma)$ to have zero mean and unit variance significantly improved the structure recovering performance, especially at large sample sizes. For this reason, $A(\Sigma)$ was standardized for all the problems and methods adopted in this study. The entries of the recovered drift matrices were then re-scaled back using the original variances of the columns of $A(\Sigma)$.

Lastly, the insights gained from this study were applied to a real-world dataset, enabling the assessment of the practical applicability of the findings.

2.5.1 Synthetic Dataset

Graphs having $p = 5, 10, 15, 20$ and 25 nodes were considered for this study. For each p, 100 signals (drift matrices, M^*, and volatility matrices, C^*) were generated. The entries of the drift matrices and the volatility matrices were generated following the procedure described by Varando and Hansen [VH20a]. All matrices were thus stable. The minimum value that the entries of the volatility matrices can take was set be 0.2. This was done to avoid estimation of very small entries in the recovered drift matrices, $\widehat{M}$. For each signal, M_i^*, the corresponding population covariance matrix, Σ_i^*, was computed by solving equation (1.7) for $i = 1, \ldots, 100$. Then samples of sizes $n = 100, 200, 500, 1000, 5000, 10^4, 10^5, \infty$ were drawn from $N(0, \Sigma_i^*)$ from each graph, $i = 1, \ldots, 100$.

Each of these set of samples were randomly split so that 80 % of it was assigned as training data and the rest as test data. The sampled data for each sample size, j, was used to construct the sample covariance matrices, $S_{(i,j)}$, for both training and test data. When $j = \infty$, $S_{(i,\infty)} := \Sigma_i^*$. The goal is to try to recover M^* from these sample covariance matrices, $S_{(i,j)}$, $j \in n$.

In addition to comparing different initialization strategies, a parametric study was conducted to compare the recovery of the graphs at different sparsity levels. For this purpose, the above procedure was repeated twice to generate two sets of signals. The first set had an edge probability of 5% ($d = 0.05p$), while the second set had an edge probability of 25 % ($0.25p$).

2.5.2 Real Dataset

In this study, isoprenoid biosynthesis in *Arabidopsis thaliana* is considered as an Ornstein-Uhlenbeck process. Isoprenoids are important components of membranes, as photosynthetic pigments and as hormones in plants. They are synthesized through condensation of the five-carbon intermediates: isopentyl diphosphate (IPP) and dimethylallyl diphosphate (DMAPP). In higher plants, there are two distinct pathways for the formation of IPP and DMAPP: Cytosolic pathway (mevalonate or MVA pathway) and plastidial (non-mevalonate or MEP) pathway.

To gain more insight into the interaction between both the pathways, Wille et al. [Wil+04] monitored gene-expression patterns under different experimental conditions using 118 GeneChip (Affymetrix) microarrays. The dataset comprises measurements of 39 genes, 15 of which were assigned to the cytosolic pathway, 19 to the plastidal pathway and 5 encode proteins located in mitochondrion. This dataset is made easily accessible by Drton et al. [Drt+23]. 80 % of the data was considered for training and the rest was used to make predictions.

There is no ground truth to the actual cross talks between the pathways. However, Wille et al. [Wil+04] constructed a gene regulatory network by applying a modified graphical Gaussian modeling (GGM). The results from this approach can be considered as a benchmark for comparison (relative ground truth). Furthermore, one does not have prior information on the true Volatility matrix, C. In this study, it was assumed to be positive definite and known. So it was chosen to be the identity. The potential implications of this choice on the results will be discussed.

2.5.3 Gurobi Settings

Bertismas et al. [BKM16] terminated the MIO model training either when optimality gap of 1 % was reached or a time limit of 15 minutes was reached. However, as mentioned before, the time limit is not practically feasible with increasing graph size. Hastie et. al [HTT17b] considered a time limit of 180 seconds for this exact reason of scalability issues. In this study, the same approach was followed, utilizing the 180-second time limit or reaching an optimality gap of 1 %. The simulations performed in this study are almost as extensive as the ones considered in Hastie et al. [HTT17b]. They considered 5 settings, each setting having 4 different coefficient variants, each data generated from these coefficients were using 10 different signal to noise ratios and the results were averaged over 10 runs for every case. In addition, they performed a cross-validation for different values of k: 10 values for the lowest setting and 50 values for the rest. This results in a total of 1,00,000 problems. In this study, there are 100 distinct signals for 5 different graph sizes, 8 sample sizes used for training using 10 different values of k. In addition to this, parametric study is done on 2 values of d. This results in a total of 80,000 problems. So maintaining a time limit of 180 seconds seemed like a reasonable choice. In general, it was noted, just like in [HTT17b], majority of the problems closed the optimality gap before the time limit. Though it can differ based on the considered problem, this time limit felt insufficient to close the optimality gap for graph sizes greater than 15, with low to medium values of k in general. For this reason, sub-optimal solutions having optimality gap of less than or equal to 5 % were accepted for further evaluation.

The gram matrix $\Gamma(\Sigma)$ is positive semidefinite [Det+22]. However Gurobi can show that this condition is not fulfilled by a small margin in certain cases. This could be because of Gurobi's presolve phase, which can potentially convert a convex problem to non-convex [Tow22]. In order to handle this, Gurobi's inbuilt approaches to solve non-convex objective functions were used [bT20; Gur23b]. This, in particular, includes performing bilinear transformation of the non-convex parts and employing spatial branching technique [QG95].

3 Results and Discussion

3.1 Numerical simulations on Synthetic Dataset

Throughout this discussion, for convenience, best-subset selection using MIQP is denoted as BS and its initialization with Algorithm 2 will be referred to as BS_{reg}, with Algorithm 3 as BS_{1edge}, with Algorithm 4 as $\text{BS}_{\text{glasso}}$ and with Algorithm 5 as BS_{lasso}.

3.1.1 Results: Performance metrics across different hyper-parameter values

The performance measures across different hyperparameters offer great insights on the structure recovering potential of different methods. Figures 3.1 to 3.5 show the variation of these metrics with graph size at different sample sizes and at edge probability of 5 % ($d = 0.05p$). One can note that it is hard to infer much from the accuracy plots apart from the fact that all the methods are quite good and robust in terms of accuracy. However, as mentioned earlier, the class imbalance problem becomes very prominent with sparse graphs and this metric may not reveal all information. For $n = 100$, both lasso and BS along with its all four initialization strategies show similar performance as seen in Figure 3.1. The performance is good, in general, for smaller graphs with $p = 5$. However, it quickly degrades with increasing signal size. AUC PR curve describes how lasso starts performing slightly worse than BS beyond $p = 15$ in terms of precision and recall trade-off. This poor performance is largely attributed to the fact that lasso selects more FPs with decreasing regularization parameter, λ. BS seems to be much more robust in performance for $p \geq 15$. However, this metric value is still below 0.5 beyond $p = 10$ for all the methods. This means that all the approaches are worse than a random guessing classifier in terms of detecting true edges. The performance of BS_{lasso} seems to be not affected by poor lasso performance in this region as it eliminates many FPs generated by lasso initialization in the attempt of getting a k-sparse solution. The AUC ROC values indicate that all the methods are as good as a random guessing classifier in differentiating between the true edges and non-edges beyond $p = 10$.

As the sample size n increases, lasso clearly seems to outperform BS in terms of precision across different values of λ (see for e.g., Figure 3.3). There is still a gradual drop in performance as the graph size increases. BS actually shows a steep drop relative to lasso, but beyond $p = 10$ the drop is relatively steadier in comparison to that of lasso. The lower precision-recall trade-off across different values of k could be due to the fact that BS uses only 10 values of k between $[p, p^2]$. This becomes a very coarse choice as the graph size increases and as a result many FP are selected. But somehow BS_{lasso} seems to select large proportion of TP among all of the positives as k is varied (see AUC PR plot for e.g., Figure 3.3). Furthermore, one can infer from the AUC ROC graphs that the discriminating power of lasso and BS_{lasso} across different values of hyper-parameters quickly decreases with increasing graph size (see Figures 3.2 to 3.5). For BS_{lasso}, this behaviour is attributed to the fact that lasso itself has low AUC ROC for larger graphs and thus provides poor initializations to BS. The other initializations strategies lead to quite robust performance, especially when $n \geq 10,000$. Furthermore, the AUC PR values are still below 0.5 for most cases. Only when $n \geq 10,000$, lasso and BS_{lasso} crosses the value of 0.5 for $p = 10$. Though discriminating power of all the methods improve for large sample sizes, none of them have a good precision-recall trade-off for large graph sizes ($p \geq 15$) across different values of hyper-parameters. Among the variants of marginal regression initialization strategies, BS_{1edge} clearly performs the

best in correctly differentiating between true edges and non-edges, and BS_{reg} performs the worst. However, in terms of precision they perform equally poor.

When the population covariance matrix, Σ^*, is used ($n = \infty$), the trend remains the same. However, BS_{lasso} dominates the AUC PR plot. Furthermore, it is the only BS method that consistently achieves AUC PR and AUC ROC values above 0.5 across all graph size.

Figures 3.6 to 3.10 show the variation of these metrics with signal size at different sample sizes and at edge probability of 25 % ($d = 0.25p$). The situation is generally similar to that of $d = 0.05p$, but is more magnified. Some key highlights are: Lasso's performance in terms of precision, as observed in AUC PR graphs, is worse than that of BS for graphs with $p \geq 10$, and it outperforms the others completely in this aspect only when sample size exceeds $n = 1,00,000$. This is slightly worse than $d = 0.05p$ case, where lasso was performing poor for graphs with $p \geq 15$ and outperformed all the methods from $n = 10,000$. Like in $d = 0.05p$ case, BS_{lasso} does not seem to be affected by this poor performance of lasso. The accuracy of all the methods have dropped by almost 10 % as a result of making the graphs less sparse. The AUC PR values are greater than 0.5 for $p = 5$ only when sample size exceeds $n = 1000$, while it was possible for all sample sizes when $d = 0.05p$. While this metric exceeded 0.5 mark for $p = 10$ when $n \geq 10,000$ at $d = 0.05p$ using lasso and BS_{lasso}, the same was not possible at $d = 0.25p$ until the sample size crossed $n = 1,00,000$. So the general observation is that the recovery of the graphs became harder when sparsity level was reduced.

For low sample size ($n \leq 1000$), the performance of BS remains quite similar for all the initialization strategies, except BS_{lasso}. The TPR-FPR trade-off estimated across different hyperparameters are relatively poor in comparison to $d = 0.05p$ case. The metrics are all close to 0.5 except for graphs with $p = 5$, indicating how poorly all of the methods distinguish a true edge from a non-edge. The AUC PR values are below 0.5 across all signal sizes in general, indicating poor precision along different values of hyperparameters.

As the sample size increases, the metrics improved. Again, lasso and BS_{lasso} started to outperform all the other methods. Just like in $d = 0.05p$ case, lasso and BS_{lasso} are dominant in the AUC ROC plots for $p \leq 10$, but quickly degrade in performance beyond that. All the other BS strategies seem again robust in this aspect and they have a metric value above 0.5 across different signal sizes. This, in particular, is noticeable when sample sizes exceed $n = 10,000$ (see Figure 3.8). Among the variants of marginal regression initialization strategies, BS_{1edge} seems to superior performer, followed by BS_{glasso} and then BS_{reg} is the poorest performer. In the AUC PR plots, however, all these methods seem to be performing similar in terms of precision across different values of k. This is again similar to $d = 0.05p$ case. Lasso, BS_{lasso} and BS_{1edge} are the only three methods who cross the 0.5 mark in the AUC PR plots for signals of size $p = 10$ when $n = 1,00,000$ (see Figure 3.9).

When the population covariance matrix, Σ^*, is used ($n = \infty$), lasso and BS_{lasso} seem to be the most dominant in terms of precision and discrimination power. BS_{1edge} remains the second most powerful method in this aspect. Compared to $d = 0.05p$ case, all the methods except BS_{reg} have relatively good precision-recall trade-off across different hyperparameter values for all graph sizes (see AUC PR in Figure 3.10). Furthermore, the F1-Score shows that BS based on marginal regression initialization strategies are slightly more robust in terms of precision and recall across different signal sizes.

In order to view the results under different perspective, the variation of same performance metrics with sample sizes at different graph size and at $d = 0.05p$ were plotted in Figures 3.11 to 3.15. Here one can quite clearly see that, in general, lasso seems to be the most superior method out of all, followed by BS_{lasso}, BS_{1edge}, BS_{glasso} and BS_{reg}.

AUC values are greater than 0.5 for small graphs with $p = 5$ across all sample sizes and all methods show good potential. All the projected gradient descent initialization based methods perform

similarly in terms of precision and maximum F1-Scores across different values of k. However, the AUC ROC plot clearly shows the dominance of BS_{1edge} and BS_{glasso} (see Figure 3.11).

As graph size increases, the metric values become poorer in general. AUC ROC plots are typically dominated by lasso, followed by BS_{lasso}, BS_{1edge} and BS_{glasso}. The AUC ROC values tend towards 0.5 and do not increase until one has enough samples, which is 1000 or more for graphs with $p = 25$ (see Figure 3.15). For AUC PR values to be above 0.5, one needs more than 1,00,000 samples (for $p = 25$). Even then, only lasso and BS_{lasso} are able to achieve relatively good precision across different hyper-parameter values (see for e.g., Figure 3.12). With good precision across different k, BS_{lasso} still has an F1-Score similar to other BS strategies. This could indicate that recall rate is poor for BS_{lasso} in general. When $p \geq 20$, all the methods tend to perform similarly. But both, BS_{lasso} and lasso, perform poorer than other projected gradient descent BS methods when there are not enough samples (see for e.g., Figure 3.14). BS_{1edge} also has the potential to perform good. However, it seems to be needing more samples ($\geq 1,00,000$).

For sake of completeness, similar results are shown for $d = 0.25p$ case (see Figures 3.16 to 3.20). As noted before, one can now clearly see that AUC PR value is not greater than 0.5 across all sample sizes when $p = 5$ unlike $d = 0.05p$ case. It only crosses the mark when $n = 1000$ for lasso and BS_{lasso}, and when $n = 5000$ for all the other BS strategies. Both, lasso and BS_{lasso}, seem to dominate all the other methods in AUC plots. However, all of the methods have similar F1-score (see Figure 3.11).

As graph size increases, the AUC ROC values do not show any improvement for small sample sizes just like in $d = 0.05p$ case (but worse). For $p = 10$, the average AUC ROC value does not cross 0.5 mark until one has $n = 1,00,000$ samples for $p = 25$ (see Figure 3.20). AUC PR values were in general lower than 0.5. Even with enough samples, only lasso was able to cross the 0.5 mark for sample size $n \geq 1,00,000$. The other methods only achieved a good AUC PR metric when the true population covariance matrix was used.

3.1.2 Plots and Summary: Performance metrics across different hyper-parameter values

Since the study was very extensive, the results are summarized as follows:

- Lasso gives better results than BS in general. However, lasso's precision across different values of tuning parameters as graph size increases is worse than BS when sample size is low. BS_{lasso} does not seem to be affected by this. In fact, at this region, BS_{lasso} performs better than lasso and other BS strategies in terms of having a good precision-recall trade-off.
- projected gradient descent BS approaches have similar performance in terms of precision-recall trade-off across different values of k. However, TPR-FPR trade-offs are clearly better for BS_{1edge} among all the three methods. Lasso and BS_{lasso} have poorer TPR-FPR trade-off for large signal sizes in comparison.
- In most cases, the values of AUC PR barely make up to 0.5. This implies how poor the methods are in comparison to a random guessing classifier in trying to detect true edges. For low signal sizes ($p = 5$), all the methods generally perform well. However for medium and large signals ($p \geq 10$), it becomes harder to identify true edges without having high FP's (which results in lower precision). One typically needs more than $n = 1,00,000$ samples to get an average AUC PR value above 0.5. The methods are not even able to distinguish between a TP and TN - the discriminating potential of all the methods quickly degrades with increasing signal size and one often needs more samples (typically greater than $n = 1000$) to get an average AUC ROC value greater than 0.5.
- The situation is worse when the sparsity level of the graph is reduced. It needs more samples to achieve similar performance like in $d = 0.05p$ case. Overall accuracy drops for all the signal sizes for a given sample size.
- As graph size increases, the performance of all methods degrades. In general, projected gradient descent BS approaches are very robust in terms of change in performance across different signal size. Furthermore, their performance does not decay as much as the lasso and BS_{lasso}. This is certainly one of the biggest advantages of BS.
- Lastly, it is important to consider that the poor performance of BS in terms of precision can be attributed to the fact that only 10 values of k were used, which is quite coarse as graph size increases. Due to this, it can select many FPs in this process. Furthermore, the time-limit was set to 180 seconds due to practical restrictions, which may not have sufficient to close the optimality gap. This, in particular, is true for large signal sizes and low sample size. Due to this sub-optimal solutions having 5 % optimality gap were accepted. All of this plays an important role in the performance of BS.

In the next section, different model selection criterion discussed in section 2.3 are applied to the solution obtained through these methods.In particular, the optimum parameters selected (k or λ) and used to recover the drift matrix from the test data set. Only results from a total sample size of $n = 10000$ are shown. All the other results will be attached as additional results in the Appendix A.

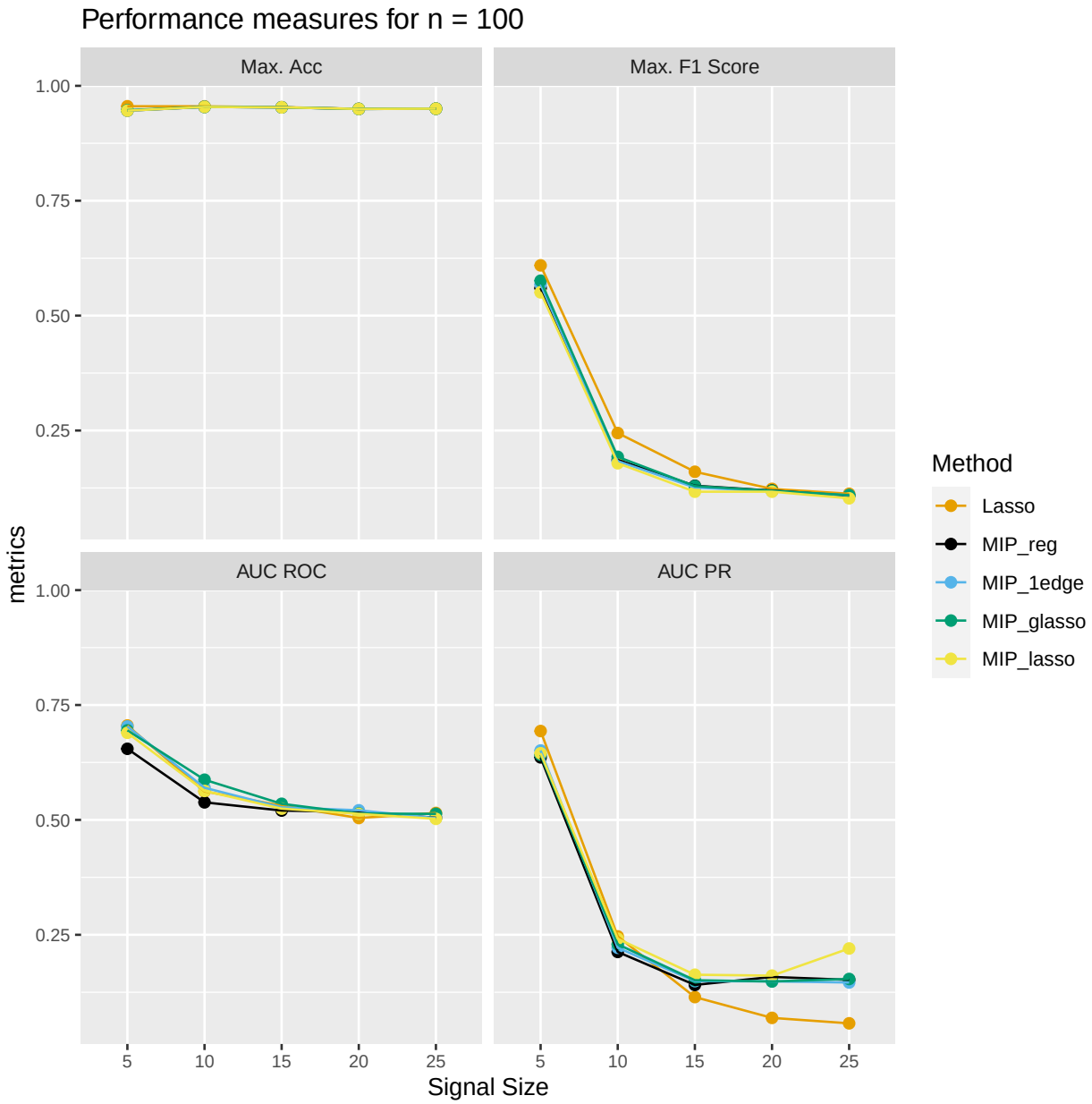


Figure 3.1 Variation of performance metrics with graph size for a total sample size of 100 (training data size of 80). The metrics were computed along the hyper-parameter path and are averaged over 100 different signals with edge probability of 5 %.

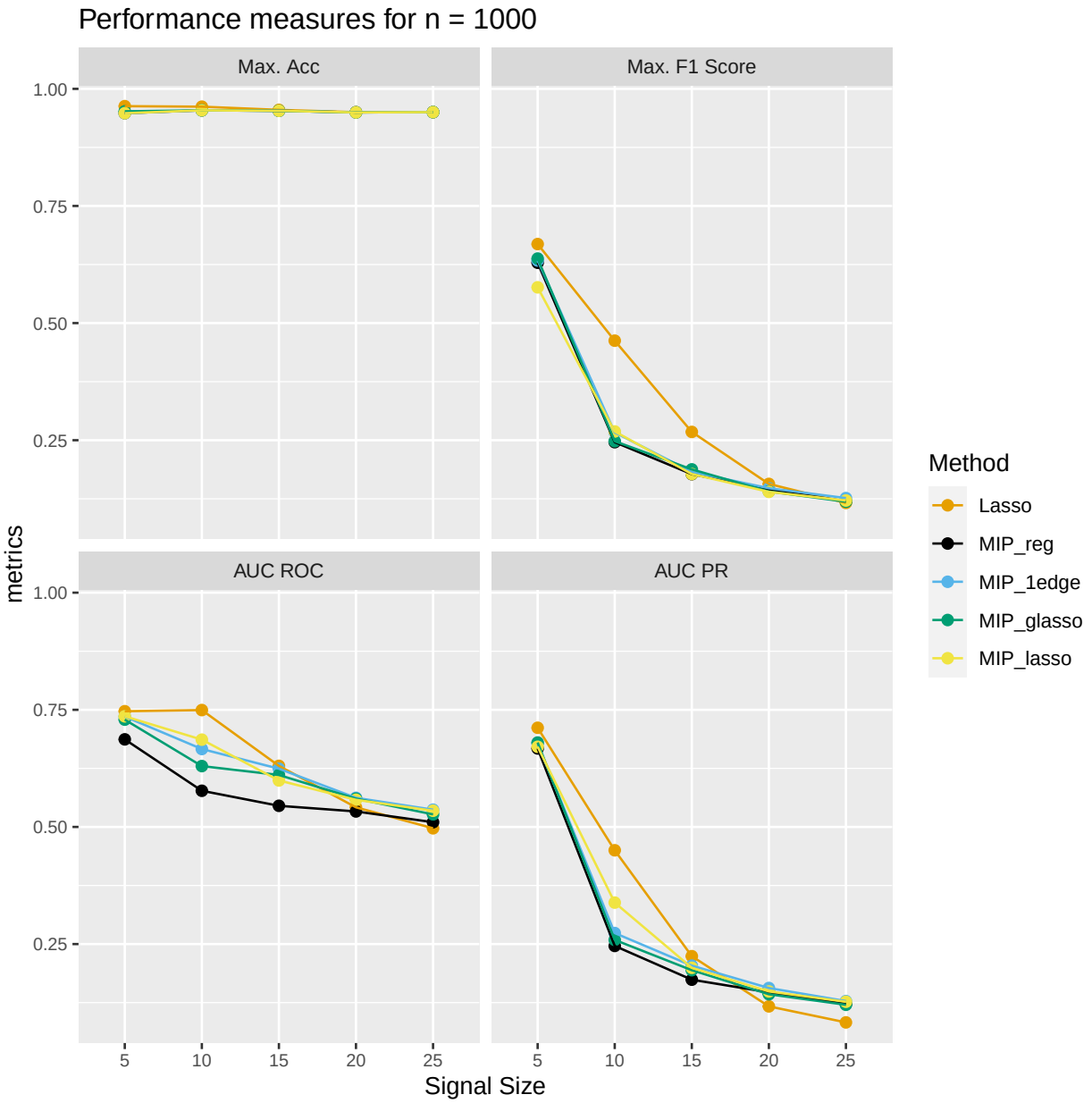


Figure 3.2 Variation of performance metrics with graph size for a total sample size of 1000 (training data size of 800). The metrics were computed along the hyper-parameter path and are averaged over 100 different signals with edge probability of 5 %.

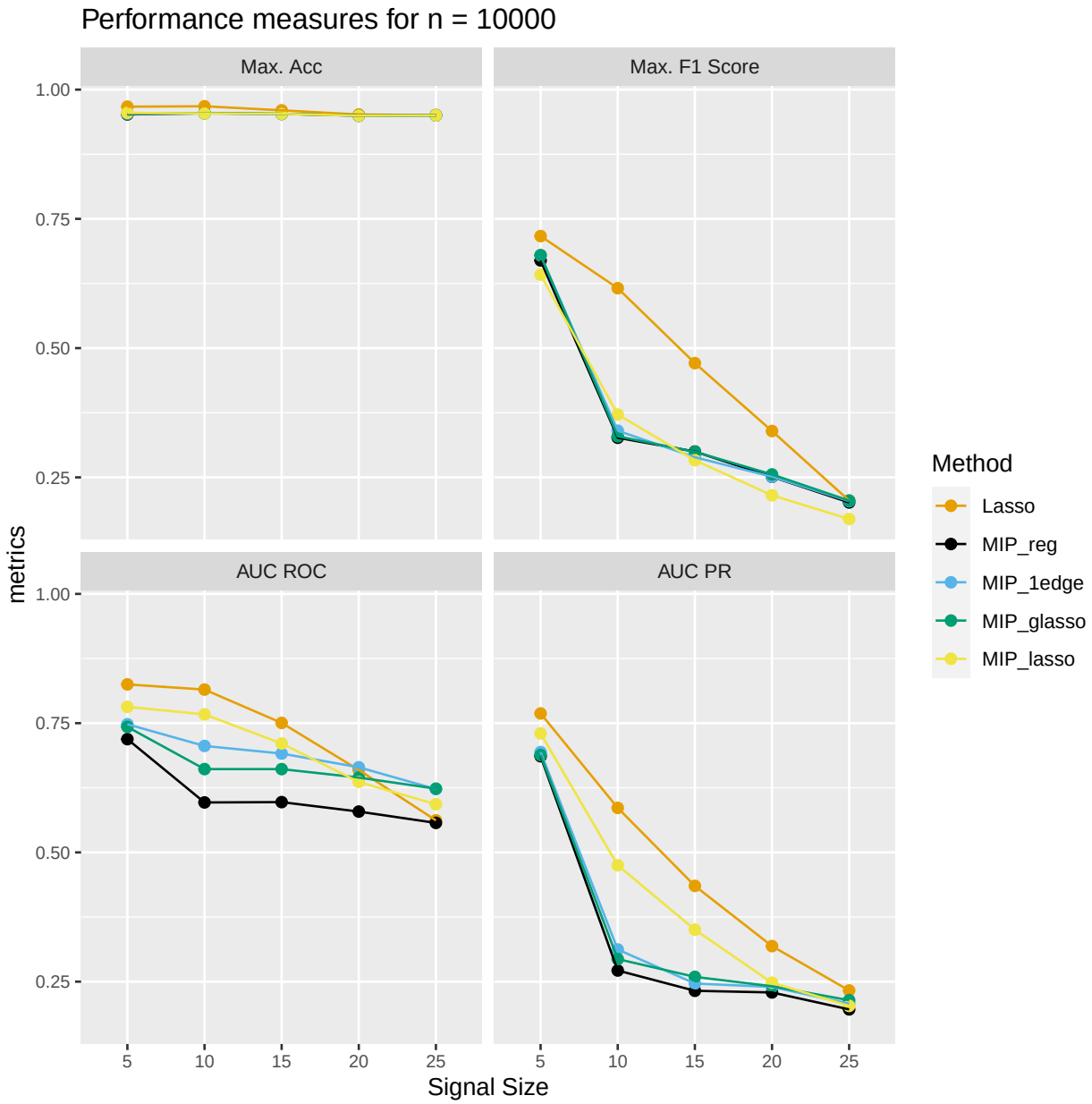


Figure 3.3 Variation of performance metrics with graph size for a total sample size of 10000 (training data size of 8000). The metrics were computed along the hyper-parameter path and are averaged over 100 different signals with edge probability of 5 %.

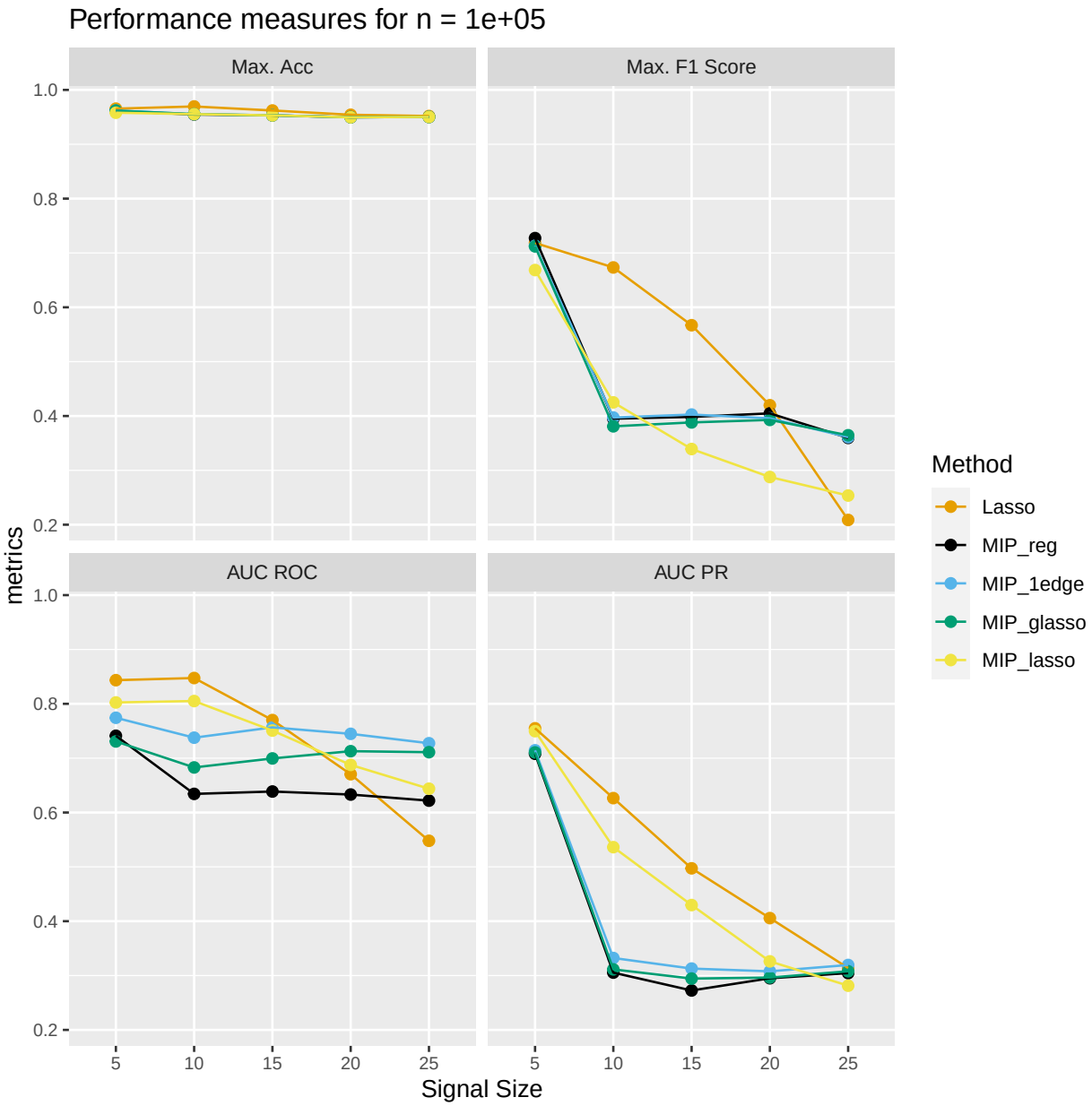


Figure 3.4 Variation of performance metrics with graph size for a total sample size of 100,000 (training data size of 80,000). The metrics were computed along the hyper-parameter path and are averaged over 100 different signals with edge probability of 5 %.

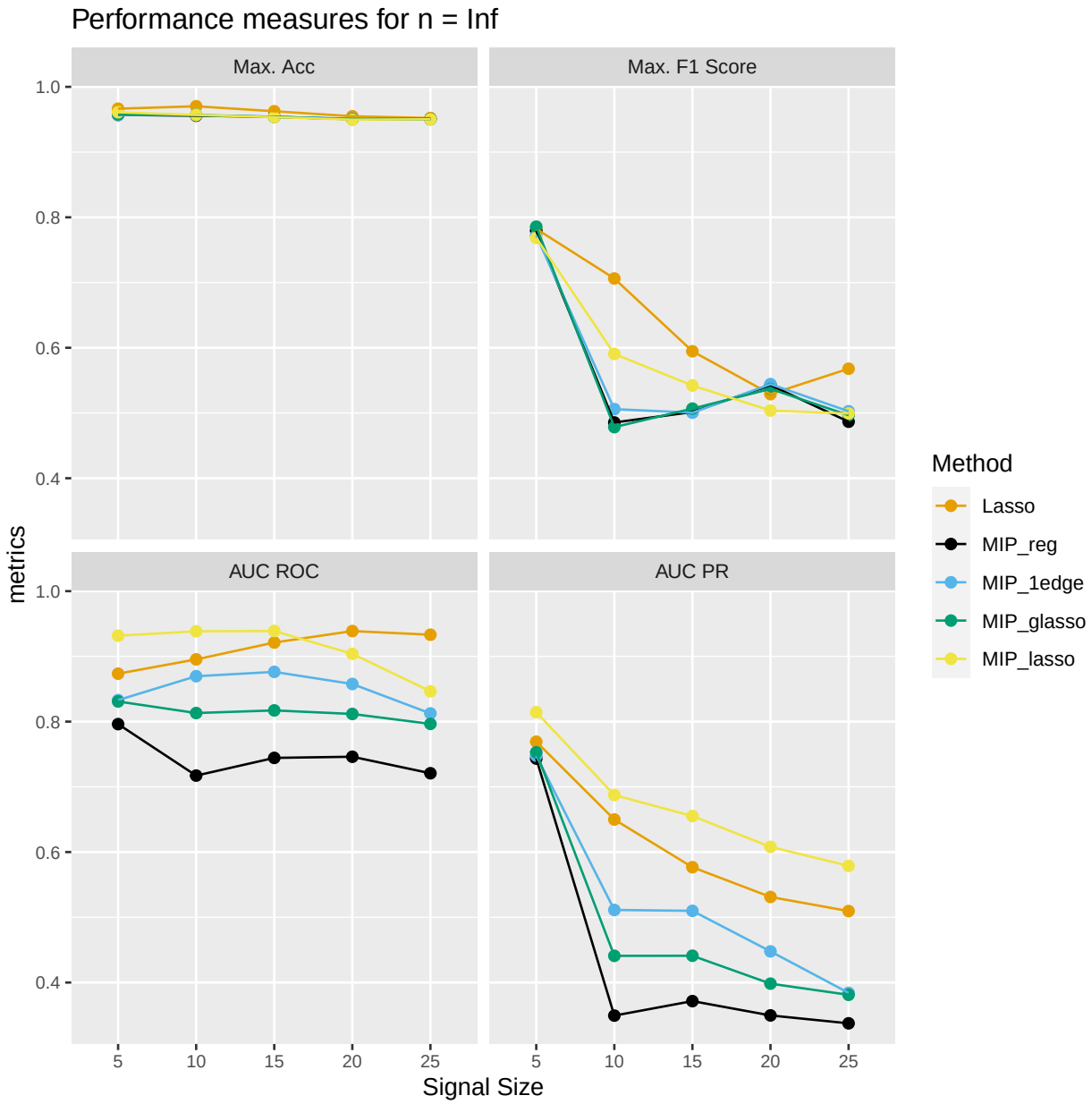


Figure 3.5 Variation of performance metrics with graph size directly using the population covariance matrix. The metrics were computed along the hyper-parameter path and are averaged over 100 different signals with edge probability of 5 %.

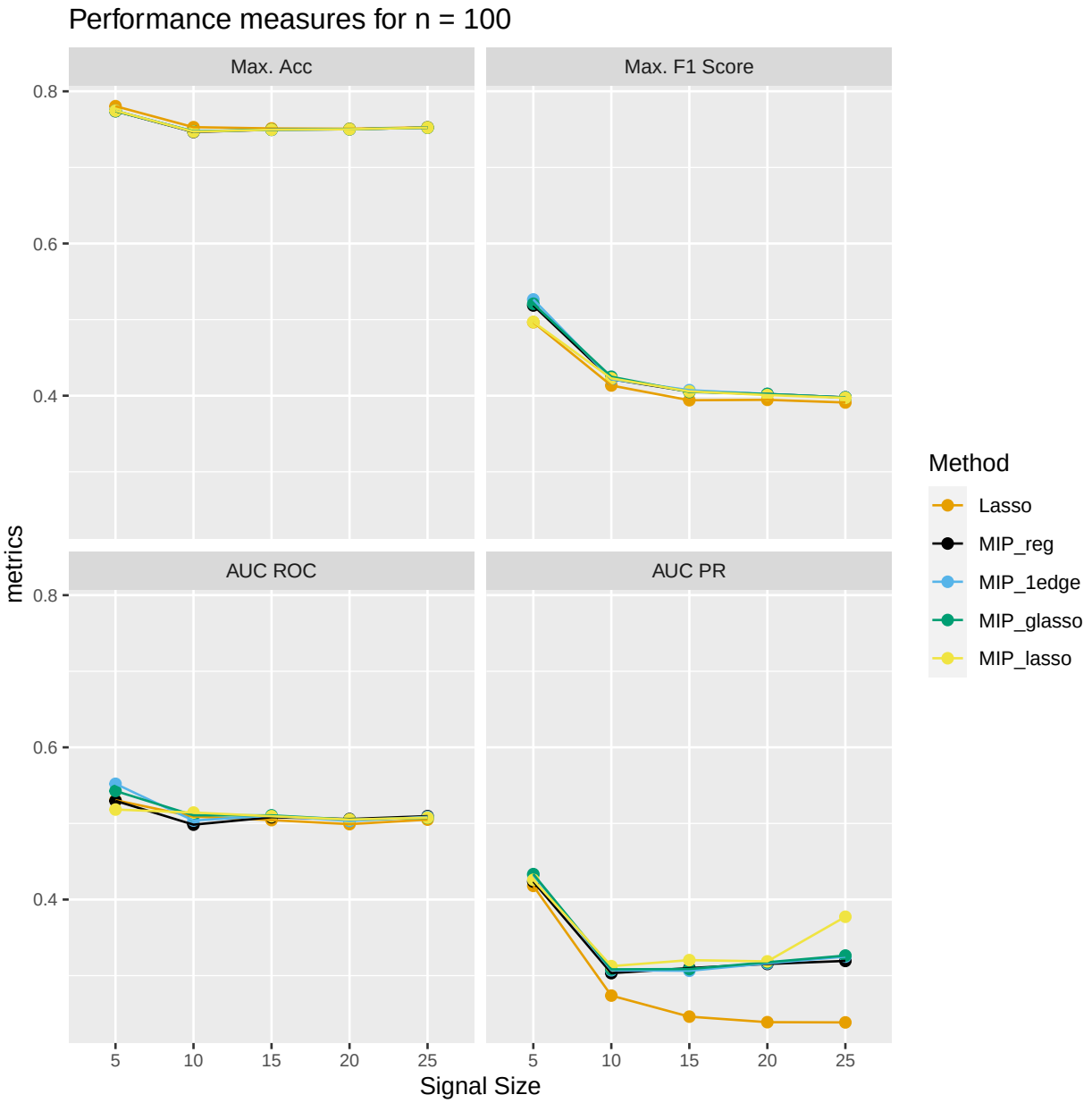


Figure 3.6 Variation of performance metrics with graph size for a total sample size of 100 (training data size of 80). The metrics were computed along the hyper-parameter path and are averaged over 100 different signals with edge probability of 25 %.

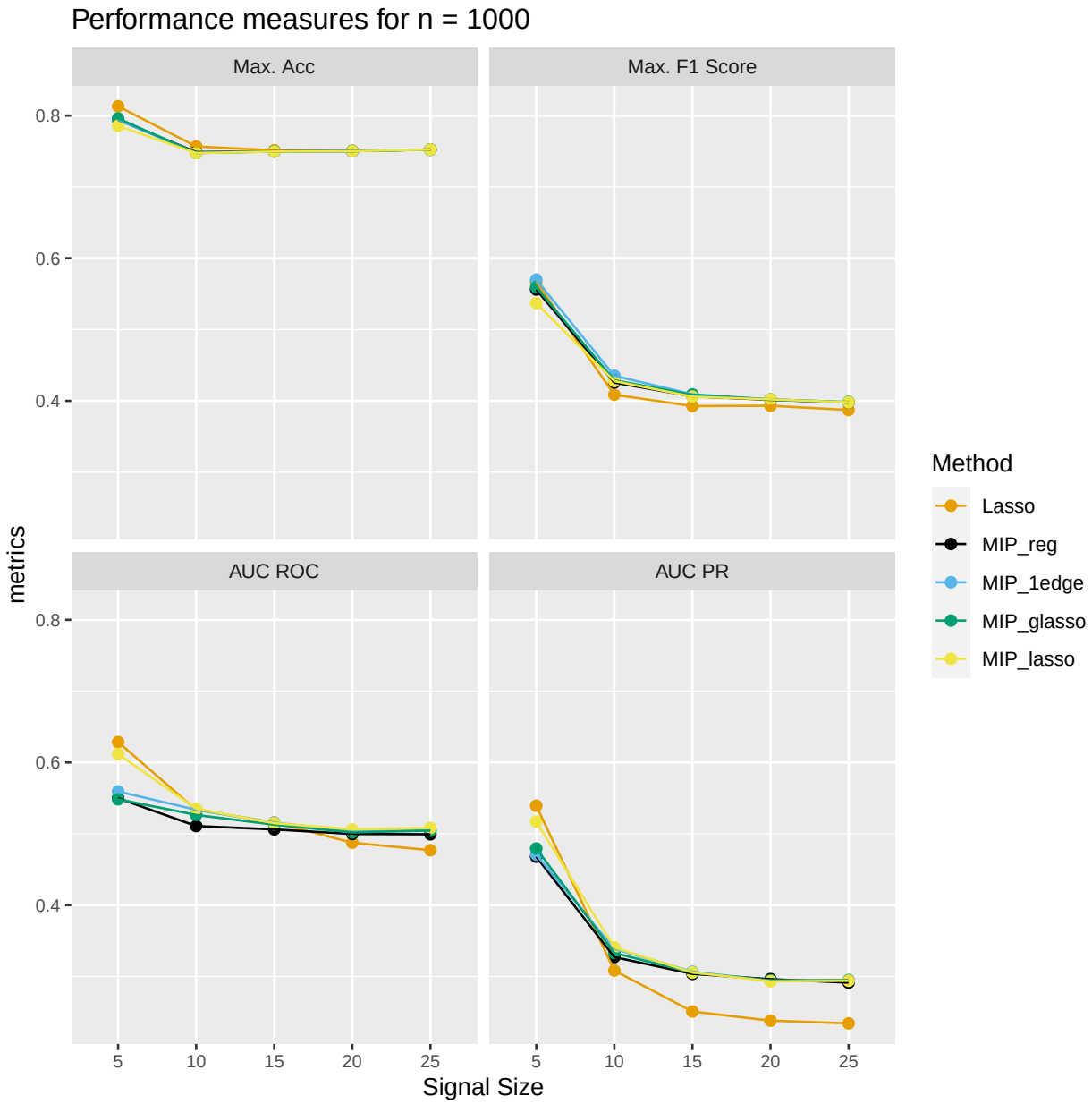


Figure 3.7 Variation of performance metrics with graph size for a total sample size of 1000 (training data size of 800). The metrics were computed along the hyper-parameter path and are averaged over 100 different signals with edge probability of 25 %.

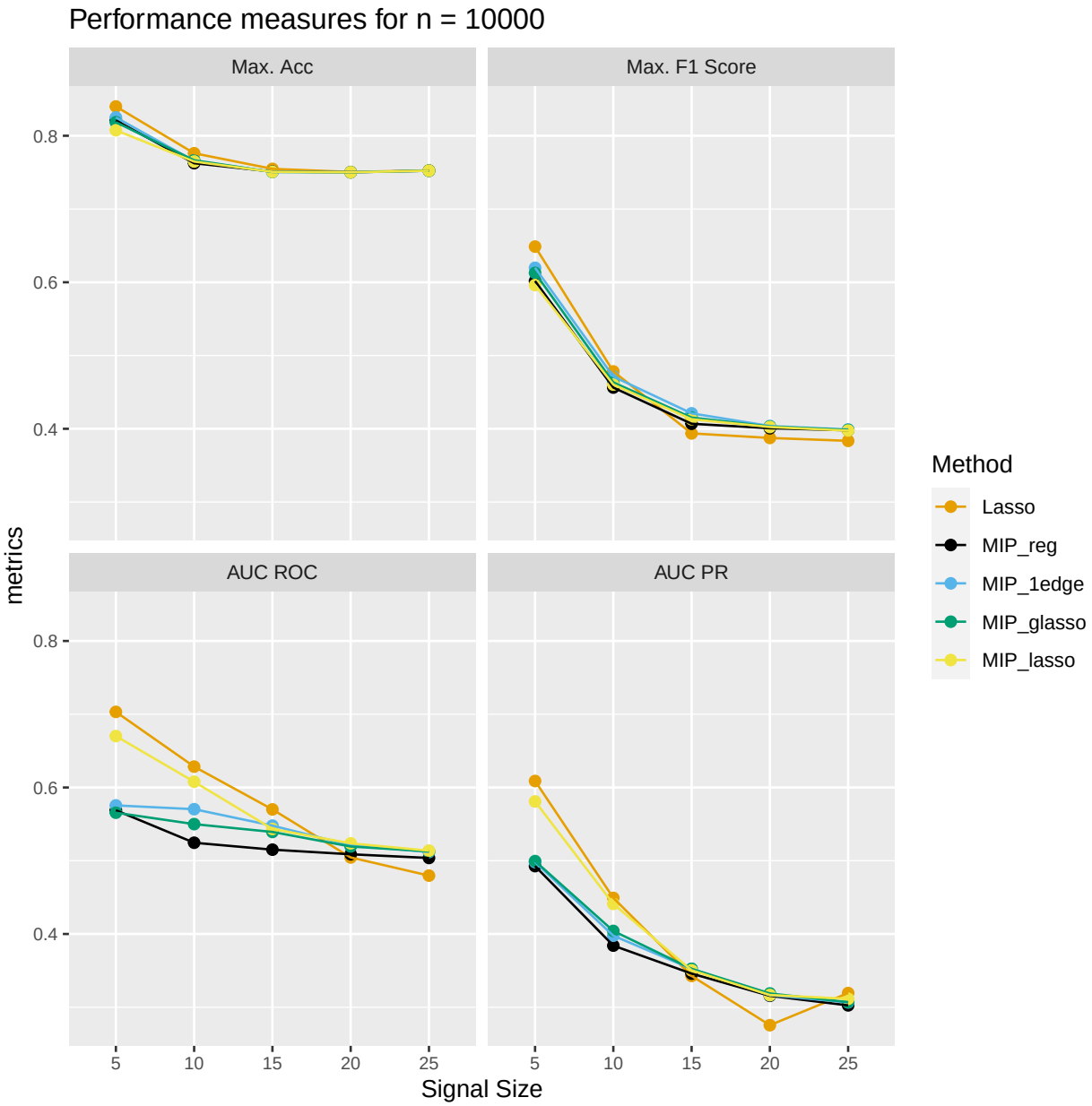


Figure 3.8 Variation of performance metrics with graph size for a total sample size of 10000 (training data size of 8000). The metrics were computed along the hyper-parameter path and are averaged over 100 different signals with edge probability of 25 %.

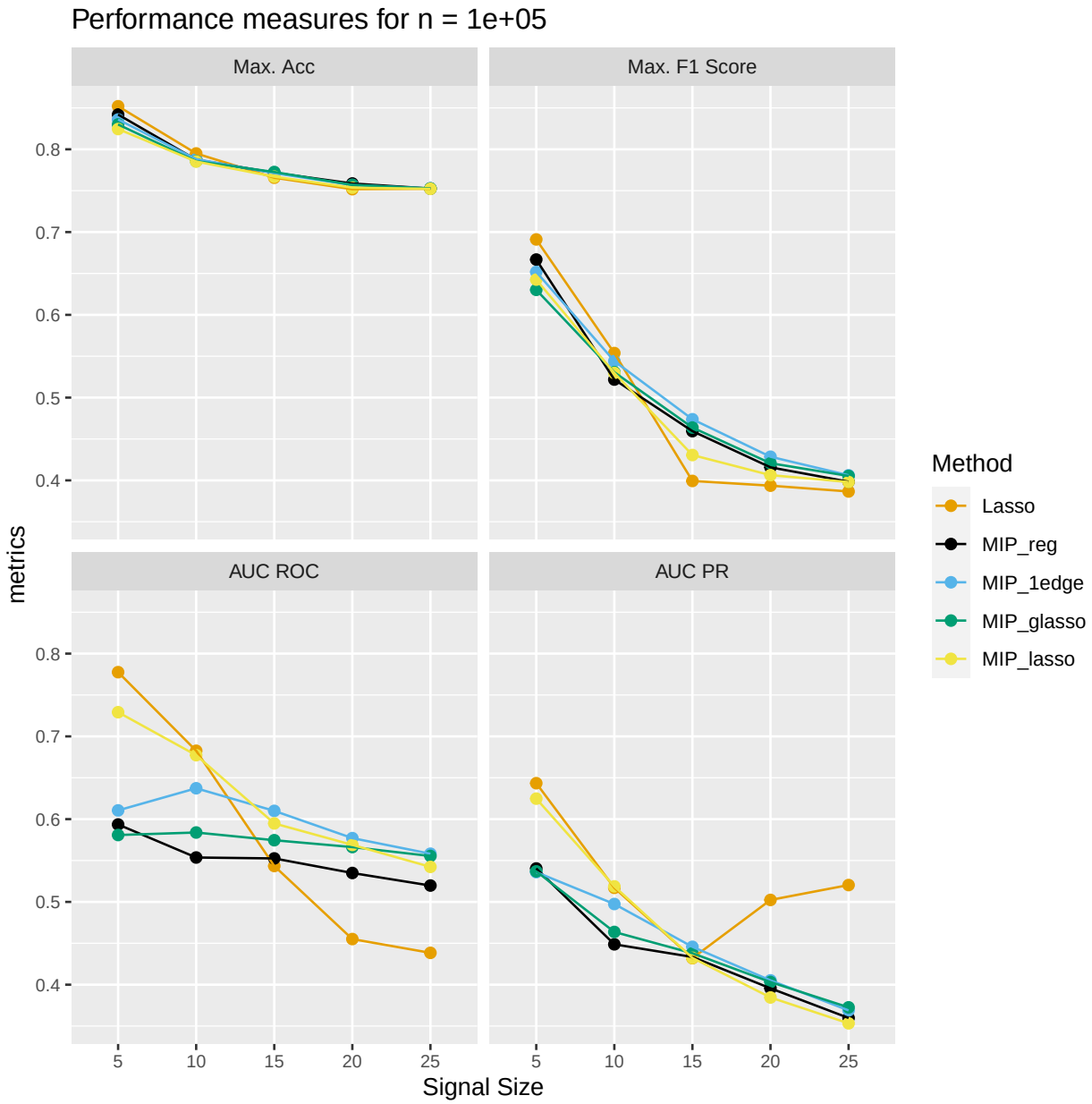


Figure 3.9 Variation of performance metrics with graph size for a total sample size of 100,000 (training data size of 80,000). The metrics were computed along the hyper-parameter path and are averaged over 100 different signals with edge probability of 25 %.

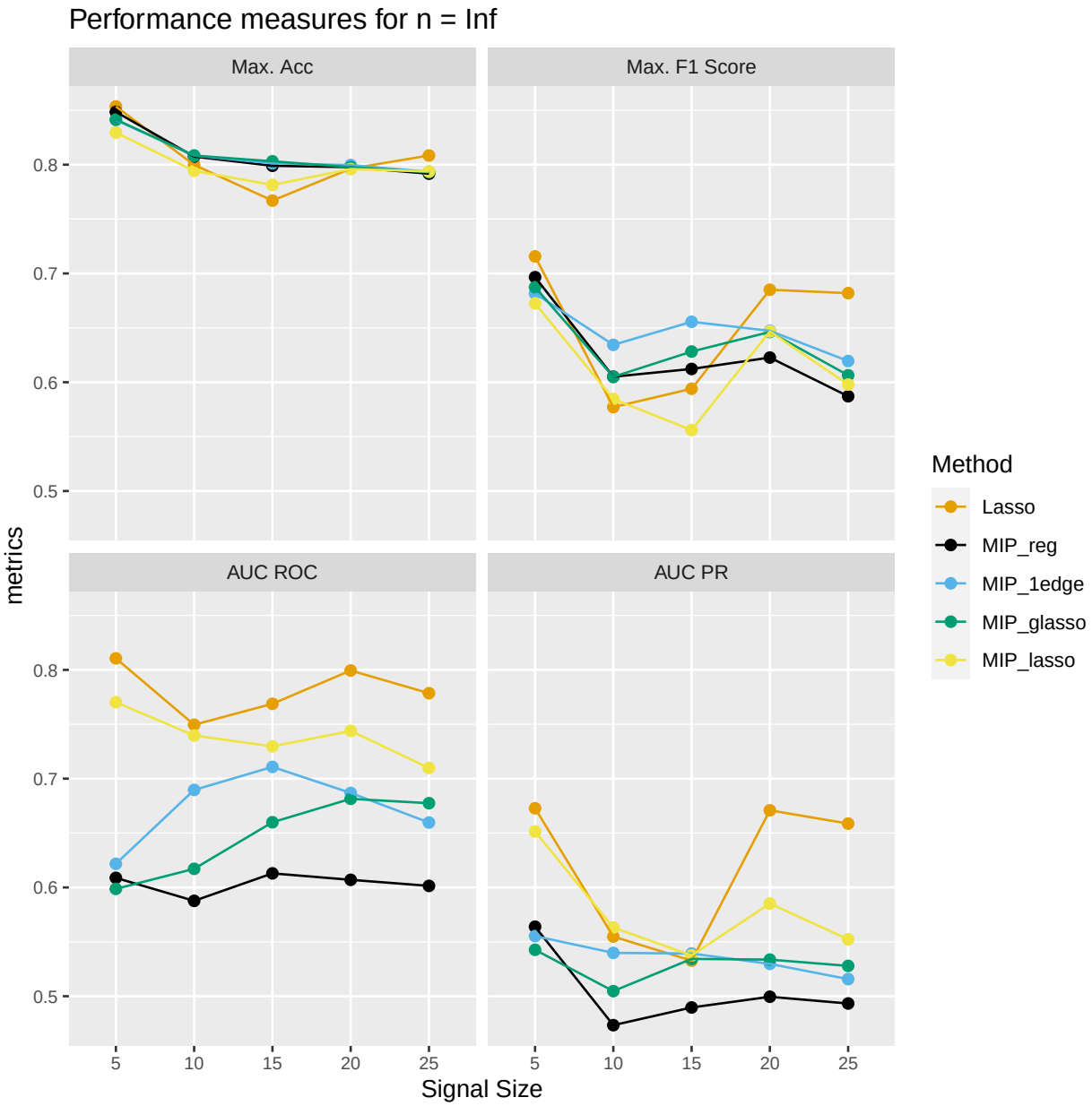


Figure 3.10 Variation of performance metrics with graph size directly using the population covariance matrix. The metrics were computed along the hyper-parameter path and are averaged over 100 different signals with edge probability of 25 %.

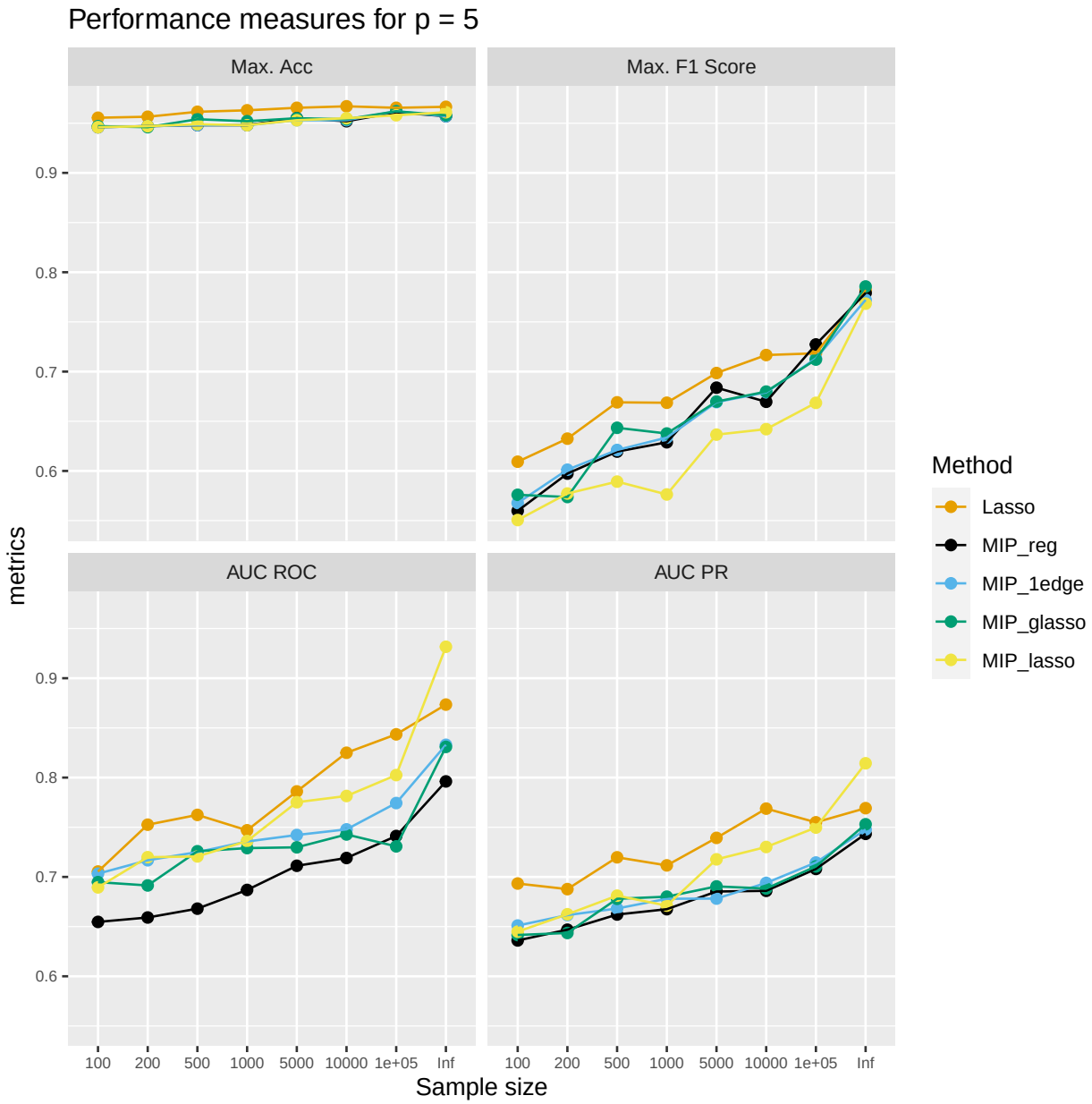


Figure 3.11 Variation of performance metrics with the total sample size (80 % training and 20 % test). The metrics were computed using the results from training along the hyper-parameter path and are averaged over 100 different signals with edge probability of 5 %.

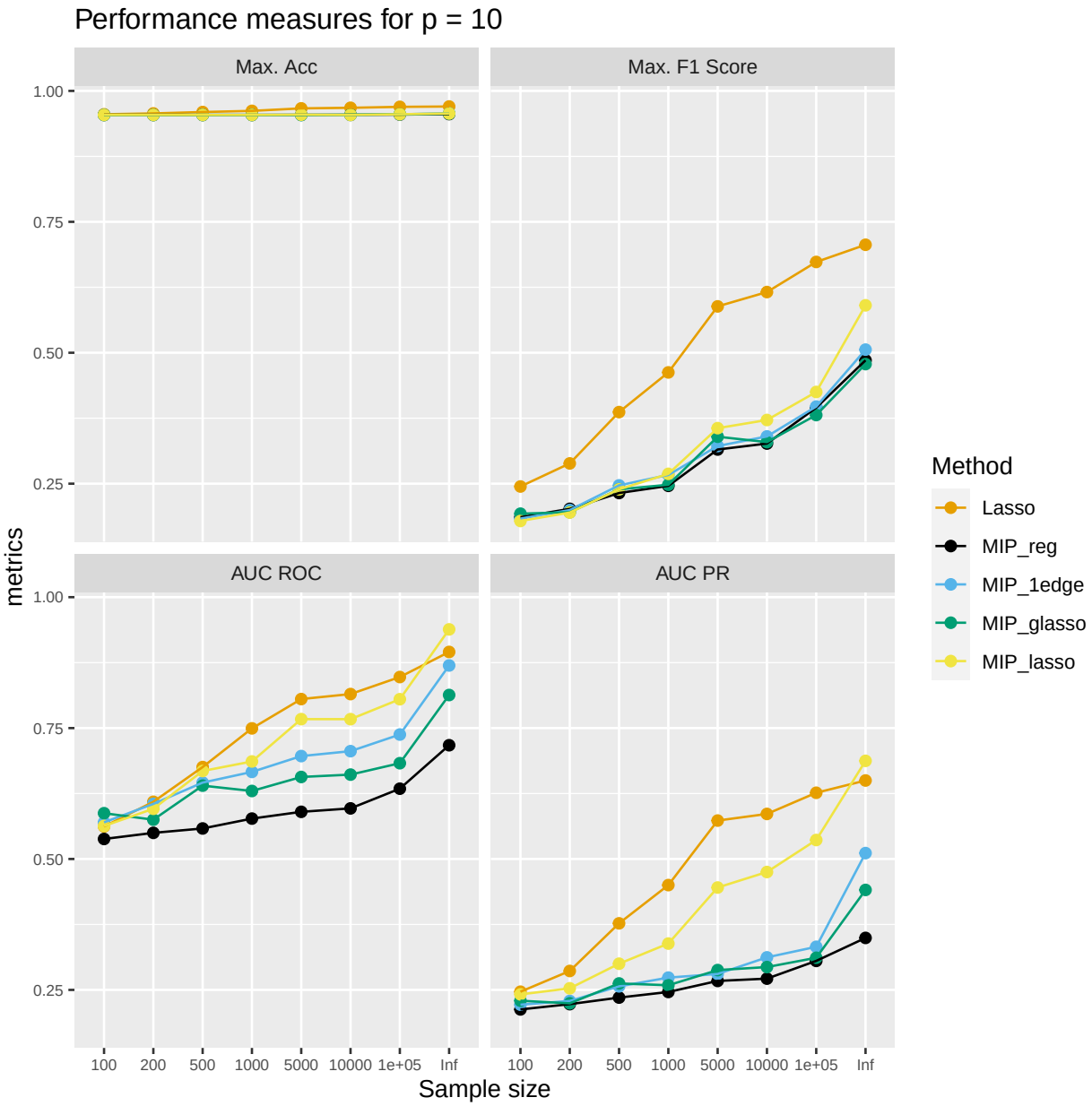


Figure 3.12 Variation of performance metrics with the total sample size (80 % training and 20 % test). The metrics were computed using the results from training along the hyper-parameter path and are averaged over 100 different signals with edge probability of 5 %.

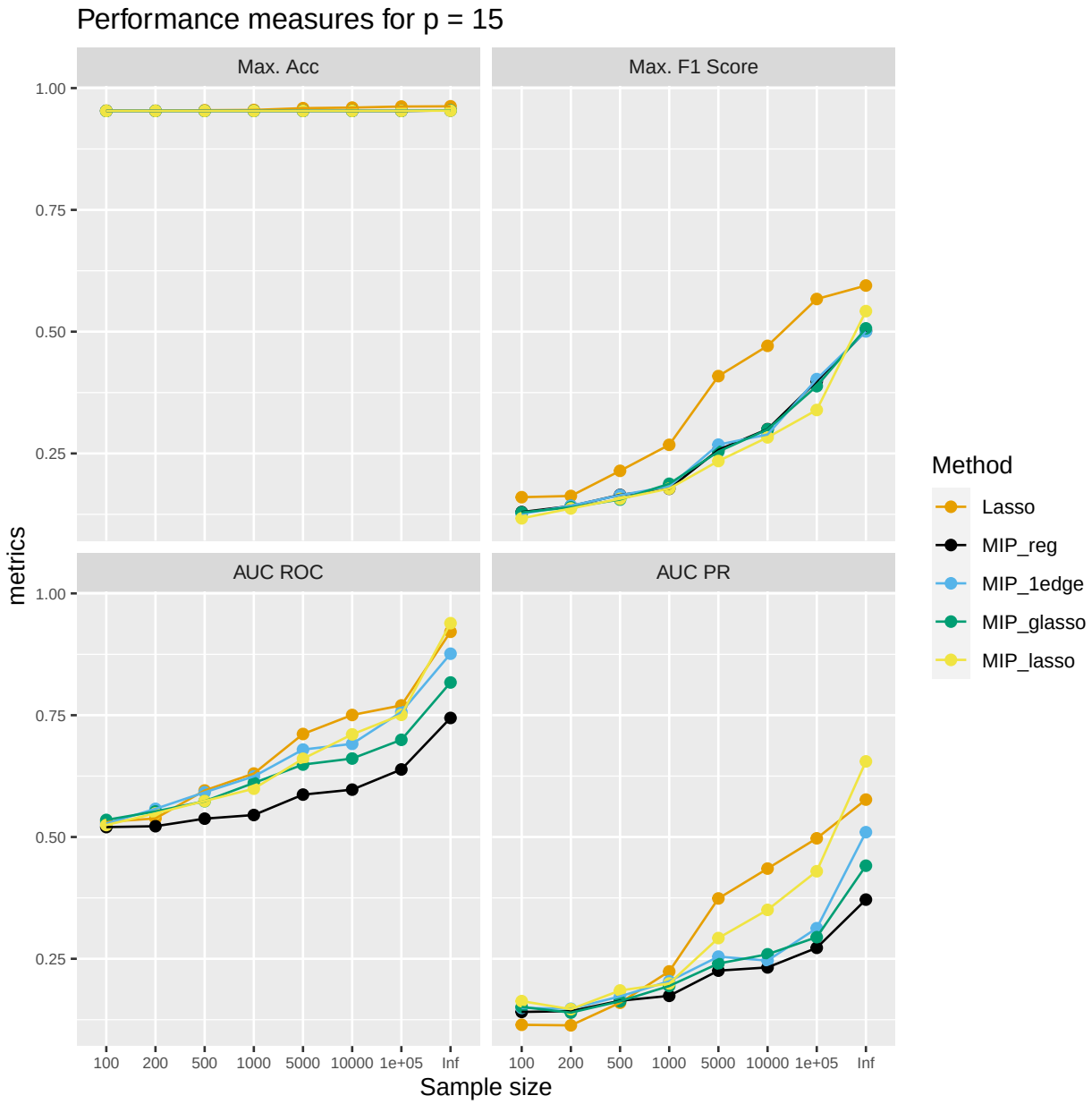


Figure 3.13 Variation of performance metrics with the total sample size (80 % training and 20 % test). The metrics were computed using the results from training along the hyper-parameter path and are averaged over 100 different signals with edge probability of 5 %.

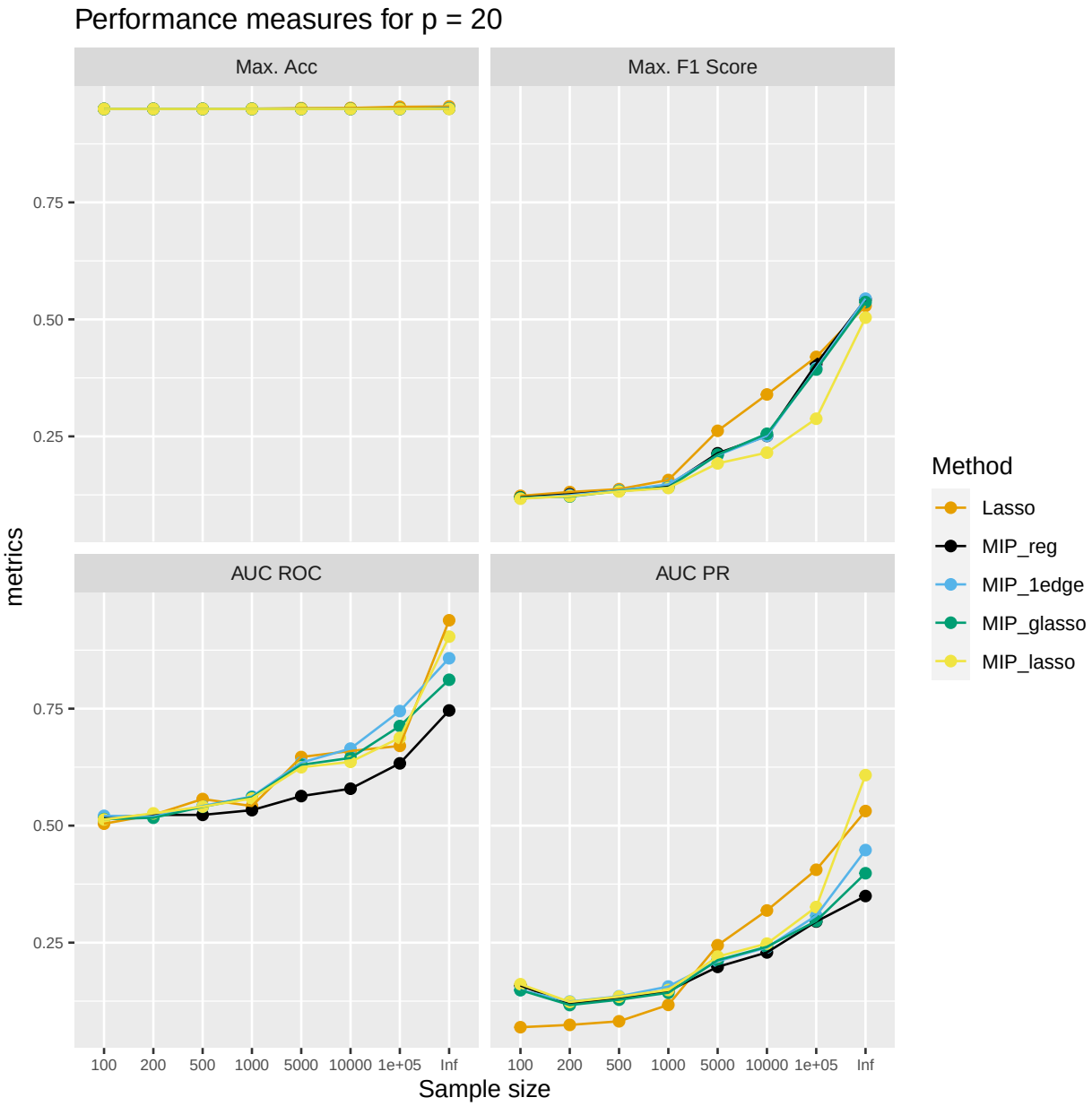


Figure 3.14 Variation of performance metrics with the total sample size (80 % training and 20 % test). The metrics were computed using the results from training along the hyper-parameter path and are averaged over 100 different signals with edge probability of 5 %.

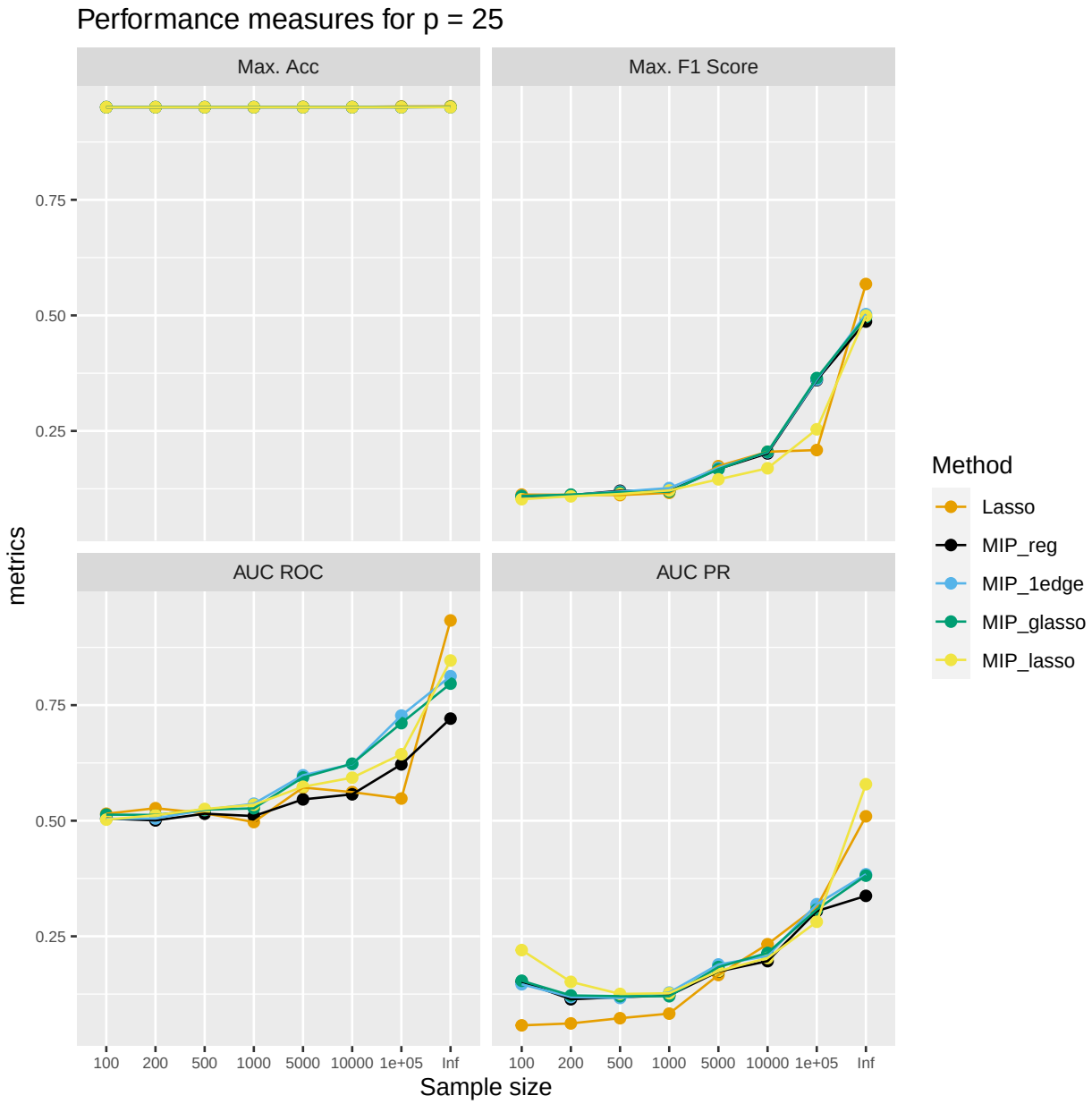


Figure 3.15 Variation of performance metrics with the total sample size (80 % training and 20 % test). The metrics were computed using the results from training along the hyper-parameter path and are averaged over 100 different signals with edge probability of 5 %.

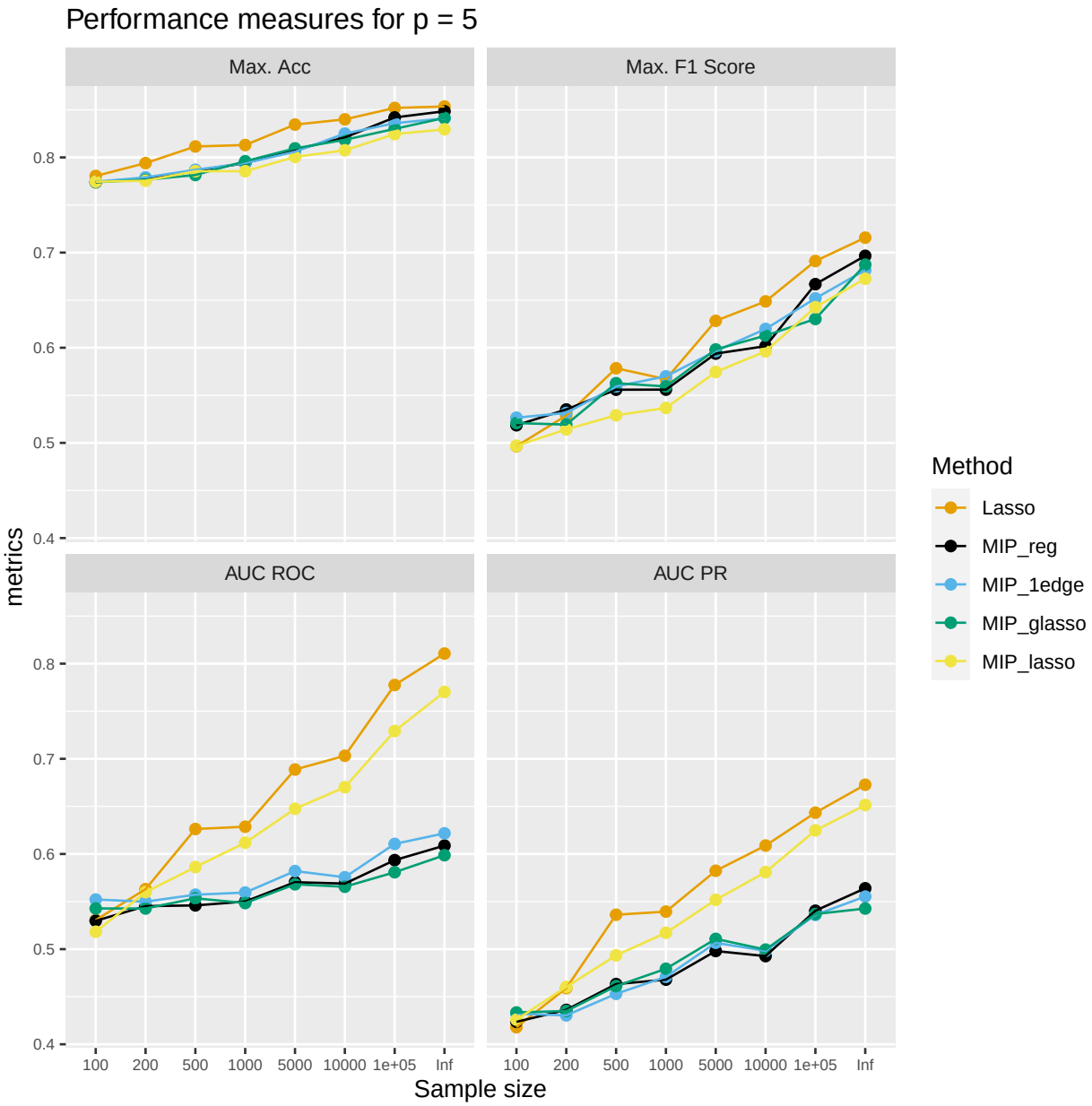


Figure 3.16 Variation of performance metrics with the total sample size (80 % training and 20 % test). The metrics were computed using the results from training along the hyper-parameter path and are averaged over 100 different signals with edge probability of 25 %.

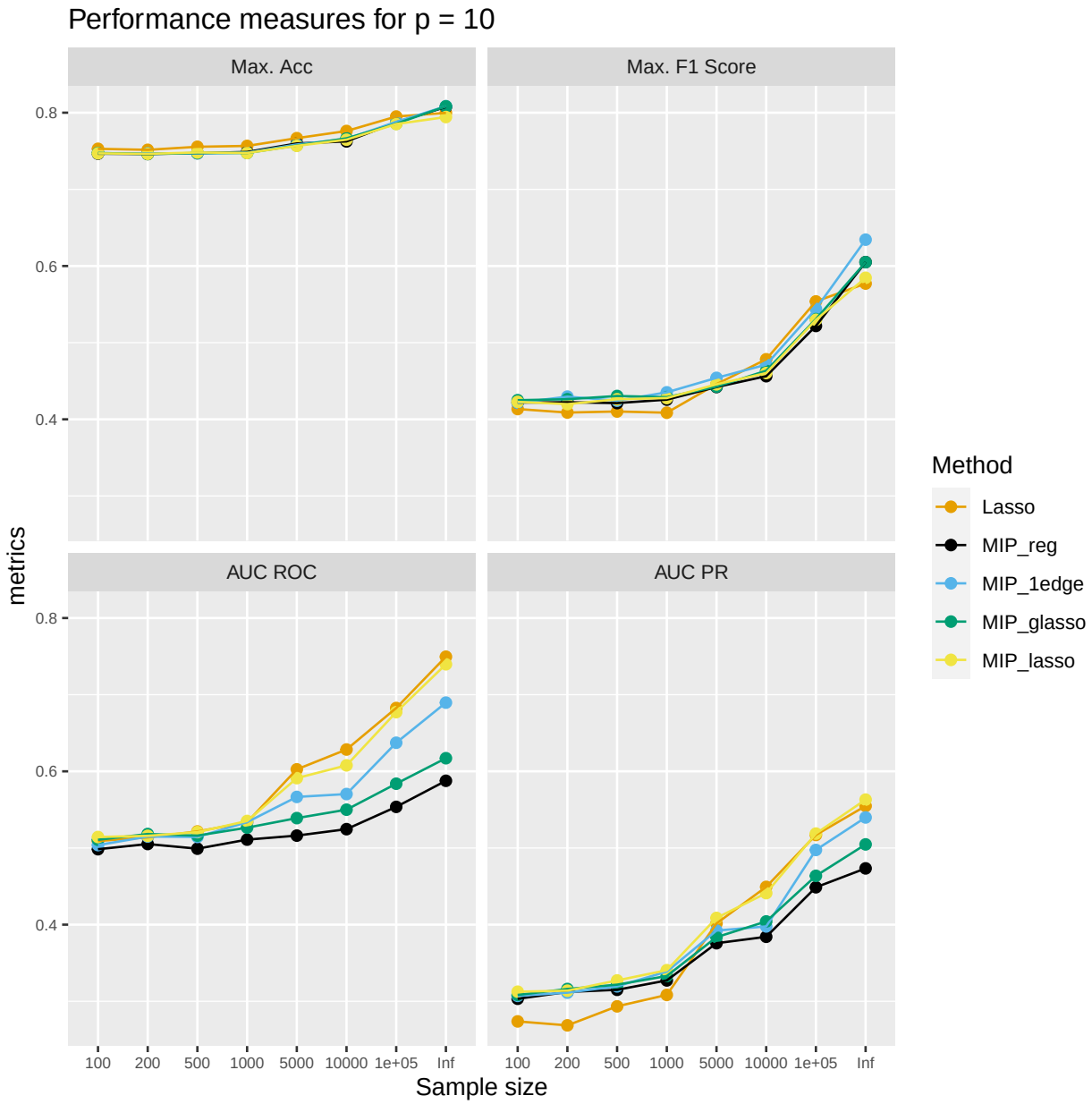


Figure 3.17 Variation of performance metrics with the total sample size (80 % training and 20 % test). The metrics were computed using the results from training along the hyper-parameter path and are averaged over 100 different signals with edge probability of 25 %.

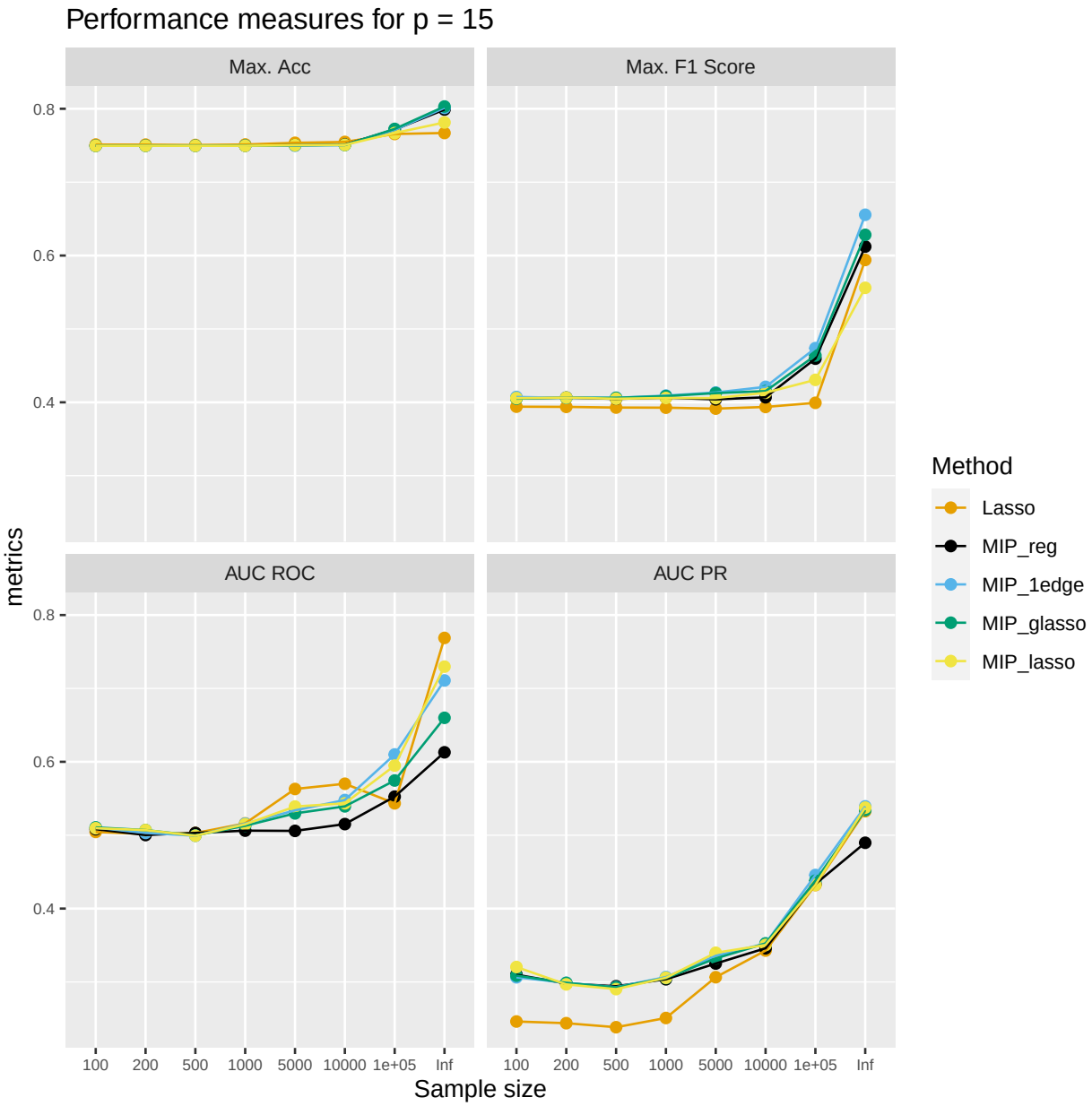


Figure 3.18 Variation of performance metrics with the total sample size (80 % training and 20 % test). The metrics were computed using the results from training along the hyper-parameter path and are averaged over 100 different signals with edge probability of 25 %.

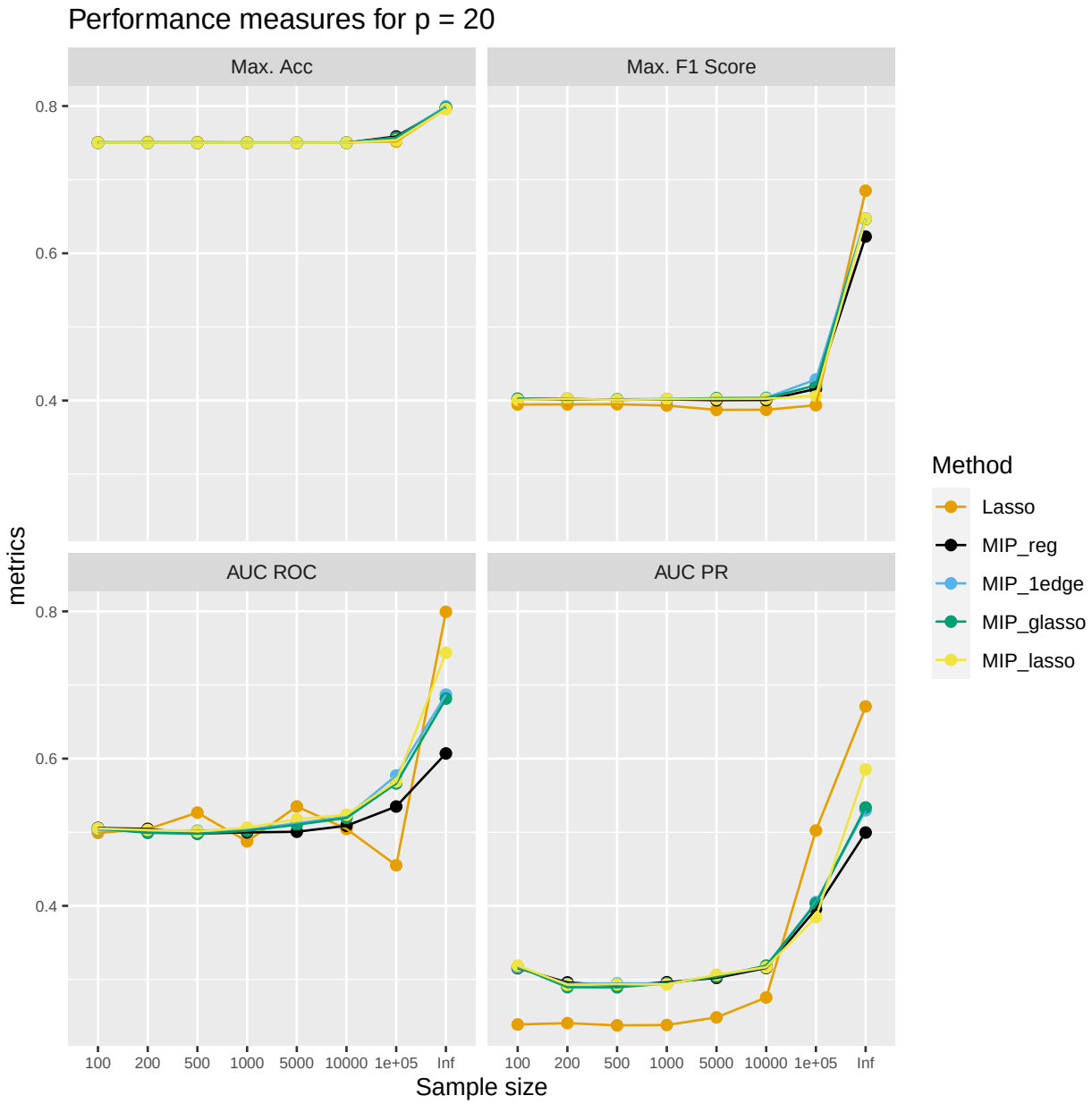


Figure 3.19 Variation of performance metrics with the total sample size (80 % training and 20 % test). The metrics were computed using the results from training along the hyper-parameter path and are averaged over 100 different signals with edge probability of 25 %.

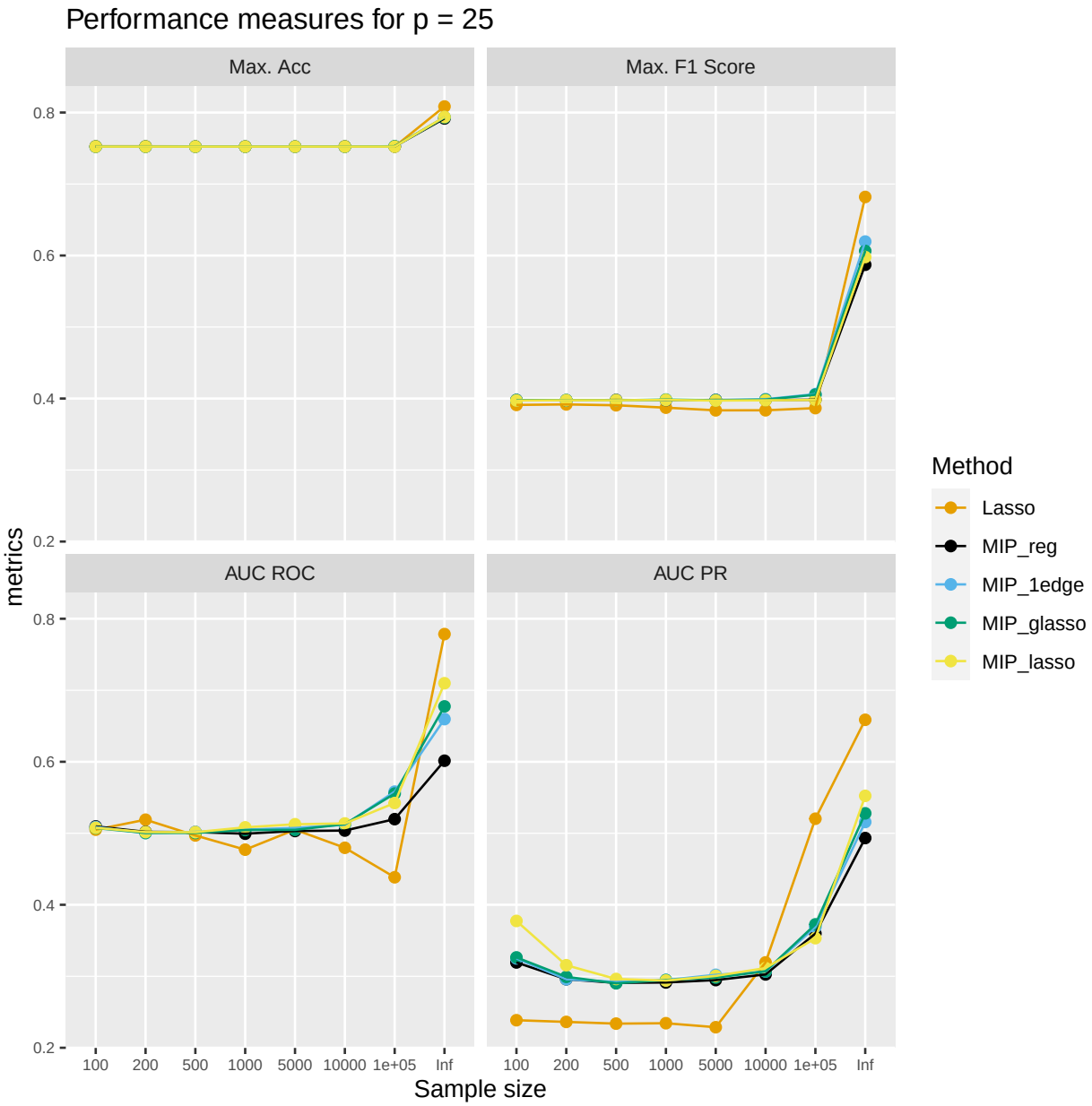


Figure 3.20 Variation of performance metrics with the total sample size (80 % training and 20 % test). The metrics were computed using the results from training along the hyper-parameter path and are averaged over 100 different signals with edge probability of 25 %.

3.1.3 Results: Model selection and prediction

Figures 3.21 to 3.23 show the performance metrics on test data after model selection using AIC, BIC, and eBIC respectively at a sample size of $n = 10,000$ and at edge probability of 5 %.

For $d = 0.05p$, parameters selected by AIC for lasso and BS_{lasso} give poor accuracy than other methods. This behaviour is attributed to an increase in number of FPs for small graph size ($p \leq 10$). The projected gradient descent initialization based BS approaches produce relatively less FPs in this region. But due to higher TPR of lasso and BS_{lasso}, all the methods are relatively similar in terms of F1-Score and AUC PR.

As the graph size increases, all the metrics decrease. Lasso dominates all the methods in terms of F1-Score due to decrease in number of FPs (and thus, relatively higher precision and recall). AUC PR values are less than or equal to 0.5 for all the methods when $p \geq 10$. AUC ROC values too drop to 0.5 beyond $p \geq 20$. This highlights the difficulty in recovering signals beyond $p \geq 20$. Lasso is slightly more superior in discriminating between true edges and non-edges as seen in the AUC ROC plot. All the projected gradient descent initialization based BS methods perform alike. BS_{reg} is the only BS method that has a TPR value of greater than zero for $p \geq 20$. It is also slightly superior than other projected gradient descent initialization based BS methods in the TPR plot in general. However, both of this comes at the cost of having many FPs (see Figure 3.21). But FPR rate of BS_{reg} is very robust in comparison to others across different signal sizes.

All the methods give similar accuracy with BIC and eBIC (see Figures 3.22 and 3.23). In fact, The results from BIC and eBIC are very similar. Lasso clearly dominates all the other methods in terms of precision and recall. The TPR and FPR values are lower than in AIC. This is expected, as the AIC has a weaker penalty on the sparsity compared to BIC and eBIC. However, this leads to no edge detection beyond $p \geq 20$. The AUC PR plots show similar performance for all the models selected by BIC and eBIC. Lasso's performance in the AUC ROC plot seems to be similar to that when AIC was used. But all of the methods except lasso degrade in performance quickly than when AIC was used. Lasso has relatively slower decay rate due to its good TPR-FPR trade-off. Though BS drops to an AUC ROC value of 0.5 at $p = 10$, the same drop happens with lasso only beyond $p = 20$.

When $d = 0.25p$, the results are similar but more prominent with AIC. Lasso and BS_{lasso} perform very poor in terms of accuracy (and precision) due to high number of FPs for signals with $p \leq 15$ (see Figure 3.24). Both of these methods dominate the TPR plot again. Among projected gradient descent initialization based methods, BS_{reg} is again the superior performer. But this is again because it selects many FPs. However, the FPR is more robust for this method in comparison to lasso or BS_{lasso} across different signal sizes.

In general with increasing signal size, none of the methods detect any edges beyond $p \geq 15$. This is worse than $d = 0.05p$ case. BS_{reg} is the only method that has a positive TPR in this region. While AUC ROC values dropped to 0.5 beyond $p = 10$, AUC PR values are always greater than or equal to 0.5 across all signal sizes. This indicates relatively good ability in detecting true edges in comparison to $d = 0.05p$ case. But they do not differentiate between TP and TN that well. Moreover, this is not very significant and could be treated as an interpolation error. The performance of all the methods are still very poor and no better than a random classifier. It was observed that the solutions were mostly diagonal matrix in many cases with no edges. For BS, the poor performance could also be attributed to the accepted sub-optimal solutions due to the time-limit.

With BIC and eBIC, the situation is less severe than with AIC, but more than $d = 0.05p$ case. Lasso's performance in terms of accuracy is not good in comparison to the rest. For small graph size ($p = 5$), it still chooses high number of FPs. All the methods perform similar in terms of F1-Score. None of the methods detect any edges beyond $p \geq 15$. Even BS_{reg} is only marginally able to detect something and its TPR value tends to zero. The AUC PR values are however, above 0.5

across all signal sizes. But this is only due to the fact that there are simply more true edges in the graphs in general, so precision-recall trade-off is slightly better than in $d = 0.05p$ case. Of course, since AUC PR values are only upper bound to the actual value due to approximations done by interpolation, this seems reasonable. The AUC ROC values are similar for all methods and are 0.5 beyond $p = 10$, which indicates that the methods are still not better in identifying true edges from non-edges.

3.1.4 Plots and Summary: Model selection and prediction

In general, the results from the prediction can be summarized as follows:

- Lasso and BS_{lasso} have poor performance in terms of accuracy and precision for $p \leq 10$ and it is much more noticeable in graphs with $d = 0.25p$. Lasso has good TPR-FPR trade-off. As signal size increases, it continues to have the highest recall rate among all methods while having low FPR. Its AUC ROC value drops to 0.5 only when $p \geq 20$.
- All the BS methods have slightly lower AUC ROC values and drop to 0.5 at $p = 15$. Furthermore, all the methods have similar performance in terms of precision-recall trade-off. Every projected gradient based BS methods, except BS_{reg}, have similar performance in general. BS_{reg}'s typically has high FPR than other BS methods. Beyond $p = 10$, none of the methods are actually very good in performance as seen in the AUC plots.
- BIC and EBIC give sparser solutions as expected in comparison to AIC. However, for large signal sizes, the model chosen by them do not detect any edges beyond $p = 20$. In this case, AIC is better. Even then, only BS_{reg} is able to detect some edges. But of course, it comes with a cost of more FPs.

3.1.5 Results: Computational Time

All simulations were performed on `CoolMUC-2` linux cluster with Intel Xeon E5-2690 v3 (Haswell based nodes) having 28 core per node and 64 GB RAM per node at Leibniz-Rechenzentrum (LRZ) super computing facility (`www.lrz.de`) in Technical University of Munich (TUM), Germany. Signals with $p \leq 15$, were run using a single core on a shared memory node. Due to high memory requirements, signals with $p > 15$ were run as MPI jobs using inter-node parallelization [Lei23].

As seen in tables 3.1 and 3.2, the power of parallel computing has really made BS a practically feasible alternative. In general, among all the BS methods, BS_{lasso} clearly seems to be the fastest. This indicates that the bounds provided by lasso solutions lead to quick closing of optimality gap, and thus shorter MIO training time. BS_{reg} seems to be relatively faster than all the other projected gradient descent initialization based BS methods. $\text{BS}_{\text{glasso}}$ and BS_{1edge} are slower in general. However, for large signal sizes, it seems to be faster than (or similar to) BS_{reg}. For $d = 0.05p$, this effect is really noticeable. But when $d = 0.25p$, it is not so clear to see this. In general, $d = 0.05p$ computations are faster than $d = 0.05p$ case.

It is important to note the increase in computation time for large graph sizes ($p \geq 15$) and low sample sizes ($n \leq 1000$). This increase is due to the fact that large proportion of problems within this category cross the time-limit before fully closing the optimality gap. This is, in particular, true for medium range of values of k from the grid $[p, p^2]$. In such cases, only sub-optimal results are returned by Gurobi.

Nevertheless, lasso is still the fastest among all. It is important to mention that while BS trains over 10 different values of k, lasso trains over 100 different values of λ (twice, with coarse and fine grid - see Algorithm 5) within a time 10 times faster than BS, even with parallelization. For $p = 25$, one can see that lasso and BS are comparable. However, it is important to note that, signals with

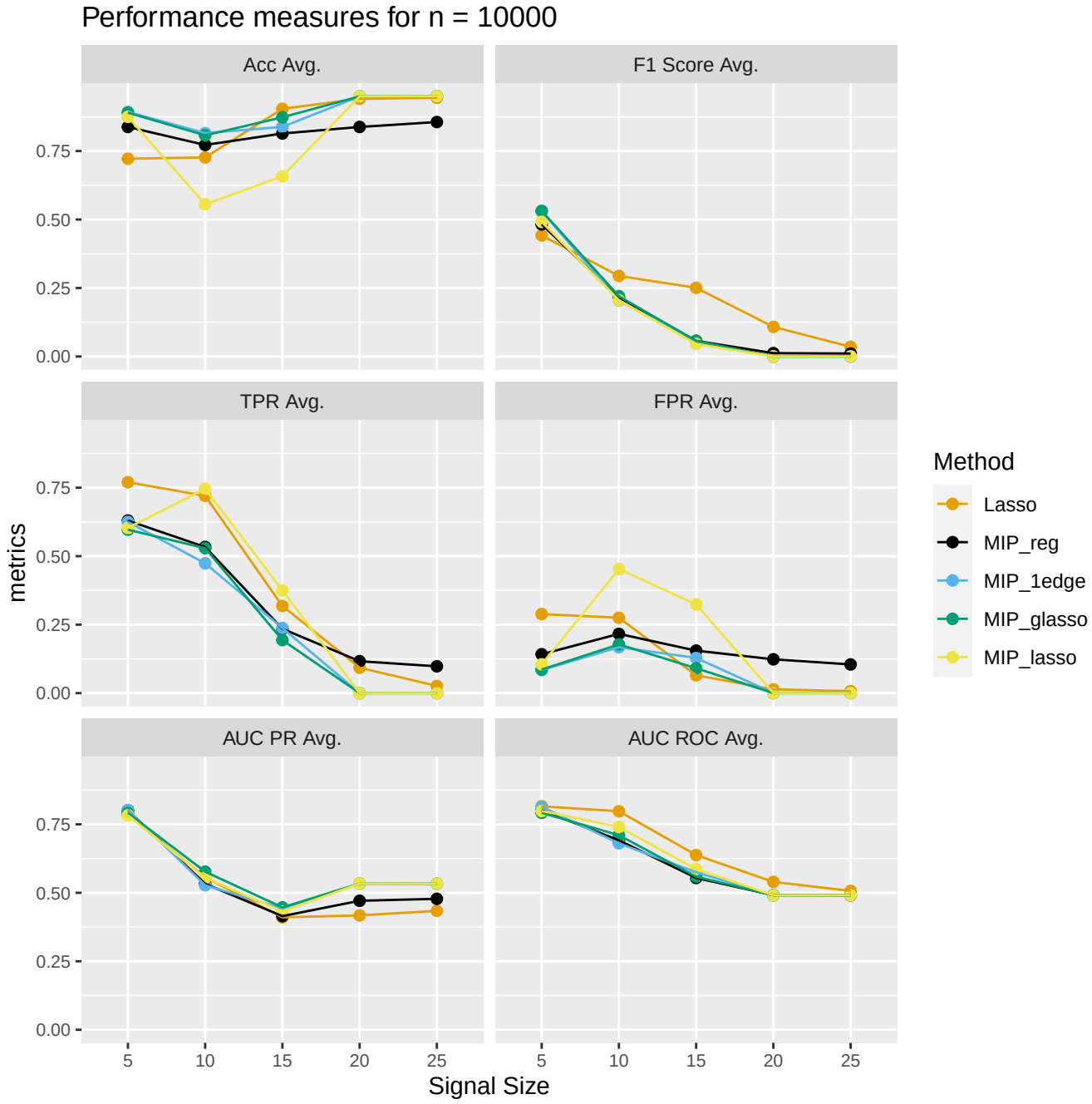


Figure 3.21 Variation of performance metrics with graph size and at edge probability 5 % on unseen dataset drawn from a sample size of 10,000 (test data size 20 %). The metrics are computed for graphs selected via AIC and are averaged results over 100 different signals.

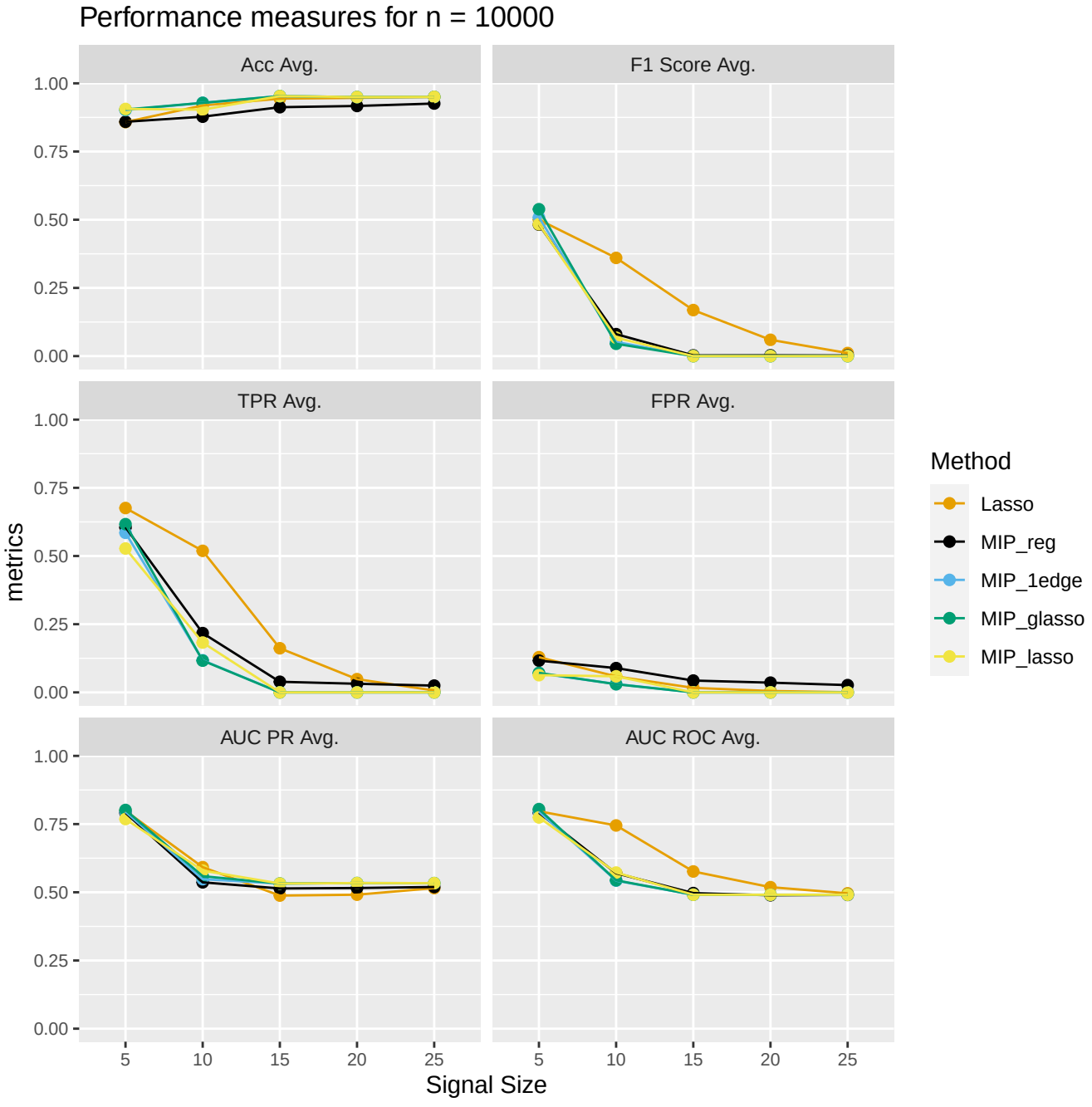


Figure 3.22 Variation of performance metrics with graph size and at edge probability 5 % on unseen dataset drawn from a sample size of 10,000 (test data size 20 %). The metrics are computed for graphs selected via BIC and are averaged results over 100 different signals.

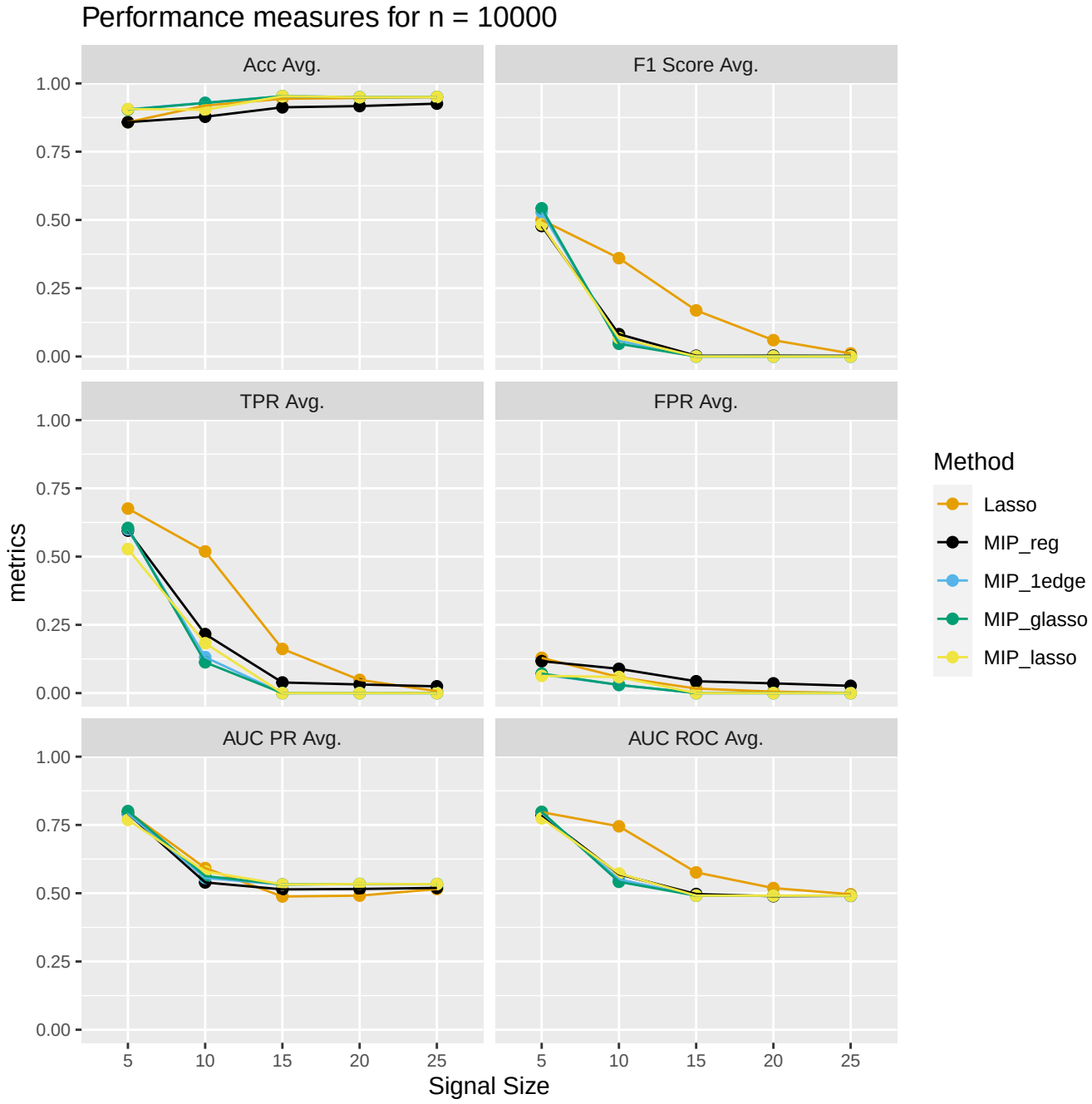


Figure 3.23 Variation of performance metrics with graph size and at edge probability 5 % on unseen dataset drawn from a sample size of 10,000 (test data size 20 %). The metrics are computed for graphs selected via EBIC and are averaged results over 100 different signals.

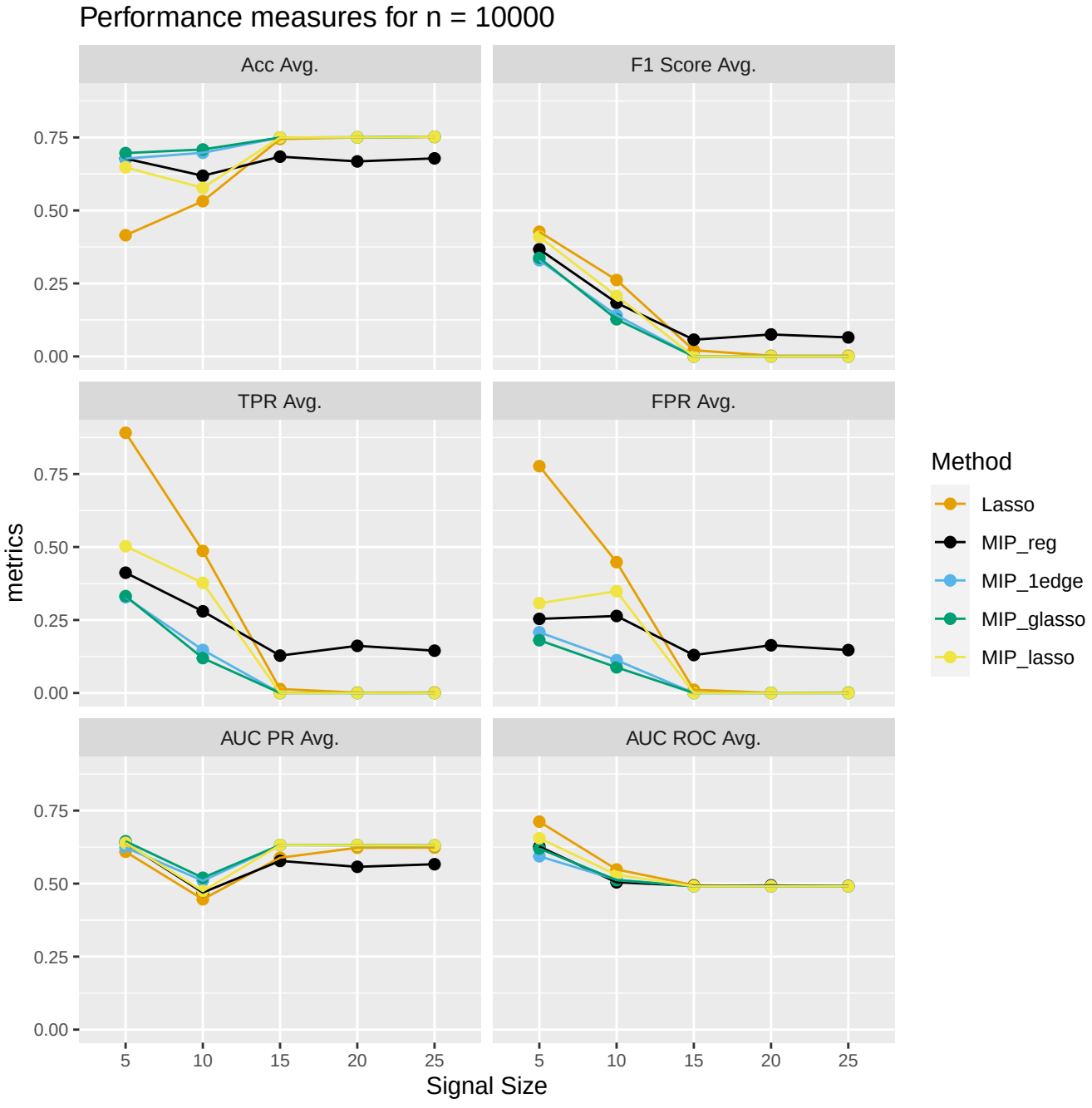


Figure 3.24 Variation of performance metrics with graph size and at edge probability 25 % on unseen dataset drawn from a sample size of 10,000 (test data size 20 %). The metrics are computed for graphs selected via AIC and are averaged results over 100 different signals.

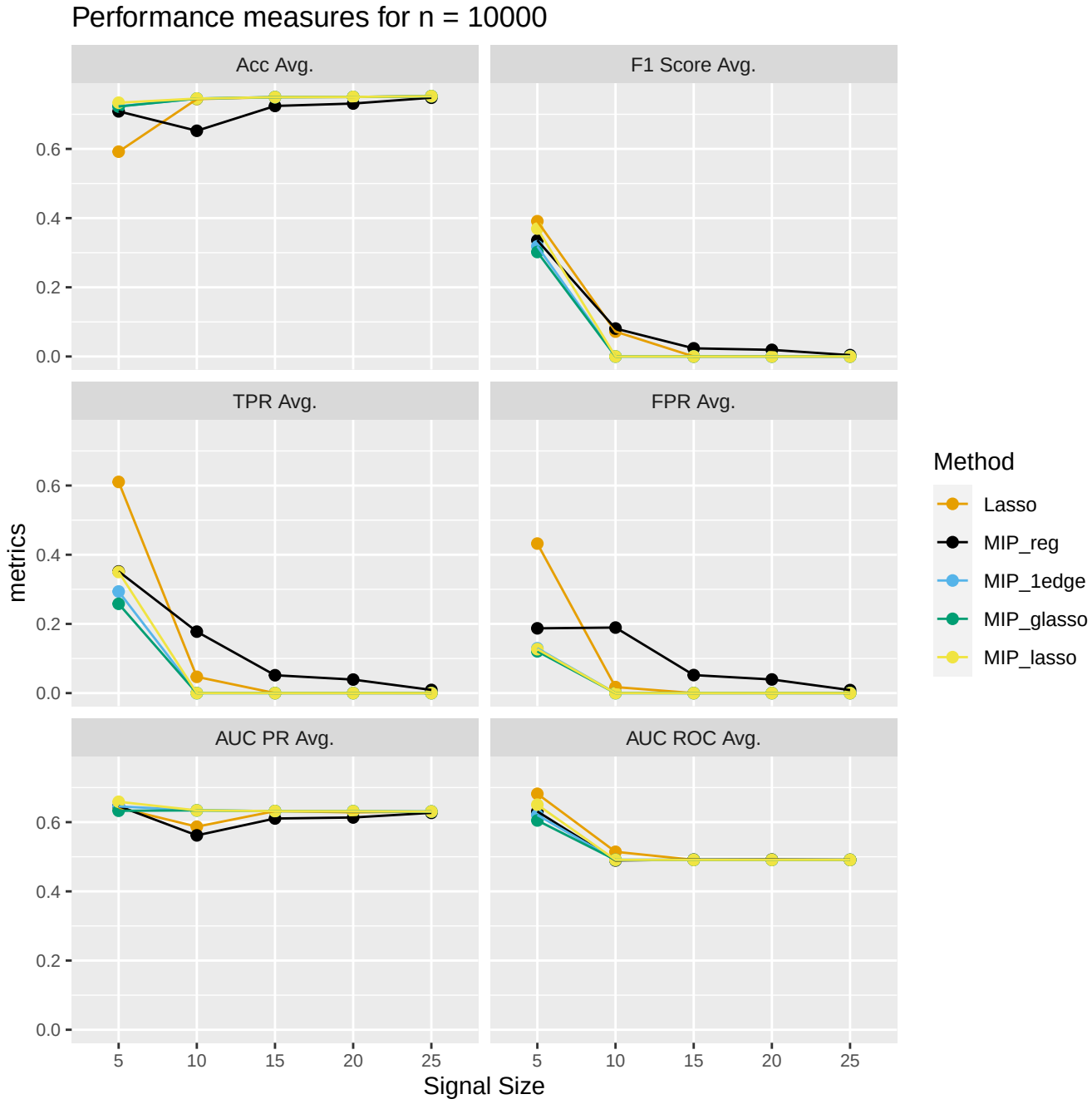


Figure 3.25 Variation of performance metrics with graph size and at edge probability 25 % on unseen dataset drawn from a sample size of 10,000 (test data size 20 %). The metrics are computed for graphs selected via BIC and are averaged results over 100 different signals.

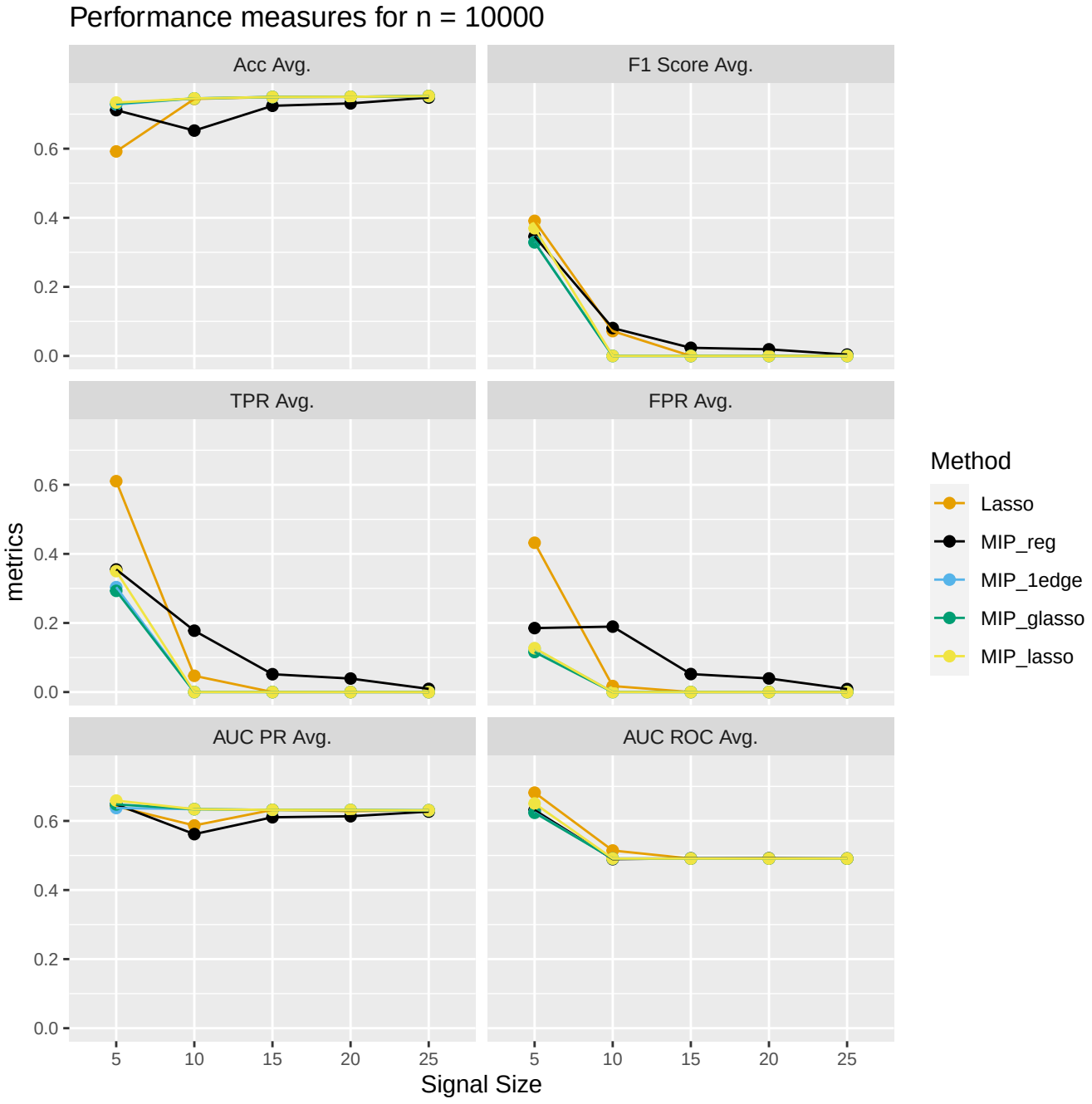


Figure 3.26 Variation of performance metrics with graph size and at edge probability 25 % on unseen dataset drawn from a sample size of 10,000 (test data size 20 %). The metrics are computed for graphs selected via EBIC and are averaged results over 100 different signals.

$p \geq 20$ were run using inter-node parallelization, where there is inter-node communication time to be considered. This can be higher than the time taken by lasso in actually solving the problems.

p	n	lasso	BS_{1edge}	BS_{glasso}	BS_{reg}	BS_{lasso}
5	100	0.020	0.273	0.294	0.274	0.136
	1000	0.020	0.149	0.171	0.150	0.091
	10000	0.020	0.138	0.156	0.136	0.082
	100000	0.020	0.135	0.152	0.133	0.082
	∞	0.020	0.183	0.202	0.185	0.116
15	100	0.300	26.442	27.005	9.120	8.823
	1000	0.200	3.200	3.288	2.799	2.661
	10000	0.200	3.211	3.280	2.738	2.651
	100000	0.200	3.140	3.272	2.715	2.637
	∞	0.200	3.169	3.260	2.699	2.524
25	100	1.335	22.500	10.660	22.330	8.14
	1000	1.319	11.270	7.550	11.510	5.730
	10000	1.319	6.920	4.628	4.910	4.720
	100000	1.318	4.740	3.855	4.560	3.570
	∞	1.318	4.090	3.708	3.870	2.968

Table 3.1 Training time (in seconds) along a path of hyper-parameters averaged across 100 different signals with edge probability 5 % for each (p, n) pair.

p	n	lasso	BS_{1edge}	BS_{glasso}	BS_{reg}	BS_{lasso}
5	100	0.025	0.259	0.279	0.255	0.132
	1000	0.025	0.146	0.169	0.144	0.085
	10000	0.028	0.143	0.159	0.142	0.084
	100000	0.029	0.141	0.157	0.140	0.083
	∞	0.023	0.157	0.177	0.155	0.094
15	100	0.32	5.608	5.332	5.826	5.159
	1000	0.202	2.936	2.685	2.963	2.601
	10000	0.204	2.736	2.959	2.984	2.919
	100000	0.210	2.902	2.933	2.663	2.900
	∞	0.197	2.557	2.939	2.816	2.879
25	100	2.618	12.510	11.640	11.650	5.316
	1000	2.623	7.150	7.540	7.580	3.804
	10000	2.625	4.450	4.033	3.929	3.359
	100000	2.624	3.520	4.294	3.883	3.273
	∞	2.624	3.500	3.4	3.624	3.045

Table 3.2 Training time (in seconds) along a path of hyper-parameters averaged across 100 different signals with edge probability 25 % for each (p, n) pair.

3.2 Numerical simulations on the *Arabidopsis thaliana* dataset

From the results obtained so far, it is clear that it is quite ambitious to aim for a support recovery with $p = 39$ variables and only a sample size of $n = 118$ using the methods presented in this study. It was observed that none of the methods were able to detect any edges when all of the genes were considered together. Due to this reason, problem of partial recovery of the gene regulatory network was considered. In particular, the interaction between genes assigned to the cystolic pathway and plastidal pathway, encoding proteins in mitochondrion were considered individually. Figures 3.27a, 3.28a and 3.29a show the interaction within each of the three modules obtained by the modified GGM approach applied by Wille et al [Wil+04]. The interactions recovered using this approach served as a benchmark for comparison. With 15 genes part of the cystolic pathway, 19 in plastidal and 5 encoding proteins within mitochondrion, all range of signal sizes considered in this study will be covered.

At a sample size of $n = 118$, it was observed that there were no differences between the models selected by different model selection criterion described in section 2.3. On closer analysis, it was observed that the the likelihood term of the information criterion dominated the penalty terms by a huge factor. This indicates that the models fit the observed data very well. Thus, a change in penalty on the non-zero entries of the inverse covariance matrix had very little effect on the model selection. In other words, drift matrices selected by AIC, BIC and eBIC were almost similar.

Figures 3.27 to 3.29 show the comparison of recovered connections by the BS methods that produced the sparsest solution (and lasso) with the connections obtained by the GGM approach. The recovered drift matrices in this study were very dense and complex, with a lot of non-zeroes. For clarity, comparison is done only for the connections that were present in the GGM approach. A summary of results obtained by all the methods are provided in tables 3.3 to 3.5, where a connection is represented by the presence of directed edges between any two genes (either bi- or uni-directed).

In Figure 3.27, one can see that all the methods recovered almost all existing connections. One can also gain additional information about the various interactions between genes. $DXPS2 \leftrightarrow HDR, GPPS \leftrightarrow PPDS2$ and $PPDS1 \leftrightarrow PPDS2$ are some of the prominent interactions between genes discovered across all methods. However, there were many FPs and the solutions are not sparse (see table 3.3). This indicates a high recall rate, but poor performance in terms of precision in general. The sparsest solution was provided by $\mathrm{BS}_{\mathrm{lasso}}$, followed by lasso and then $\mathrm{BS}_{\mathrm{reg}}$. However, $\mathrm{BS}_{\mathrm{lasso}}$ did not detect connections that were present in the GGM (FN). This indicates a lower recall value in comparison to other methods. $\mathrm{BS}_{\mathrm{reg}}$ detects everything while having same number of FPs as $\mathrm{BS}_{\mathrm{lasso}}$, thus having relatively higher precision.

Figure 3.28 shows the dependency between genes that are involved in encoding proteins in mitochondrion. Except $\mathrm{BS}_{\mathrm{glasso}}$, all the methods are in agreement with the relationship established by the GGM (see table 3.4). Given the size of the problem ($p = 5$), this is expected. All the methods, generally, perform well for small graph sizes.

In Figure 3.29, all of the connections within the module were recovered. Here, in this module, interactions $AACT2 \leftrightarrow MK, HMGS \leftrightarrow FPPS1$ and $FPPS1 \leftrightarrow MPDC2$ are always identified across all methods. However, just like in cystolic pathway results, there were again too many FPs. Lasso performs very poorly in terms of giving a sparse solution. $\mathrm{BS}_{\mathrm{lasso}}$ still remains to be the only method that provides sparsest solution along with $\mathrm{BS}_{\mathrm{glasso}}$ and $\mathrm{BS}_{\mathrm{1edge}}$. Among the BS methods, $\mathrm{BS}_{\mathrm{glasso}}$ has relatively low FPs (thus high precision). With this gene module, $\mathrm{BS}_{\mathrm{lasso}}$ does not agree with 2 existing connections (FN) between genes and identifies 2 additional connections (FP), thus leading to a relatively lower precision. Furthermore, $\mathrm{BS}_{\mathrm{reg}}$ also performs poorly in terms of precision, even though it detects all existing connections.

The high FPs with BS could be attributed to the fact that there were many solutions for medium values of k, that were sub-optimal. To tackle this, a finer grid containing 20 values of k withing the interval $[p, p^2]$ was considered for this dataset to obtain solutions of wide range of sparsity. It was observed that the optimality gap did not go below 5 % for many solutions within the specified time and hence, were not considered for further evaluation. However, this is an optimistic view. Another reason could be that the choice of idenitity matrix as the volatility matrix, C. In reality, this matrix is not the identity due to large correlations between variables. This is indeed true in this case as well [Wil+04]. Gaïffas and Matulewicz [GM19] used the identity matrix for all analysis, but for real application they calculate the squared variations to compute an estimate of the diagonal entries of this matrix. However, they recommend to jointly estimate M and a non-diagonal C for such cases, which is not a trivial task in general.

Overall, these results seem to agree with the results from simulations performed on synthetic dataset. For large graph sizes and low sample size, lasso performs worse in terms of precision. However, BS is more robust. Out of all BS methods, BS_{lasso} performs the best in providing a sparse solution.

	lasso	BS_{1edge}	BS_{glasso}	BS_{reg}	BS_{lasso}
Total number of edges (including self-loops)	300	341	322	305	253
Existing connections (as bi-directed)	14	14	15	11	7
Existing connections (as uni-directed)	3	3	2	6	8
Existing connections (not present)	0	0	0	0	2
New connections (bi-direction)	100	138	117	99	76
New connections (uni-direction)	50	15	37	48	60

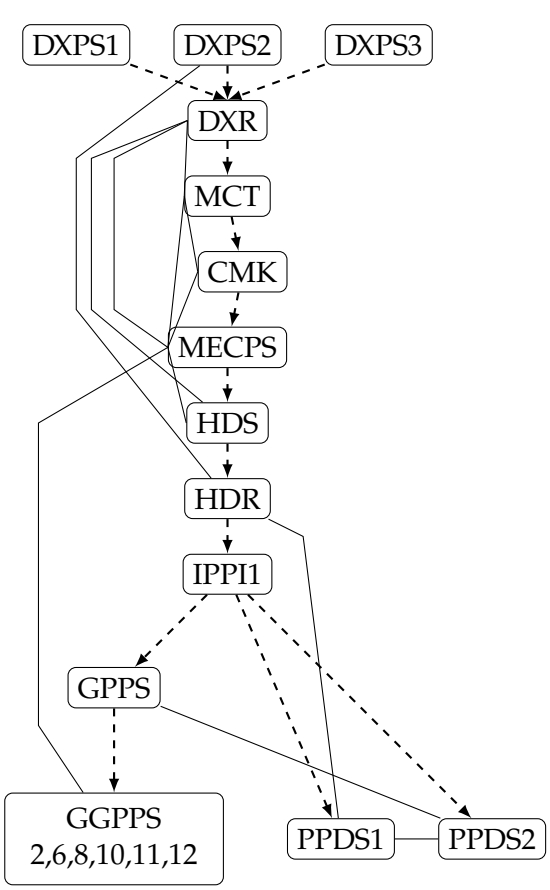
Table 3.3 Gene connections detected by each method in the Cystolic pathway in comparison to the connections detected by the modified GGM approach by Wille et al. [Wil+04]

	lasso	BS_{1edge}	BS_{glasso}	BS_{reg}	BS_{lasso}
Total number of edges (including self-loops)	5	5	16	5	5
Existing connections (as bi-directed)	0	0	0	0	0
Existing connections (as uni-directed)	0	0	0	0	0
Existing connections (not present)	0	0	0	0	0
New connections (bi-direction)	0	0	4	0	0
New connections (uni-direction)	0	0	7	0	0

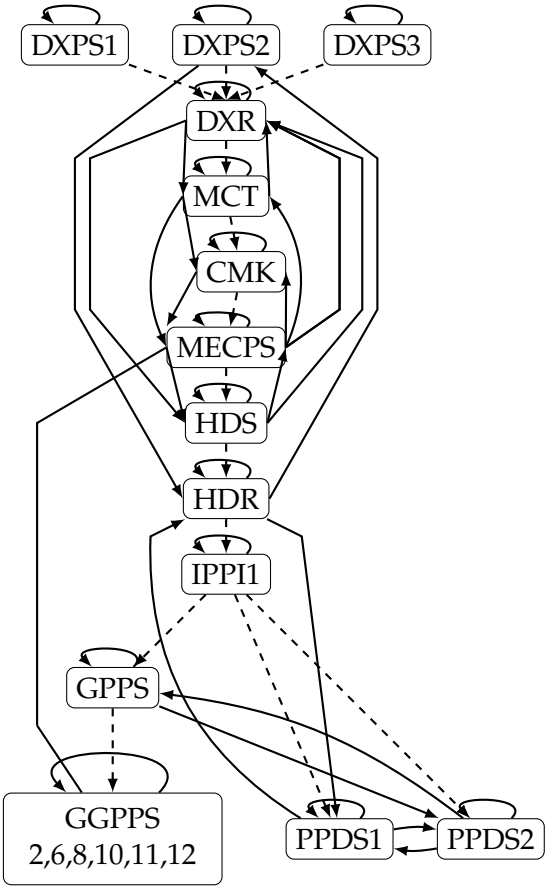
Table 3.4 Gene connections detected by each method, that are responsible for protein sythesis within the mitochondrion, in comparison to the connections detected by the modified GGM approach by Wille et al. [Wil+04]

	lasso	BS_{1edge}	BS_{glasso}	BS_{reg}	BS_{lasso}
Total number of edges (including self-loops)	221	186	157	225	156
Existing connections (as bi-directed)	12	11	7	12	5
Existing connections (as uni-directed)	0	1	4	0	5
Existing connections (not present)	0	0	1	0	2
New connections (bi-direction)	89	60	48	93	48
New connections (uni-direction)	4	28	28	0	30

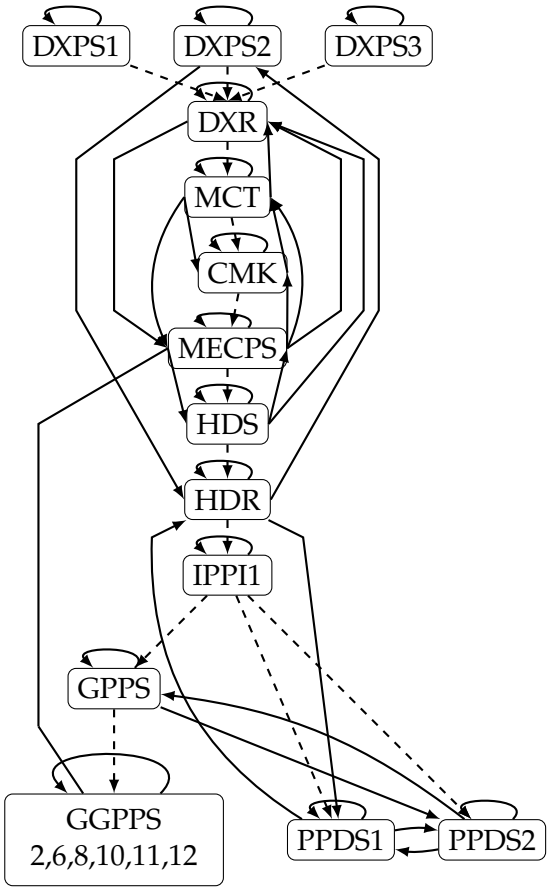
Table 3.5 Gene connections detected by each method in the plastidal pathway in comparison to the connections detected by the modified GGM approach by Wille et al. [Wil+04]. Dotted directed edges mark the metabolic network, and were not part of the GGM and not part of comparison.


DXPS1
DXPS2
DXPS3
DXR
MCT
CMK
MECPS
HDS
HDR
IPPI1
GPPS
GGPPS
2,6,8,10,11,12
PPDS1
PPDS2

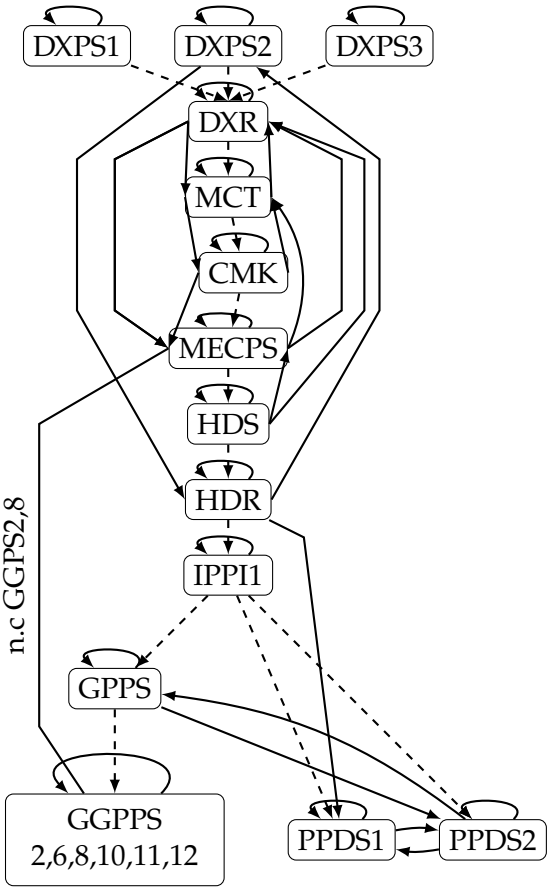
(a)


DXPS1
DXPS2
DXPS3
DXR
MCT
CMK
MECPS
HDS
HDR
IPPI1
GPPS
GGPPS
2,6,8,10,11,12
PPDS1
PPDS2

(b)

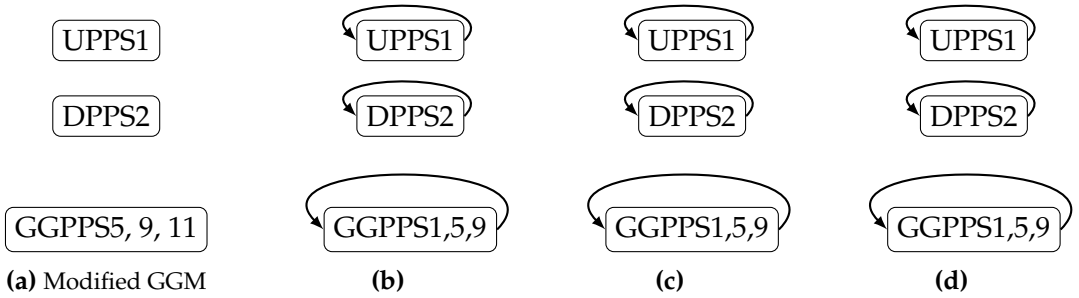

DXPS1
DXPS2
DXPS3
DXR
MCT
CMK
MECPS
HDS
HDR
IPPI1
GPPS
GGPPS
2,6,8,10,11,12
PPDS1
PPDS2

(c)


DXPS1
DXPS2
DXPS3
DXR
MCT
CMK
MECPS
HDS
HDR
IPPI1
GPPS
GGPPS
2,6,8,10,11,12
PPDS1
PPDS2
n.c GGPS2,8

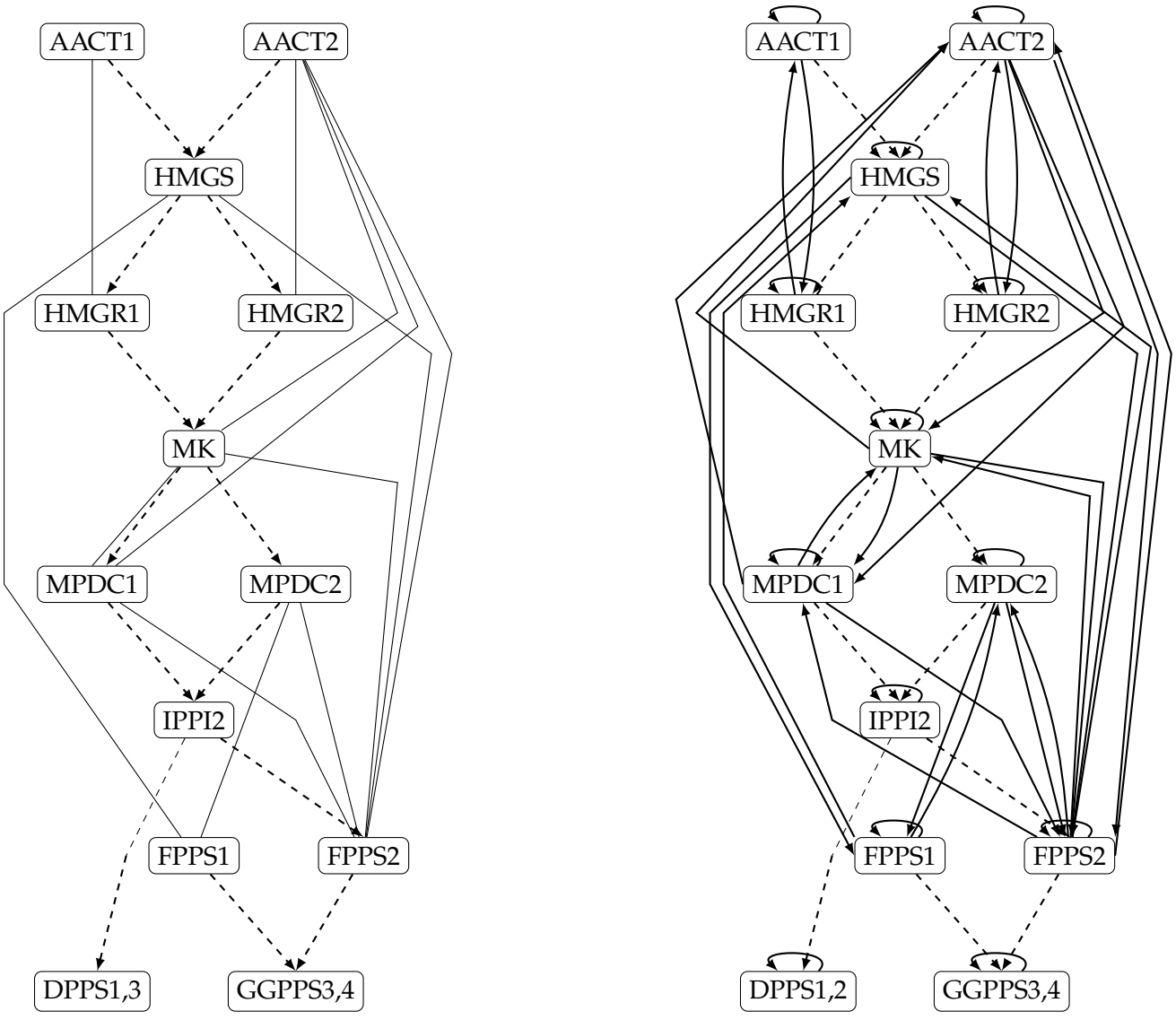
(d)

Figure 3.27 Dependencies between genes in the cystolic pathway module. (a) by the modified GGM approach (Wille et al.); (b) by lasso ; (c) by BS_{reg} and (d) by BS_{lasso}. The solid undirected lines in the GGM approach represent the connections, whereas the undirected edge between MECPS and GGPPS in the other methods represent a bundle of directed edges (both bi- and uni-directed). n.c refers to "no connection" and it is followed by gene names for which there is no connection among a bundle of directed edges. Dotted directed edges mark the metabolic network, and were not part of the GGM and not part of comparison.



(a) Modified GGM **(b)** **(c)** **(d)**

Figure 3.28 Dependencies between genes responsible for protein sythiesis within mitochondrion. (a) by the modified GGM approach (Wille et al.); (b) by lasso ; (c) by BS_{reg} and (d) by BS_{lasso}. The solid undirected lines in the GGM approach represent the connections.



(a) Modified GGM **(b)**

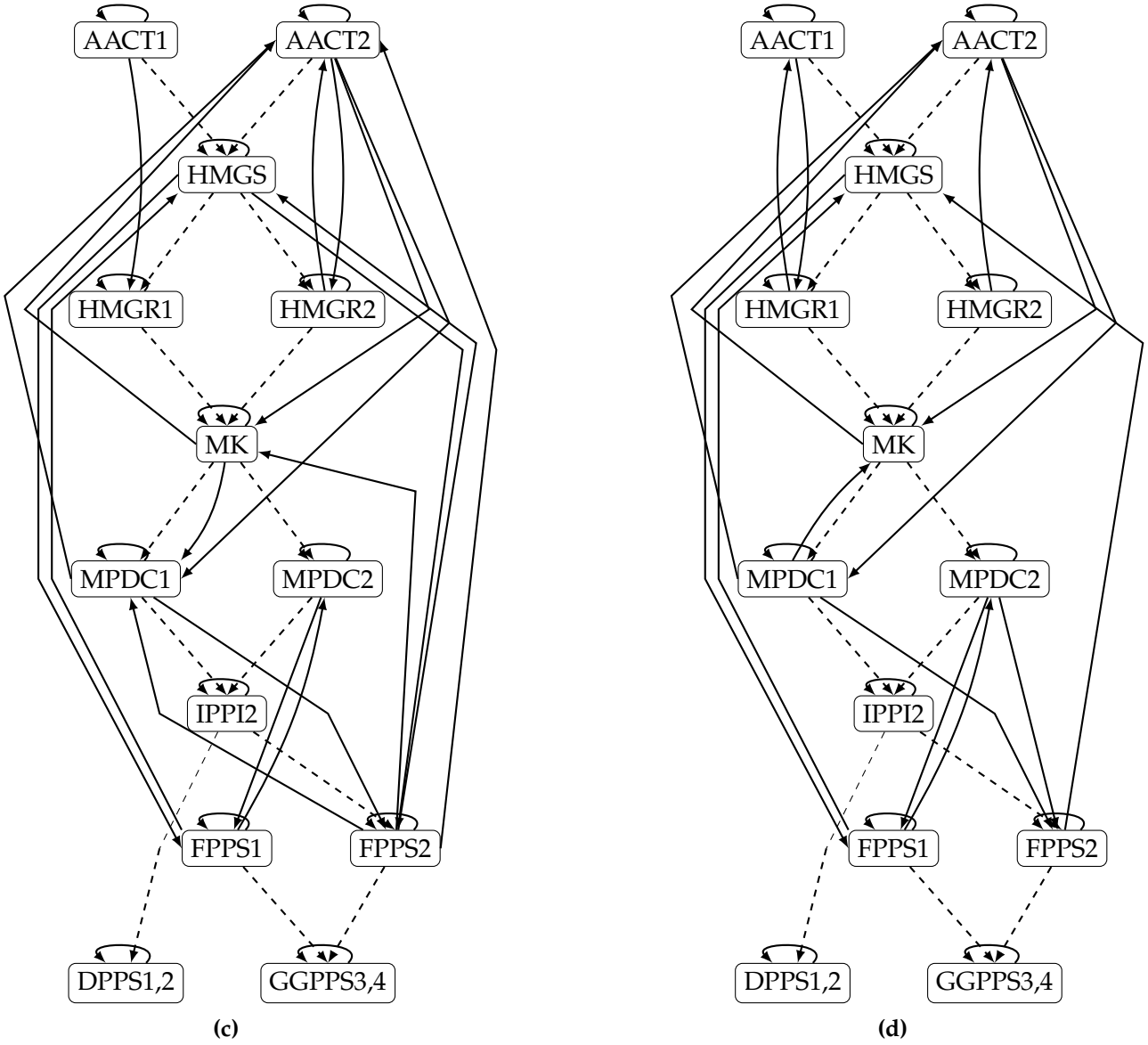


Figure 3.29 Dependencies between genes in the plastidal pathway module. (a) by the modified GGM approach (Wille et al.); (b) by lasso ; (c) by BS_{glasso} and (d) by BS_{lasso}. The solid undirected lines in (a) connecting genes represent the GGM. Dotted directed edges mark the metabolic network, and were not part of the GGM and not part of comparison.

4 Conclusion

This study focused on using mixed integer programming technique to recover drift matrices, M, from graphical lyapunov models. In particular, the problem of support recovery of the drift matrix was treated as a best-subset selection problem and converted into a MIO problem. Parallel computing has made solving MIO problems practical and feasible for large scale applications. Moreover, they have the advantage of always providing an optimality certificate to the obtained solution and this can be very useful in analyzing results.

For graphs with $p < 10$, the differences between BS and lasso are not so significant. However, as the problem size increases, lasso clearly outperforms BS provided one has sufficiently large samples. With low sample size, BS tends to perform better. This effect becomes more and more prominent with increasing graph size because the performance of lasso degrades for larger signals with low sample size. This degradation rate is typically faster than degradation rate of BS beyond certain value of p. The drop in performance of BS could be attributed to insufficient time for closing the optimality gap or the coarse grid chosen for model selection. It was observed that the choices of the initializations provided to MIO formulations are very crucial. BS_{lasso} consistently performs as good as lasso in giving a sparse solution. However its performance is sensitive to the quality of lasso solutions. So when lasso performs poor, it tends to perform poor. But with low sample size, BS_{lasso} is not affected by poor lasso performance. For low sample sizes, BS_{lasso} is as good as, or better than, other BS methods. Among the projected gradient initialization based BS, BS_{1edge} and $\text{BS}_{\text{glasso}}$ showed relatively good potential in general. However, in comparison to BS_{lasso}, they are still inferior. This implies that lasso provides better lower bounds to the MIO than projected gradient descent. Beyond certain graph size and/or below certain sample size, all the methods perform very poor and are worse than a random guessing classifier. In addition, when the graphs' sparsity level is reduced, the support recovery becomes harder. At the end, everything boils down to choosing a method that is least bad.

In general, one needs to be cautious with model selection. Their usage may depend on the type of dataset and its size. The choice of a specific model selection method could have a significant impact on the performance and generalizability of the final model. Different datasets may have varying degrees of complexity, noise, or underlying patterns, and these factors can influence which model selection approach is most appropriate. AIC, BIC and eBIC were able to choose distinct sparse solutions with the synthetic dataset. However, model selection with the results from the simulation of the *Arabidopsis thaliana* dataset resulted in dense estimates of drift matrices. This implies that these selection methods may not be the best for this dataset at such a low sample size. One often needs additional domain expertise to filter the entries and derive meaningful relationships.

A Appendix

A.1 Prediction with AIC when edge probability 5 %: Comparison along signal sizes

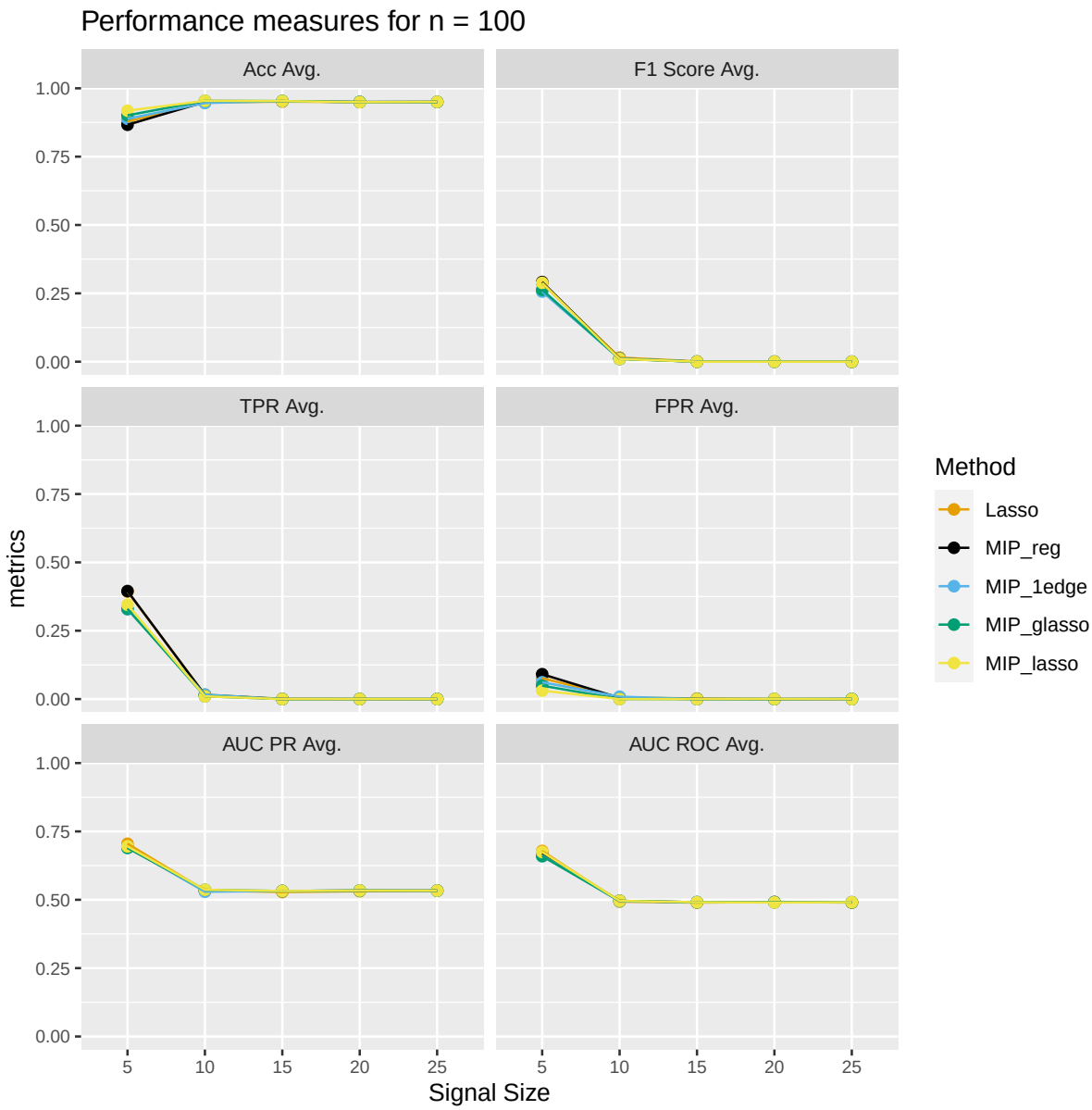


Figure A.1 Variation of performance metrics with graph size and at edge probability 5 % on unseen dataset drawn from a sample size of 100 (test data size 20 %). The metrics are computed for graphs selected via AIC and are averaged results over 100 different signals.

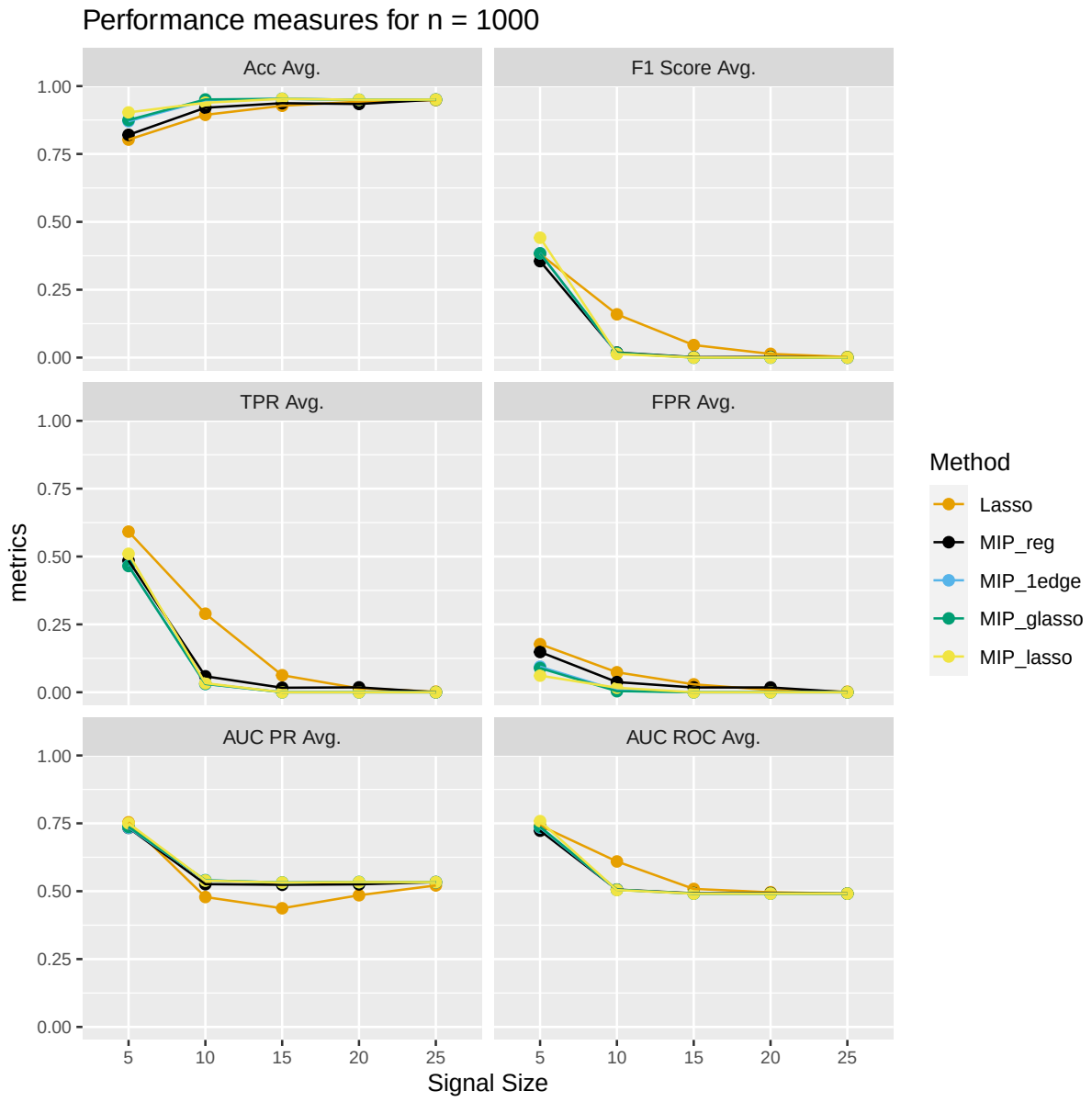


Figure A.2 Variation of performance metrics with graph size and at edge probability 5 % on unseen dataset drawn from a sample size of 1000 (test data size 20 %). The metrics are computed for graphs selected via AIC and are averaged results over 100 different signals.

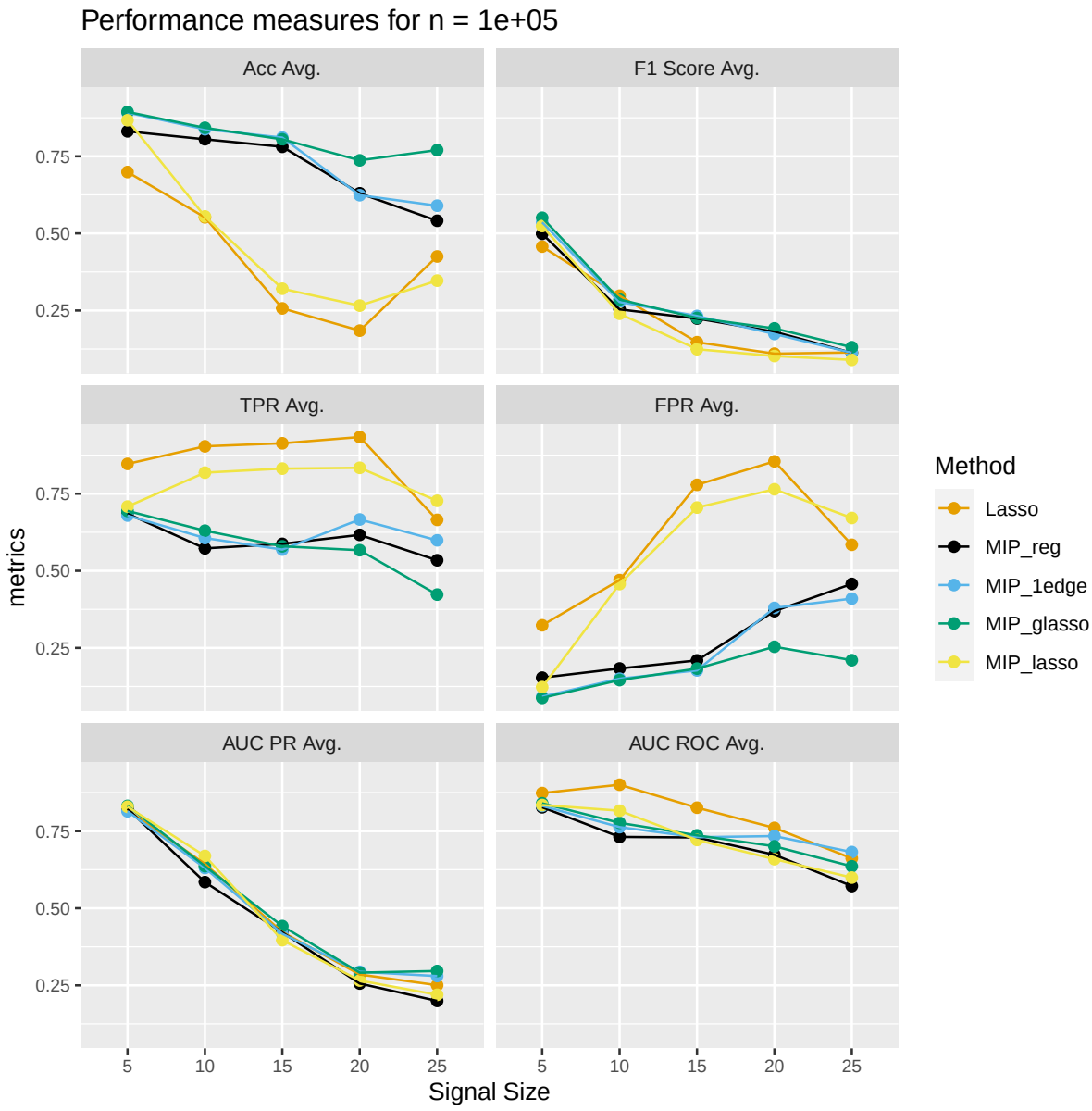


Figure A.3 Variation of performance metrics with graph size and at edge probability 5 % on unseen dataset drawn from a sample size of 100,000 (test data size 20 %). The metrics are computed for graphs selected via AIC and are averaged results over 100 different signals.

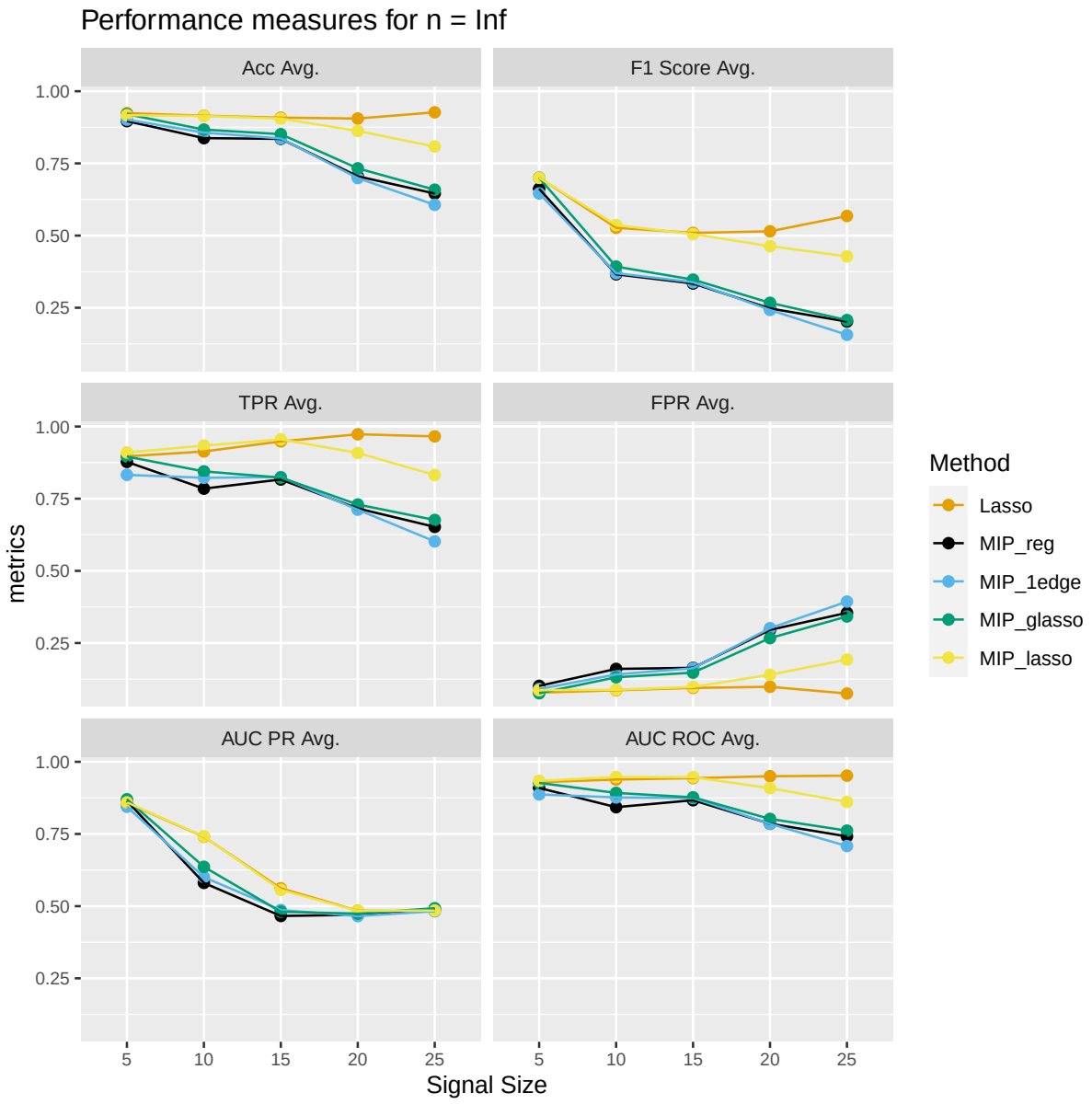


Figure A.4 Variation of performance metrics with graph size and at edge probability 5 % on unseen dataset drawn from population covariance matrix (test data size 20 %). The metrics are computed for graphs selected via AIC and are averaged results over 100 different signals.

A.2 Prediction with AIC when edge probability 25 %: Comparison along signal sizes

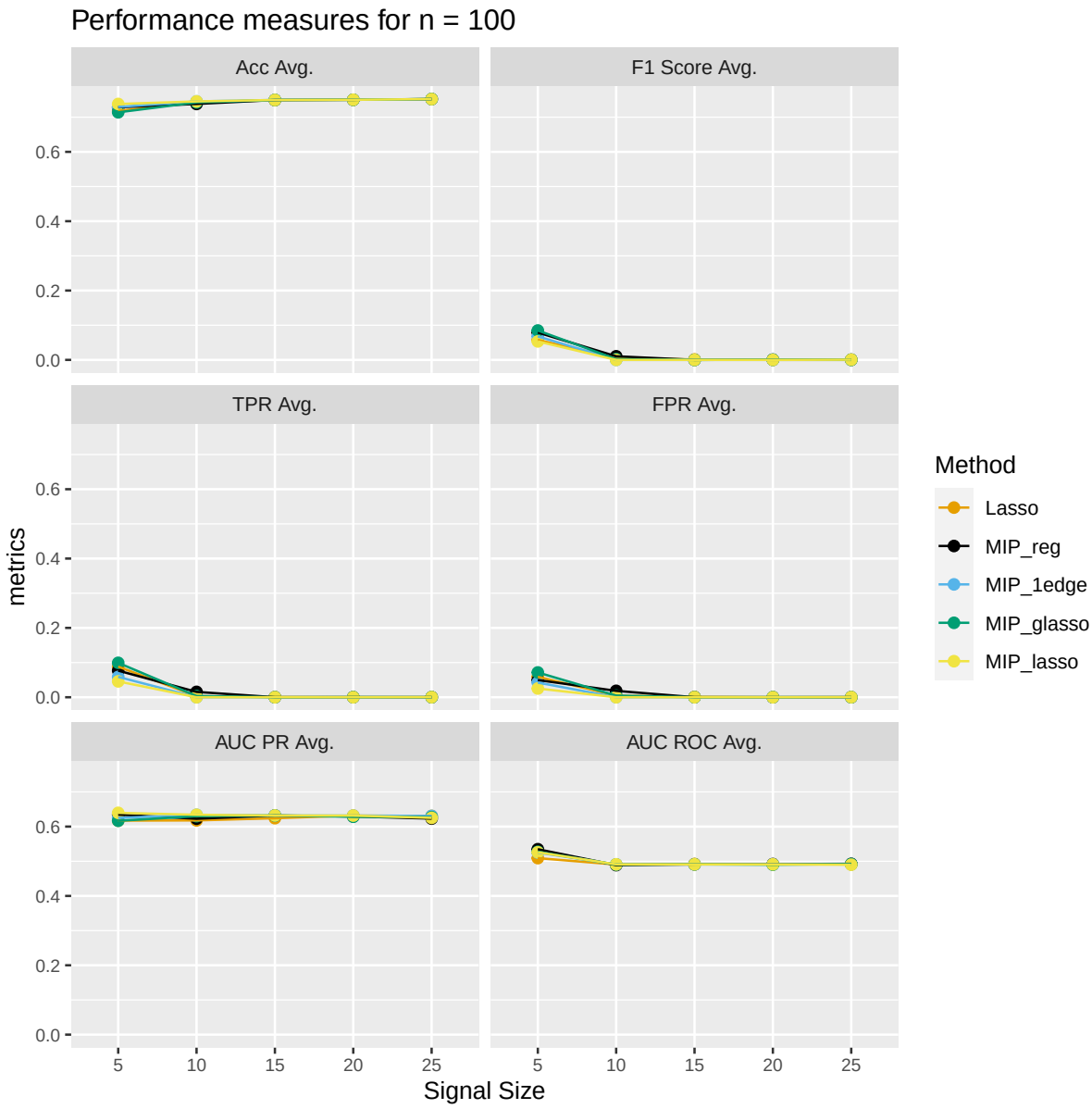


Figure A.5 Variation of performance metrics with graph size and at edge probability 25 % on unseen dataset drawn from a sample size of 100 (test data size 20 %). The metrics are computed for graphs selected via AIC and are averaged results over 100 different signals.

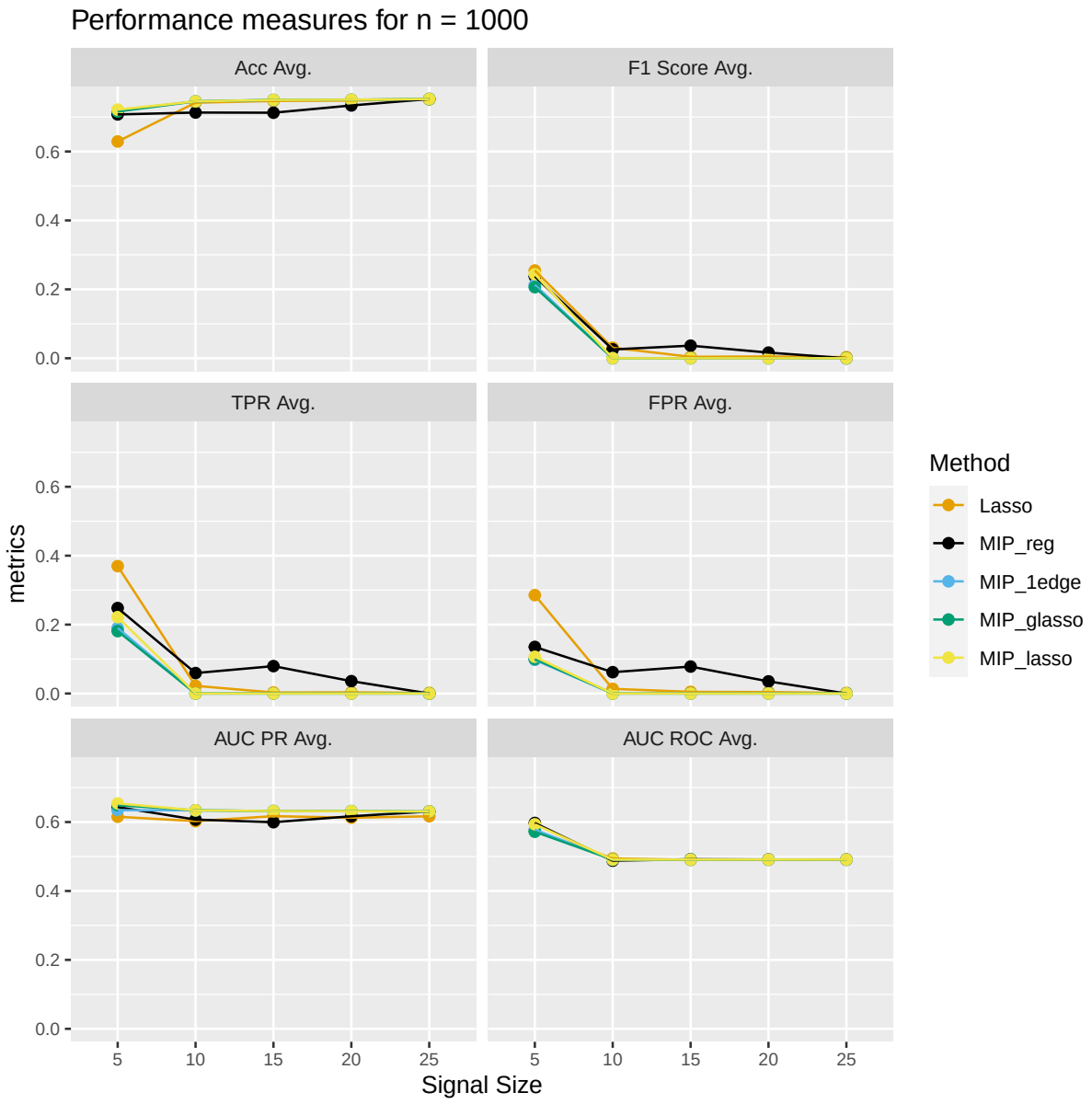


Figure A.6 Variation of performance metrics with graph size and at edge probability 25 % on unseen dataset drawn from a sample size of 1000 (test data size 20 %). The metrics are computed for graphs selected via AIC and are averaged results over 100 different signals.

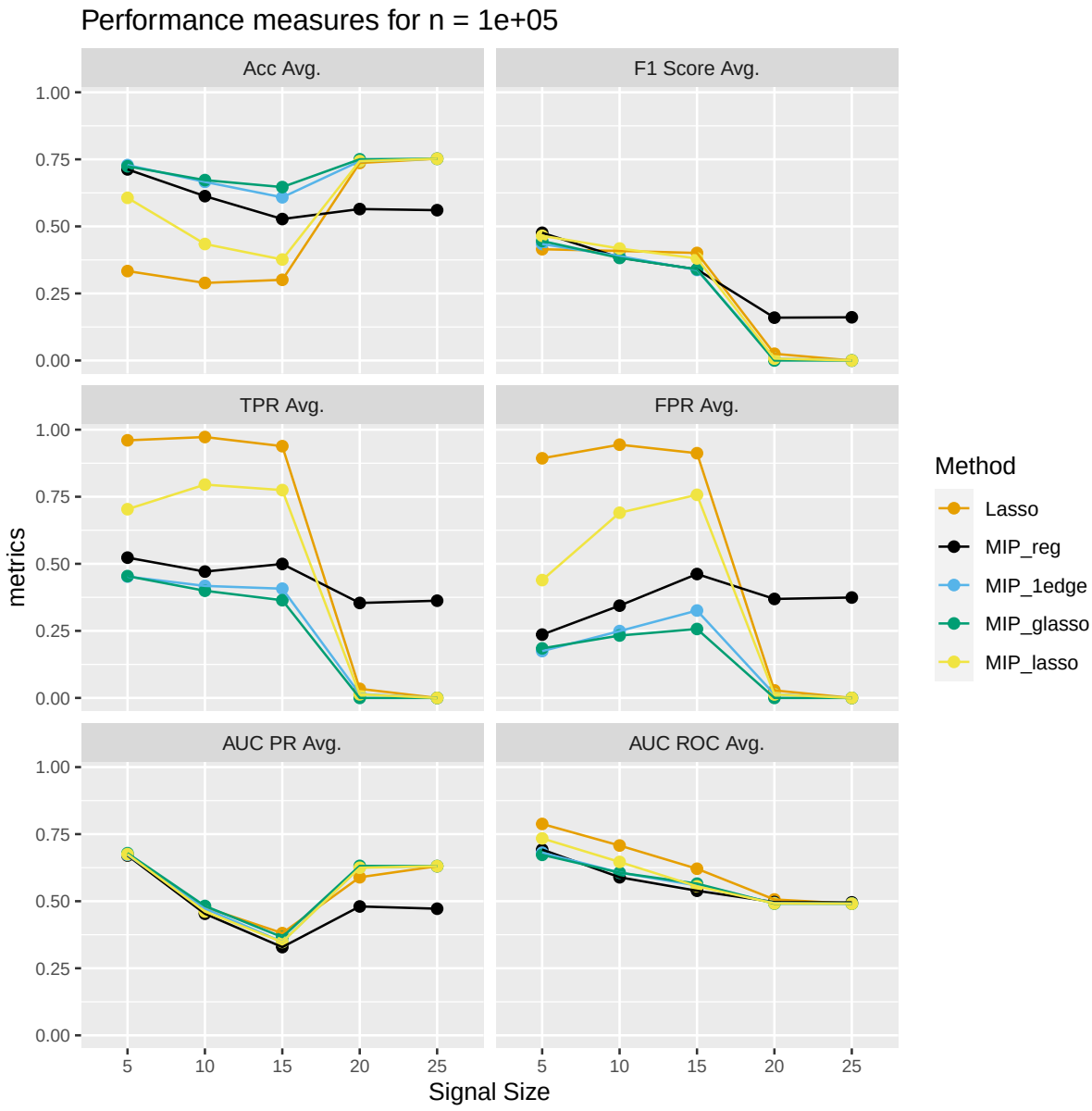


Figure A.7 Variation of performance metrics with graph size and at edge probability 25 % on unseen dataset drawn from a sample size of 100,000 (test data size 20 %). The metrics are computed for graphs selected via AIC and are averaged results over 100 different signals.

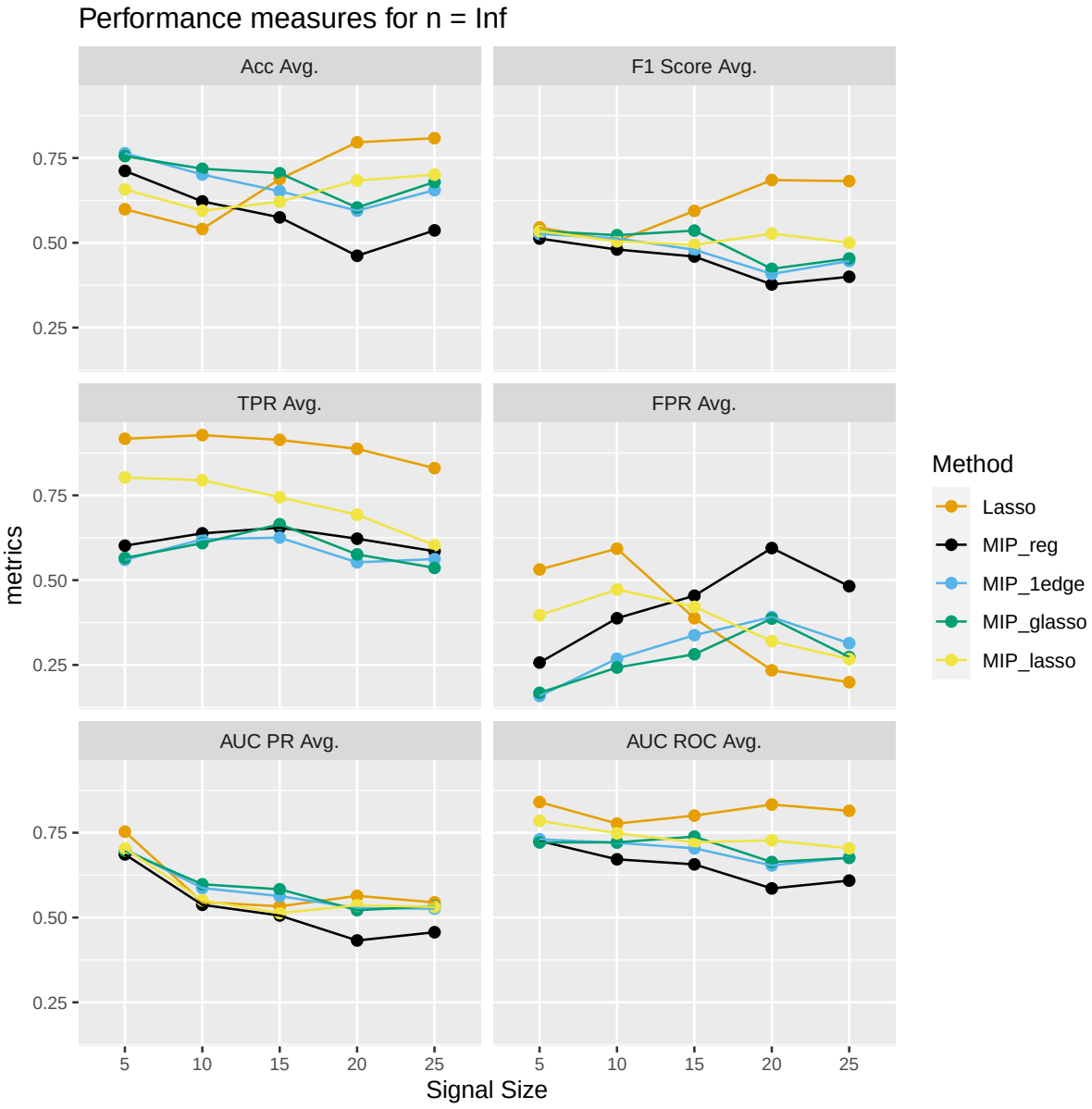


Figure A.8 Variation of performance metrics with graph size and at edge probability 25 % on unseen dataset drawn from population covariance matrix (test data size 20 %). The metrics are computed for graphs selected via AIC and are averaged results over 100 different signals.

A.3 Prediction with AIC when edge probability 5 %: Comparison along sample size

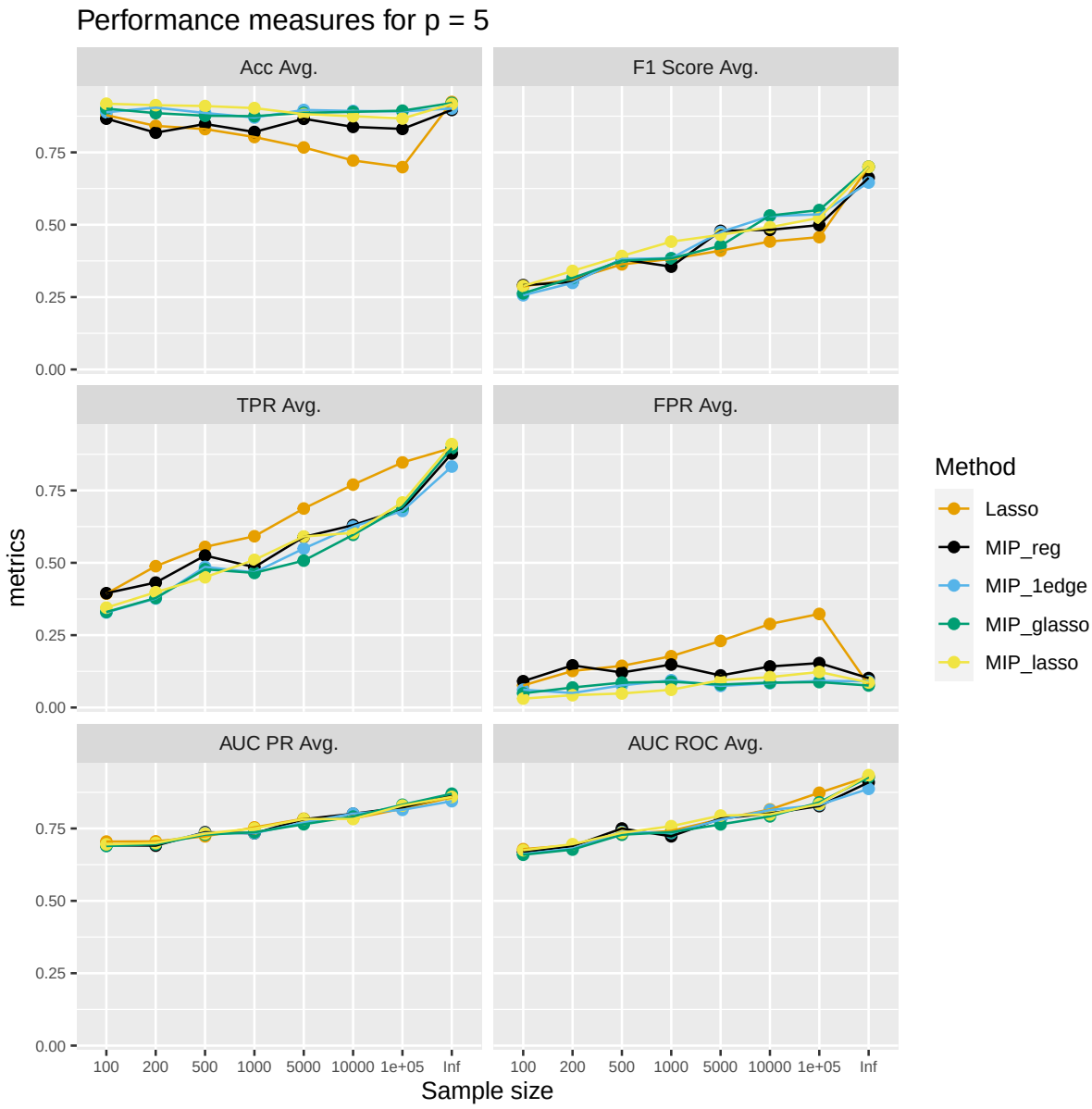


Figure A.9 Variation of performance metrics with sample size and at edge probability 5 % on unseen dataset drawn from a graph size $p = 5$ (test data size 20 %). The metrics are computed for graphs selected via AIC and are averaged results over 100 different signals.

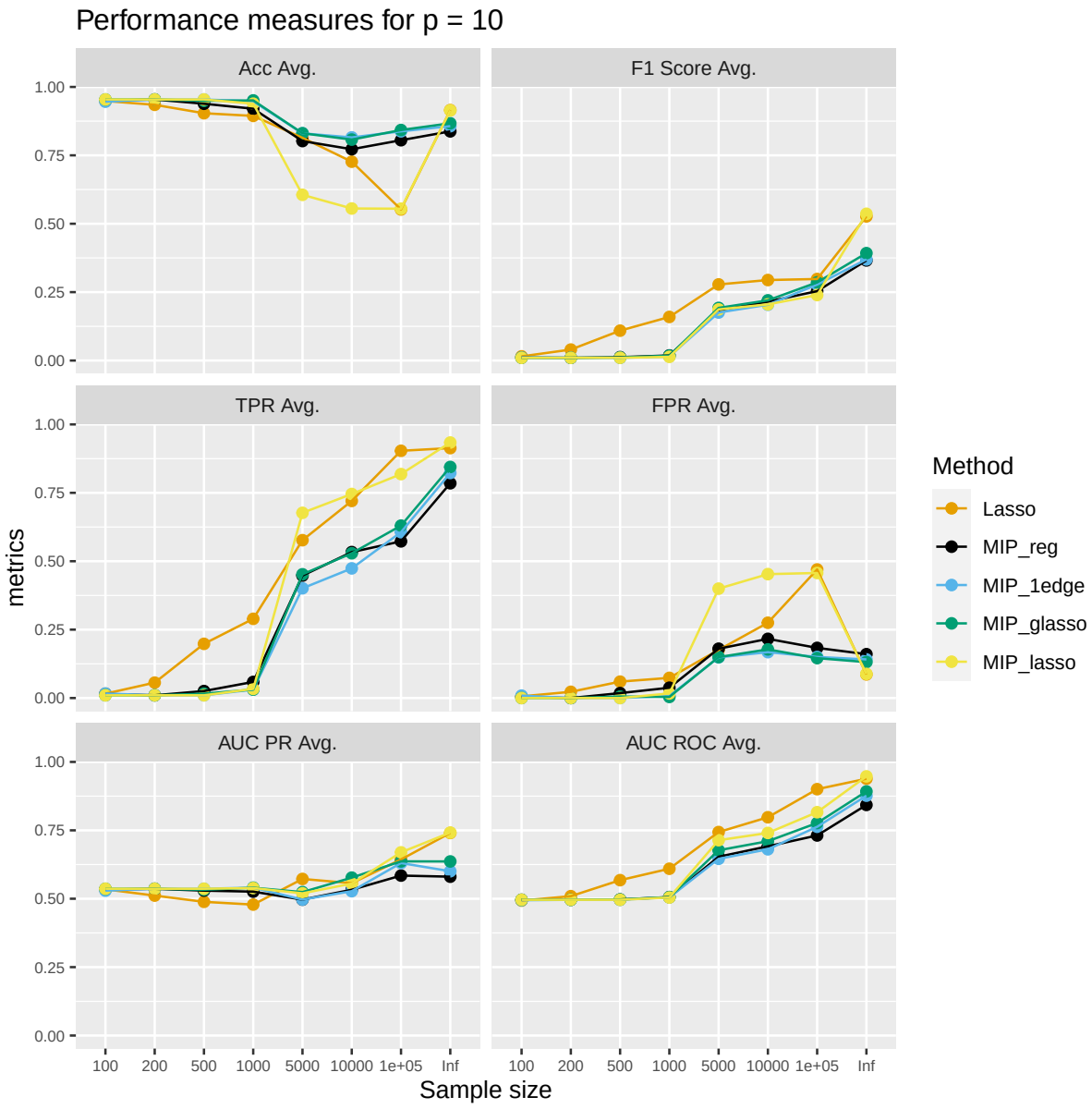


Figure A.10 Variation of performance metrics with graph size and at edge probability 5 % on unseen dataset drawn from a graph size $p = 10$ (test data size 20 %). The metrics are computed for graphs selected via AIC and are averaged results over 100 different signals.

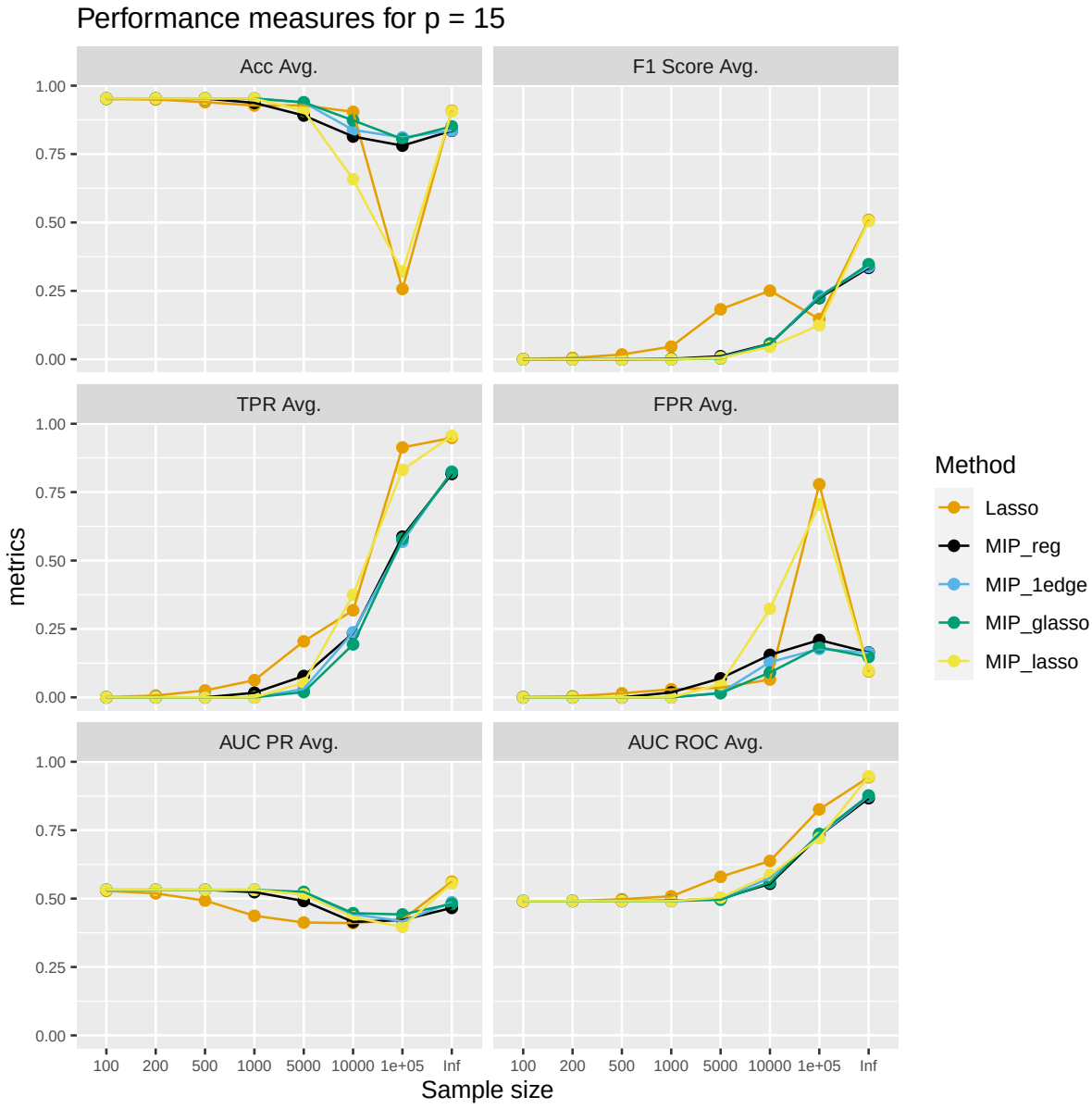


Figure A.11 Variation of performance metrics with graph size and at edge probability 5 % on unseen dataset drawn from a graph size $p = 15$ (test data size 20 %). The metrics are computed for graphs selected via AIC and are averaged results over 100 different signals.

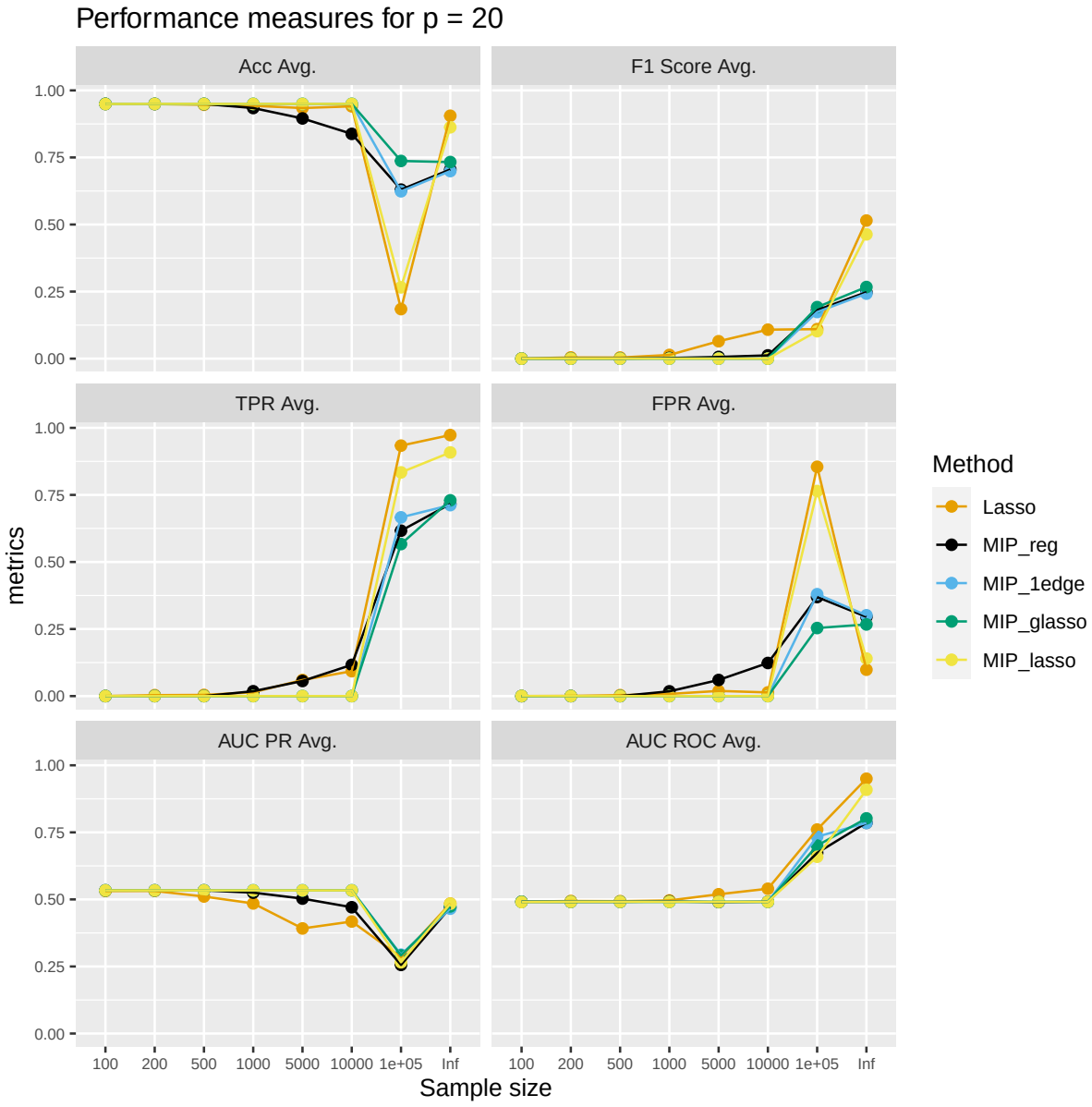


Figure A.12 Variation of performance metrics with graph size and at edge probability 5 % on unseen dataset drawn from a graph size $p = 20$ (test data size 20 %). The metrics are computed for graphs selected via AIC and are averaged results over 100 different signals.

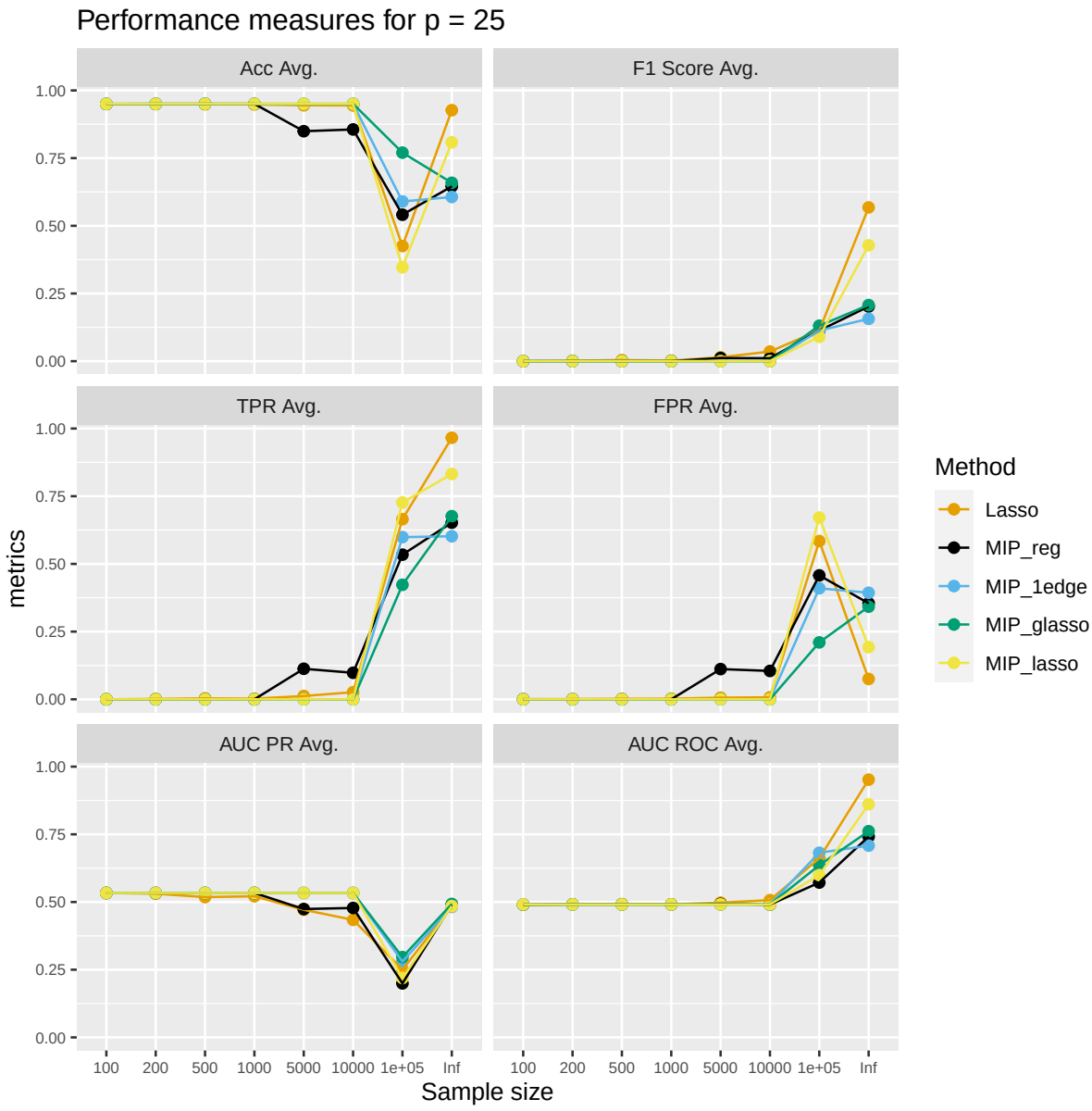


Figure A.13 Variation of performance metrics with graph size and at edge probability 5 % on unseen dataset drawn from a graph size $p = 25$ (test data size 20 %). The metrics are computed for graphs selected via AIC and are averaged results over 100 different signals.

A.4 Prediction with AIC when edge probability 25 %: Comparison along sample sizes

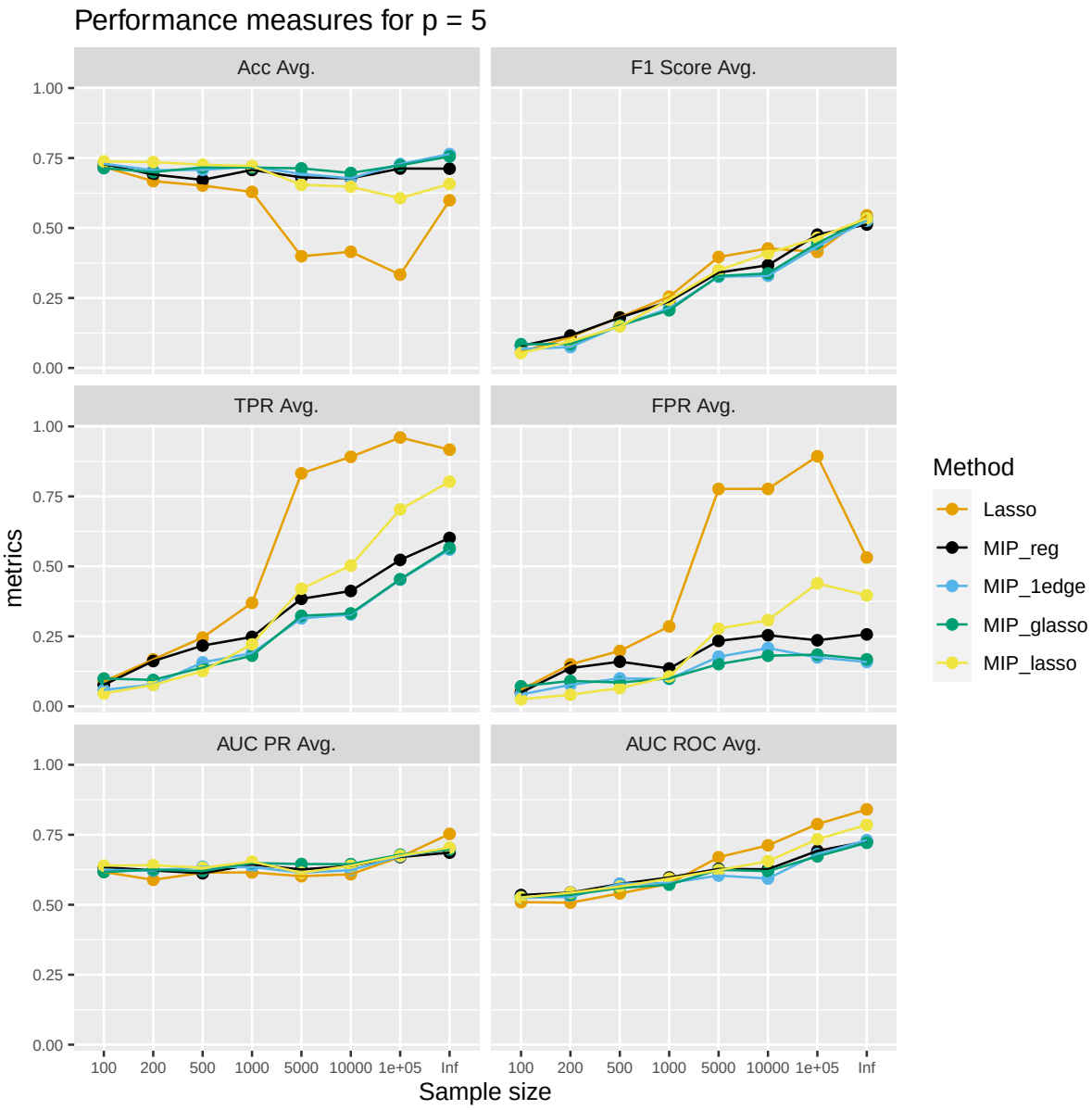


Figure A.14 Variation of performance metrics with graph size and at edge probability 25 % on unseen dataset drawn from a graph size $p = 5$ (test data size 20 %). The metrics are computed for graphs selected via AIC and are averaged results over 100 different signals.

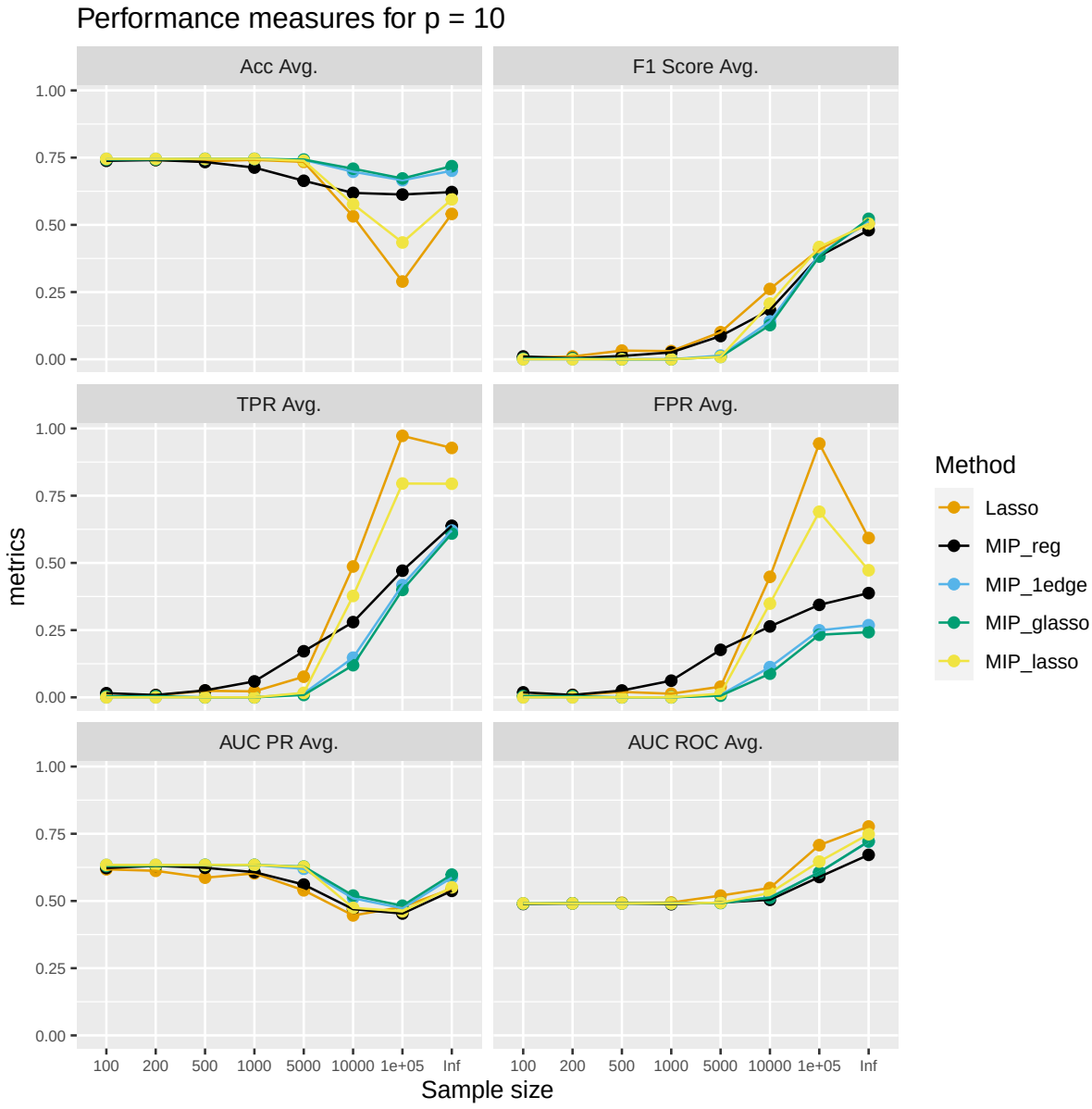


Figure A.15 Variation of performance metrics with graph size and at edge probability 25 % on unseen dataset drawn from a graph size $p = 10$ (test data size 20 %). The metrics are computed for graphs selected via AIC and are averaged results over 100 different signals.

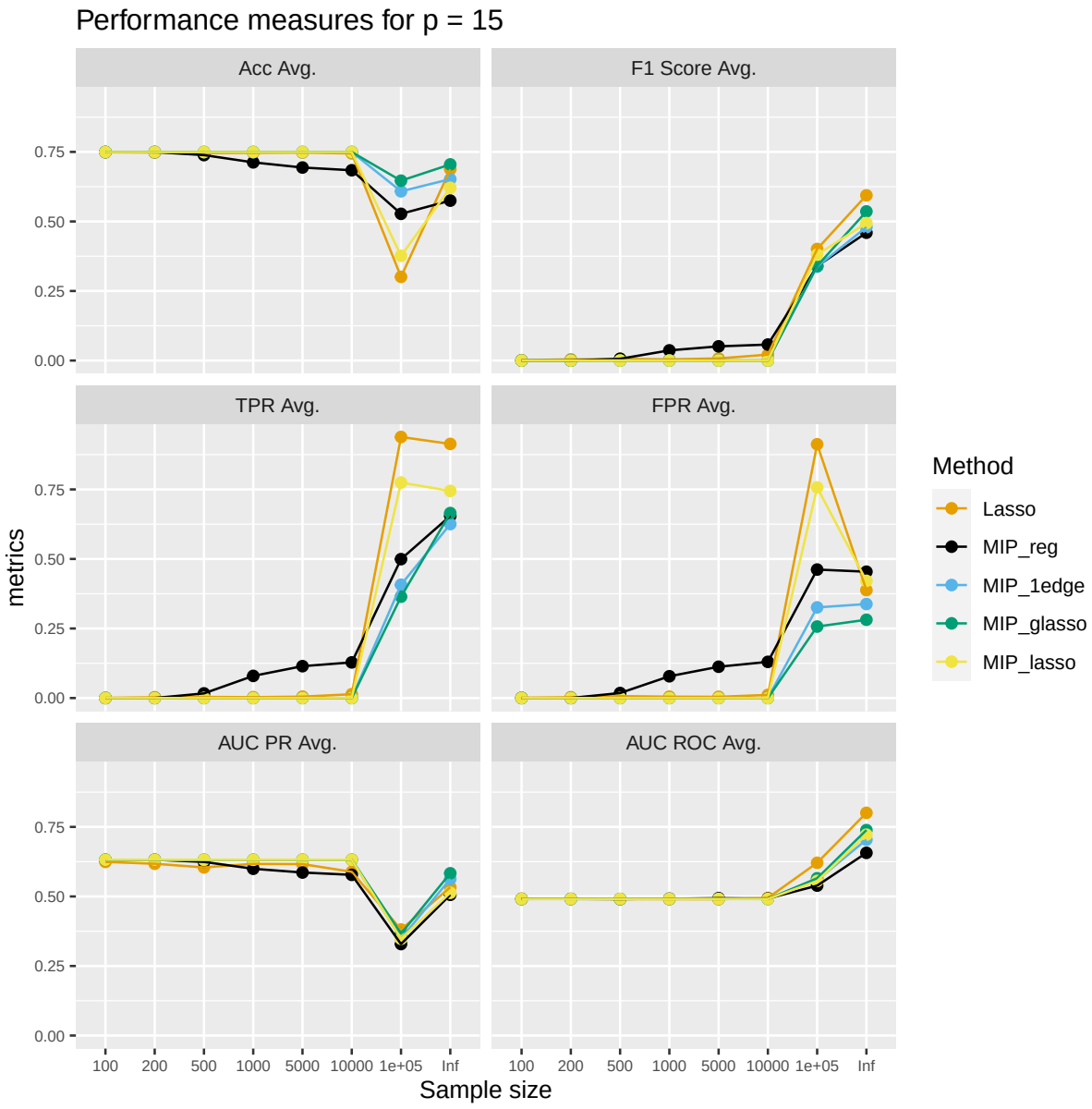


Figure A.16 Variation of performance metrics with graph size and at edge probability 25 % on unseen dataset drawn from a graph size $p = 15$ (test data size 20 %). The metrics are computed for graphs selected via AIC and are averaged results over 100 different signals.

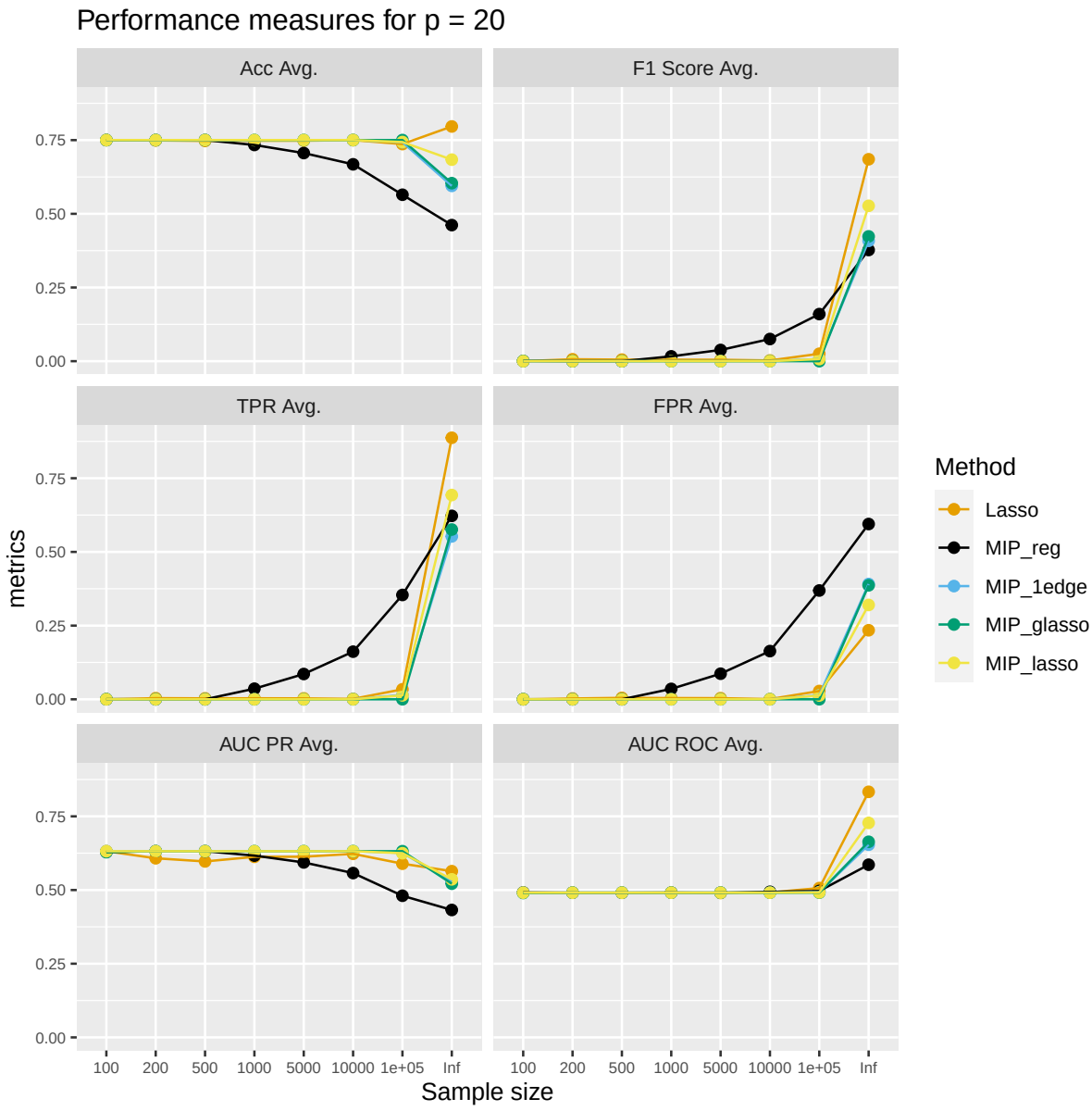


Figure A.17 Variation of performance metrics with graph size and at edge probability 25 % on unseen dataset drawn from a graph size $p = 20$ (test data size 20 %). The metrics are computed for graphs selected via AIC and are averaged results over 100 different signals.

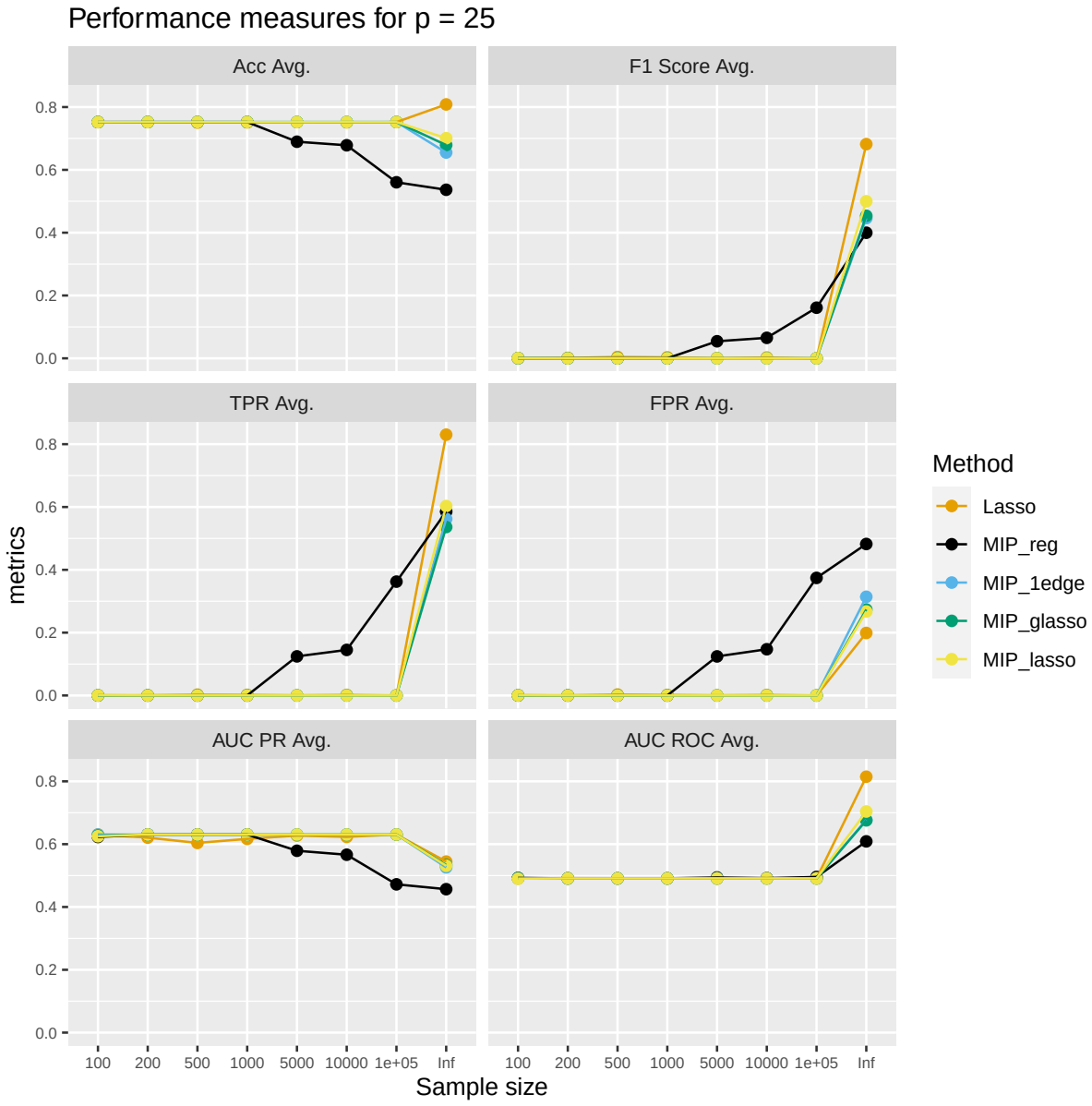


Figure A.18 Variation of performance metrics with graph size and at edge probability 25 % on unseen dataset drawn from a graph size $p = 25$ (test data size 20 %). The metrics are computed for graphs selected via AIC and are averaged results over 100 different signals.

A.5 Prediction with BIC when edge probability 5 %: Comparison along signal sizes

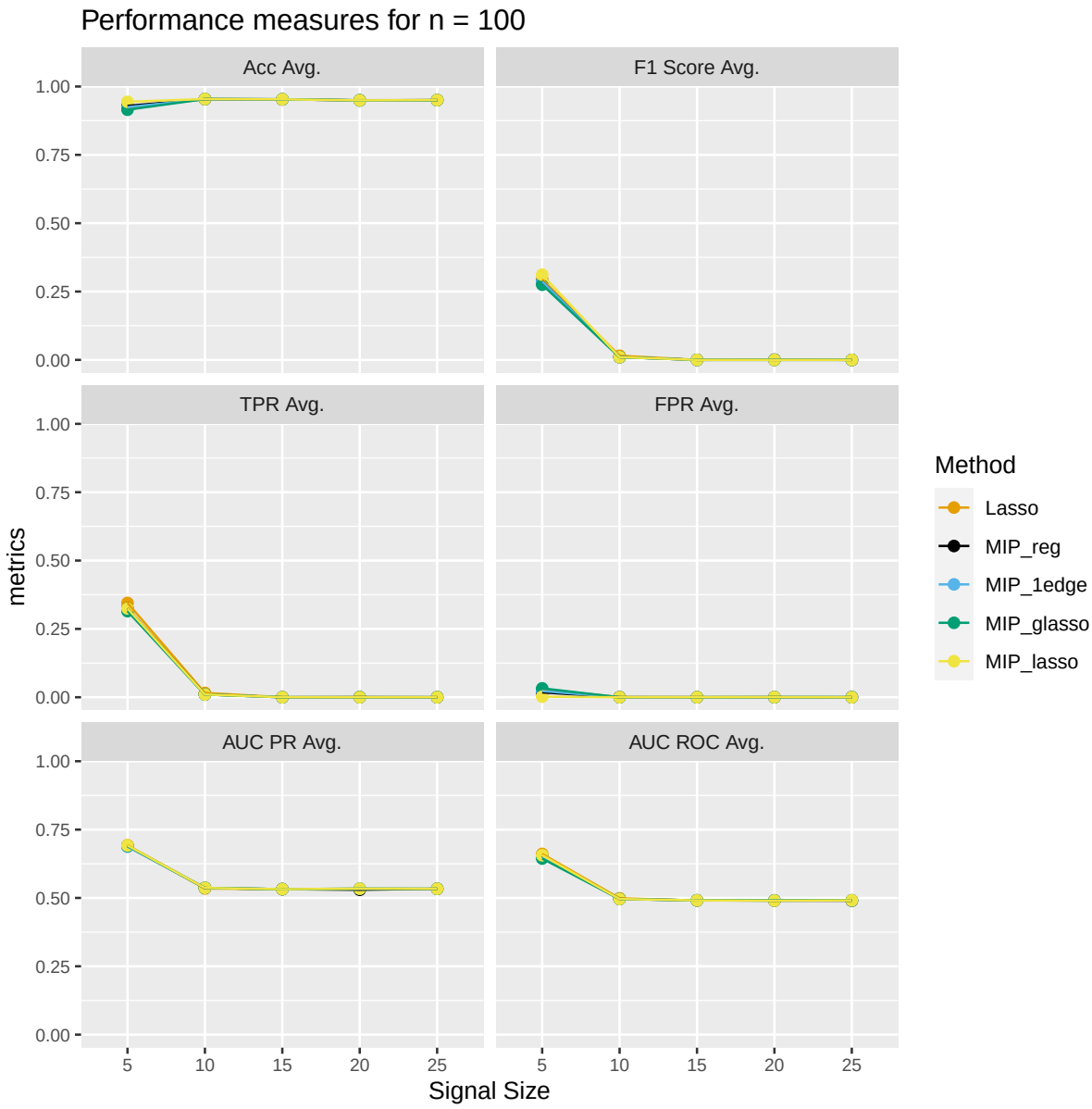


Figure A.19 Variation of performance metrics with graph size and at edge probability 5 % on unseen dataset drawn from a sample size of 100 (test data size 20 %). The metrics are computed for graphs selected via BIC and are averaged results over 100 different signals.

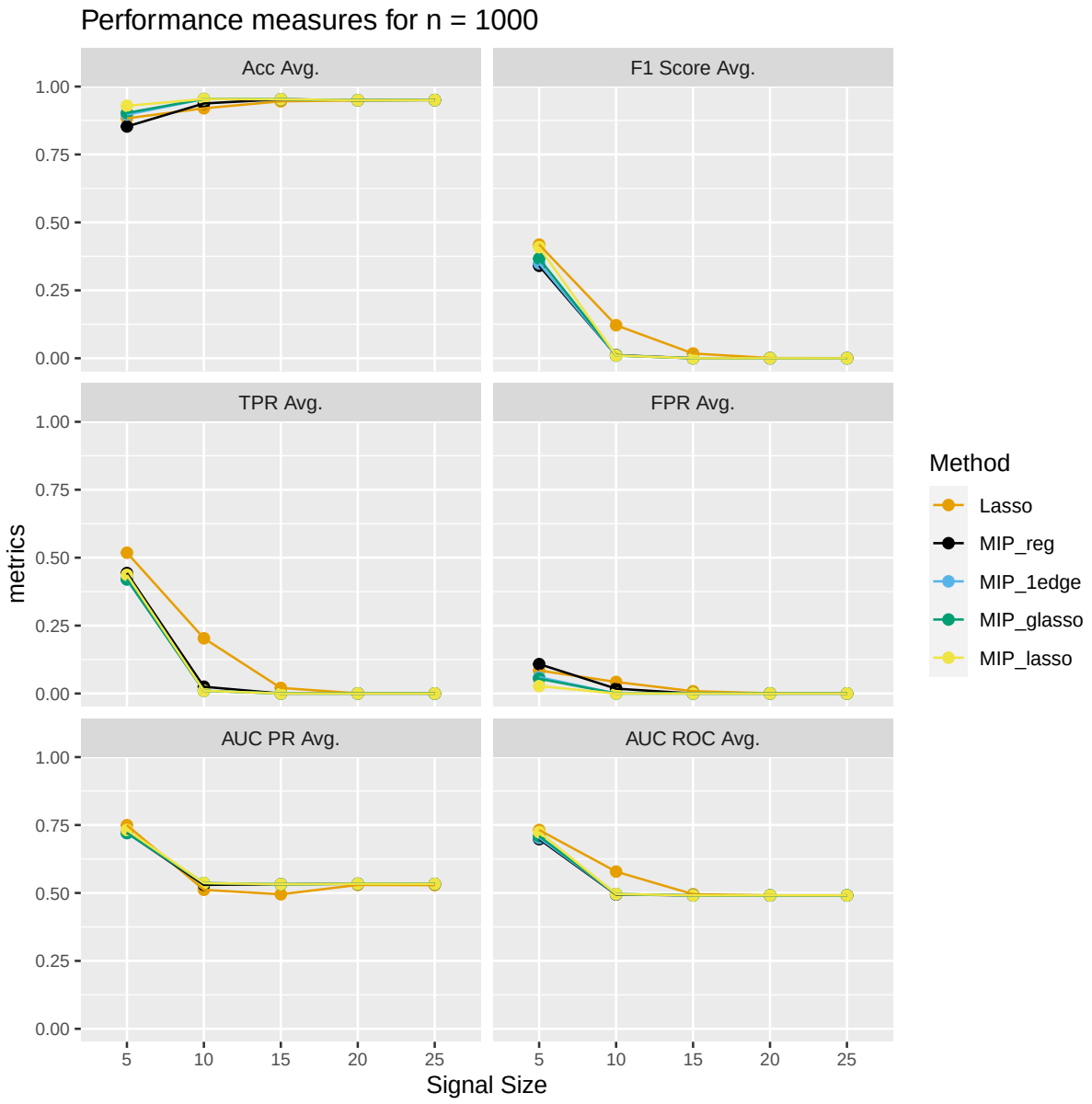


Figure A.20 Variation of performance metrics with graph size and at edge probability 5 % on unseen dataset drawn from a sample size of 1000 (test data size 20 %). The metrics are computed for graphs selected via BIC and are averaged results over 100 different signals.

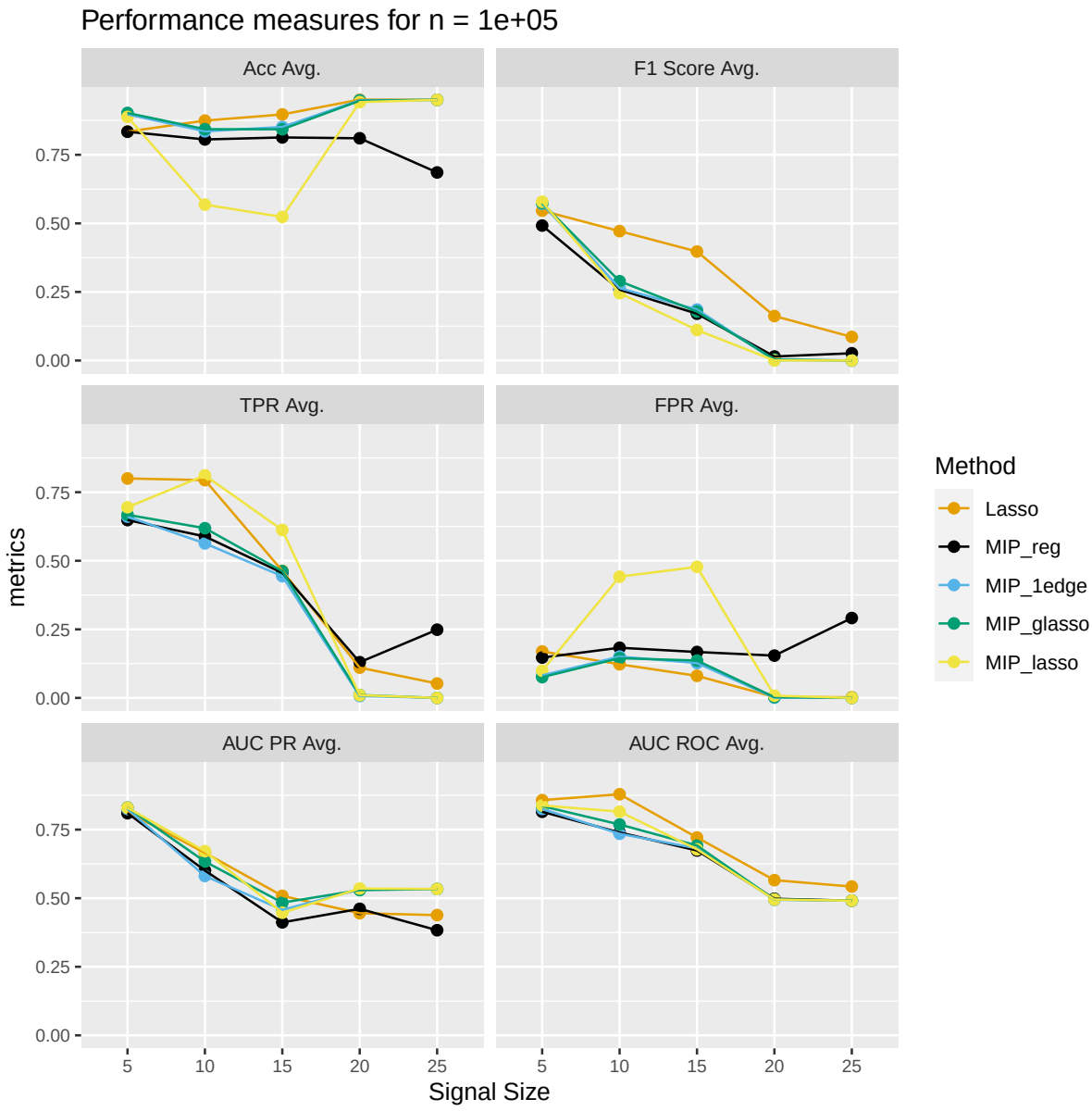


Figure A.21 Variation of performance metrics with graph size and at edge probability 5 % on unseen dataset drawn from a sample size of 100,000 (test data size 20 %). The metrics are computed for graphs selected via BIC and are averaged results over 100 different signals.

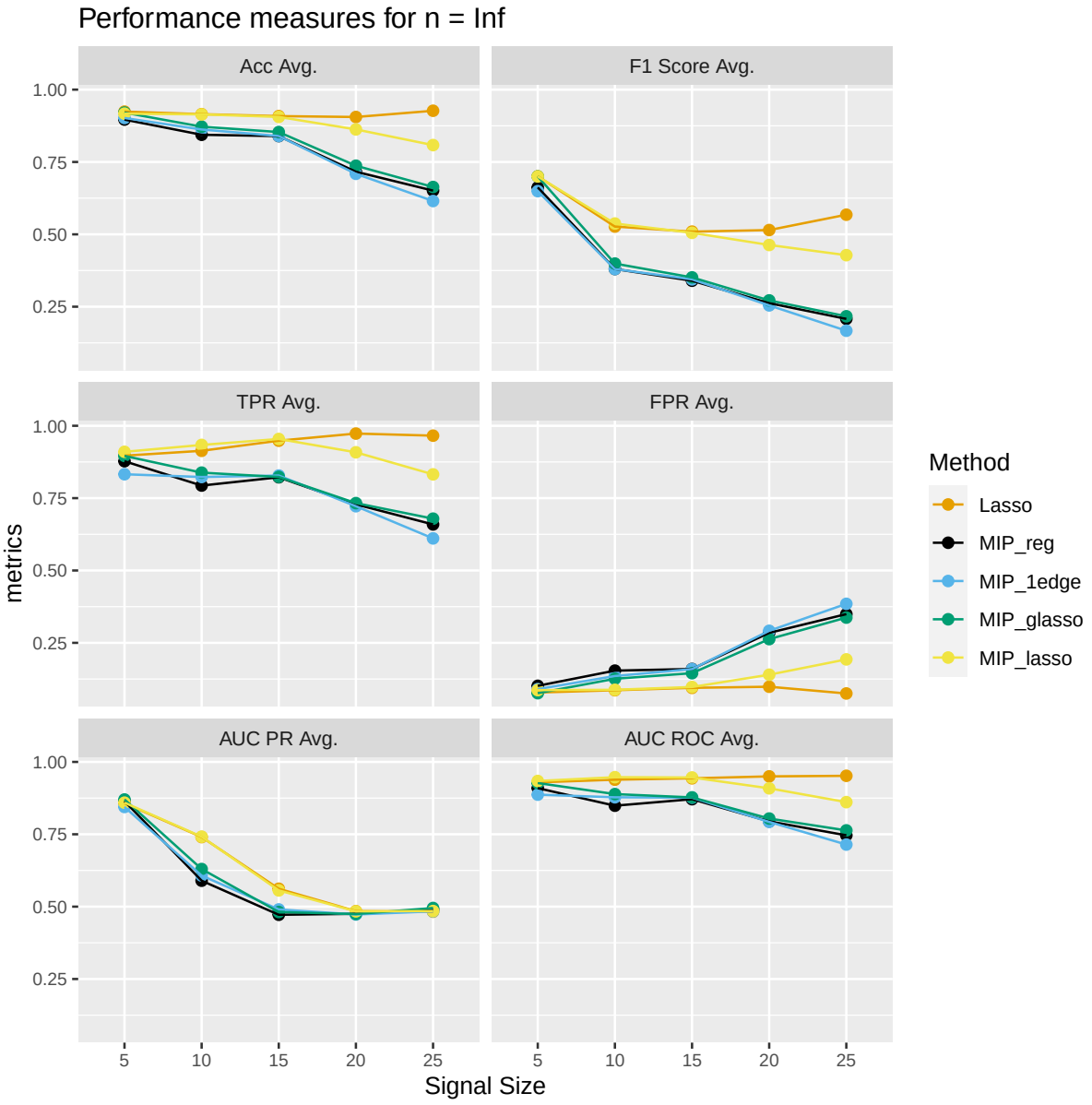


Figure A.22 Variation of performance metrics with graph size and at edge probability 5 % on unseen dataset drawn from population covariance matrix (test data size 20 %). The metrics are computed for graphs selected via BIC and are averaged results over 100 different signals.

A.6 Prediction with BIC when edge probability 25 %: Comparison along signal sizes

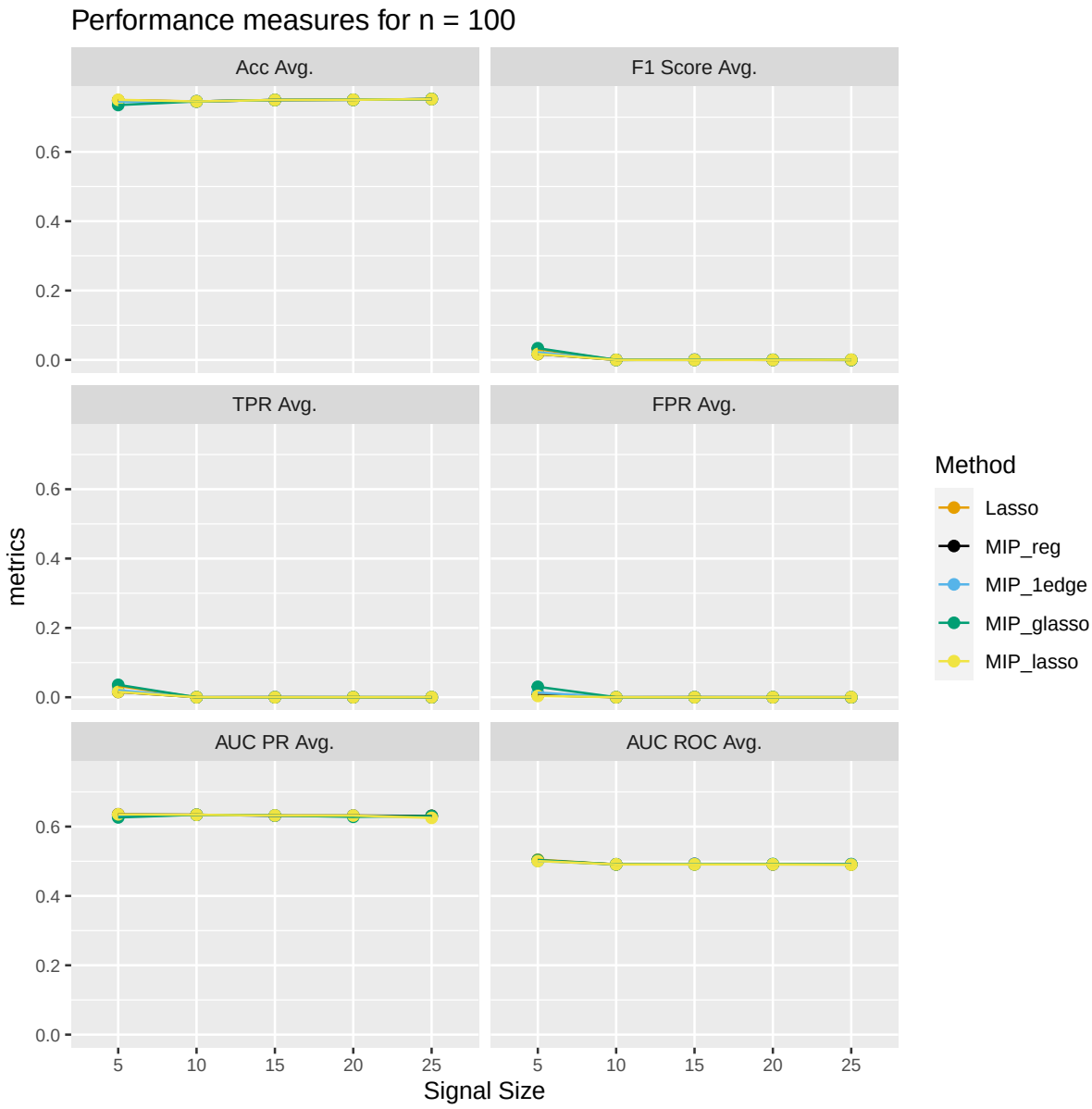


Figure A.23 Variation of performance metrics with graph size and at edge probability 25 % on unseen dataset drawn from a sample size of 100 (test data size 20 %). The metrics are computed for graphs selected via BIC and are averaged results over 100 different signals.

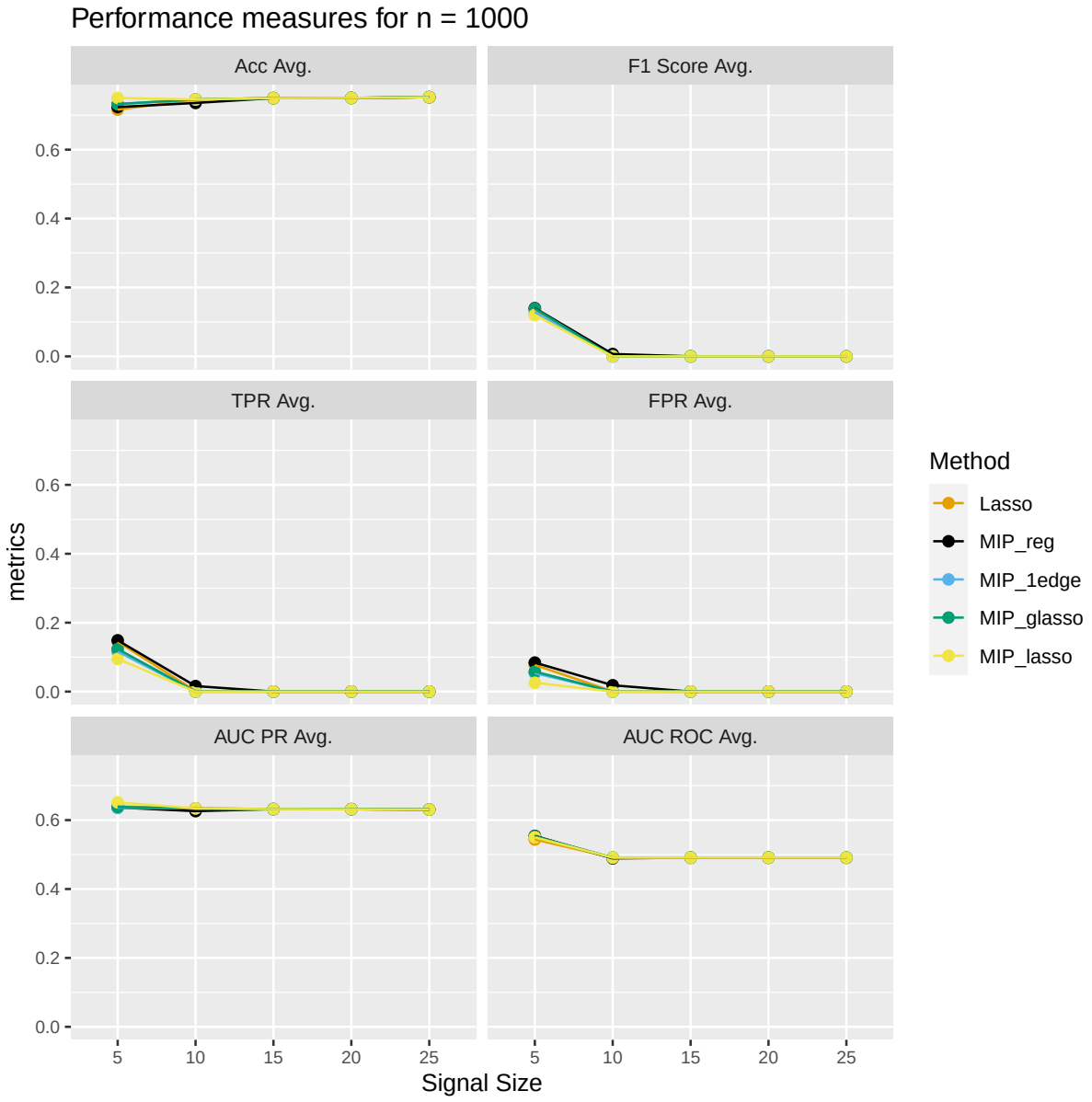


Figure A.24 Variation of performance metrics with graph size and at edge probability 25 % on unseen dataset drawn from a sample size of 1000 (test data size 20 %). The metrics are computed for graphs selected via BIC and are averaged results over 100 different signals.

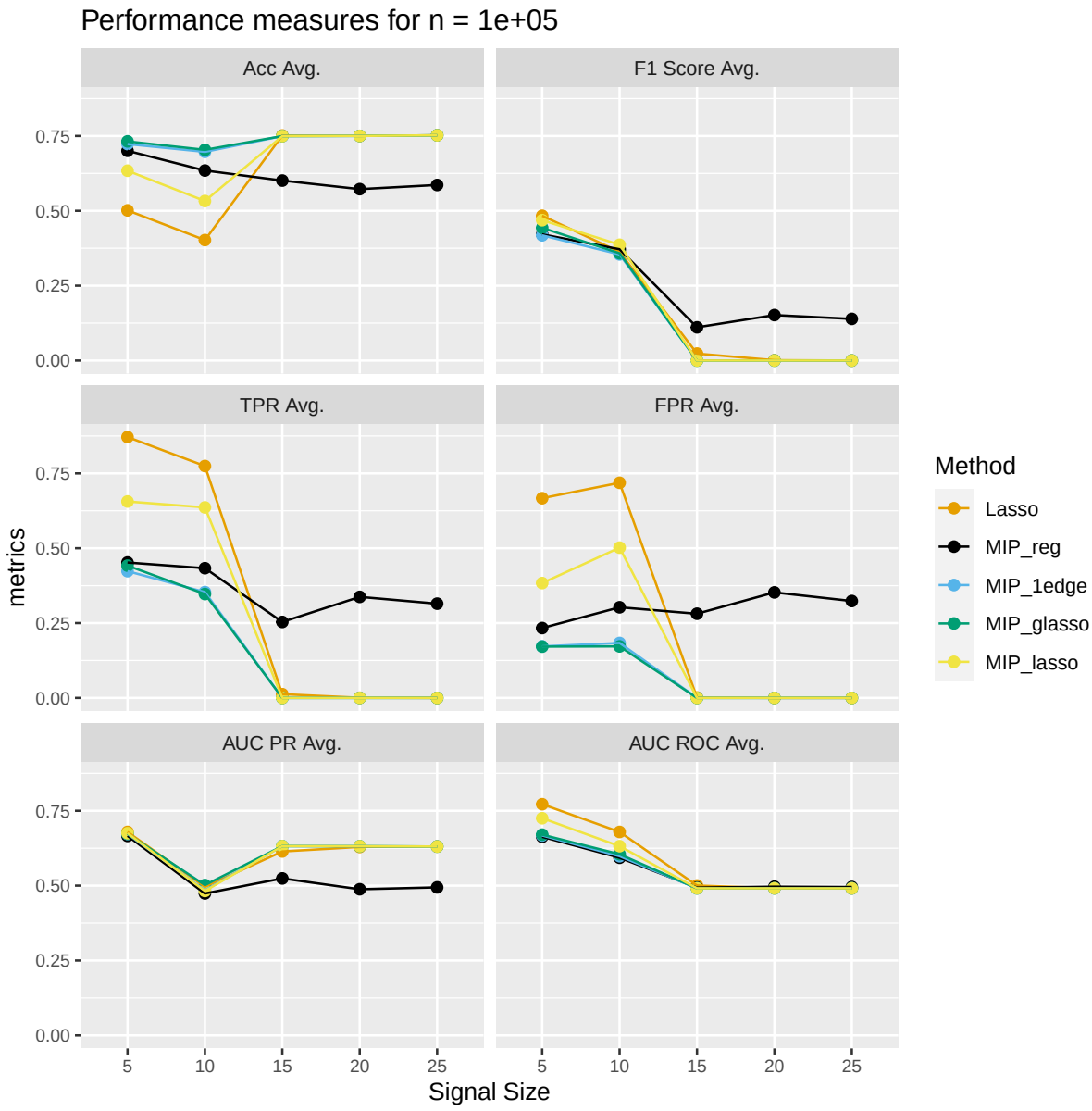


Figure A.25 Variation of performance metrics with graph size and at edge probability 25 % on unseen dataset drawn from a sample size of 100,000 (test data size 20 %). The metrics are computed for graphs selected via BIC and are averaged results over 100 different signals.

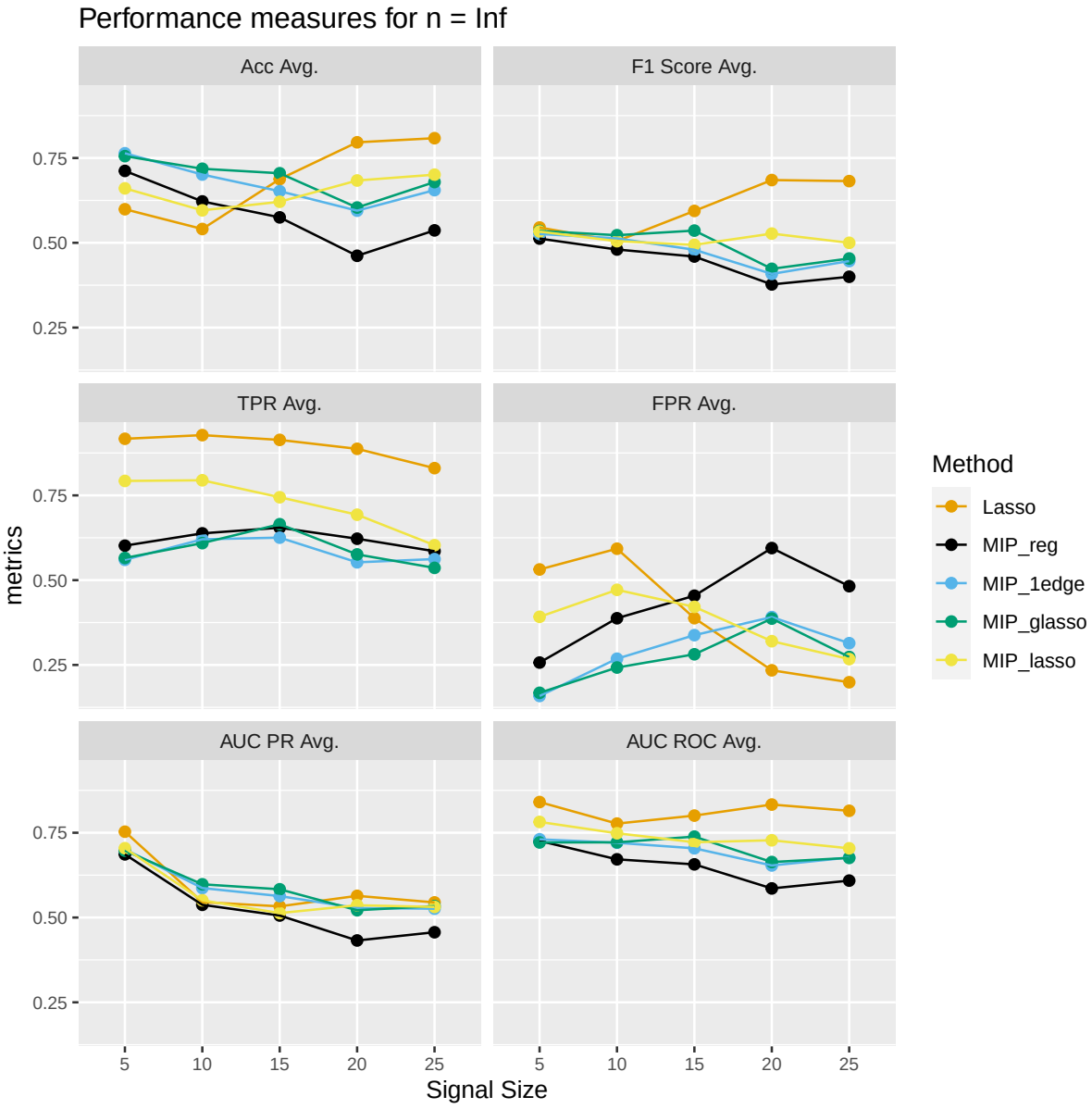


Figure A.26 Variation of performance metrics with graph size and at edge probability 25 % on unseen dataset drawn from population covariance matrix (test data size 20 %). The metrics are computed for graphs selected via BIC and are averaged results over 100 different signals.

A.7 Prediction with BIC when edge probability 5 %: Comparison along sample size

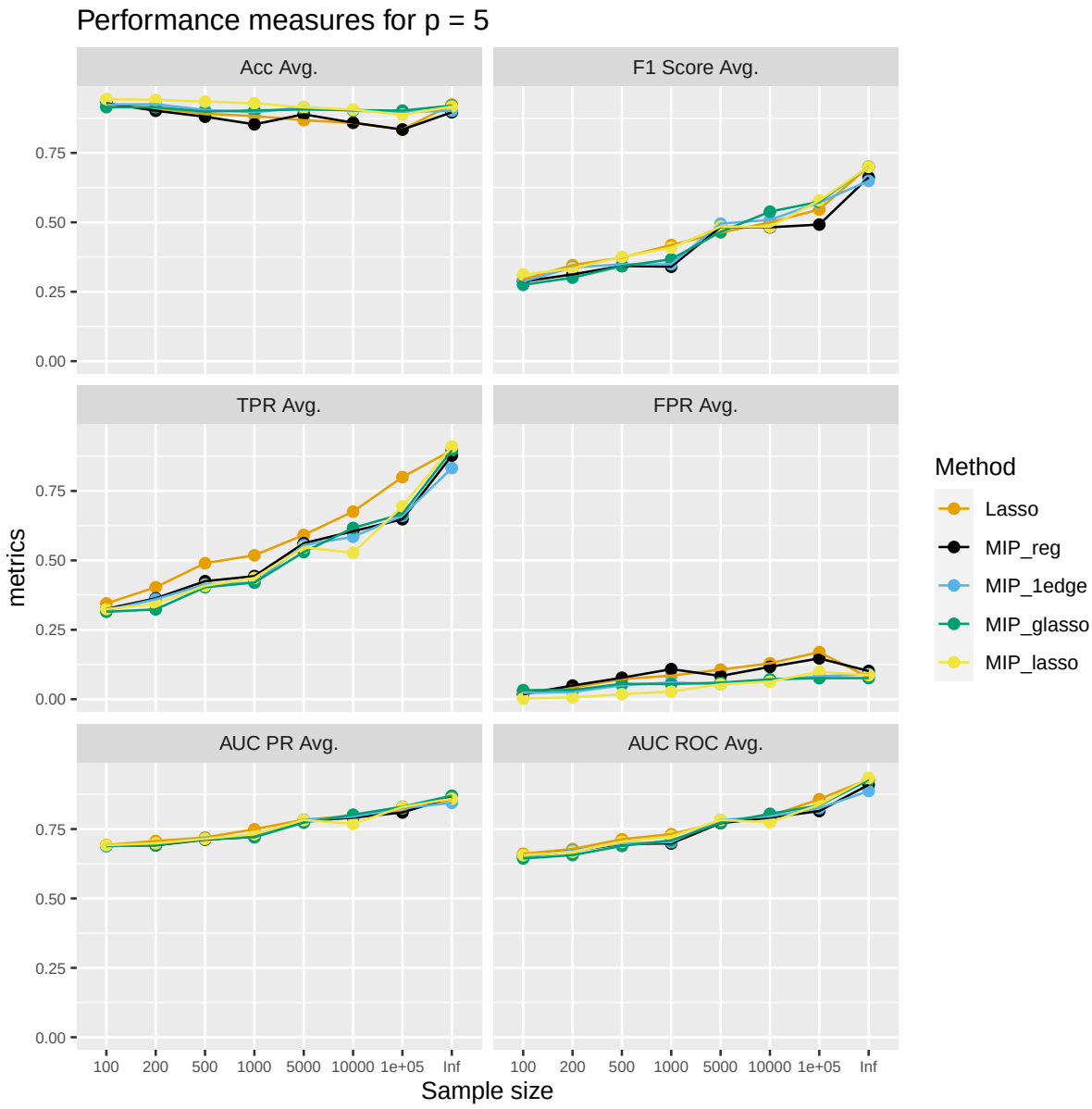


Figure A.27 Variation of performance metrics with sample size and at edge probability 5 % on unseen dataset drawn from a graph size $p = 5$ (test data size 20 %). The metrics are computed for graphs selected via BIC and are averaged results over 100 different signals.

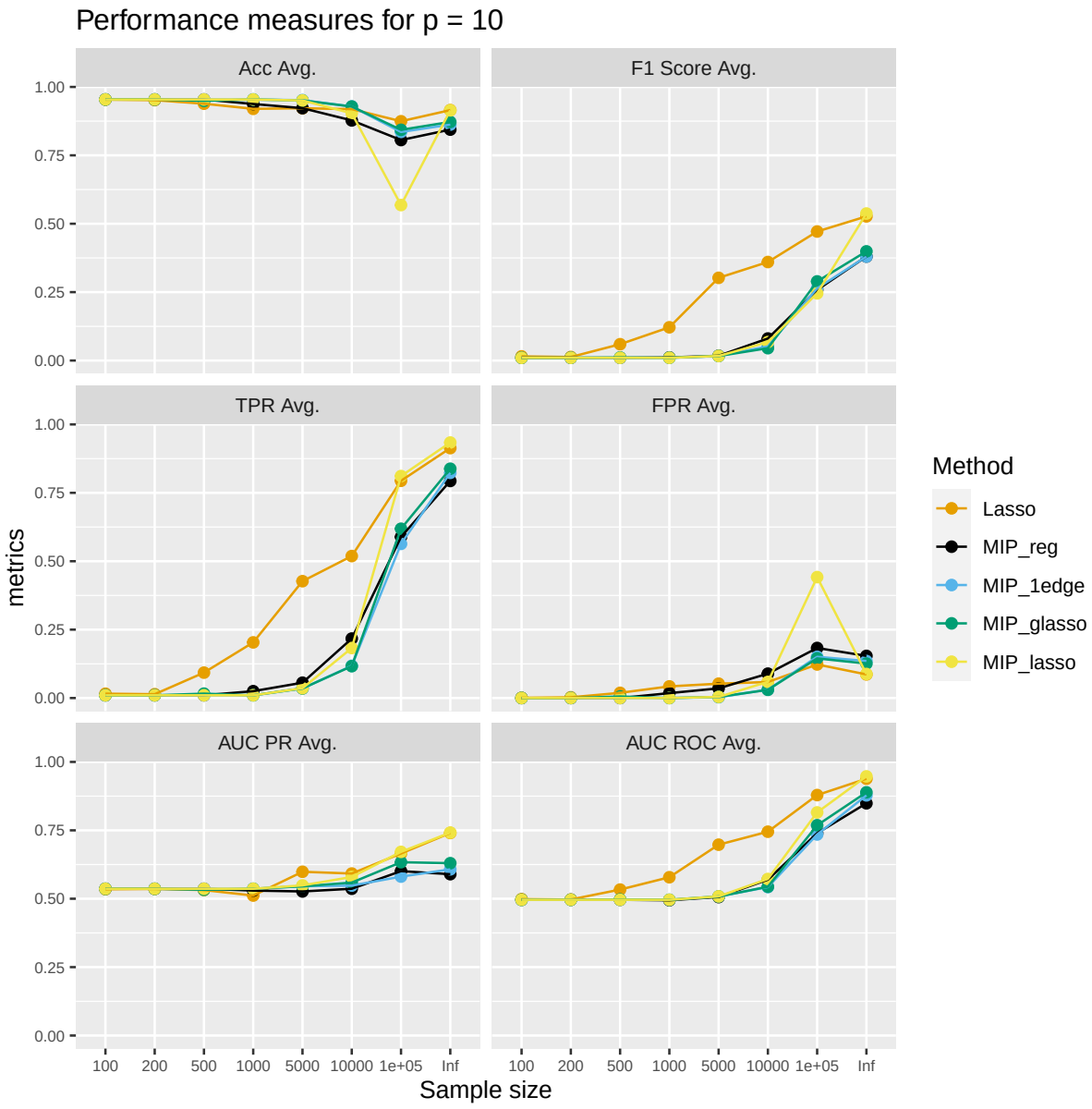


Figure A.28 Variation of performance metrics with graph size and at edge probability 5 % on unseen dataset drawn from a graph size $p = 10$ (test data size 20 %). The metrics are computed for graphs selected via BIC and are averaged results over 100 different signals.

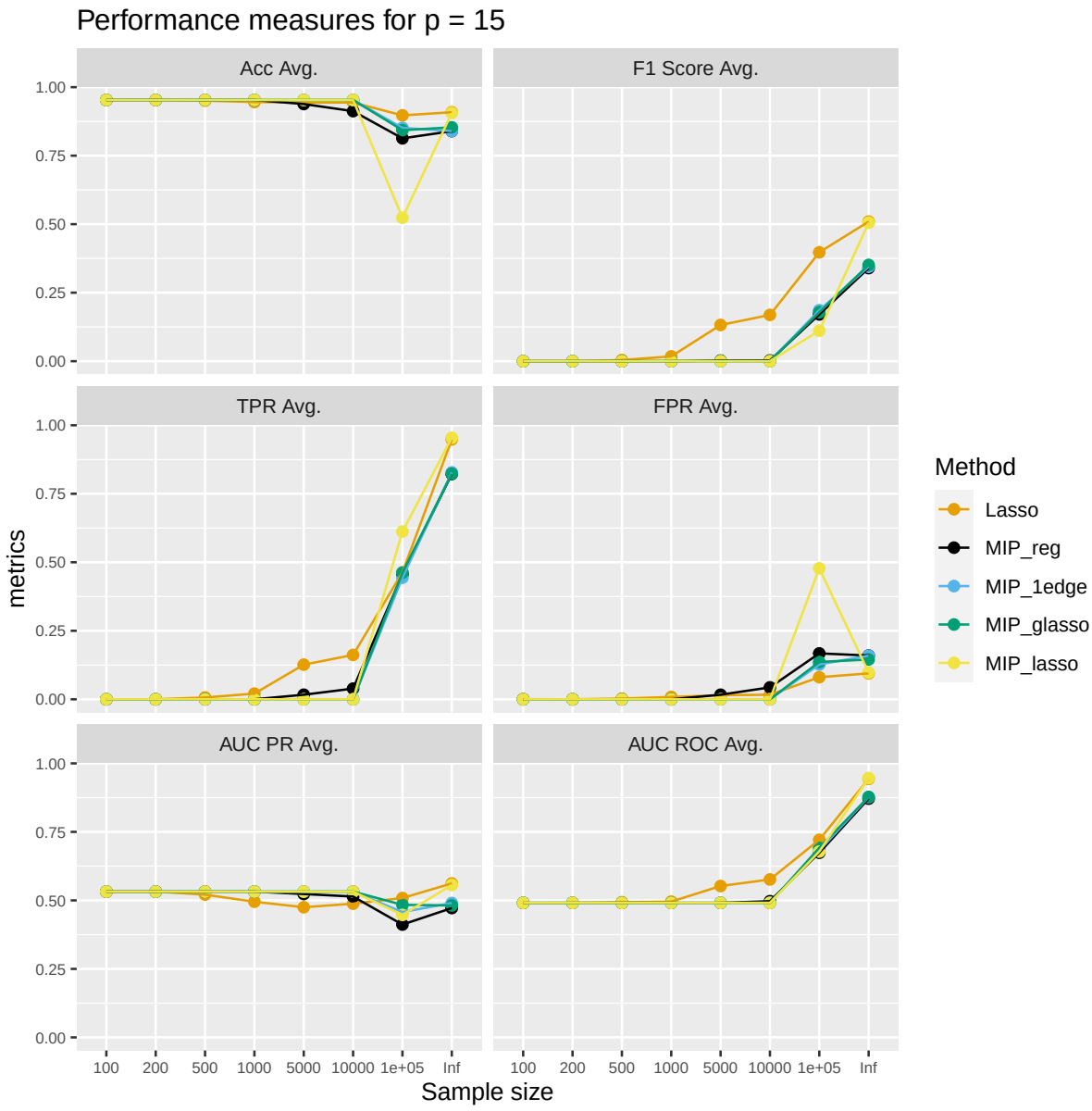


Figure A.29 Variation of performance metrics with graph size and at edge probability 5 % on unseen dataset drawn from a graph size $p = 15$ (test data size 20 %). The metrics are computed for graphs selected via BIC and are averaged results over 100 different signals.

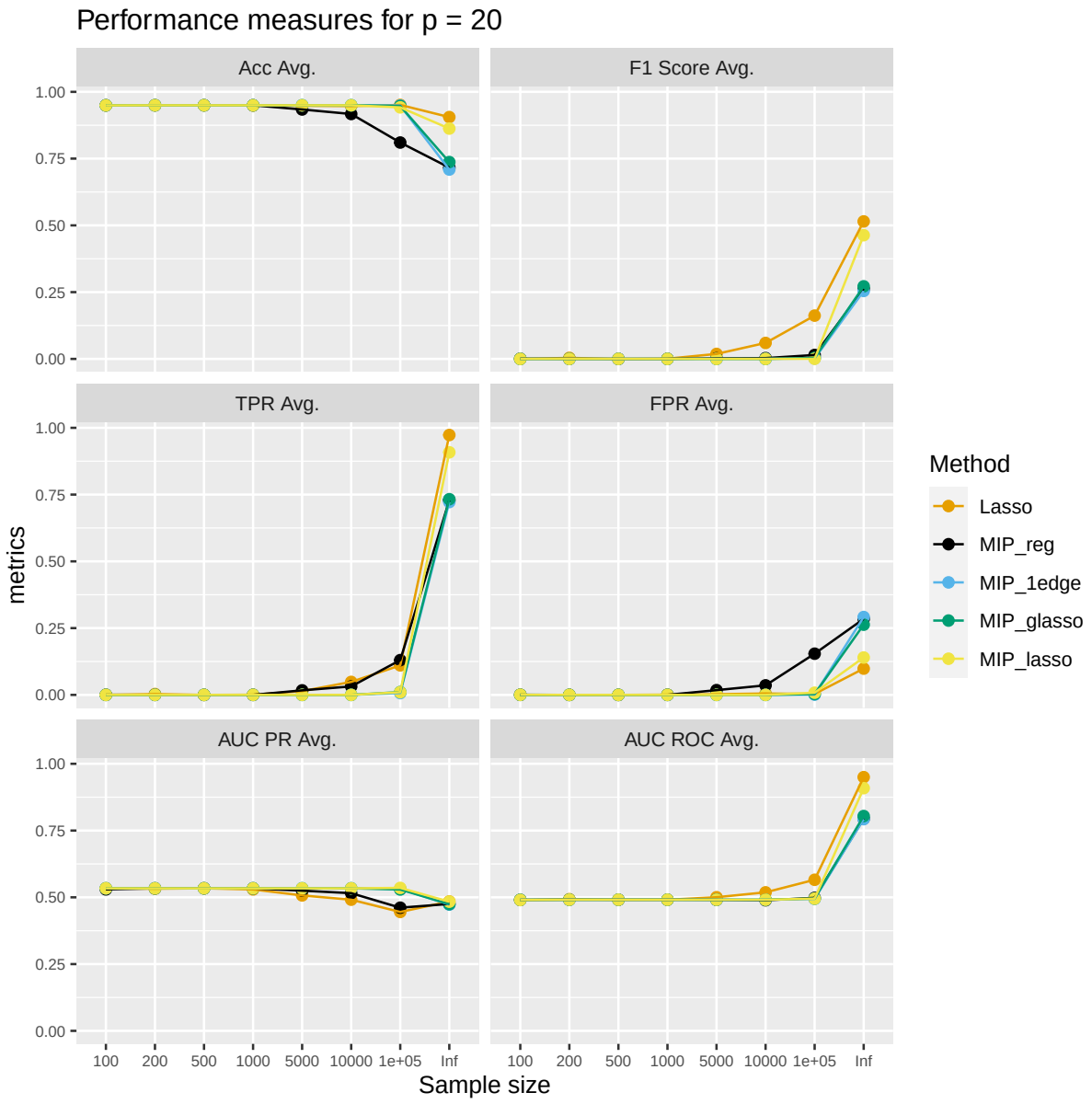


Figure A.30 Variation of performance metrics with graph size and at edge probability 5 % on unseen dataset drawn from a graph size $p = 20$ (test data size 20 %). The metrics are computed for graphs selected via BIC and are averaged results over 100 different signals.

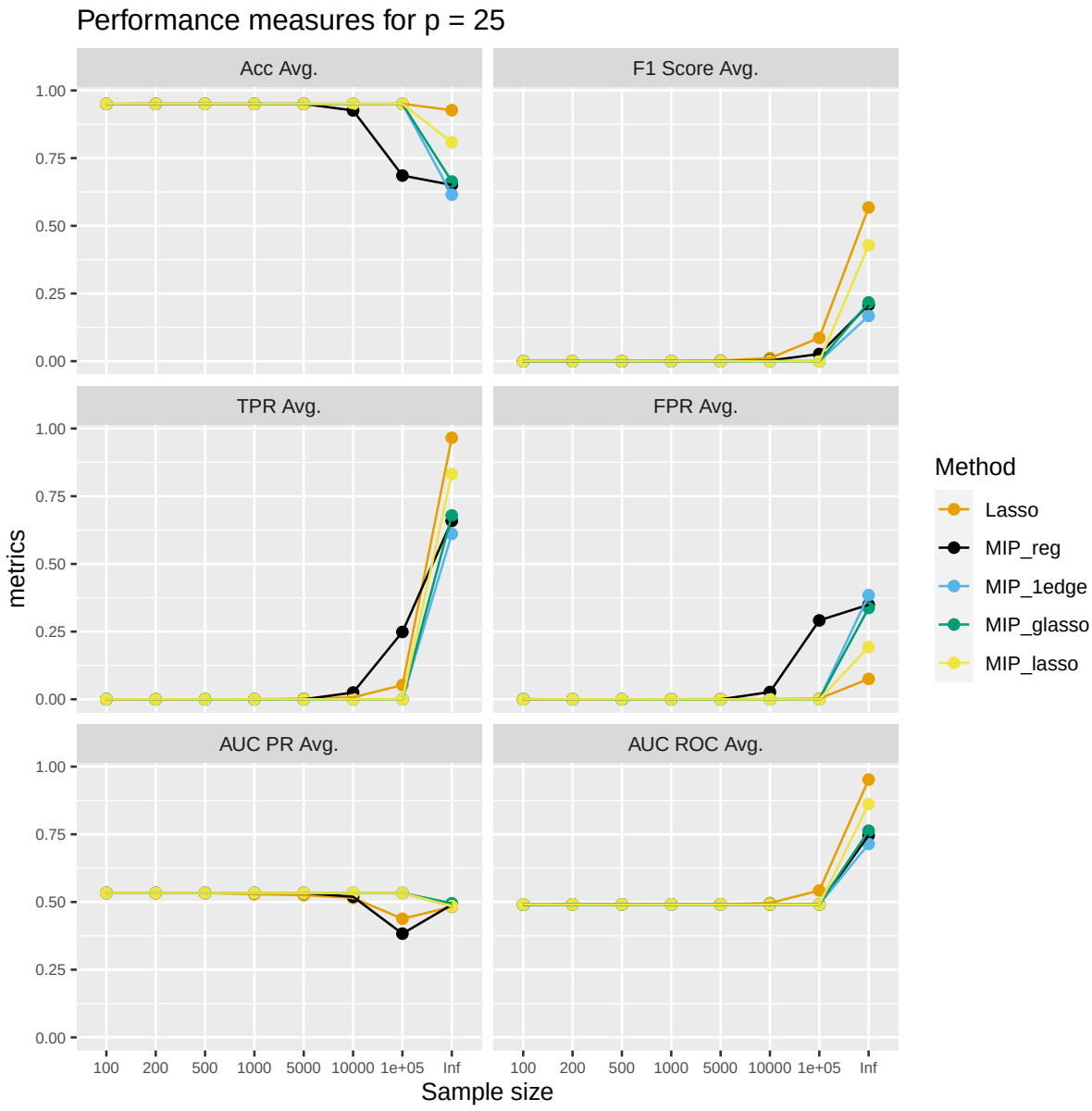


Figure A.31 Variation of performance metrics with graph size and at edge probability 5 % on unseen dataset drawn from a graph size $p = 25$ (test data size 20 %). The metrics are computed for graphs selected via BIC and are averaged results over 100 different signals.

A.8 Prediction with BIC when edge probability 25 %: Comparison along sample sizes

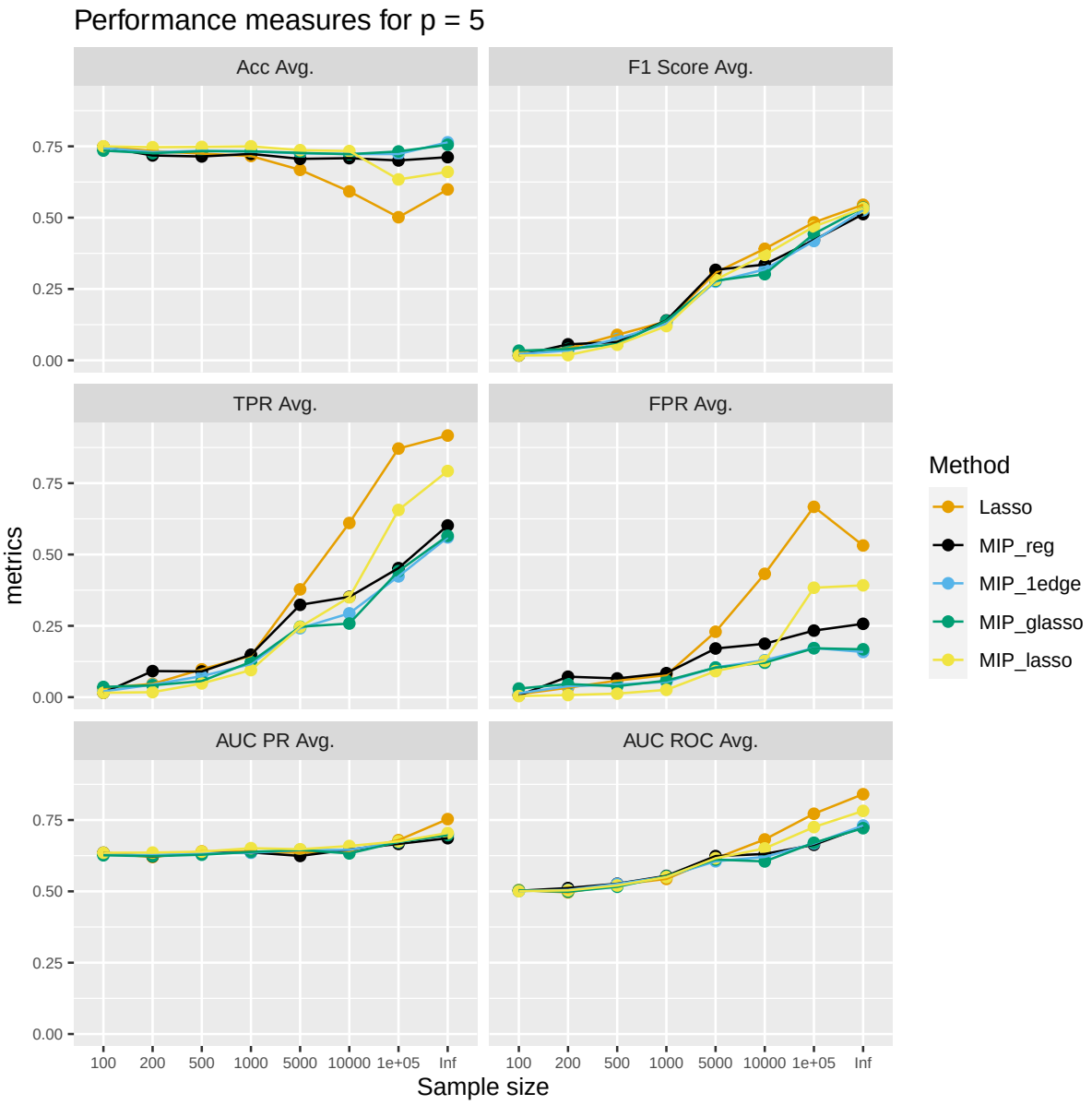


Figure A.32 Variation of performance metrics with graph size and at edge probability 25 % on unseen dataset drawn from a graph size $p = 5$ (test data size 20 %). The metrics are computed for graphs selected via BIC and are averaged results over 100 different signals.

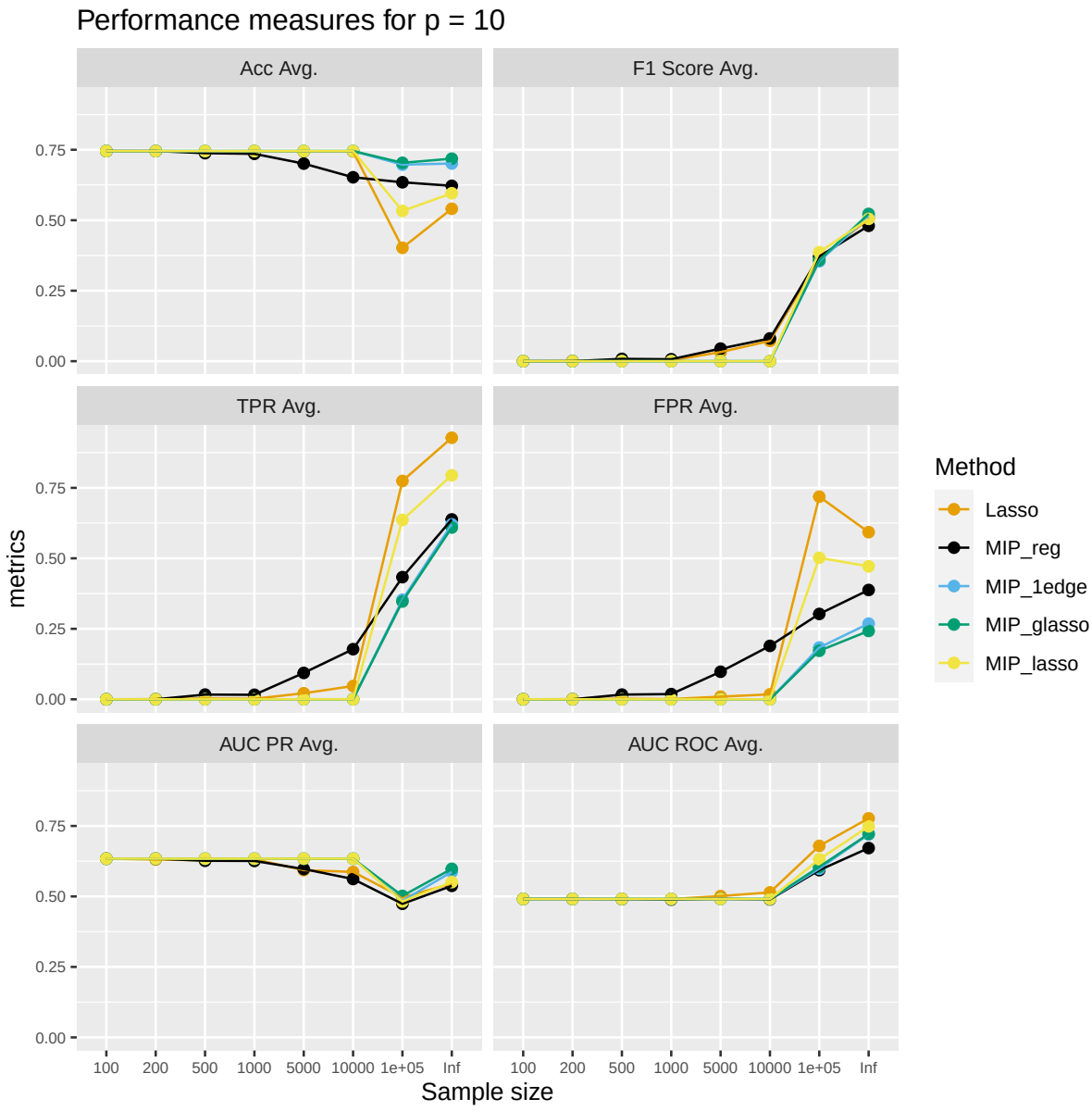


Figure A.33 Variation of performance metrics with graph size and at edge probability 25 % on unseen dataset drawn from a graph size $p = 10$ (test data size 20 %). The metrics are computed for graphs selected via BIC and are averaged results over 100 different signals.

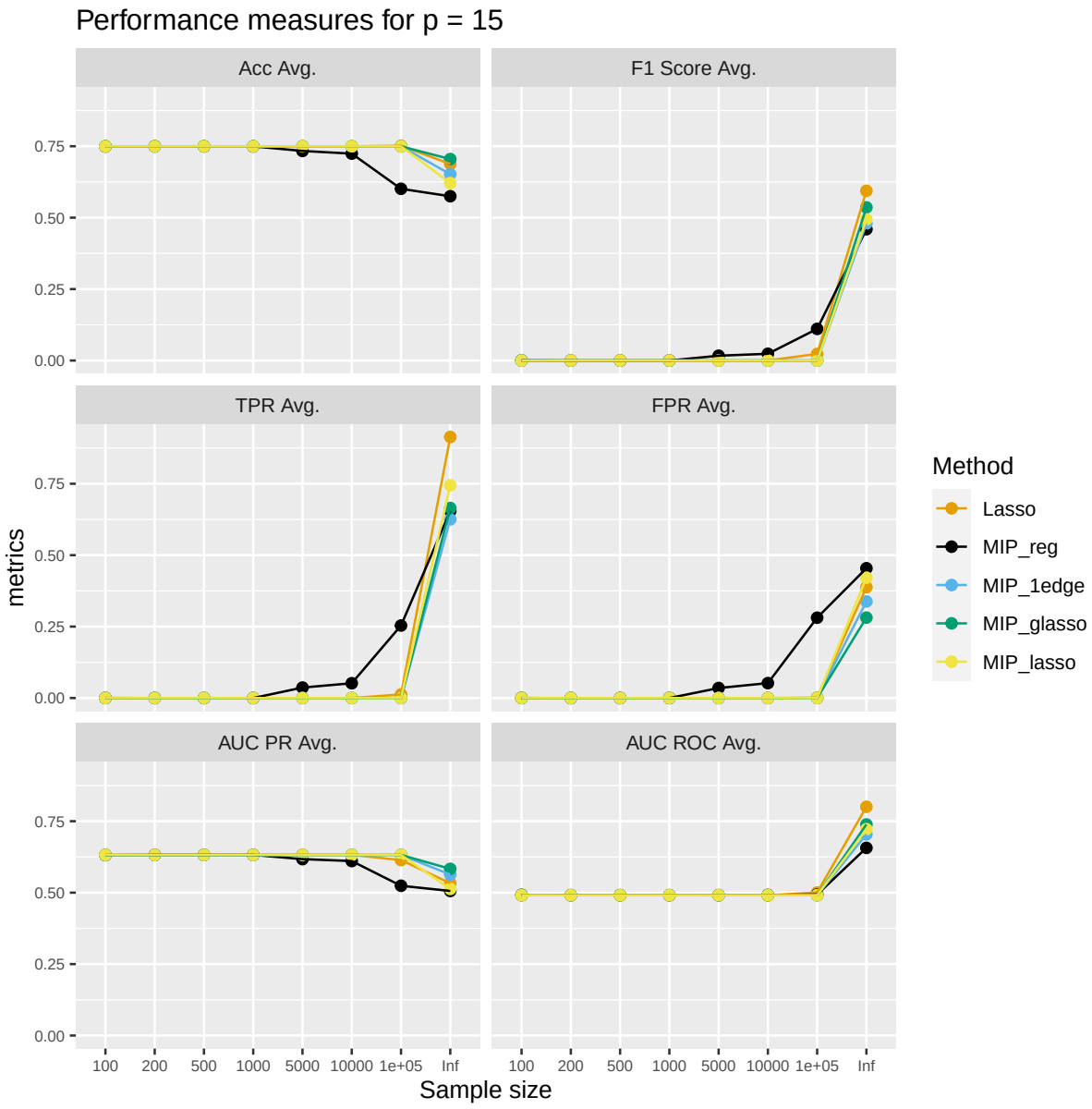


Figure A.34 Variation of performance metrics with graph size and at edge probability 25 % on unseen dataset drawn from a graph size $p = 15$ (test data size 20 %). The metrics are computed for graphs selected via BIC and are averaged results over 100 different signals.

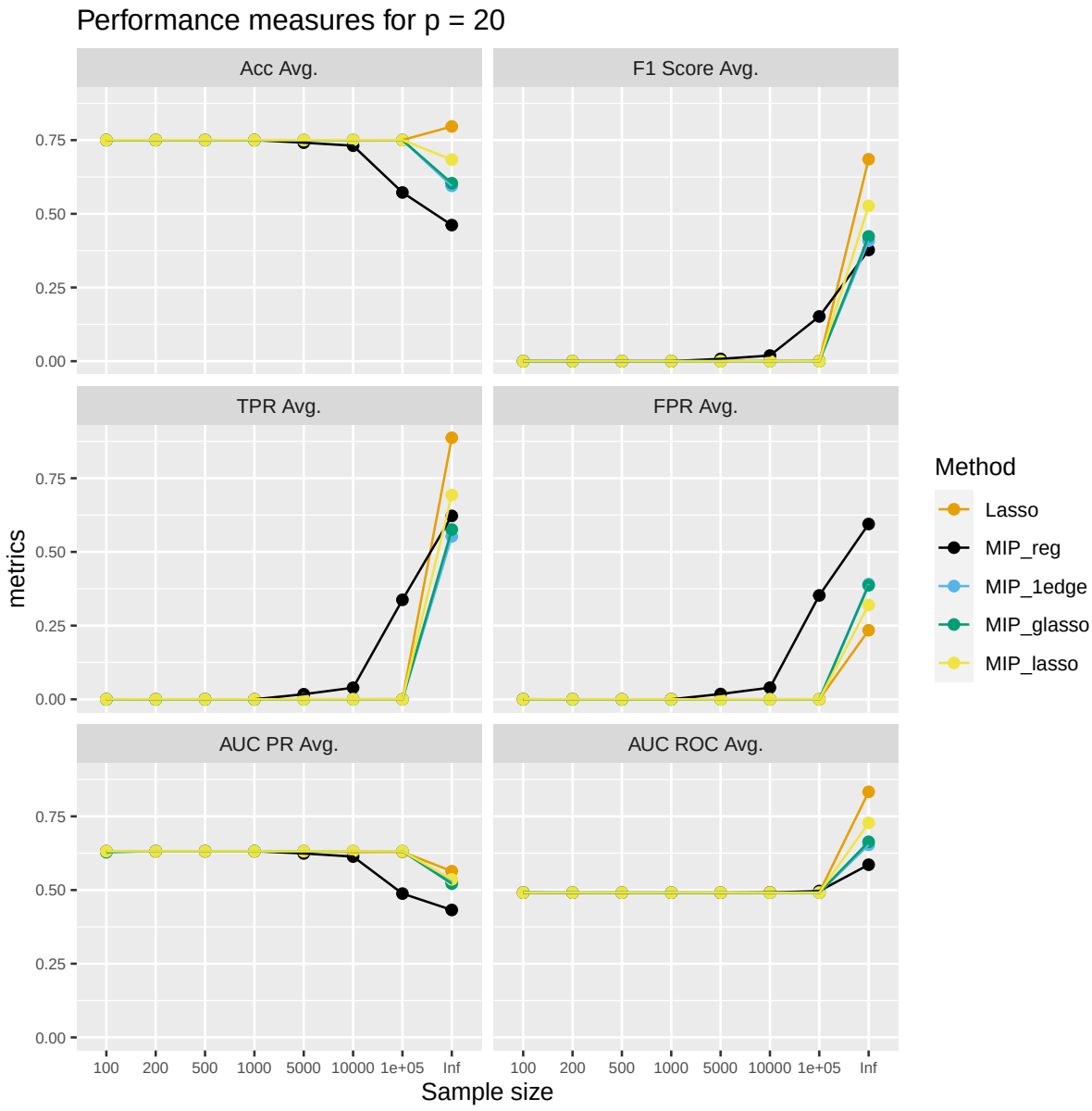


Figure A.35 Variation of performance metrics with graph size and at edge probability 25 % on unseen dataset drawn from a graph size $p = 20$ (test data size 20 %). The metrics are computed for graphs selected via BIC and are averaged results over 100 different signals.

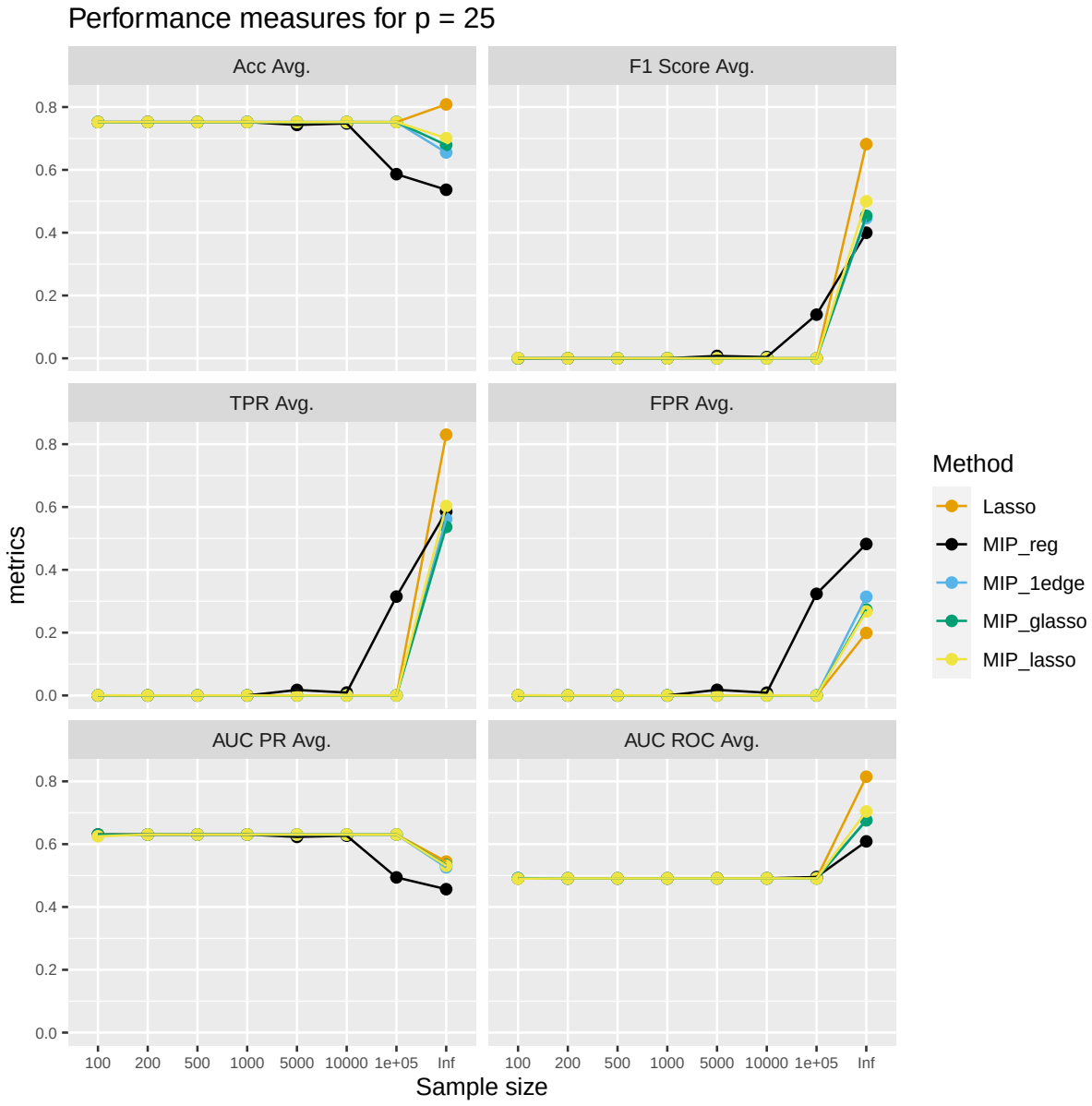


Figure A.36 Variation of performance metrics with graph size and at edge probability 25 % on unseen dataset drawn from a graph size $p = 25$ (test data size 20 %). The metrics are computed for graphs selected via BIC and are averaged results over 100 different signals.

A.9 Prediction with eBIC when edge probability 5 %: Comparison along signal sizes

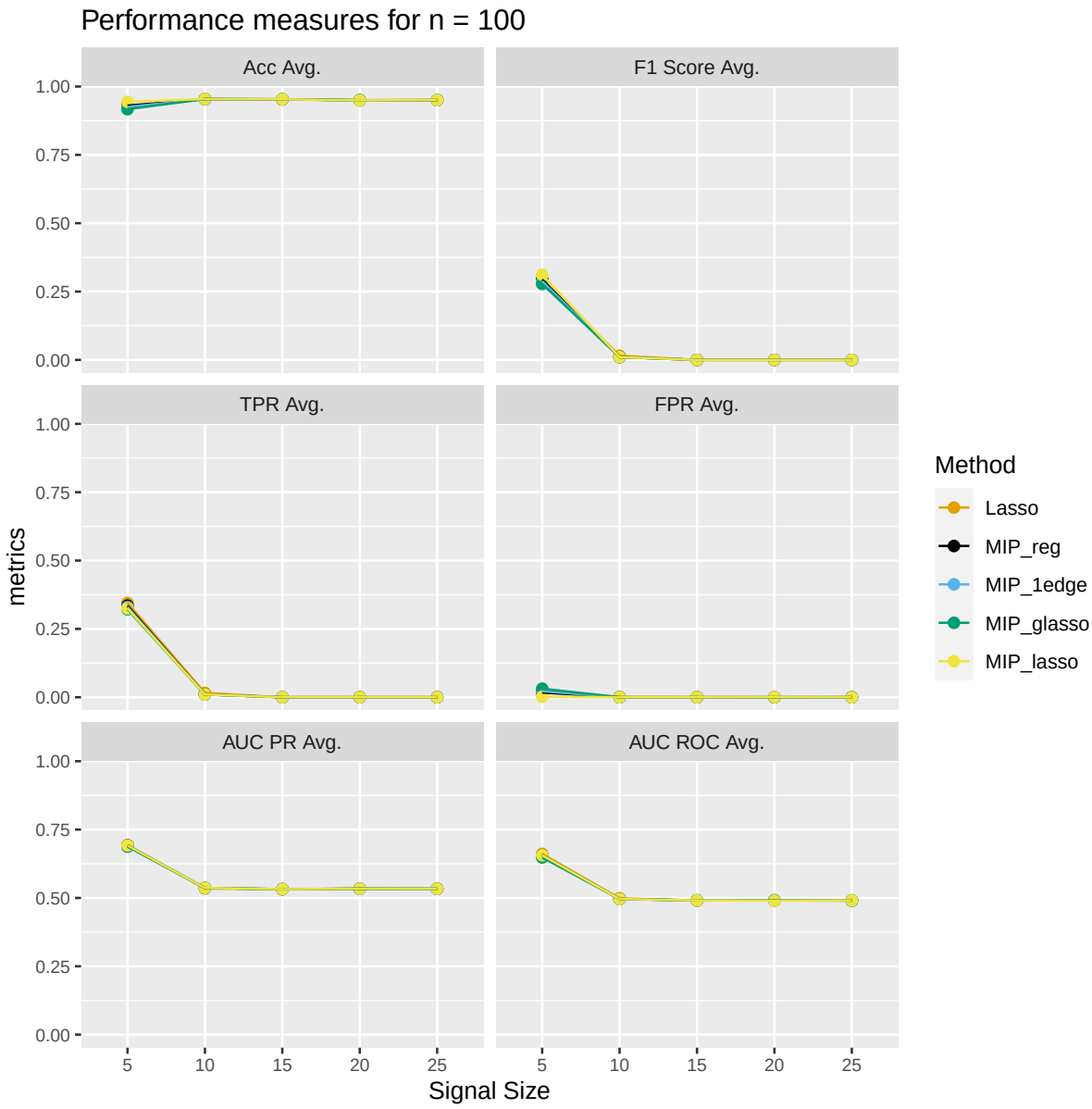


Figure A.37 Variation of performance metrics with graph size and at edge probability 5 % on unseen dataset drawn from a sample size of 100 (test data size 20 %). The metrics are computed for graphs selected via eBIC and are averaged results over 100 different signals.

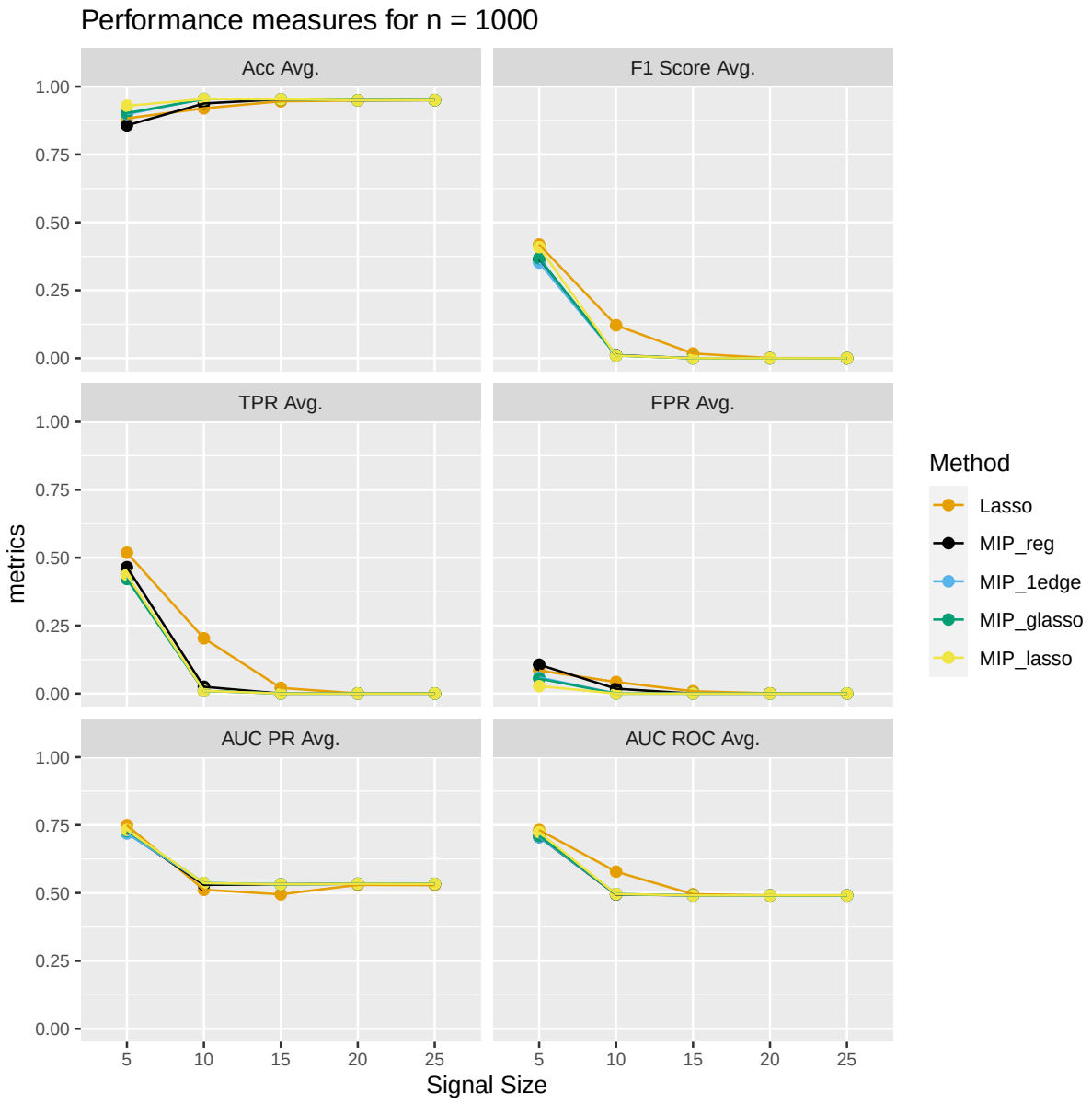


Figure A.38 Variation of performance metrics with graph size and at edge probability 5 % on unseen dataset drawn from a sample size of 1000 (test data size 20 %). The metrics are computed for graphs selected via eBIC and are averaged results over 100 different signals.

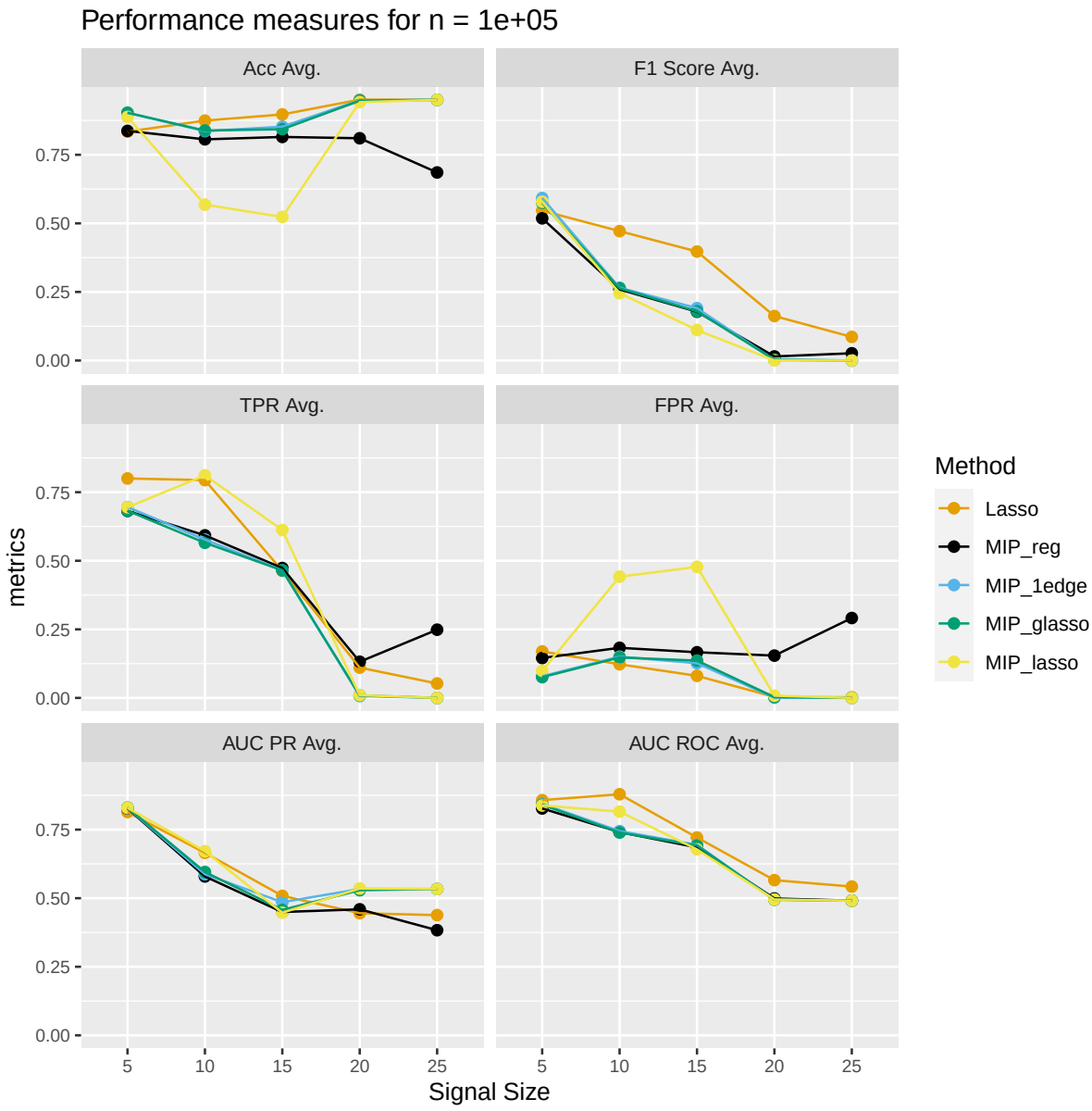


Figure A.39 Variation of performance metrics with graph size and at edge probability 5 % on unseen dataset drawn from a sample size of 100,000 (test data size 20 %). The metrics are computed for graphs selected via eBIC and are averaged results over 100 different signals.

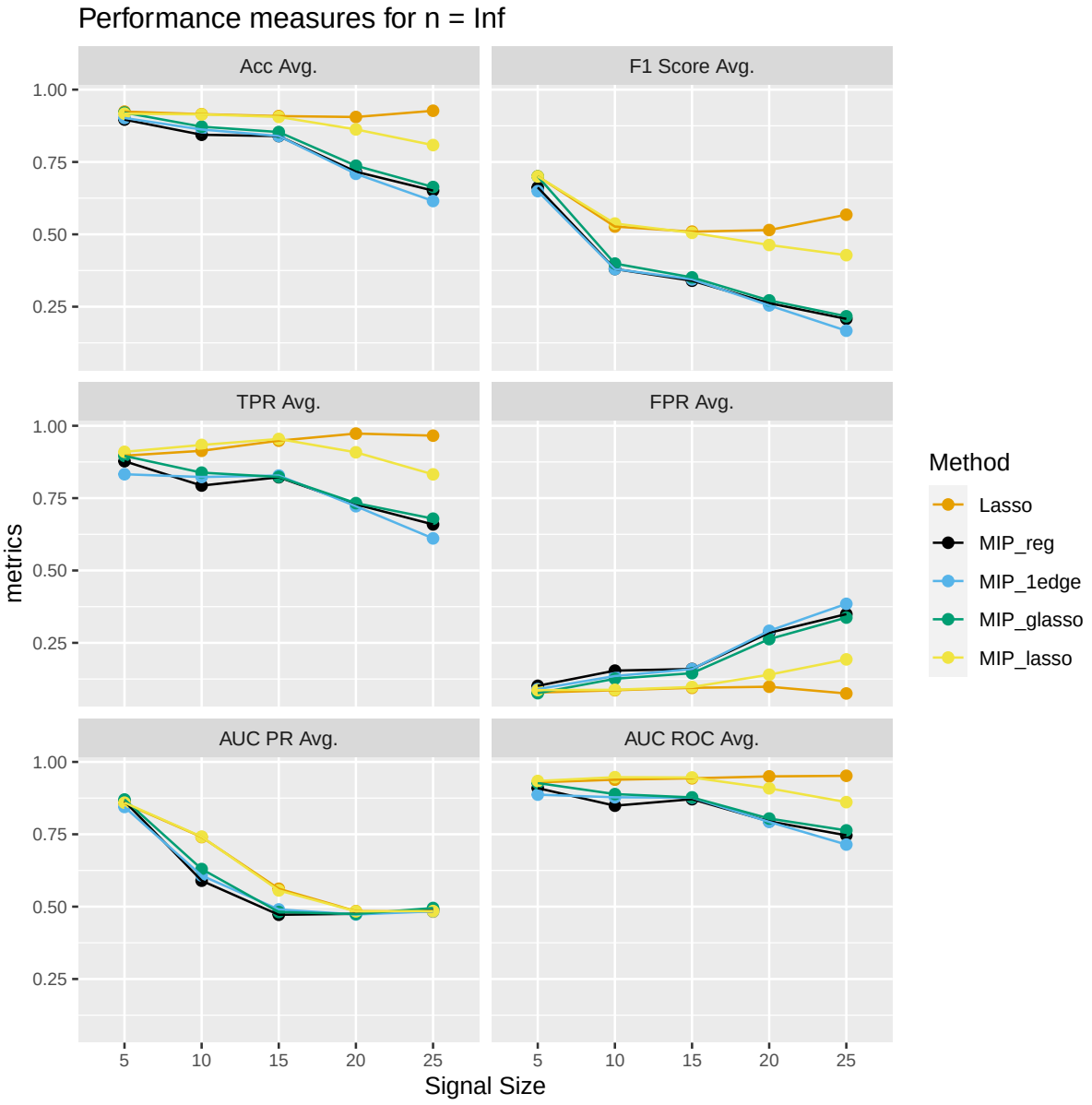


Figure A.40 Variation of performance metrics with graph size and at edge probability 5 % on unseen dataset drawn from population covariance matrix (test data size 20 %). The metrics are computed for graphs selected via eBIC and are averaged results over 100 different signals.

A.10 Prediction with eBIC when edge probability 25 %: Comparison along signal sizes

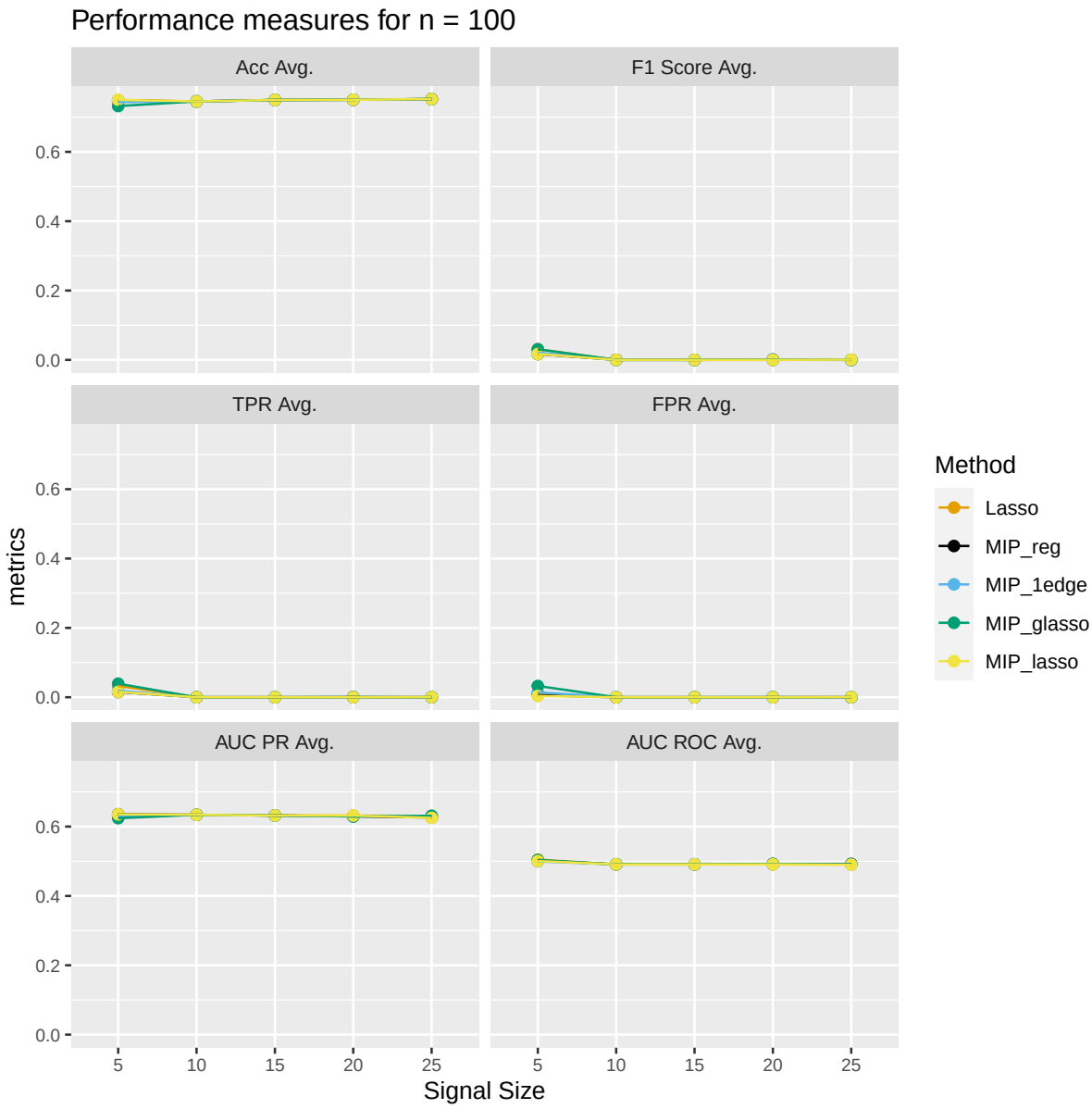


Figure A.41 Variation of performance metrics with graph size and at edge probability 25 % on unseen dataset drawn from a sample size of 100 (test data size 20 %). The metrics are computed for graphs selected via eBIC and are averaged results over 100 different signals.

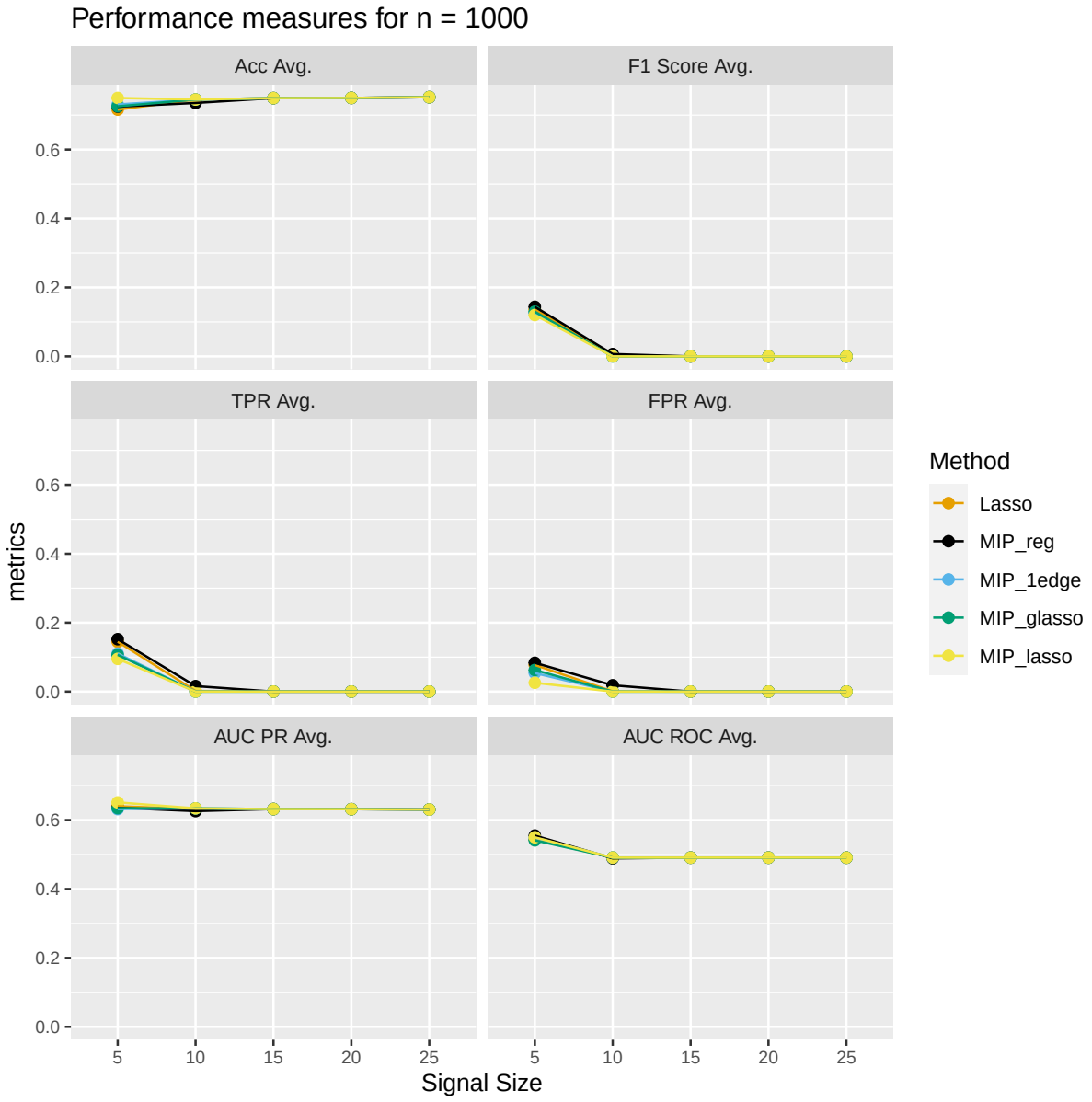


Figure A.42 Variation of performance metrics with graph size and at edge probability 25 % on unseen dataset drawn from a sample size of 1000 (test data size 20 %). The metrics are computed for graphs selected via eBIC and are averaged results over 100 different signals.

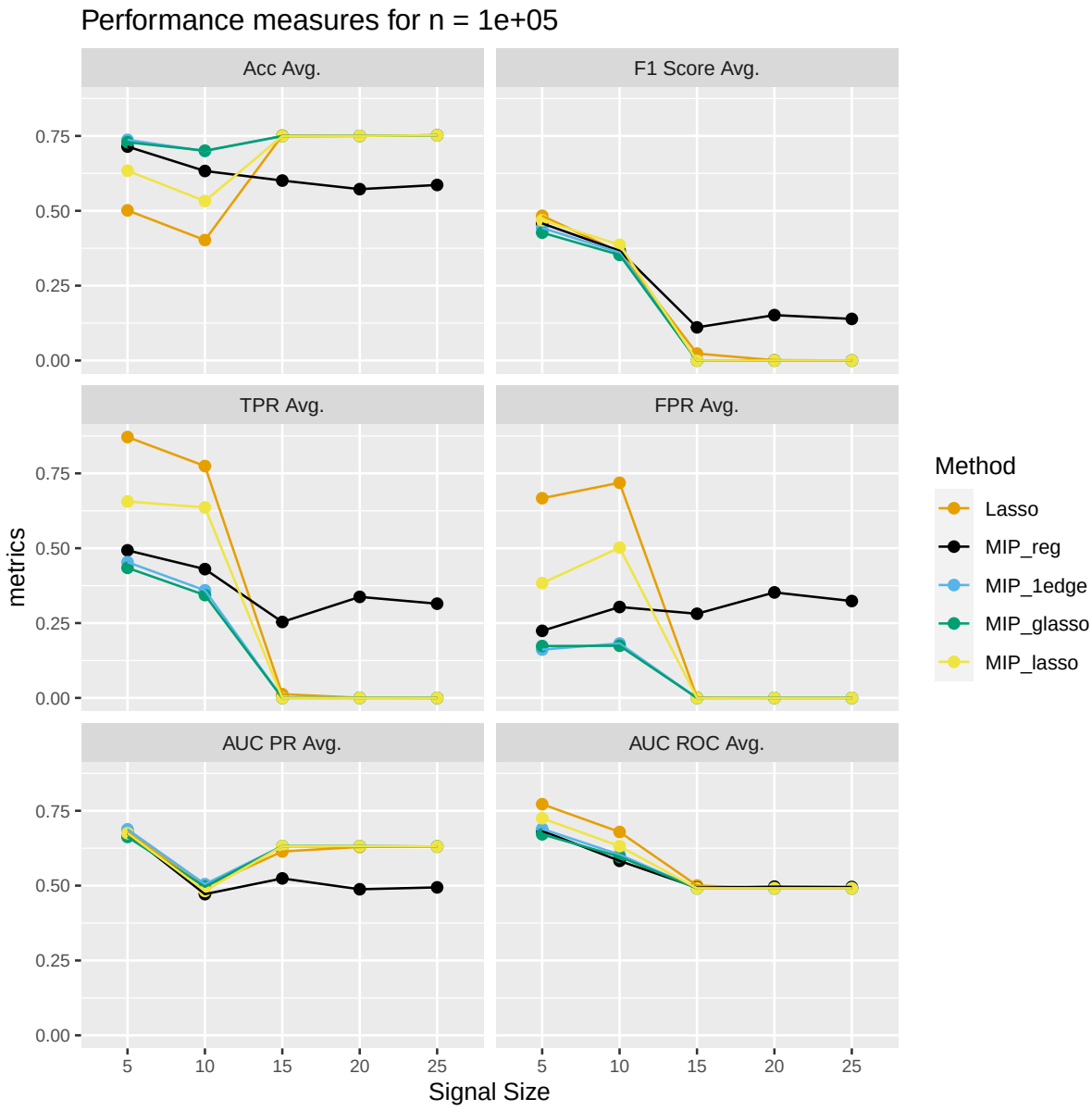


Figure A.43 Variation of performance metrics with graph size and at edge probability 25 % on unseen dataset drawn from a sample size of 100,000 (test data size 20 %). The metrics are computed for graphs selected via eBIC and are averaged results over 100 different signals.

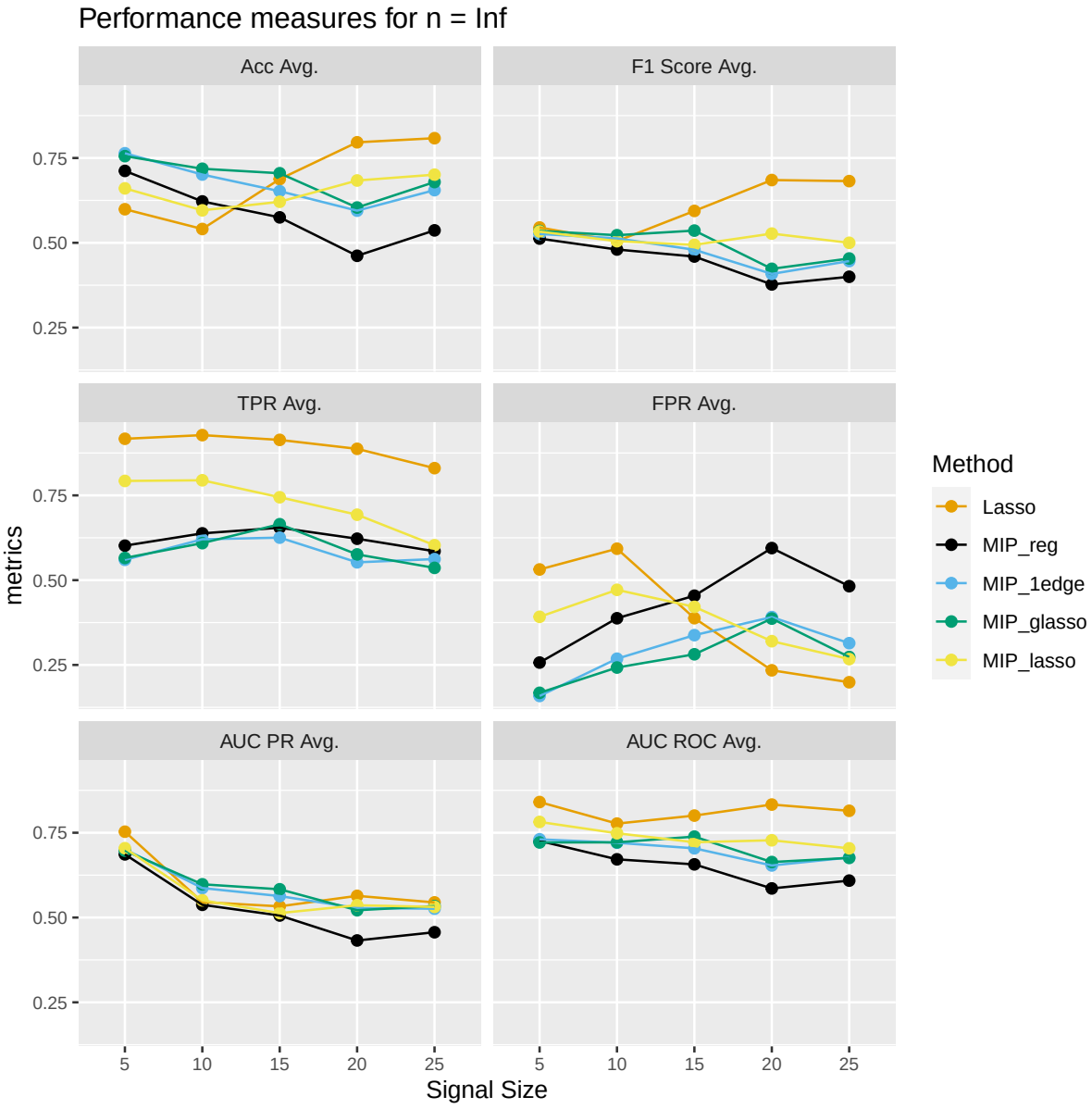


Figure A.44 Variation of performance metrics with graph size and at edge probability 25 % on unseen dataset drawn from population covariance matrix (test data size 20 %). The metrics are computed for graphs selected via eBIC and are averaged results over 100 different signals.

A.11 Prediction with eBIC when edge probability 5 %: Comparison along sample size

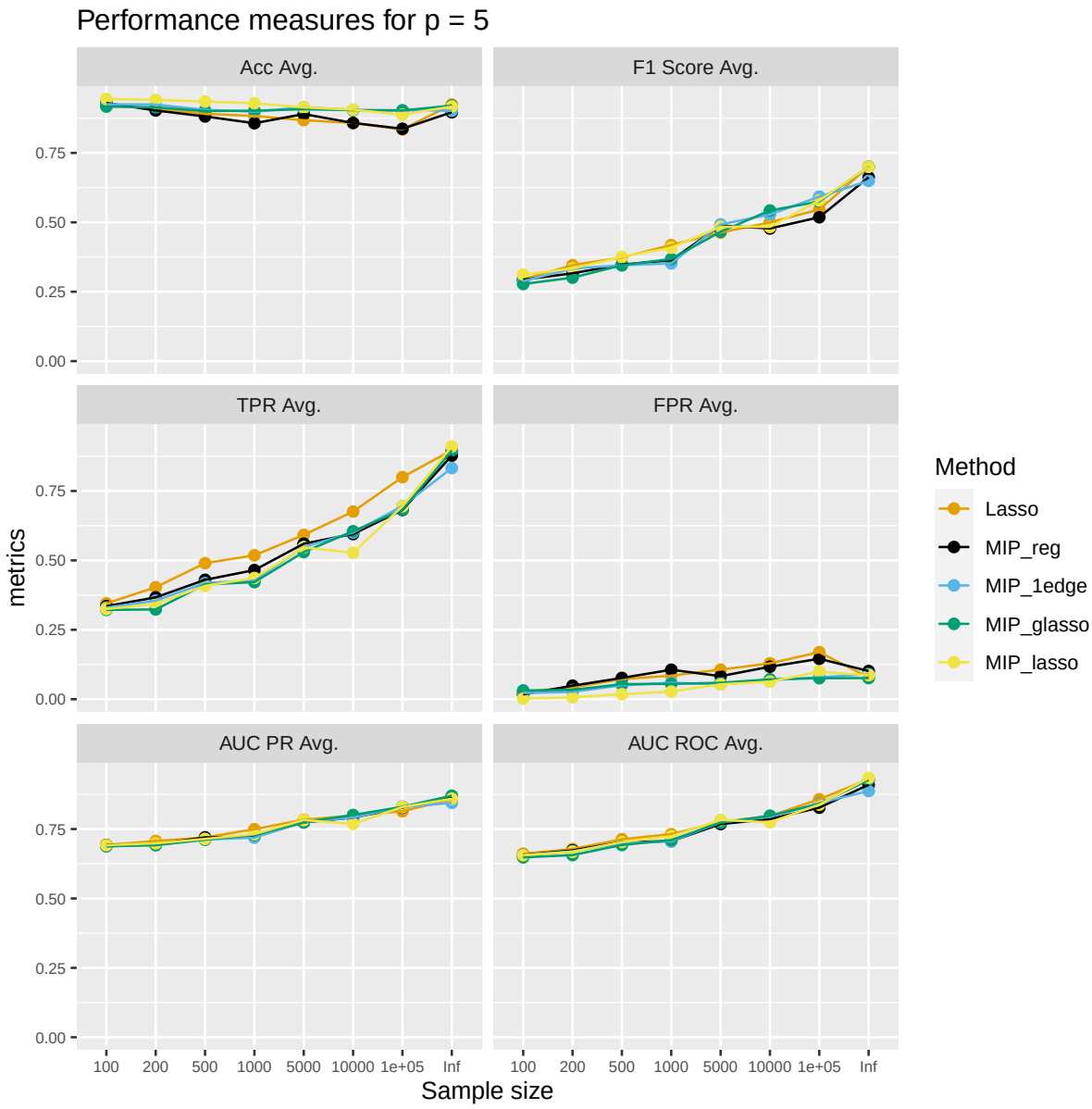


Figure A.45 Variation of performance metrics with sample size and at edge probability 5 % on unseen dataset drawn from a graph size $p = 5$ (test data size 20 %). The metrics are computed for graphs selected via eBIC and are averaged results over 100 different signals.

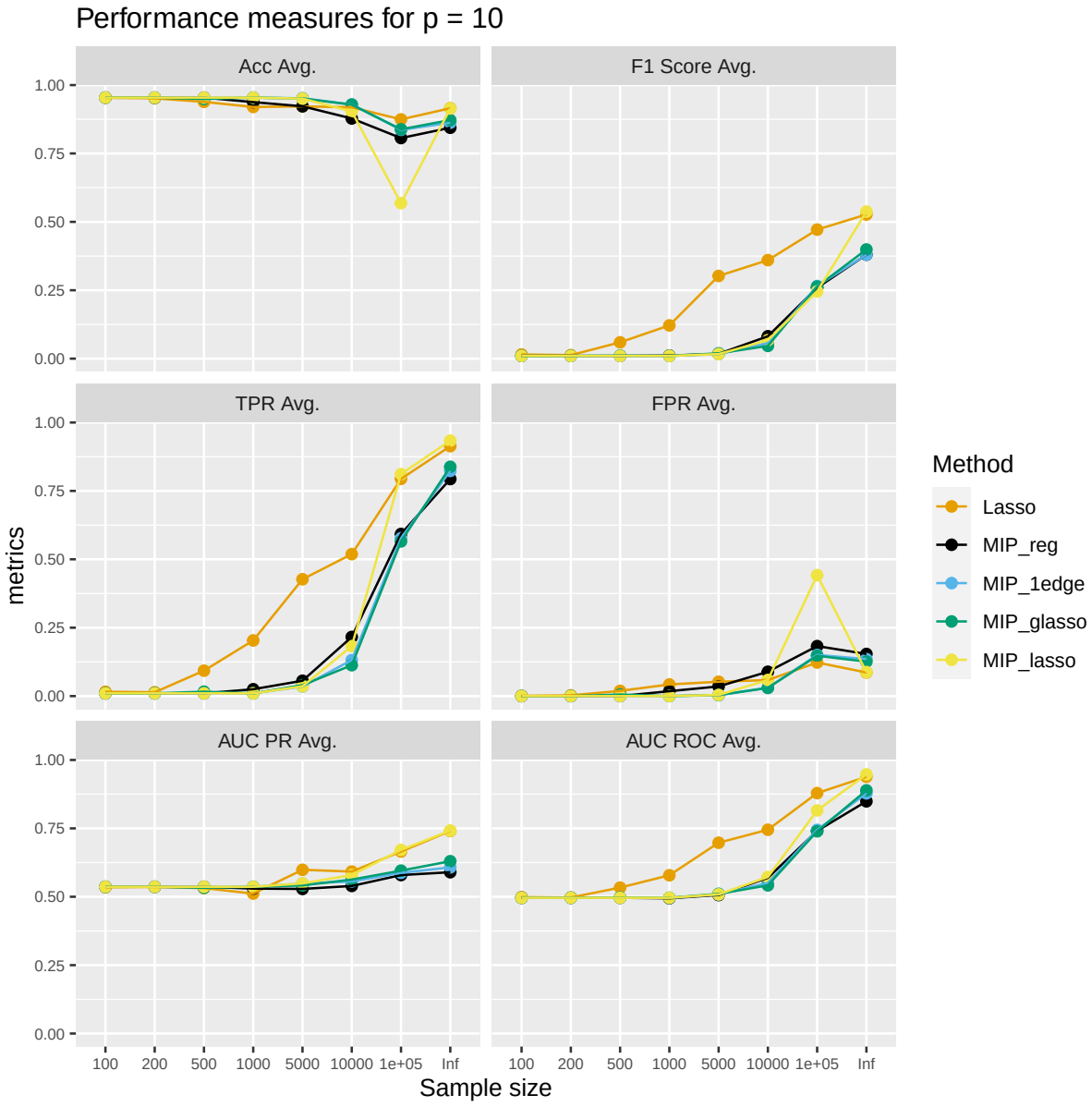


Figure A.46 Variation of performance metrics with graph size and at edge probability 5 % on unseen dataset drawn from a graph size $p = 10$ (test data size 20 %). The metrics are computed for graphs selected via eBIC and are averaged results over 100 different signals.

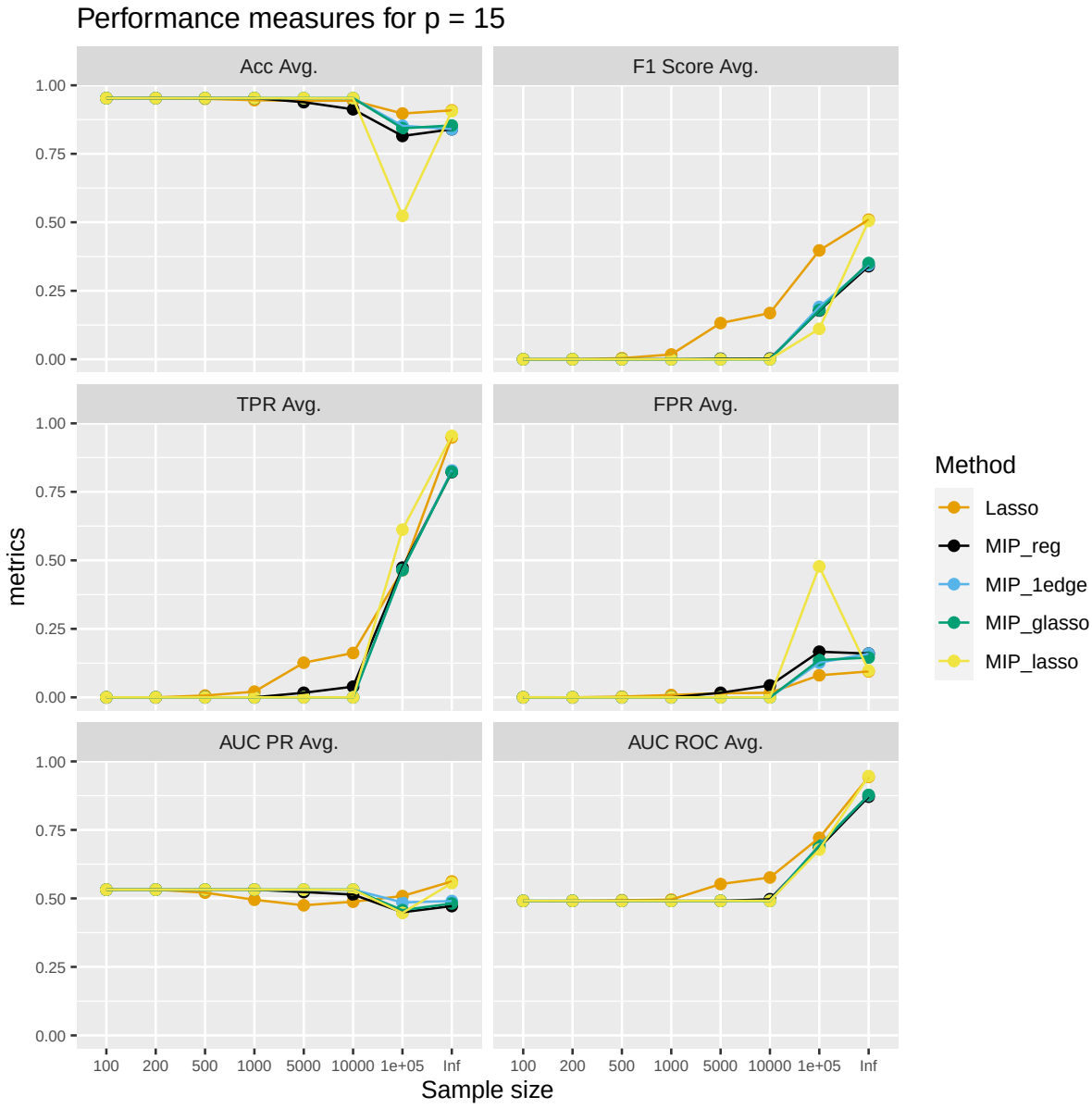


Figure A.47 Variation of performance metrics with graph size and at edge probability 5 % on unseen dataset drawn from a graph size $p = 15$ (test data size 20 %). The metrics are computed for graphs selected via eBIC and are averaged results over 100 different signals.

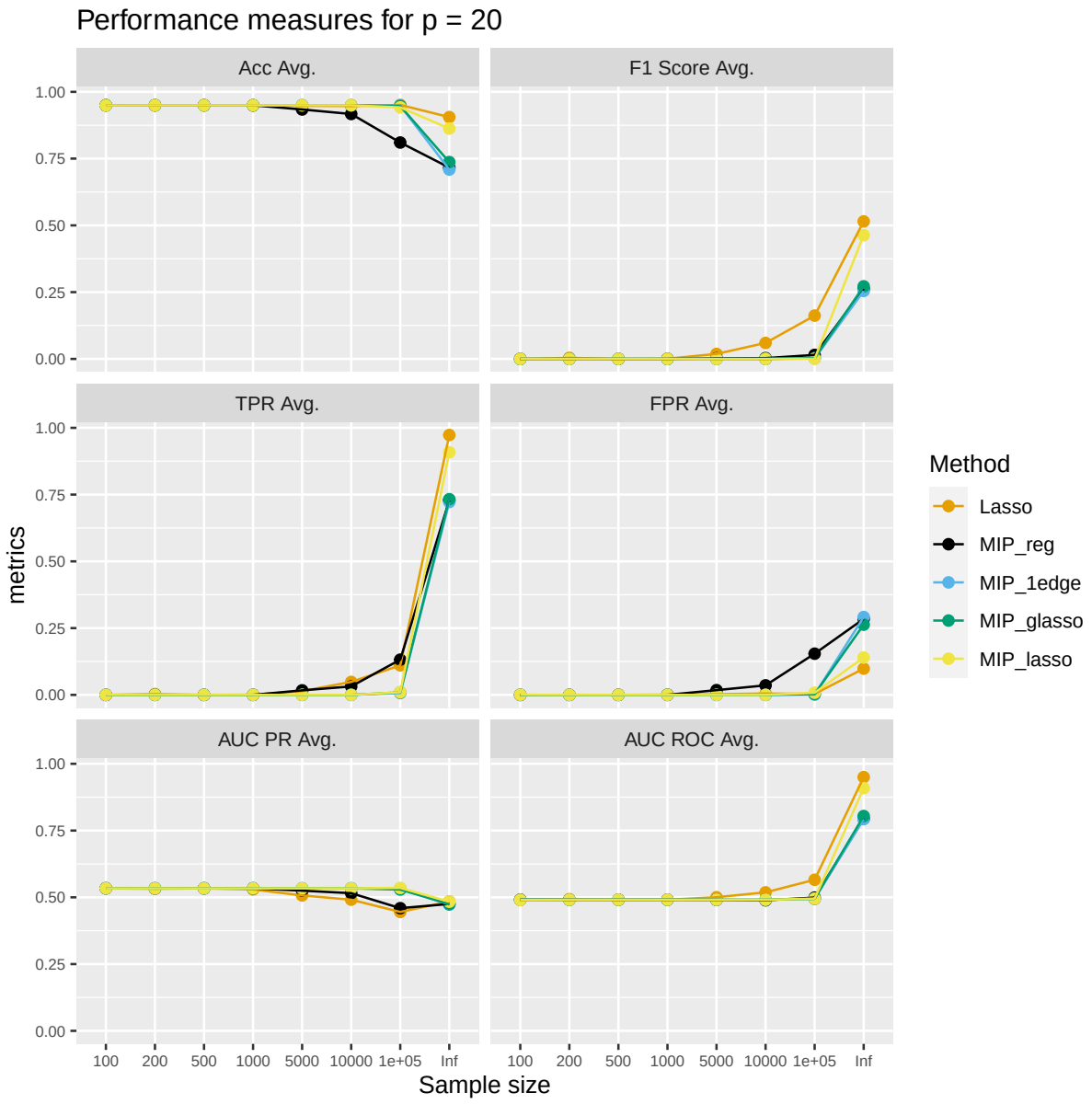


Figure A.48 Variation of performance metrics with graph size and at edge probability 5 % on unseen dataset drawn from a graph size $p = 20$ (test data size 20 %). The metrics are computed for graphs selected via eBIC and are averaged results over 100 different signals.

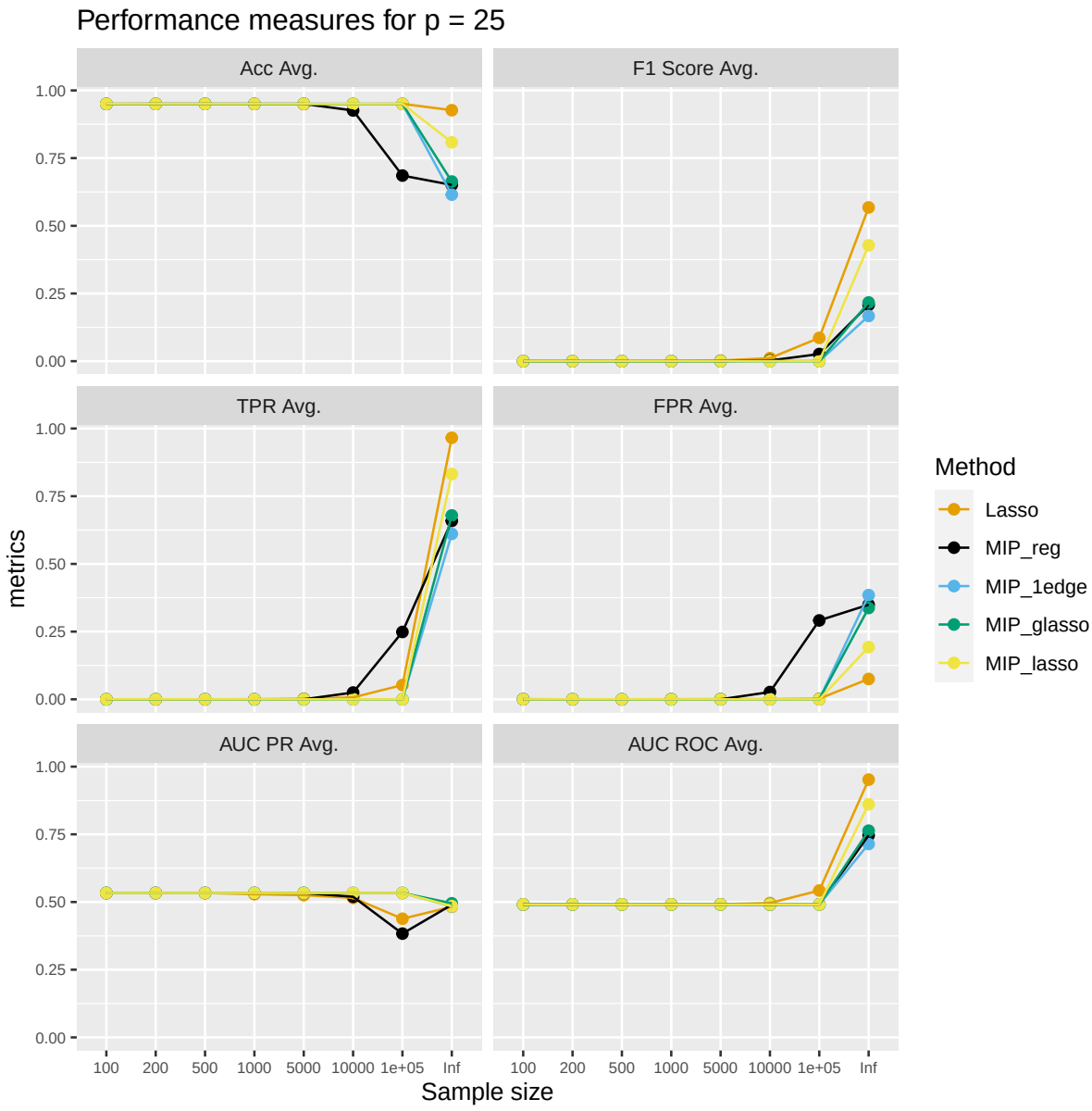


Figure A.49 Variation of performance metrics with graph size and at edge probability 5 % on unseen dataset drawn from a graph size $p = 25$ (test data size 20 %). The metrics are computed for graphs selected via eBIC and are averaged results over 100 different signals.

A.12 Prediction with eBIC when edge probability 25 %: Comparison along sample sizes

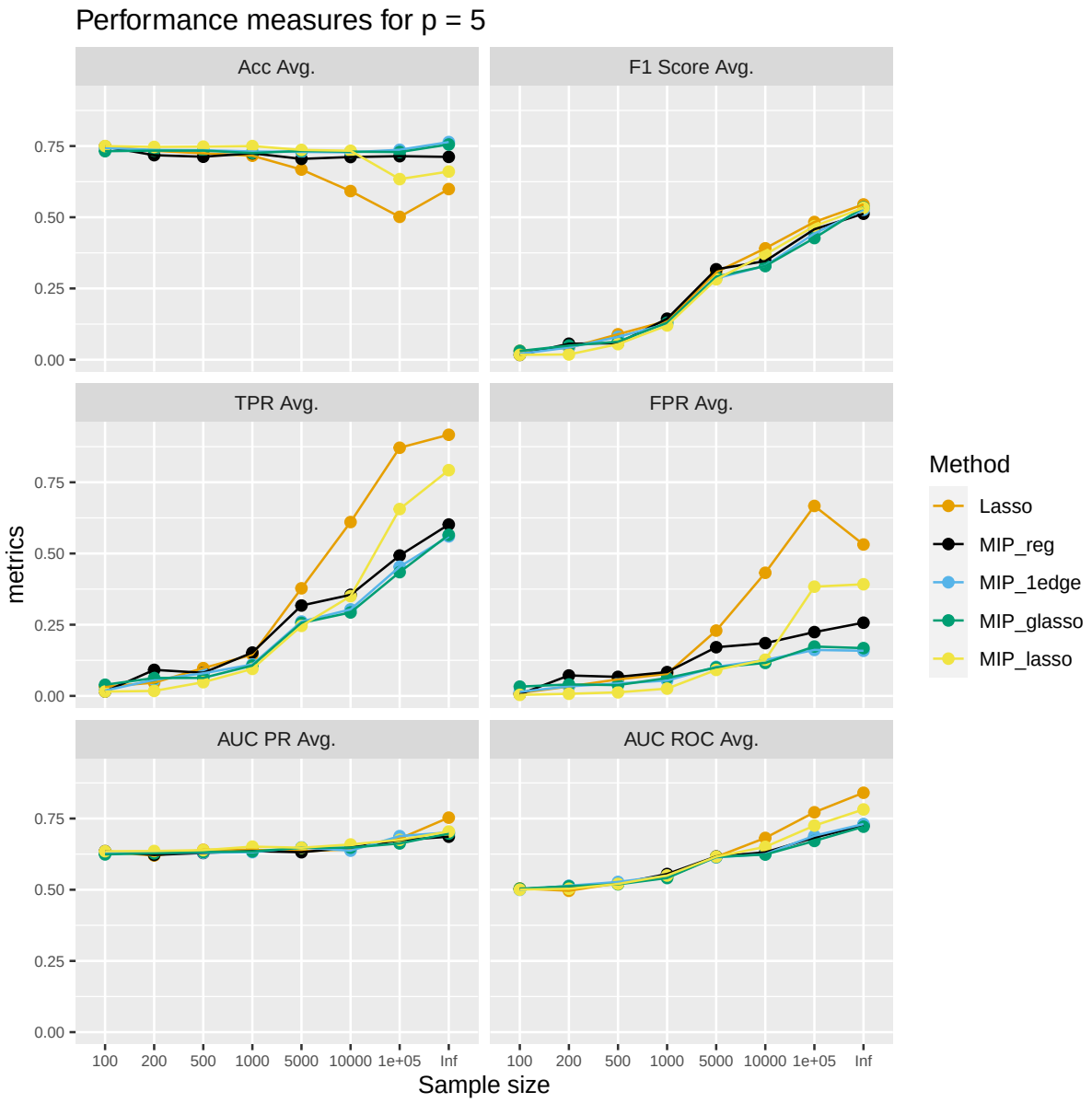


Figure A.50 Variation of performance metrics with graph size and at edge probability 25 % on unseen dataset drawn from a graph size $p = 5$ (test data size 20 %). The metrics are computed for graphs selected via eBIC and are averaged results over 100 different signals.

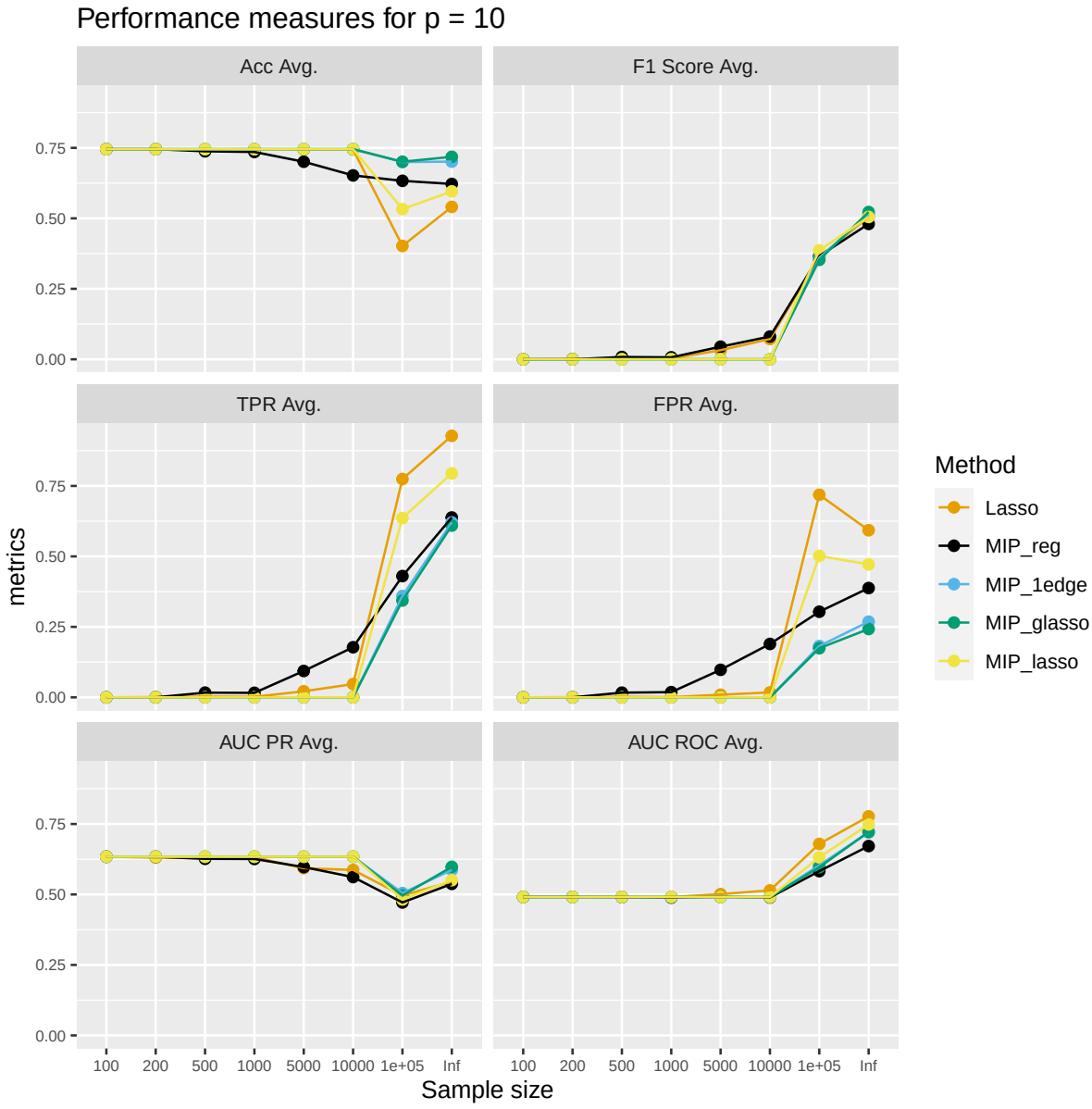


Figure A.51 Variation of performance metrics with graph size and at edge probability 25 % on unseen dataset drawn from a graph size $p = 10$ (test data size 20 %). The metrics are computed for graphs selected via eBIC and are averaged results over 100 different signals.

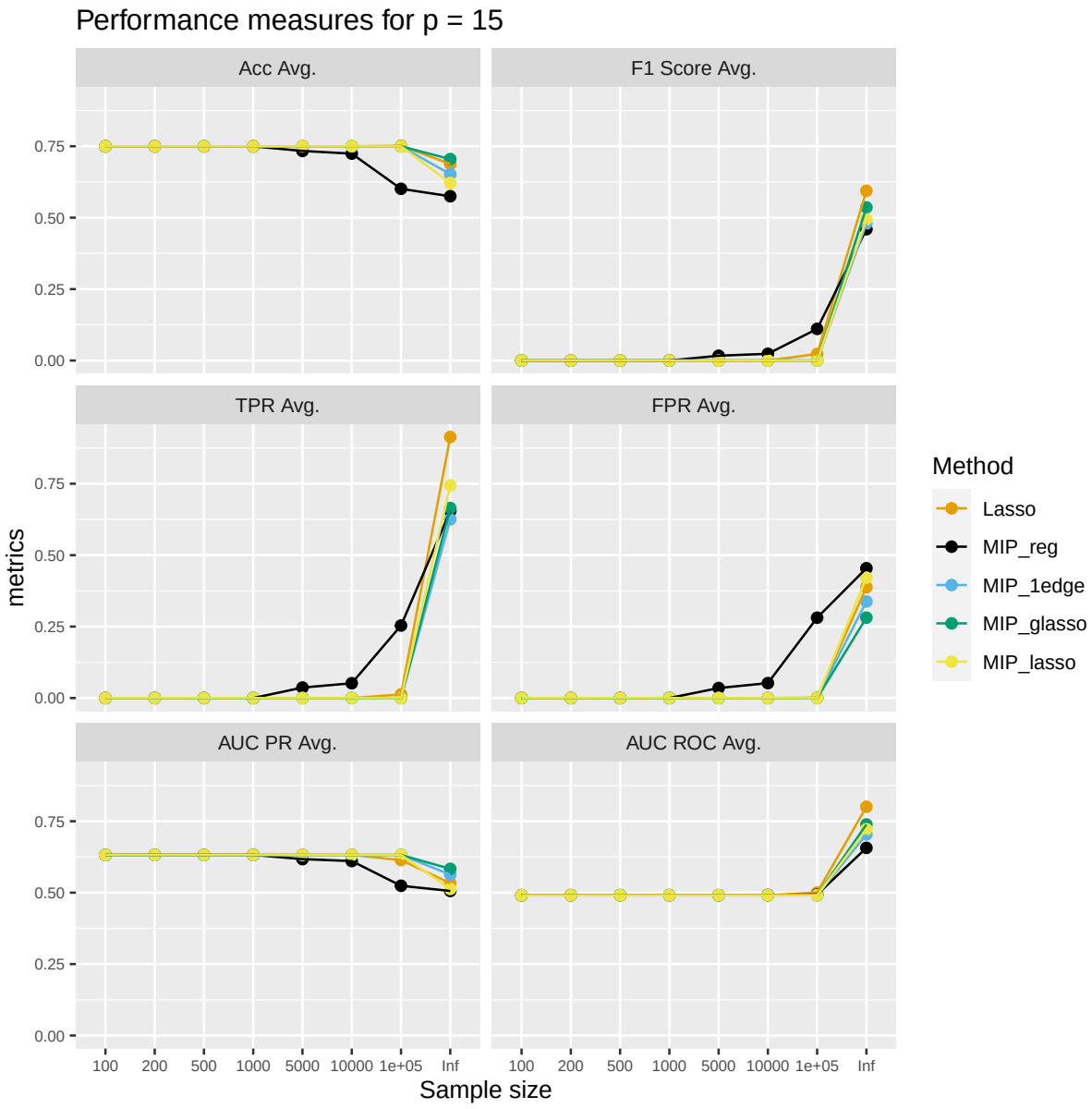


Figure A.52 Variation of performance metrics with graph size and at edge probability 25 % on unseen dataset drawn from a graph size $p = 15$ (test data size 20 %). The metrics are computed for graphs selected via eBIC and are averaged results over 100 different signals.

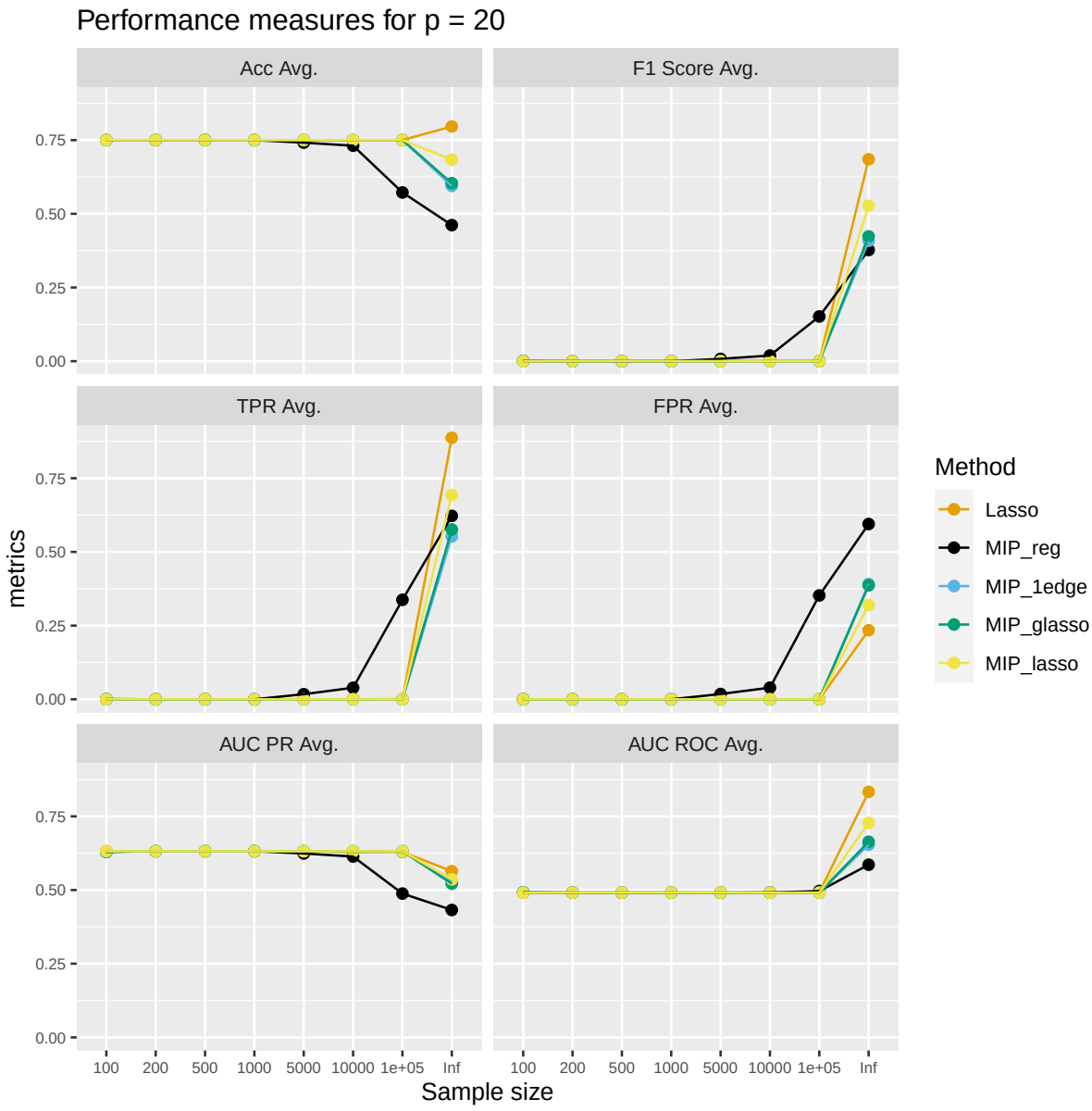


Figure A.53 Variation of performance metrics with graph size and at edge probability 25 % on unseen dataset drawn from a graph size $p = 20$ (test data size 20 %). The metrics are computed for graphs selected via eBIC and are averaged results over 100 different signals.

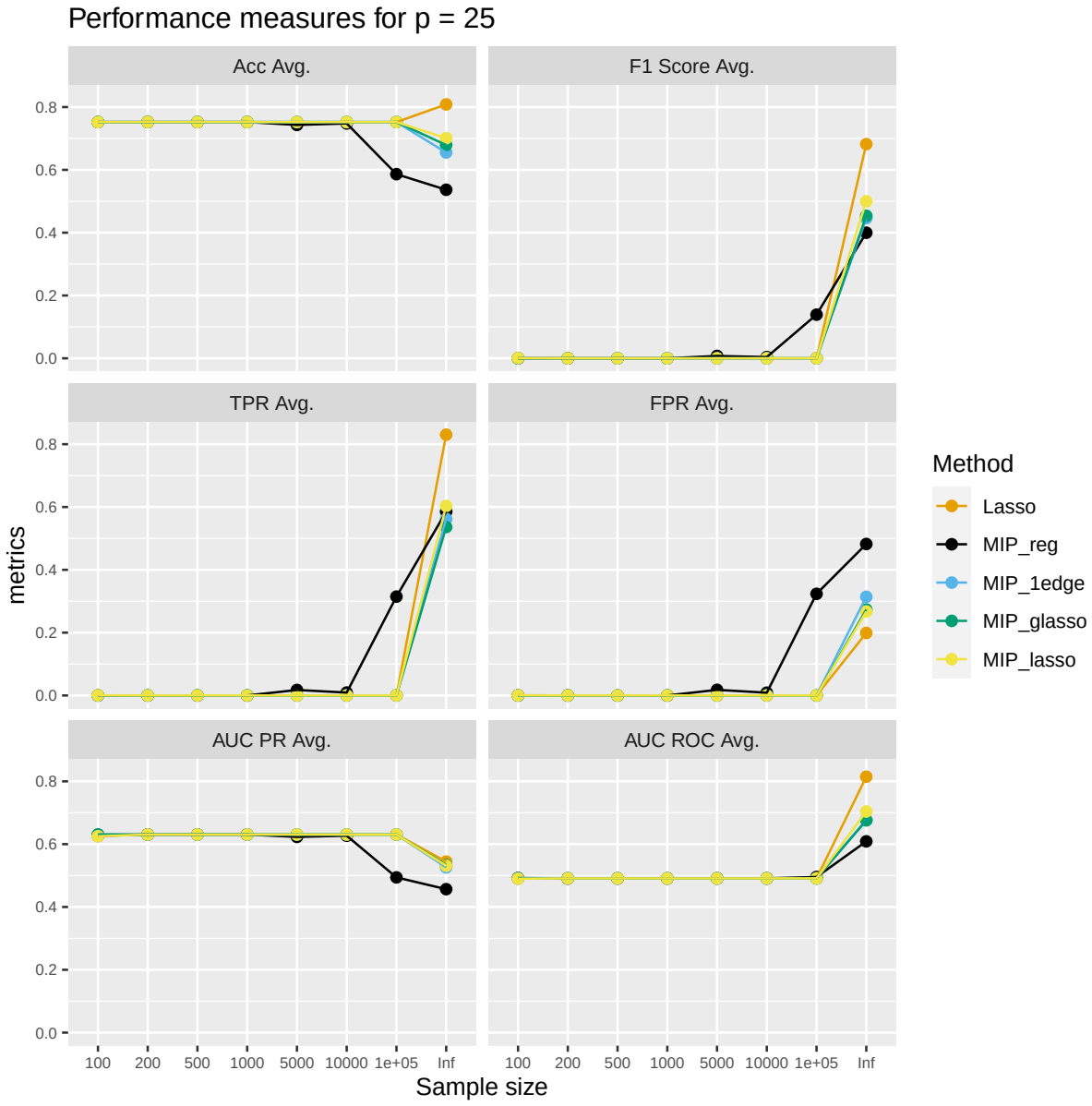


Figure A.54 Variation of performance metrics with graph size and at edge probability 25 % on unseen dataset drawn from a graph size $p = 25$ (test data size 20 %). The metrics are computed for graphs selected via eBIC and are averaged results over 100 different signals.

List of Figures

List of Tables

Bibliography

[AW13] T. Achterberg and R. Wunderling. "Mixed integer programming: Analyzing 12 years of progress". In: *Facets of combinatorial optimization: Festschrift for martin grötschel*. Springer, 2013, pp. 449–481.

[Ach+20] T. Achterberg et al. "Presolve reductions in mixed integer programming". In: *INFORMS Journal on Computing* 32.2 (2020), pp. 473–506.

[Aka73] H Akaike. "Information theory and an extension of the maximum likelihood principle". In: *Second International Symposium on Information Theory*. Akademia Kiadom. 1973, pp. 267–281.

[Amé+20] C. Améndola et al. "Structure learning for cyclic linear causal models". In: *Conference on Uncertainty in Artificial Intelligence*. PMLR. 2020, pp. 999–1008.

[BKM67] E. M. L. Beale, M. G. Kendall, and D. Mann. "The discarding of variables in multivariate analysis". In: *Biometrika* 54.3-4 (1967), pp. 357–366.

[bT20] T. berg and E. Towle. *Non-Convex Quadratic Optimization*. 2020.

[BKM16] D. Bertsimas, A. King, and R. Mazumder. "Best subset selection via a modern optimization lens". In: (2016).

[BKM15] D. Bertsimas, A. King, and R. Mazumder. "Supplement to "Best subset selection via a modern optimization lens."" In: (2015).

[BW05] D. Bertsimas and R. Weismantel. "Optimization over integers". In: *(No Title)* (2005).

[BK10] H. S. Bhat and N. Kumar. "On the derivation of the bayesian information criterion". In: *School of Natural Sciences, University of California* 99 (2010).

[Bix+04] R. E. Bixby et al. "Mixed-integer programming: A progress report". In: *The sharpest cut: the impact of Manfred Padberg and his work*. SIAM, 2004, pp. 309–325.

[Blu+87] A. Blumer et al. "Occam's razor". In: *Information processing letters* 24.6 (1987), pp. 377–380.

[CC08] J. Chen and Z. Chen. "Extended Bayesian information criteria for model selection with large model spaces". In: *Biometrika* 95.3 (2008), pp. 759–771.

[DDK22] P. Dettling, M. Drton, and M. Kolar. "On the Lasso for Graphical Continuous Lyapunov Models". In: *arXiv preprint arXiv:2208.13572* (2022).

[Det+22] P. Dettling et al. "Identifiability in Continuous Lyapunov Models". In: *arXiv preprint arXiv:2209.03835* (2022).

[DJ94] D. L. Donoho and I. M. Johnstone. "Ideal spatial adaptation by wavelet shrinkage". In: *biometrika* 81.3 (1994), pp. 425–455.

[DS98] N. R. Draper and H. Smith. *Applied regression analysis*. Vol. 326. John Wiley & Sons, 1998.

[Drt18] M. Drton. "Algebraic problems in structural equation modeling". In: *The 50th anniversary of Gröbner bases*. Vol. 77. Mathematical Society of Japan, 2018, pp. 35–87.

[DFW19] M. Drton, C. Fox, and Y. S. Wang. "Computation of maximum likelihood estimates in cyclic structural equation models". In: (2019).

[Drt+23] M. Drton et al. *Isoprenoid gene network in Arabidopsis thaliana*. 2023.

[Efr66] M Efroymson. "Stepwise regression–a backward and forward look". In: *Eastern Regional Meetings of the Institute of Mathematical Statistics*. 1966, pp. 27–29.

[Fit19] K. Fitch. "Learning directed graphical models from Gaussian data". In: *arXiv preprint arXiv:1906.08050* (2019).

[FL98] R. Fletcher and S. Leyffer. "Numerical experience with lower bounds for MIQP branch-and-bound". In: *SIAM Journal on Optimization* 8.2 (1998), pp. 604–616.

[FD10] R. Foygel and M. Drton. "Extended Bayesian information criteria for Gaussian graphical models". In: *Advances in neural information processing systems* 23 (2010).

[FHT08] J. Friedman, T. Hastie, and R. Tibshirani. "Sparse inverse covariance estimation with the graphical lasso". In: *Biostatistics* 9.3 (2008), pp. 432–441.

[Fri+07] J. Friedman et al. "Pathwise coordinate optimization". In: (2007).

[GM19] S. Gaïffas and G. Matulewicz. "Sparse inference of the drift of a high-dimensional Ornstein–Uhlenbeck process". In: *Journal of Multivariate Analysis* 169 (2019), pp. 1–20.

[Gir21] C. Giraud. *Introduction to high-dimensional statistics*. CRC Press, 2021.

[Guo+15] P. Guo et al. "Improved variable selection algorithm using a LASSO-type penalty, with an application to assessing hepatitis B infection relevant factors in community residents". In: *PloS one* 10.7 (2015), e0134151.

[Gur23a] Gurobi Optimization, LLC. *Gurobi Optimizer Reference Manual*. 2023.

[Gur23b] Gurobi Optimization, LLC. *Gurobi optimizer reference manual*. 2023.

[HTT17a] T. Hastie, R. Tibshirani, and R. J. Tibshirani. *Best Subset Selection and Related Tools*. `https://github.com/ryantibs/best-subset`. 2017.

[HTT17b] T. Hastie, R. Tibshirani, and R. J. Tibshirani. "Extended comparisons of best subset selection, forward stepwise selection, and the lasso". In: *arXiv preprint arXiv:1707.08692* (2017).

[HL67] R. R. Hocking and R. Leslie. "Selection of the best subset in regression analysis". In: *Technometrics* 9.4 (1967), pp. 531–540.

[HEH12] A. Hyttinen, F. Eberhardt, and P. O. Hoyer. "Learning linear cyclic causal models with latent variables". In: *The Journal of Machine Learning Research* 13.1 (2012), pp. 3387–3439.

[L92] A. L. *Stochastic Differential Equations: Theory and Applications*. Kreiger Publishing Company, 1992.

[Lei23] Leibniz-Rechenzentrum (LRZ). *Parallelization using R: LRZ Documentation*. 2023.

[MFH11] R. Mazumder, J. H. Friedman, and T. Hastie. "Sparsenet: Coordinate descent with nonconvex penalties". In: *Journal of the American Statistical Association* 106.495 (2011), pp. 1125–1138.

[Moz+22] Z. Mozafari et al. "Application of the LAD-LASSO as a dimensional reduction technique in the ANN-based QSAR study: Discovery of potent inhibitors using molecular docking simulation". In: *Chemometrics and Intelligent Laboratory Systems* 222 (2022), p. 104510.

[Nes13] Y. Nesterov. "Gradient methods for minimizing composite functions". In: *Mathematical programming* 140.1 (2013), pp. 125–161.

[Nes03] Y. Nesterov. *Introductory lectures on convex optimization: A basic course*. Vol. 87. Springer Science & Business Media, 2003.

[Pfe+22] P. Pfeiffer et al. "Weighted LASSO variable selection for the analysis of FTIR spectra applied to the prediction of engine oil degradation". In: *Chemometrics and Intelligent Laboratory Systems* 228 (2022), p. 104617.

[QG95] I. Quesada and I. E. Grossmann. "A global optimization algorithm for linear fractional and bilinear programs". In: *Journal of Global Optimization* 6 (1995), pp. 39–76.

[Rad23] R. Radhakrishnan. *Learning Graphical Lyapunov models using best-subset selection methods*. `https://github.com/bluearrow98/Master-Thesis`. 2023.

[Ric13] T. S. Richardson. "A discovery algorithm for directed cyclic graphs". In: *arXiv preprint arXiv:1302.3599* (2013).

[Sch78] G. Schwarz. "Estimating the dimension of a model". In: *The annals of statistics* (1978), pp. 461–464.

[Sha97] J. Shao. "An asymptotic theory for linear model selection". In: *Statistica sinica* (1997), pp. 221–242.

[Sto77] M. Stone. "An asymptotic equivalence of choice of model by cross-validation and Akaike's criterion". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 39.1 (1977), pp. 44–47.

[Tib96] R. Tibshirani. "Regression shrinkage and selection via the lasso". In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 58.1 (1996), pp. 267–288.

[Tow22] E. Towle. *Why does Gurobi report that my convex model is non-convex?* 2022.

[VH20a] G. Varando and N. R. Hansen. "Graphical continuous Lyapunov models". In: *Conference on Uncertainty in Artificial Intelligence*. PMLR. 2020, pp. 989–998.

[VH20b] G. Varando and N. R. Hansen. *Simulations and Experiments for Graphical Continuous Lyapunov Models*. `https://github.com/gherardovarando/gclm_experiments`. 2020.

[Wel82] W. J. Welch. "Algorithmic complexity: three NP-hard problems in computational statistics". In: *Journal of Statistical Computation and Simulation* 15.1 (1982), pp. 17–25.

[Wil+04] A. Wille et al. "Sparse graphical Gaussian modeling of the isoprenoid gene network in Arabidopsis thaliana". In: *Genome biology* 5.11 (2004), pp. 1–13.

[ZY06] P. Zhao and B. Yu. "On model selection consistency of Lasso". In: *The Journal of Machine Learning Research* 7 (2006), pp. 2541–2563.