# POSITIVE SEMI DEFINITE MATRIX COMPLETION VIA CONVEX OPTIMIZATION

MAX RESNICK*

**Abstract.** Given a subset of the entries of a low rank positive semi definite matrix, can we recover the matrix exactly? Problems of this form are pivotal in recommender systems, survey analysis, sparse sensing, and exotic financial derivative pricing. The general matrix completion problem is NP-Hard; however, approximating rank conditions with the atomic norm transforms the problem into a nearby optimization problem which is convex and can therefore be solved exactly. I outline an exact alternating direction method of multipliers algorithm (ADMM) that can be used to complete arbitrary positive semi definite matrices from only a subset of their entries, and implement it in Julia. I present the results of numerical experiments and compare the algorithm with fast iterative thresholding (FISTA). The two algorithms perform similarly in the absence of noise. But, the ADMM algorithm performs better once noise is introduced and when completing higher rank matrices.

**Key words.** Convex Optimization, Matrix Completion, Sparse Sensing, Survey Analysis, Nuclear Norm Minimization

**AMS subject classifications.** 68Q25, 68R10, 68U05

## 1. Introduction.

An accurate stochastic model of the motion of a basket of underlying assets is the crucial ingredient in the pricing of options, and other more exotic financial derivatives. Many of the parameter's in these stochastic models can be derived, under some assumptions on the class of process, from the states of options and swaps markets; however, in practice, many of these options and swaps markets are not liquid and so cannot be used for this purpose. To learn the variance covariance matrix of the underlying process, you would need to know the prices of options markets, not only for each asset with the numeraire, but also for each of the pairwise swaps. Most of these pairwise swaps are likely to be traded exclusively over the counter if they are traded at all, leaving us with only part of the covariance matrix filled in. The problem then becomes one of completing the covariance matrix from the sparse subset of it's entries that are known.

This is just one of the many real world problems that can be solved through matrix completion. Other applications include sparse sensing [9], survey evaluation[10], recommender systems [6], and much more.

Generic matrix completion is underspecified. That is, if we are given $k < n^2$ entries of a matrix $M \in \mathbb{R}^{n\times n}$, without any structural restrictions on $M$, there is an $n^2 - k$ dimensional subspace of matrices that are viable completions. Since the unstructured problem has too many degrees of freedom, a low rank condition is often introduced. The low rank restriction allows us to learn some information about the unknown entries of column $i$ from the known entries of column $j$ and vice versa. Therefore, we have some hope of completing the unknown entries of a low rank matrix. But the problem is still not tractable in general, even for extremely low rank matrices.

If the revealed entries are chosen by an adversary for example, it becomes extremely difficult. Even when the entries are sampled uniformly at random, certain

*Massachusetts Institute of Technology, Cambridge, MA (resnickm@mit.edu)

low rank matrices are difficult to complete. Consider:

$$M = e_1 e_n^* = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix} \tag{1.1}$$

Clearly this matrix cannot be recovered without access to nearly every entry, despite the fact that it is of rank 1 [4].

But these edge cases are the exception not the rule. It is possible to recover, with very high probability most $n \times n$ matrices of rank $r$ provided that

$$m \geq C n^{1.2} r \log(n) \tag{1.2}$$

Where $m$ is the number of sampled entries, $C$ is a constant, and $r$ is the rank[4]. Intuitively, problems with more structure require fewer samples to complete. For example, the bounds for positive semi definite matrices are slightly less restrictive[8].

**1.1. Matrix Completion as an Optimization Problem.** Ideally, low rank positive semi definite matrix completion is the solution to the minimization problem

$$\begin{aligned} \min \quad & \text{Rank}(X) \\ \text{s.t.} \quad & X_{ij} = M_{ij}, \qquad \forall (i,j) \in \Omega \\ & X \succeq 0 \end{aligned}$$

Where $X \in \mathbb{R}^{m \times n}$ and $\Omega$ is the index set of the $p$ revealed elements of $X$.

Define the masking operator, $\mathscr{P}$, as the projection onto the subspace of sparse matrices with non-zeros indexed by $\Omega$:

$$\mathscr{P}_\Omega(X)_{ij} = \begin{cases} X_{ij}, & \textbf{if } (i,j) \in \Omega, \\ 0, & \textbf{otherwise} \end{cases} \tag{1.3}$$

Since the Rank function is combinatorial in nature, the above problem is NP-hard in general [10]. Instead, the rank function can be approximated by the nuclear norm

$$\|X\|_* = \sum_{i=1}^{n} \sigma_i(X),$$

where $\sigma_i$ is the $i$th largest singular value. When the singular values of $X$ are near 1 this approximation is reasonable.

The resulting minimization problem,

$$\begin{aligned} \min \quad & \|X\|_*, \\ \text{s.t.} \quad & \mathscr{P}_\Omega(X) = \mathscr{P}_\Omega(M), \\ & X \succeq 0 \end{aligned}$$

is convex, and can therefore be solved exactly.

In section 2, I discuss an algorithm for solving this optimization problem based on [10]. The algorithm was implemented in Julia and results of numerical experiments as well as a comparison to a competing algorithm are presented in section 3. section 4 discusses practical impediments to applying this algorithm to some real world problems. section 5 concludes.

**2. Algorithm.** The algorithm is based on the Alternating direction method of multipliers (ADMM), an iterative process for solving convex optimization problems [3].

The following algorithm was described in [10]. Note that $\|X\|_* = \operatorname{tr}(X)$ when $X$ is positive semi definite. When the revealed entries of $M$ are exact, the minimization problem in the previous section can be solved exactly; however, when there is some noise in the observations, we must relax the requirement that $\mathcal{P}_\Omega(X) = \mathcal{P}_\Omega(M)$ so that we are solving

$$\begin{aligned} \min \quad & \operatorname{tr}(X), \\ \text{s.t.} \quad & \|\mathscr{P}_\Omega(X) - \mathscr{P}_\Omega(M)\|_2 \le \delta, \\ & X \succeq 0 \end{aligned}$$

It turns out that for an appropriately chosen $\mu$, we can express the problem as

$$\begin{aligned} \min_X \quad & \mu \operatorname{tr}(X) + \frac{1}{2}\|\mathscr{P}_\Omega(X) - \mathscr{P}_\Omega(M)\|_2^2, \\ \text{s.t.} \quad & X \succeq 0 \end{aligned}$$

Introducing a splitting variable $Y$, along with the constraint $X = Y$, gives us the equivalent problem:

$$\begin{aligned} \min_X \quad & \mu \operatorname{tr}(X) + \frac{1}{2}\|\mathscr{P}_\Omega(Y) - \mathscr{P}_\Omega(M)\|_2^2, \\ \text{s.t.} \quad & X = Y \\ & X \succeq 0 \end{aligned}$$

The augmented Lagrangian of the above problem is

$$\mathscr{L}(X, Y, \Pi) = \mu \operatorname{tr}(X) + \frac{1}{2}\|\mathscr{P}_\Omega(Y) - \mathscr{P}_\Omega(M)\|_2^2 + \langle \Pi, X - Y\rangle + \frac{\rho}{2}\|X - Y\|_2^2$$

the alternating method of multipliers for this problem comes from minimizing $\mathscr{L}$ with respect to $X$ and $Y$ repeatedly. That is:

$$X_{k+1} := \arg\min_{X \succeq 0} \mathscr{L}(X, Y_k, \Pi_k) \tag{2.1}$$

$$Y_{k+1} := \arg\min_{Y \succeq 0} \mathscr{L}(X_{k+1}, Y, \Pi_k) \tag{2.2}$$

$$\Pi_{k+1} := \Pi_k + \gamma\rho(X_{k+1} - Y_{k+1}) \tag{2.3}$$

Rearanging the terms in (2.1) we have

$$\min_{X \succeq 0} \mu \operatorname{tr}(X) + \frac{\rho}{2}\|X - G_k\|_2^2, \tag{2.4}$$

where $G_k = Y_k - (1/\rho)\Pi_k$. Let $G_k = U_k \operatorname{Diag}(w_k) U_k^T$ be the eigenvalue decomposition of $G_k$. We will impose sparsity by applying the shrinkage operator $S_v(\cdot)$,

$$S_v(w) = \overline{w}, \qquad \text{with } \overline{w}_i = \begin{cases} w_i - v, & \textbf{if } w_i - v > 0, \\ 0, & \textbf{otherwise} \end{cases} \tag{2.5}$$

So

$$X_{k+1} := U_k \operatorname{Diag}(S_{\mu/\rho}(w_k))U_k^T \tag{2.6}$$

Gives a solution to (2.1).

Turning now to (2.2), we can see that it can be split into two sub-problems, one relating to the unmasked region and one relating to the masked region.

$$\min \quad \frac{1}{2}\|(Y)_- \mathscr{P}_\Omega(M)\|_2^2 + \frac{p}{2}\|(Y)_- \mathscr{P}_\Omega(X_k + \frac{1}{\rho}\Pi_k)\|_2^2, \tag{2.7}$$

$$\min \quad \|\mathscr{P}_{\hat{\Omega}}(Y)_- \mathscr{P}_{\hat{\Omega}}(X_k + \frac{1}{\rho}\Pi_k)\|_2^2 \tag{2.8}$$

Where $\hat{\Omega}$ is the complement of $\Omega$ corresponding to the unknown region of the matrix. Solving these sub-problems yields:

$$(Y_{k+1})_\Omega = \frac{1}{\rho+1}\mathscr{P}_\Omega(M + \rho E_k), \tag{2.9}$$

$$(Y_{k+1})_{\hat{\Omega}} = \mathscr{P}_{\hat{\Omega}}(E_k), \tag{2.10}$$

Where $E_k = X_k + (1/\rho)\Pi_k$. Our analysis leads to Algorithm 2.1 which is guaranteed to converge since the problem is convex.

**Algorithm 2.1** Exact ADMM-based PSDMC

Input $\mathscr{P}_\Omega(M)$, $I_m \geq 0$, and tol $\geq 0$.
Set $\mu$, $\gamma$, and $\rho \geq 0$. Set $X_0$ and $Y_0$ as random matrices, and $\Pi_0$ as the zero matrix.

**while** *not converged* **do**
  Update $G_k := Y_k - (1/\rho)\Pi_k$
  Compute the Eigen Value Decomposition $G_k = U_k \operatorname{Diag}(w_k)U_k^T$
  Update $X_{k+1} := U_k \operatorname{Diag}(S_{\mu/\rho}(w_k))U_k^T$
  Update $E_{k+1} := X_{k+1} + (1/\rho)\Pi_k$
  Update $(Y_{k+1})_\Omega := \frac{1}{\rho+1}\mathscr{P}_\Omega(M + \rho E_{k+1})$
  Update $(Y_{k+1})_{\hat{\Omega}} := \mathscr{P}_{\hat{\Omega}}(E_{k+1})$
  Update $\Pi_{k+1} := \Pi_k + \gamma\rho(X_{k+1} - Y_{k+1})$
**end while**
**return** $X$

**2.1. Time Complexity.** The run-time of 2.1 Is dominated by the computation of the Eigen Decomposition in the iteration so the run-time is $O(n^3)$ for each iteration. In practice, a fast and numerically stable Eigen Decomposition is the main bottleneck for this algorithm and therefore the starting point for run-time optimization. In later iterations, the matrices tend to be positive semi definite and of low rank, imposing some structure that can be useful in computing the eigenvalues. Similar bottlenecks appear in the low rank matrix approximation literature and some work has been done to address them [5].

**3. Numerical Experiments.** In each experiment, a random positive semidefinite matrix $M$ is formed. To ensure that it has rank $r$, we construct this by drawing then entries of $M_L \in \mathbb{R}^{n\times r}$ i.i.d from $N(0,1)$. Then we construct $M = M_L M_L'$. We let $\Sigma$ be a symmetric but otherwise *i.i.d* $N(0,1)$, noise matrix. The matrix $\tilde{M} = M + \sigma\Sigma$ is constructed. Finally, the mask $\Omega$ is chosen according to a series of independent weighted coin flips (one for each entry in the upper diagonal of the matrix). The masked matrix $\mathscr{P}_\Omega(\tilde{M})$ is then passed to the algorithm as the matrix to be completed and the resulting completion is denoted $\hat{X}$. The solution is scored based on the absolute error

$$\text{Err} = \|\hat{X} - M\|_2 \tag{3.1}$$

and the rank

$$\text{Rank}(\hat{X}) \tag{3.2}$$

**3.1. Implementation.** I use the parameters outlined in [2]. Those are $I_m = 300$, $tol = 0.002$, $\gamma = 1.618$. $\mu$ is difficult to set because it can neither be too small, in which case the resulting matrix will not have low rank, nor too big, in which case the rank condition will be too aggressive. Optimal performance is achieved by adjusting $\mu$ downwards in each iteration, allowing the model more flexibility to take the final steps towards convergence in later iterations. For example, I set:

$$\mu_{k+1} := \frac{\mu_k}{1.01},$$

according to their suggestions.

Figure 1 Shows this stopping conditions converging to 0 as the solver iterates.

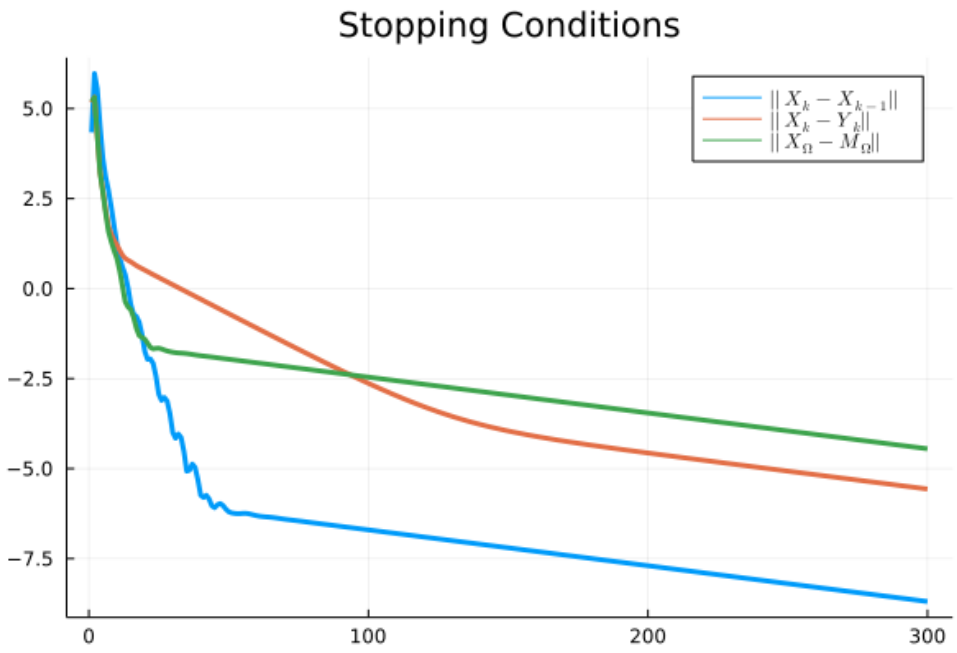


FIG. 1. *Stopping Conditions for* $r = 5$, $n = 100$, $\sigma = 0$. *Plotted on a log scale.*

Speaking of convergence, we still have to define what the *converge* condition means in 2.1. There are three conditions that must be satisfied for the algorithm to have converged. First, $X$ must not be moving very much. Second, $X$ must be very close to $Y$. Third, $X$ must be nearly in agreement with $\tilde{M}$ on the revealed entries. Stating these conditions more formally, we have:

$$\|X_{k+1} - X_k\| \leq \mathbf{tol} \tag{3.3}$$

$$\|X_{k+1} - Y_{k+1}\| \leq \mathbf{tol} \tag{3.4}$$

$$\|\mathscr{P}_\Omega(X) - \mathscr{P}_\Omega(\tilde{M})\| \leq \mathbf{tol} \tag{3.5}$$

**3.2. Results.** Table 1 Shows that when $\mu$ is set too small, the low rank condition fails to hold and the replication fails. When $\mu$ grows, the rank condition becomes agressive and absolute error increases. Recall that $\mu$ must be set according to the level of noise. In Table 3, we set $\mu = 1$ to better accommodate the noise.

TABLE 1
*Numerical results for different Values of* $\mu$ $(n = 100, r = 5, \sigma = 0)$

| SR | $\mu = 10^{-3}$ | | $\mu = 10^{-2}$ | | $\mu = 10^{-1}$ | | $\mu = 1$ | |
|---|---|---|---|---|---|---|---|---|
| | Rank | Err | Rank | Err | Rank | Err | Rank | Err |
| 0.25 | 44 | 76.5 | 5 | 1.65e-03 | 5 | 1.62e-02 | 5 | 1.71e-01 |
| 0.35 | 44 | 60.7 | 5 | 1.70e-03 | 5 | 1.77e-02 | 5 | 1.67e-01 |
| 0.45 | 44 | 66.3 | 5 | 1.73e-03 | 5 | 1.62e-02 | 5 | 1.84e-01 |
| 0.55 | 44 | 74.2 | 5 | 1.71e-03 | 5 | 1.72e-02 | 5 | 1.71e-01 |
| 0.65 | 44 | 76.8 | 5 | 1.58e-03 | 5 | 1.81e-02 | 5 | 1.63e-01 |
| 0.75 | 44 | 73.8 | 5 | 1.67e-03 | 5 | 1.63e-02 | 5 | 1.81e-01 |
| 0.85 | 44 | 78.3 | 5 | 1.74e-03 | 5 | 1.66e-02 | 5 | 1.68e-01 |

In Table 2, we compare the performance of the ADMM algorithm to Fast Iterative Shrinkage-Thresholding (FISTA) [1]. FISTA fails to complete some matrices of moderate rank that ADMM performs well on. At higher ranks, both algorithm's fail due to the natural bounds of the problem.

TABLE 2
*Sensitivity to truth matrix rank* ($n = 100, SR = .45, \sigma = 0$)

| Target Rank | ADMM | | FISTA | |
|---|---|---|---|---|
| | Rank | Err | Rank | Err |
| 3 | 3 | 1.27e-03 | 3 | 5.02e-02 |
| 5 | 5 | 1.67e-03 | 5 | 6.13e-02 |
| 7 | 7 | 2.23e-03 | 9 | 3.55e-01 |
| 10 | 10 | 2.72e-03 | 14 | 1.59e-01 |
| 15 | 19 | 1.17 | 49 | 14.7 |
| 20 | 55 | 39.5 | 49 | 52.1 |

Table 3 Compares ADMM and FISTA with varying levels of noise. Notice that FISTA begins to fail once any noise is introduced. This is because FISTA enforces $\mathscr{P}_\Omega(X) = \mathscr{P}_\Omega(\tilde{M})$ exactly which is too rigid a constraint when noise is introduced [4]. Adjusting $\mu$ can yield even better results than those presented for the ADMM algorithm.

TABLE 3
*Sensitivity to noise*

| $\sigma$ | ADMM | | FISTA | |
|---|---|---|---|---|
| | Rank | Err | Rank | Err |
| 0 | 5 | 1.65e-01 | 5 | 6.77e-02 |
| $10^{-3}$ | 5 | 1.79e-01 | 16 | 9.27e-02 |
| $10^{-2}$ | 29 | 12.3 | 50 | 1.03 |
| $10^{-1}$ | 46 | 12.3 | 54 | 11.2 |

Overall the ADMM algorithm dominates FISTA on all elements except speed, which may be a product of less optimized code.

**4. Discussion.** In many real world applications of matrix completion, the observed entries are corrupted by noise. This is particularly common in survey analysis, sparse sensing, and derivatives pricing. The ADDM algorithm presented here performs well in the presence of noise; however, its performance depends on parameter tuning. That is, $\mu$ must be set correctly. In some applications, parameter tuning is easy. For instance, in sparse sensing, test cases can be manufactured relatively easily, and the value of $\mu$ can be adjusted until the desired performance is met. But in other cases, such as implied covariance sensing, test cases are much more difficult to manufacture and therefore the optimal $\mu$ may be difficult to find in practice. Even backtesting may not be sufficient because the covariance matrix is a moment of the underlying joint distribution so unless market conditions are similar in many time steps, the 'true' noise level may be hard to recover and therefore impossible to adjust to.

Another problem arises in implied covariance applications. What happens when the observed entries are sampled not uniformly at random but endogenously. For example if we only observe entries from pairs of asset's who's swap markets are liquid does that render the algorithm we have just discussed irrelevant? Some algorithm's have been proposed for sensing when the entries of the matrix are deterministic [2]. They often require that the observed entries form principle submatrices, which can be restrictive when implied covariance is taken from a sparse exchange graph, who's sparsity structure need not, and in practice does not have such a structure.

Endogenous selection of revealed entries poses another problem. When revealed entries are sampled endogenously, how do we know that revealed entries do not differ qualitatively from unrevealed entries? In practice, the presence of a liquid swap market does effect the behavior of the two assets prices. For a simple example of how derivatives markets can change the underlying price movement of an asset. Consider a market in which significant volume is traded as forwards. The Canonical example is energy markets. The existence of long dated forward contracts means that demand is essentially fixed on any given day, regardless of price. This causes demand to be highly inelastic, which means the market price (the intersection of supply and demand) is highly sensitive to supply shocks and therefore has higher variance. Similar stories can be told about options and swap markets, and the direction in which these influence variance are not uniform in direction or magnitude [7].

In summary, the ADMM algorithm works well for some applications, but other applications have structural challenges which may require adjustment or different approaches.

**5. Conclusion.** While the problem of matrix completion is hard in general, the ADMM algorithm offers a solution that can be used for many real world applications. It is particularly suited to applications in which the observed entries have been corrupted by noise. It performs better than a competing algorithm, FISTA, on all observed metrics and can be tuned to arbitrary noise levels.

## REFERENCES


[1] A. Beck and M. Teboulle, *A fast iterative shrinkage-thresholding algorithm for linear inverse problems*, SIAM Journal on Imaging Sciences, 2 (2009), pp. 183–202, https://doi.org/10.1137/080716542, https://doi.org/10.1137/080716542, https://arxiv.org/abs/https://doi.org/10.1137/080716542.

[2] W. E. Bishop and B. M. Yu, *Deterministic symmetric positive semidefinite matrix completion*, in Advances in Neural Information Processing Systems, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, eds., vol. 27, Curran Associates, Inc., 2014, https://proceedings.neurips.cc/paper/2014/file/56468d5607a5aaf1604ff5e15593b003-Paper.pdf.

[3] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, *Distributed optimization and statistical learning via the alternating direction method of multipliers*, Foundations and Trends® in Machine Learning, 3 (2011), pp. 1–122, https://doi.org/10.1561/2200000016, http://dx.doi.org/10.1561/2200000016.

[4] E. J. Candès and B. Recht, *Exact matrix completion via convex optimization*, CoRR, abs/0805.4471 (2008), http://arxiv.org/abs/0805.4471, https://arxiv.org/abs/0805.4471.

[5] G. Chen, *Matrix norm and low-rank approximation*, https://www.sjsu.edu/faculty/guangliang.chen/Math253S20/lec7matrixnorm.pdf.

[6] Y. Chen and Y. Chi, *Spectral compressed sensing via structured matrix completion*, in Proceedings of the 30th International Conference on Machine Learning, S. Dasgupta and D. McAllester, eds., vol. 28 of Proceedings of Machine Learning Research, Atlanta, Georgia, USA, 17–19 Jun 2013, PMLR, pp. 414–422, https://proceedings.mlr.press/v28/chen13g.html.

[7] P. Cramton, *Electricity market design*, Oxford Review of Economic Policy, 33 (2017), pp. 589–612, https://doi.org/10.1093/oxrep/grx041, https://doi.org/10.1093/oxrep/grx041, https://arxiv.org/abs/https://academic.oup.com/oxrep/article-pdf/33/4/589/21515797/grx041.pdf.

[8] J. Narasimhan, *Low-rank matrix completion for positive semidefinite matrices*, (2016), https://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-79.pdf.

[9] A. Ramlatchan, M. Yang, Q. Liu, M. Li, J. Wang, and Y. Li, *A survey of matrix completion methods for recommendation systems*, Big Data Mining and Analytics, 1 (2018), pp. 308–323, https://doi.org/10.26599/BDMA.2018.9020008.

[10] Q.-W. Wang, F. Xu, and P. Pan, *A new algorithm for positive semidefinite matrix completion*, Journal of Applied Mathematics, 2016 (2016), p. 1659019, https://doi.org/10.1155/2016/1659019, https://doi.org/10.1155/2016/1659019.