Journal of the Korean Data & Information Science Society
2016, **27**(2), 381–394

http://dx.doi.org/10.7465/jkdi.2016.27.2.381
한국데이터정보과학회지

유사가능도 기반의 네트워크 추정 모형에 대한 GPU 병렬화 BCDR 알고리즘[†]

김병수[1] · 유동현[2]

[1]영남대학교 통계학과 · [2]계명대학교 통계학과

접수 2016년 2월 25일, 수정 2016년 3월 22일, 게재확정 2016년 3월 23일

요 약

그래피컬 모형은 변수들 사이의 조건부 종속성을 노드와 연결선을 통하여 그래프로 나타낸다. 변수들 사이의 복잡한 연관성을 표현하기 위하여 그래피컬 모형은 물리학, 경제학, 생물학을 포함하여 다양한 분야에 적용되고 있다. 조건부 종속성은 공분산 행렬의 역행렬의 비대각 성분이 0인 것과 대응하는 두 변수의 조건부 독립이 동치임에 기반하여 공분산 행렬의 역행렬로부터 추정될 수 있다. 본 논문은 공분산 행렬의 역행렬을 희박하게 추정하는 유사가능도 기반의 CONCORD (convex correlation selection method) 방법에 대하여 기존의 BCD (block coordinate descent) 알고리즘을 랜덤 치환을 활용한 갱신 규칙과 그래픽 처리 장치 (graphics processing unit)의 병렬 연산을 활용하여 고차원 자료에 대하여 보다 효율적인 BCDR (block coordinate descent with random permutation) 알고리즘을 제안하였다. 두 종류의 네트워크 구조를 고려한 모의실험에서 제안하는 알고리즘의 효율성을 수렴까지의 계산 시간을 비교하여 확인하였다.

주요용어: 그래피컬 모형, 그래픽 처리 장치, 랜덤 치환, 유사가능도, BCD 알고리즘.

1. 서론

네트워크 구조에 관한 추론 문제는 사회학, 경제학, 물리학 등 다양한 분야에서 관심을 가져온 주제로 최근 실험 장비 및 기술의 발달에 따라 생물학 분야에서 많이 연구되고 있는 주제이다. 통계학 분야에서는 관측된 자료로부터 변수들 사이의 조건부 독립 및 종속의 관계를 네트워크의 구조로 추론하는 그래피컬 모형 (graphical model)에 대하여 많은 연구가 이루어지고 있다. 특히, 그래피컬 모형에서 관측된 자료가 정규 분포를 따르는 경우에 가우시안 그래피컬 모형 (Gaussian graphical model)으로 표현하며 조건부 의존성 (conditional dependence)의 관계를 연결선 (edge)을 통하여 나타낸다 (Lauritzen, 1996). 연결선을 추정하기 위하여 가우시안 그래피컬 모형에서는 두 변수의 조건부 의존성과 공분산 행렬의 역행렬의 비대각 원소 (off-diagonal element)가 0이 아님이 동치인 성질을 활용하여 공분산 행렬의 역행렬의 추정 문제를 집중적으로 다룬다. 이 경우, 공분산 행렬의 역행렬의 개별 원소의 추정뿐만 아니라 조건부 독립성과 의존성 (즉, 0인 성분과 0이 아닌 성분)을 구분하는 문제 또한 매우 중요하다. 가우시안 그래피컬 모형에서는 공분산 행렬의 역행렬을 강조하기 위하여 집중 행렬 (concentration

[†] 이 논문은 2015년도 정부 (미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (NRF-2015R1C1A1A02036312).
[1] (38541) 경상북도 경산시 대학로 280, 영남대학교 통계학과, 조교수.
[2] 교신저자: (42601) 대구광역시 달서구 달구벌대로 1095, 계명대학교 통계학과, 조교수.
E-mail: dyu3@kmu.ac.kr

matrix) 또는 정밀 행렬 (precision matrix)의 용어를 통하여 나타낸다. 본 논문에서는 정밀 행렬 표현을 사용한다.

초기 연구에서는 표본 공분산 행렬 (sample covariance matrix)로부터 정밀 행렬을 계산하여 각각의 원소에 대한 가설 검정을 통하여 네트워크를 추정하는 방법이 개발되었으나 (Drton과 Perlman, 2004), 현재 주로 관측되고 연구가 이루어지는 고차원 자료 (변수의 수가 표본의 수보다 많은 자료)의 경우에는 표본 공분산 행렬의 역행렬이 정의되지 않으므로 이를 해결하기 위하여 벌점화 방법 (penalized method)에 기반한 여러 방법들이 제안되었다 (Meinshausen과 Bühlmann, 2006; Yuan과 Lin, 2007; Friedman 등, 2008; Peng 등, 2009; Cai 등, 2011; Witten 등, 2011; Khare 등, 2015).

기존의 벌점화 방법에 기반한 네트워크 추정 모형들은 크게 가능도 함수 (likelihood function) 기반의 추정 모형, 회귀 모형 (regression model) 기반의 추정 모형, ℓ_1 최소화 (ℓ_1 minimization) 방법의 3가지로 나눌 수 있다. 가능도 함수 기반의 추정 모형은 다변량 정규분포의 가능도 함수와 정밀 행렬의 원소에 대한 ℓ_1 노름 (norm) 벌점을 고려하였으며 (Yuan과 Lin, 2007; Friedman 등, 2008), 회귀 모형 기반의 추정 모형은 두 변수에 대응하는 정밀 행렬의 성분이 0인 것과 두 변수 중 하나를 반응변수로 나머지 변수를 설명 변수로 하는 회귀 모형에서 대응하는 회귀 계수가 0인 것이 동치인 성질을 이용한다. 일반적으로 0이 아닌 회귀 계수의 식별을 위하여 LASSO 회귀 모형 (Tibshirani, 1996)과 같이 ℓ_1 노름 벌점을 고려한다 (Meinshausen과 Bühlmann, 2006; Peng 등, 2009). LASSO 벌점에 대한 개괄적인 연구 흐름과 정밀 행렬의 추정에 대한 응용은 Kwon 등 (2013)을 참조하기 바란다. 마지막으로 제약조건이 있는 ℓ_1 최소화 방법 (constrained ℓ_1 minimization)은 Dantzig 선택기 (Dantzig selector; Candes와 Tao, 2007) 형식의 최적화 문제를 고려하여 다른 모형과 비교할 때, 상대적으로 약한 분포가정을 지니며 적률에 대한 조건하에서 최소최대 최적 수렴성 (minimax optimal convergence rate)을 지님이 증명되었다 (Cai 등, 2011). 제약조건이 있는 ℓ_1 최소화 방법이 좋은 이론적인 성질을 가짐이 알려져 있으나 실제 변수의 수가 매우 큰 경우에는 계산 상의 효율성 문제로 회귀 모형 기반의 방법이 널리 적용되고 있다. 특히, 부분상관계수 (partial correlation)을 통하여 네트워크를 추정하는 SPACE (sparse partial correlation estimation) 방법이 여러 응용 분야에 적용되었다 (Dong 등, 2010; Tang 등, 2013). 하지만, 최근에 SPACE 방법이 추정량의 계산에 있어서 수렴성에 대한 문제가 발생할 수 있음이 보고되었고 이를 개선한 유사가능도 (pseudo-likelihood) 기반의 CONCORD (convex correlation selection method) 방법이 제안되었다 (Khare 등, 2015).

본 논문에서는 가장 최근에 발표되었으며 여러 수치적인 실험에서 다른 방법들에 비하여 우수한 연결선 추정의 성질을 보이는 CONCORD 방법에 대하여 그래픽 처리 장치 (graphics processing unit, GPU)의 병렬화 계산과 랜덤 치환 기반의 갱신 규칙을 이용한 효율적인 BCDR (Block Coordinate Descent with Random permutation) 알고리즘을 제안하고자 한다. 본 논문에서는 두 종류의 네트워크를 고려한 모의실험에서 CONCORD의 기존 알고리즘과 GPU 병렬화 BCDR 알고리즘의 효율성을 계산 시간을 통하여 비교하였다.

본 논문의 구성은 다음과 같다. 2절에서는 앞서 설명한 네트워크 추정 모형들 중에서 대표적인 방법들과 알고리즘을 설명하였다. 본 논문에서 제안하고자 하는 유사가능도 기반의 모형에 대한 GPU 병렬화 BCDR 알고리즘을 3절에서 설명하였으며 알고리즘의 효율성을 보이기 위하여 4절에서 두 종류의 네트워크 구조 하에서 수렴까지의 계산 시간을 비교하였다. 마지막으로 5절에서 본 연구의 결과를 요약하고 결론을 정리하였다.

2. 기존의 네트워크 추정 모형 및 알고리즘

본 절에서는 앞서 간략히 소개하였던 벌점화 방법 기반의 네트워크 추정 모형 및 알고리즘에 대하여 설명하고자 한다. 모형을 정의하기 위하여, $\mathbf{X} = (\mathbf{X}_1, \mathbf{X_2}, \ldots, \mathbf{X}_p) = (\mathbf{X}^1, \mathbf{X}^2, \ldots, \mathbf{X}^n)^T$는 $n \times p$ 차원

의 관측된 자료 행렬을 나타내며, $j = 1, 2, \ldots, p$에 대하여 $\mathbf{X}_j = (X_j^1, X_j^2, \ldots, X_j^n)^T$, $k = 1, 2, \ldots, n$에 대하여 $\mathbf{X}^k = (X_1^k, X_2^k, \ldots, X_p^n)^T$이다. 일반적으로 관측된 n개의 벡터 $\mathbf{X}^k$ $(k = 1, 2, \ldots, n)$는 서로 독립이며 평균이 0이고 공분산 행렬 $\Sigma = (\sigma_{ij})_{1 \le i,j \le p}$인 동일한 분포를 따른다고 가정한다. 또한, 그래피컬 모형에서 관심을 두는 정밀 행렬을 표현하기 위하여 $\Omega = \Sigma^{-1} = (\omega_{ij})_{1 \le i,j \le p}$의 표기를 사용한다. 가우시안 그래피컬 모형에서는 주어진 자료 $\mathbf{X}$로 부터 Ω의 추정을 통하여 조건부 종속과 독립의 관계를 통하여 그래프 (graph) $G = (V, E)$를 통하여 변수들 사이의 관계를 요약한다. 여기서, $V = \{1, 2, \ldots, p\}$는 노드 (node) 집합을 $E = \{(i, j) \mid \omega_{ij} \neq 0\}$로 정의되며 연결선 (edge) 집합을 나타낸다. 가우시안 그래피컬 모형에서는 Ω의 대칭성으로 인하여 연결선이 방향성을 갖지 않는 무방향 그래프 (undirected graph)를 가정한다 (즉, $(i, j) \in E \iff (j, i) \in E$).

가능도 함수 기반의 접근 방법은 관측된 자료가 다변량 정규분포 (multivariate normal distribution)을 따른다고 가정하고 이에 기반하여

$$L(\Omega; \mathbf{X}, \lambda) = \frac{n}{2} \log \det \Omega - \frac{n}{2} \mathrm{tr}(\mathbf{S}\Omega) - \lambda \sum_{i \neq j} |\omega_{ij}| \tag{2.1}$$

의 벌점화 로그가능도 함수 (penalized log-likelihood function)을 고려한다. 여기서, $\det A$는 행렬 A의 행렬식 (determinant), $\mathrm{tr}(A)$는 행렬 A의 대각합 (trace)를 나타내며, $\mathbf{S} = \mathbf{X}^T\mathbf{X}/n$는 관측된 자료 $\mathbf{X}$의 표본 공분산 행렬, λ는 비음의 조율 모수 (tuning parameter)를 나타낸다. Yuan과 Lin (2007)은 위의 문제 (2.1)를 공분산 행렬의 조건인 양정치성 (positive definiteness)를 제약조건으로 고려하여 MAXDET 알고리즘 (Vandenberghe 등, 1998)을 이용하여 추정량을 계산하였다. 하지만, MAXDET 알고리즘의 경우 양정치성을 보장하지만 계산 복잡도 (computational complexity)의 차수가 $O(p^6)$에 해당하여 p가 큰 고차원의 문제에는 효율성이 매우 떨어지는 단점이 있다. 따라서, 목적함수 (2.1)를 최대화하는 보다 효율적인 알고리즘들이 개발되었으며 (Banerjee 등, 2008; Friedman 등, 2008), 이 중에서 graphical lasso (glasso) 알고리즘이 효율적이며 널리 사용되고 있다 (Friedman 등, 2008). 최근에는 사전 블록화 절차를 통하여 보다 효율적으로 계산할 수 있도록 개선되었다 (Witten 등, 2011). 개선된 glasso 알고리즘은 기존에 발표된 R 패키지 `glasso`를 통하여 제공되고 있다.

회귀 모형 기반의 접근 방법은 하나의 변수를 반응 변수로 여기고 나머지 다른 변수들을 설명 변수로 적용한 아래의 회귀 모형에서

$$\sum_{k=1}^{n} \left(X_i^k - \sum_{j \neq i} \beta_{ij} X_j^k \right)^2 \tag{2.2}$$

$\beta_{ij} = 0$인 것과 $\omega_{ij} = 0$인 것이 동치임을 이용하여 ℓ_1 노름을 이용한 벌점화 회귀 모형 (즉, LASSO 회귀 모형)

$$\frac{1}{2} \sum_{k=1}^{n} \left(X_i^k - \sum_{j \neq i} \beta_{ij} X_j^k \right)^2 + \lambda \sum_{j \neq i} |\beta_{ij}| \tag{2.3}$$

을 고려하여 조건부 종속과 독립을 추정하였다 (Meinshausen과 Bühlmann, 2006). Meinshausen과 Bühlmann (2006)에 의해 제안된 근방 선택 (neighborhood selection) 방법은 각 변수에 대하여 회귀 모형 (2.3)을 고려하여 p개의 LASSO 회귀 모형에 대한 해를 독립적으로 구한다. 이 경우, $\beta_{ij} = 0$이며 $\beta_{ji} \neq 0$인 경우가 발생할 수 있으므로 무방향그래프 $G = (V, E)$를 가정한 모형과 모순된 결과를 초래할 수 있는 단점이 있다. 이를 개선한 방법으로 Peng 등 (2009)이 제안한 SPACE 방법이 있다. SPACE 방법의 경우에는 모순된 연결선 문제의 해결을 위하여 부분상관계수 $\rho_{ij} = -\omega_{ij}/\sqrt{\omega_{ii}\omega_{jj}} = \beta_{ij}\sqrt{\omega_{ii}/\omega_{jj}}$의 정의와 $\rho_{ij} = \rho_{ji}$인 성질을 이용하여

$$\frac{1}{2} \sum_{i=1}^{p} w_i \sum_{k=1}^{n} \left(X_i^k - \sum_{j \neq i} \rho_{ij} \sqrt{\frac{\omega_{jj}}{\omega_{ii}}} X_j^k \right)^2 + \lambda \sum_{j \neq i} |\rho_{ij}| \quad \text{subject to:} \quad \rho_{ij} = \rho_{ji} \text{ for } i < j \tag{2.4}$$

부분상관계수 $\rho = (\rho_{12}, \rho_{13}, \ldots, \rho_{(p-1)p})^T$와 정밀 행렬의 대각성분 $\omega_D = (\omega_{11}, \omega_{22}, \ldots, \omega_{pp})^T$을 나누어 추정한다. 여기서, w_i는 i번째 변수에 대한 비음의 가중치 (weight)를 나타내며 λ는 비음의 조율 모수이다. SPACE 방법에서는 부분상관계수의 효율적인 추정을 위하여 개별 좌표 하강 (coordinate descent, CD) 알고리즘에 기반한 shooting 알고리즘 (Fu, 1998)을 개선하여 0이 아닌 반복해에 대한 갱신과 전체 반복해의 갱신을 나누어 계산하는 활성화 shooting (active shooting) 알고리즘을 적용하였다 (Peng 등, 2009). 정밀 행렬의 대각성분 ω_D의 추정은 추정된 부분상관계수를 이용하여

$$\hat{\omega}_{ii} = n \Big/ \Big\| \mathbf{X}_i - \sum_{j \neq i} \hat{\rho}_{ij} \sqrt{\hat{\omega}_{jj}^{(old)} / \hat{\omega}_{ii}^{(old)}} \mathbf{X}_j \Big\|_2^2 \text{ for } i = 1, 2, \ldots, p, \tag{2.5}$$

을 이용하여 갱신한다. SPACE 방법은 계산 상의 효율성과 가능도 함수 기반의 방법과 비교할 때 상대적으로 모형의 분포 가정에 덜 민감하여 생물학 분야의 응용 분야에 많이 적용되었다 (Dong 등, 2010; Tang 등, 2013).

하지만, SPACE 방법의 경우에 부분상관계수 ρ와 정밀 행렬의 대각성분 ω_D을 반복하여 갱신하면서 추정량을 계산하게 되는데 목적함수 (2.4)는 ρ와 ω_D대하여 양면 볼록 함수 (bi-convex function)이므로 각각이 고정된 하에서는 볼록 함수가 되어 수렴성이 보장되지만 ρ와 ω_D를 동시에 고려할 경우 수렴성이 보장되지 않는다. 따라서 SPACE의 경우와 같이 ρ와 ω_D에 대하여 추정하는 경우 수렴하지 않는 경우가 발생할 수 있음이 최근 보고 되었다 (Khare 등, 2015).

위의 문제를 해결하기 위하여, 추정량에 대하여 볼록 함수의 성질을 지니며 분포 가정이 상대적으로 완화 된 유사가능도 (pseudo-likelihood) 기반의 CONCORD 방법이 Khare 등 (2015)에 의해 제안되었다. CONCORD 방법은 ω_D에 대한 추정을 목적 함수을 이용하여 직접 추정이 이루어지며, ρ에 대한 추정으로 인하여 볼록성질 (convexity)가 위배됨을 반영하여 Ω의 성분을 이용한 목적함수

$$L(\Omega; \lambda) = -\sum_{i=1}^{p} n \log \omega_{ii} + \frac{1}{2} \sum_{i=1}^{p} \sum_{k=1}^{n} \Big(\omega_{ii} X_i^k + \sum_{j \neq i} \omega_{ij} X_j^k \Big)^2 + \lambda \sum_{i<j} |\omega_{ij}| \tag{2.6}$$

를 고려하였다. 위의 목적함수 (2.6)는 SPACE 방법의 목적함수 (2.4)에서 $w_i = \omega_{ii}^2$로 설정하고 벌점을 ω_{ij}에 고려한 모형으로 SPACE 방법을 개선한 방법이다. CONCORD 방법은 ω_D와 $(\omega_{ij},\ i < j)$를 블록화하여 각각에 대한 블록 좌표 하강법 (block coordinate descent, BCD) 알고리즘을 적용하여 추정량을 효율적으로 계산한다. 보다 자세히 살펴보면, 주어진 $(\hat{\omega}_{ij}^{(0)},\ i < j)$ 추정량을 기반으로 $\omega_D = (\omega_{ii})_{1 \leq i \leq p})$는

$$\hat{\omega}_{ii} = \frac{-\sum_{j \neq i} \hat{\omega}_{ij}^{(0)} T_{ij} + \sqrt{\big(\sum_{j \neq i} \hat{\omega}_{ij}^{(0)} T_{ij} \big)^2 + 4 n s_{ii}}}{2 T_{ii}} \tag{2.7}$$

로 계산되며, 주어진 $\hat{\omega}_D^{(0)}$에 대하여 $\hat{\omega}_{ij}$는 개별 좌표 하강법 (coordinate descent, CD) 알고리즘을 통하여

$$\hat{\omega}_{ij} = \frac{\text{Soft}_\lambda(-\sum_{j' \neq j} \tilde{\omega}_{ij'} T_{jj'} - \sum_{i' \neq i} \tilde{\omega}_{i'j} T_{ii'})}{T_{ii} + T_{jj}} \tag{2.8}$$

로 수렴할 때까지 반복하여 갱신된다. 여기서 T_{ij}는 $\mathbf{X}^T\mathbf{X}$의 (i,j)번째 성분을 나타내며, $\text{Soft}_\tau(x) = \text{sign}(x)(|x| - \tau)_+$로 부드러운 임계화 (soft-thresholding), $(x)_+ = \max(0, x)$를 나타낸다. CONCORD 방법의 계산 효율성은 glasso 알고리즘과 유사하며 연결선 식별에 대하여 더 좋은 성능을 지님이 여러 모의 실험을 통하여 보여졌다 (Khare 등, 2015). CONCORD 방법은 R 패키지 `gconcord`로 공개되어 제공되고 있다.

마지막으로 제약조건이 있는 ℓ_1 최소화 방법에서 적용하고 있는 최적화 문제의 형식은 Dantzig 선택기 (Dantzig selector)로 알려진 문제의 형태로 (Candes와 Tao, 2007), 압축 센싱 (compressed censing)

분야에서 많이 적용되고 있다 (Candes와 Plan, 2011). Cai 등 (2011)은 제약조건이 있는 ℓ_1 최소화 문제 형식을 정밀 행렬의 추정에 적용하여 CLIME (constrained ℓ_1-minimization for inverse matrix estimation) 방법을 제안하였다. CLIME 방법은

$$\min \|\Omega\| \quad \text{subject to:} \quad \|\mathbf{S}\Omega - I\|_\infty \tag{2.9}$$

의 최적화 문제를 통하여 Ω를 추정한다. 여기서 $\mathbf{S}$는 $\mathbf{X}$의 표본 공분산 행렬, I는 $p \times p$ 차원의 단위 행렬 (idnetity matrix), $\|A\|_\infty = \max_{1\le i,j\le p} |a_{ij}|$로 행렬의 ℓ_∞ 노름을 나타낸다. 위의 최적화 문제로 부터 계산된 추정량을 $\hat{\Omega}^{(0)} = (\hat{\omega}_{ij}^{(0)})$라 할 때, $\hat{\Omega}^{(0)}$는 일반적으로 대칭성 (symmetry)을 만족하지 않는다. 이를 해결하기 위하여, Cai 등 (2011)은 최종 추정량 $\hat{\Omega} = (\hat{\omega}_i j)$를

$$\hat{\omega}_{ij} = \hat{\omega}_{ij}^{(0)} I(|\hat{\omega}_{ij}^{(0)} \le \hat{\omega}_{ji}^{(0)}) + \hat{\omega}_{ji}^{(0)} I(|\hat{\omega}_{ij}^{(0)} > \hat{\omega}_{ji}^{(0)}) \tag{2.10}$$

를 이용하여 정의하였고 분포의 적률 (moment)에 대한 가정 하에서 최소최대 최적수렴성 (minimax optimal convergence rate)을 지님을 증명하였다.

CLIME 방법에서는 목적함수에 ℓ_1 노름만을 포함하며 제약조건은 오직 범위에 대한 최소 및 최대의 조건이므로 정확히 선형계획법 (linear programming)의 최적화 문제가 된다. 따라서, 선형계획법의 일반적인 알고리즘인 원-쌍대문제 내부점 (primal-dual interior point) 방법 또는 심플렉스 (simplex) 방법을 통하여 해를 찾을 수 있다 (Boyd와 Vandenberghe, 2004). Cai 등 (2011)은 행렬에 대한 최적화 문제를 p개의 독립적인 벡터의 최적화 문제로 유도하여 효율적으로 계산하였다. 하지만 선형계획법의 최적화를 독립적으로 p개를 풀고 있기 때문에 p가 커질 경우 다른 방법과 비교할 때 계산 속도가 느리다는 단점이 있다. 이를 해결하기 위하여, Pang 등 (2014)이 해의 경로를 고려한 알고리즘을 개발하여 속도를 향상 시켰으며 R 패키지 `fastclime`으로 제공되고 있다.

3. 유사가능도 기반의 모형에 대한 GPU 병렬화 BCDR 알고리즘

본 절에서는 여러 네트워크 추정 모형 중에서 참연결성 식별의 성능이 glasso 모형 보다 뛰어나며 (Khare 등, 2015), 분포에 대한 가정이 상대적으로 유연한 유사가능도 기반의 CONCORD 방법에 대한 GPU 병렬화 BCDR 알고리즘을 제안하고자 한다. GPU 병렬화 BCDR 알고리즘의 설명에 앞서 기존 CONCORD 방법의 알고리즘을 살펴보면, 대각성분과 비대각성분을 블록으로 구성하여 BCD 알고리즘을 적용하고 있으며 각 블록에 대한 CD 알고리즘은 Algorithm 1에 명시된 바와 같이 개별 좌표를 일정 순서대로 갱신하는 순환 갱신 (cyclic update)을 적용하고 있다. 일반적으로 순환 갱신을 적용한 경우에는 순차적으로 갱신이 이루어지므로 병렬 연산을 적용하여 얻는 이점이 매우 낮다. 특히, GPU를 이용한 병렬 연산의 경우에는 CPU 보다 상대적으로 낮은 연산 속도와 메모리간의 통신으로 인하여 오히려 병렬 연산이 계산 효율을 더 낮추는 효과를 가져올 수 있다.

하지만, 본 논문에서 다루고 있는 CONCORD 모형의 목적 함수에 대한 개별 좌표에 대한 갱신을 살펴보면, 식 (2.7)의 대각 성분의 블록에 대한 갱신의 경우, 각각의 $\hat{\omega}_{ii}$의 갱신에 있어 $\sum_{j\neq i} \hat{\omega}_{ij} T_{ij}$와 $T_i i$의 정보만 이용할 뿐 $(\hat{\omega}_{kk})_{k\neq i}$는 영향을 주지 않으므로 $(\hat{\omega}_{ii})_{1\le i\le p}$는 독립적으로 갱신이 가능함을 알 수 있다. 따라서, $(\hat{\omega}_{ii})_{1\le i\le p}$의 갱신을 개별 좌표에 따라 식 (2.7)에 GPU 기반의 병렬 연산을 적용하여 효율적으로 계산할 수 있다. 비록 대각 성분의 갱신에 대하여 병렬 연산의 적용이 이점을 지닌다고 볼 수 있으나 정밀 행렬의 추정에 있어서 여전히 $p(p-1)/2$개의 비대각 성분에 대하여 순환 갱신을 적용하게 되므로 계산 효율의 개선이 크지 않다.

비대각 원소에 대하여 병렬 연산을 통하여 계산 효율을 증가시키기 위하여 본 논문에서는 랜덤 갱신 (random upate) 규칙을 적용하고자 한다. 랜덤 갱신 규칙은 순환 갱신과 같은 결정적인 갱신 (deterministic update) 규칙과 다르게 각 좌표의 선택 확률로 균일한 확률 또는 Lipschitz 상수에 비례하는

확률을 고려하여 갱신 좌표를 랜덤하게 선택하여 갱신하는 규칙이다. CD 알고리즘에 대한 랜덤 갱신 규칙은 확률 CD (stochastic coordinate descent, SCD) 알고리즘 (Shalev-Shwartz와 Tewari, 2011) 및 랜덤 CD (random coordinate descent, RCD) 알고리즘 (Nesterov, 2012)에서 소개 되었으며 수렴성에 대한 증명이 이루어졌다. 특히, 결정적인 갱신 (deterministic upate)과 비교하여 수렴 속도 (convergence rate)와 같은 이론적인 성질의 유도에 대한 이점을 지닌다 (Nesterov, 2012).

본 연구에서는 랜덤 갱신 규칙과 CONCORD 모형의 비대각성분에 대한 식 (2.8)의 성질은 이용하여 병렬 연산에 이점을 지닌 갱신 규칙을 제안한다. 먼저 갱신 식 (2.8)에 대한 성질을 살펴 보면, (i,j)의 성분에 대한 갱신 식 (2.8)은 $\hat{\omega}_{ij}$의 갱신을 위하여 $\sum_{k\neq i}\hat{\omega}_{jk}T_{ki}$, $\sum_{k\neq j}\hat{\omega}_{ik}T_{kj}$, 및 (T_{ii},T_{jj})의 정보만을 사용하므로 $\hat{\Omega}$의 i행, j행을 제외한 나머지 행에 대한 값은 전혀 영향을 주지 않게 된다. 위의 성질을 통하여 p가 짝수인 경우에는 최대 $p/2$개의 좌표를 독립적으로 갱신이 가능하며 p가 홀수인 경우에도 $\lfloor p/2\rfloor$개의 좌표를 독립적으로 갱신할 수 있음을 알 수 있다. 여기서, $\lfloor x\rfloor$는 x를 소수점 이하의 값을 내림한 값을 나타낸다.

본 연구에서 고려한 병렬화 가능한 랜덤 갱신 규칙을 설명하기 위하여 I^π는 $(1,2,\ldots,p)$의 벡터를 랜덤 치환 (random permutation) π에 의해 치환한 벡터로 나타내고 $I^T=(I_1^\pi,I_2^\pi,\ldots,I_{\lfloor p/2\rfloor}^\pi)$로 정의하였다. 본 연구에서 식 (2.8)의 성질과 랜덤 갱신 규칙을 결합하여 아래의 갱신 규칙을 고려하였다.

(Step 1) 랜덤 치환 π를 수행하여 I^π를 생성한다.

(Step 2) 생성된 I^π를 기반으로 I^T를 정의한다.

(Step 3) 갱신 좌표의 설정을 위하여 아래와 같이 집합 I를 정의한다.

$$(\underbrace{I_1^\pi,I_2^\pi}_{(I_1,I_2)},\underbrace{I_3^\pi,I_4^\pi}_{(I_3,I_4)},\ldots,\underbrace{I_{\lfloor p/2\rfloor-1}^\pi,I_{\lfloor p/2\rfloor}^\pi}_{I_{\lfloor p/2\rfloor-1},I_{\lfloor p/2\rfloor}})\rightarrow I=\{(I_{2j-1},I_{2j})\mid I_{2j-1}<I_{2j},j=1,2,\ldots,\lfloor p/2\rfloor\}$$

(Step 4) 모든 $(r,s)\in I$에 대하여 $\hat{\omega}_{rs}$를 식 (2.8)를 이용하여 병렬적으로 갱신한다.

위의 랜덤 갱신 규칙을 적용할 경우에는 각 행과 열이 겹치지 않도록 선택되어 병렬 연산이 가능한 이점을 지닌다. 다만, 랜덤 치환에 의하여 $\lfloor p/2\rfloor$개의 좌표들 중에서 일부가 중복되어 갱신될 수 있으며 한 번에 $\lfloor p/2\rfloor$개의 좌표만 갱신하므로 대각 성분과 비대각 성분의 BCD 알고리즘 적용을 위하여 랜덤 치환을 $(p-1)$번 반복하는 형태로 알고리즘을 구성하였다. 본 절에서 제안한 전체 알고리즘은 Algorithm 2에 정리하였다.

Algorithm 2에 명시한 바와 같이 랜덤 치환에 의한 갱신 규칙을 통하여 $\lfloor p/2\rfloor$개의 비대각성분이 병렬적으로 갱신이 가능하며 p개의 대각 성분이 병렬적으로 갱신이 가능하다. 병렬 연산은 일반적으로 중앙처리장치 (central processing unit, CPU)의 병렬 연산을 통하여 이루어졌으나 최근에는 그래픽 처리 장치 (graphics processing unit, GPU)를 이용하여 병렬 연산이 가능하게 되어 다양한 분야에 적용되고 있다. 하지만, GPU는 CPU 보다 코어 (core)의 수가 매우 많기는 하지만 상대적으로 낮은 처리 속도를 지니므로 GPU를 이용한 병렬 연산이 CPU를 이용한 병렬 연산과 비교하여 항상 효율적인 것은 아니며 매우 많은 수의 GPU 코어를 활용할 수 있는 동인한 연산을 여러 자료에 대하여 적용하는 경우에 대하여 병렬 연산의 이점을 지닌다. 본 연구에서 고려하고 있는 갱신 식 (2.7)과 (2.8)은 다수의 개별 좌표로 병렬 계산이 가능하며 내적 (inner product)과 제곱근, 부드러운 임계화 등의 상대적으로 간단한 연산으로 이루어져 있어 GPU를 이용하면 고차원의 문제에 대한 병렬 연산에서 계산 효율을 크게 증가 시킬 수 있다. 본 연구에서는 NVIDIA 기반의 GPU를 활용하여 CUDA 7.5 Toolkit (`https://developer.nvidia.com/cuda-toolkit`)을 이용하여 병렬 연산을 진행하였다. 본 연구에서 적용한 CUDA 기반의 GPU 병렬 연산은 Yu와 Lim (2013)에 기본적인 소개와 예시가 제시되어 있다.

Algorithm 1 Coordinate descent algorithm with cyclic update

Require: $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_p), \Omega^{(0)} = (\omega_{ij}^{(0)}), \lambda, \delta_{tol}$

$k = 0, \hat{\Omega} = \Omega^{(0)}, T = \mathbf{X}^T\mathbf{X}$ ▷ initialization

repeat

  $k = k + 1$

  for $i = 1, 2, \ldots, p-1$ **do**

    for $j = i+1, \ldots, p$ **do**

      $\hat{\omega}_{ij} = \frac{\mathrm{Soft}_\lambda(-\sum_{j' \neq j} \hat{\omega}_{ij'} T_{jj'} - \sum_{i' \neq i} \hat{\omega}_{i'j} T_{ii'})}{T_{ii} + T_{jj}}$

    end for

  end for

  for $i = 1, 2, \ldots, p$ **do**

    $\hat{\omega}_{ii} = \frac{-\sum_{j \neq i} \hat{\omega}_{ij} T_{ij} + \sqrt{\left(\sum_{j \neq i} \hat{\omega}_{ij} T_{ij}\right)^2 + 4nT_{ii}}}{2T_{ii}}$

  end for

  $\delta = \|\Omega^{(k)} - \hat{\Omega}\|_\infty$

  $\Omega^{(k)} = \hat{\Omega}$

until $\delta < \delta_{tol}$

Algorithm 2 Parallel coordinate descent algorithm with stochastic update

Require: $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_p), \Omega^{(0)} = (\omega_{ij}^{(0)}), \lambda, \delta_{tol}$

$k = 0, \hat{\Omega} = \Omega^{(0)}, T = \mathbf{X}^T\mathbf{X}$ ▷ initialization

repeat

  $k = k + 1$

  for $i = 1, 2, \ldots, p-1$ **do**

    Generate an index vector I^π by random permutation π of $(1, 2, \ldots, p)$

    Define a target set $I = \{(r, s) \mid r = I^\pi_{2j-1}, s = I^\pi_{2j}, j = 1, 2, \ldots, \lfloor p/2 \rfloor\}$

    Update all $(r, s) \in I$ in parallel, $\hat{\omega}_{rs} = \frac{\mathrm{Soft}_\lambda(-\sum_{k \neq s} \hat{\omega}_{rk} T_{sk} - \sum_{k \neq r} \hat{\omega}_{ks} T_{rk})}{T_{rr} + T_{ss}}$

  end for

  Upate $i = 1, 2, \ldots, p$ in parallel, $\hat{\omega}_{ii} = \frac{-\sum_{j \neq i} \hat{\omega}_{ij} T_{ij} + \sqrt{\left(\sum_{j \neq i} \hat{\omega}_{ij} T_{ij}\right)^2 + 4nT_{ii}}}{2T_{ii}}$

  $\delta = \|\Omega^{(k)} - \hat{\Omega}\|_\infty$

  $\Omega^{(k)} = \hat{\Omega}$

until $\delta < \delta_{tol}$

4. 모의실험

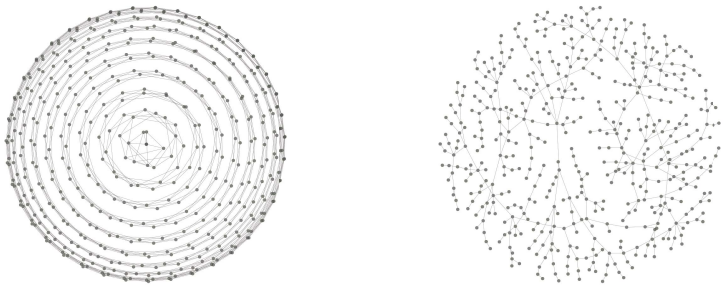
본 절에서는 랜덤 갱신 규칙 기반의 BCDR 알고리즘의 계산적인 효율성을 보이기 위하여 CONCORD의 BCD 알고리즘 (BCD)과 병렬 연산을 적용하지 않은 BCDR 알고리즘 (CPU-BCDR),

GPU를 이용한 BCDR 알고리즘 (GPU-BCDR)을 비교하였다. 계산 효율성의 공평한 비교를 위하여 CONCORD 방법의 BCD 알고리즘은 R 패키지인 `gconcord`의 C 코드를 기반으로 C 함수를 작성하였고 병렬 연산을 적용하지 않은 BCDR 알고리즘과 GPU를 이용한 BCDR 알고리즘 모두 C 함수로 작성하여 커맨드 라인 (command line)을 통하여 직접 실행 후 계산 시간을 측정하였다. 모든 알고리즘은 Intel Core i7-4790K CPU (4.00 GHz)와 32 GB RAM 사양의 데스크탑 PC (desktop PC)에서 실행되어 계산 시간이 측정되었다. 본 GPU 병렬 연산을 위하여 NVIDIA사의 GeForce GTX 970 (1.02 GHz, 4 GB RAM)을 사용하였다. 본 논문은 CONCORD의 BCD 알고리즘에 대한 병렬화 BCDR 알고리즘을 제안하고 효율성을 확인하는데 목적이 있어 실제 자료를 통한 자료 분석 절차 및 각 방법에 따른 조율 모수의 선택은 원래의 논문들을 참조하는 것으로 대신한다.

본 모의 실험에서는 참 네트워크의 모형으로 AR(2) 네트워크와 스케일프리 (scale-free) 네트워크의 두 가지 구조를 고려하였다. AR(2) 네트워크는 시계열 모형에서의 AR(2) 모형과 동일한 연관성을 갖는 네트워크로 인접한 두 변수와 조건부 종속성을 지닌 네트워크로 정의하였으며 (Figure 4.1 (a) 참조), 스케일프리 네트워크는 노드의 차수 (degree of node)가 멱법칙 (power law)

$$P(k) \propto k^{-\alpha}$$

를 따름이 알려져 있으므로 본 모의 실험에서는 $\alpha = 2.3$으로 설정하여 Barabasi-Albert 모형 (Barabasi와 Albert, 1999)을 기반으로 R 패키지인 `igraph`의 `barabasi.game()` 함수를 통하여 네트워크를 생성하였다 (Figure 4.1 (b) 참조). 여기서, $P(k)$는 k 차수를 갖는 노드의 비율을 나타내며 α는 선호적 연결 (preferential attachment) 상수로 일반적으로 $2 \sim 3$의 값을 갖는 것으로 알려져 있다.



(a) AR(2) (b) Scale-free

Figure 4.1 Structures of AR(2) and scale-free networks ($p = 500$)

AR(2) 네트워크에 대응하는 참 정밀 행렬 (true precision matrix) $\Omega^{AR} = (\omega_{ij}^{AR})$은 아래와 같이

$$\omega_{ij}^{AR} = \omega_{ji}^{AR} = \begin{cases} 0.45 & \text{for } i = 1, 2, \ldots, p-1,\ j = i+1 \\ 0.4 & \text{for } i = 1, 2, \ldots, p-2,\ j = i+2 \\ 0 & \text{otherwise} \end{cases}$$

생성하였으며, 스케일프리 네트워크에 대응하는 참 정밀 행렬은 양정치성 (positive definiteness)을 만족하기 위하여 Peng 등 (2009)에서 적용한 바와 같이 주어진 그래프 $G = (V, E)$에 대하여 참 정밀 행렬 $\Omega^{SC} = (\omega_{ij}^{SC})$을 아래의 절차를 적용하여 생성하였다.

Step 1: $\tilde{\Omega} = (\tilde{\omega}_{ij})$, $\tilde{\omega}_{ij} = \tilde{\omega}_{ji} \sim Unif\big([-1, -0.5] \cup [0.5, 1]\big)$ for $i < j$, $\tilde{\omega}_{ii} = 1$ for $i = 1, 2, \ldots, p$

Step 2: $\tilde{\omega}_{ij} = \tilde{\omega}_{ij} / \big(1.25 \sum_{j \neq i} \tilde{\omega}_{ij}\big)$

Step 3: $\Omega^{SC} = (\tilde{\Omega} + \tilde{\Omega}^T)/2$

여기서, 연결선의 존재를 보장하기 위하여 ω_{ij}^{SC}의 절댓값이 0.1보다 작은 경우에는 $0.1 \cdot \text{sign}(\omega_{ij}^{SC})$로 설정하였다.

본 모의 실험에서는 표본의 수 (n)는 200과 500을, 변수의 수 (p)는 500, 1000, 2000 및 3000을 고려하였으며, 관측 자료 $\mathbf{X} = (\mathbf{X}^1, \mathbf{X}^2, \ldots, \mathbf{X}^n)^T$는 $\mathbf{X}^k \sim N(0, \Omega^{-1})$을 통하여 생성하였다. 여기서, $\mathbf{X}^k$는 p차원의 벡터로 k번째 표본을 나타내며 Ω는 각 네트워크 구조에 대응하는 참 정밀 행렬이다. 벌점화 모형에서는 고려하는 조율 모수에 따라 알고리즘의 수렴 속도가 영향을 받으므로 $\lambda/n = 0.1$, 0.3의 두 가지 경우를 고려하여 비교하였다. 알고리즘의 계산 효율성 비교를 위하여 10개의 자료를 독립적으로 생성하여 실험을 진행하였다. 또한 각 알고리즘의 수렴 기준은 동일하게 $\|\Omega^{(old)} - \hat{\Omega}^{(new)}\|_\infty < 10^{-5}$를 적용하였다. 여기서, $\Omega^{(old)}$는 이전 반복의 해, $\Omega^{(new)}$는 현재 반복의 해를 나타낸다.

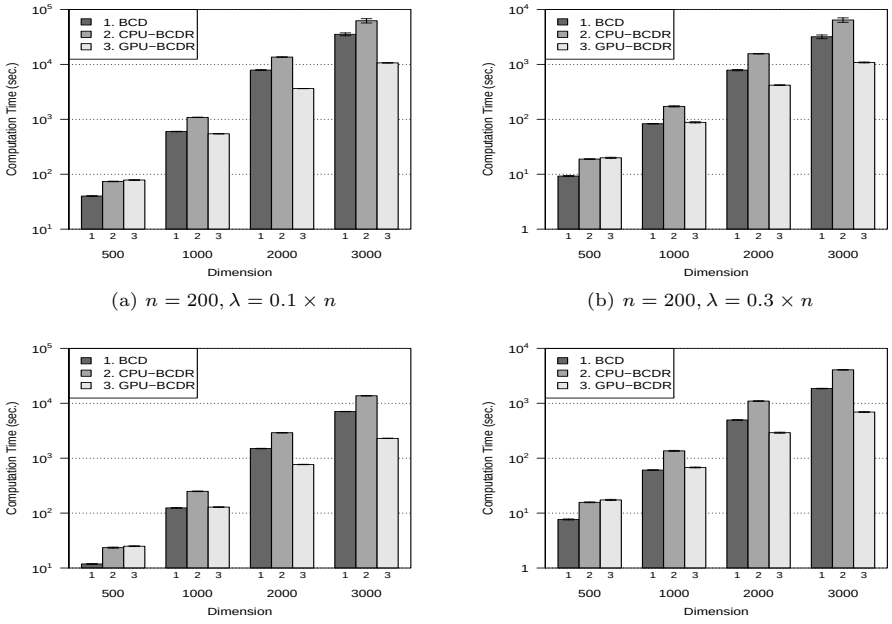
모의 실험의 결과는 평균 계산 시간, 반복 수 및 수렴 후 목적함수의 값과 각 측도에 대한 표준오차 (standard error)를 Table 4.1과 Table 4.2에 요약하였다. 먼저, 병렬화 연산을 적용하지 않은 BCDR 알고리즘과 BCD 알고리즘의 결과를 비교하면, AR2 및 스케일프리 네트워크 구조 하에서 평균 반복 수는 모의 실험에서 고려한 모든 경우에서 BCDR 알고리즘이 BCD 알고리즘보다 약 1.5 $\sim$ 2.0 배 정도 더 반복 되는 것을 확인할 수 있다. 이러한 패턴은 BCDR 알고리즘의 특성에서 기인한 것으로 BCDR 알고리즘이 각 반복마다 독립적으로 랜덤 치환을 $p-1$회 수행하기 때문에 각 반복에서 중복 갱신된 성분과 갱신이 이루어지지 않는 성분들의 영향 때문이다. 이에 따른 결과로 BCDR 알고리즘을 적용한 평균 계산 시간이 본 모의 실험의 모든 경우에 대하여 BCD 알고리즘 보다 약 1.5 $\sim$ 2.0 배 가량 느리게 계산 된 것을 확인할 수 있다. BCDR 알고리즘은 반복 수가 증가하기는 하지만 수렴된 목적함수의 값을 비교하면 BCD 알고리즘과 동일한 함수값을 갖음을 알 수 있다. 여기서, $L(\Omega; \lambda)$는 $\lambda > 0$인 경우 강볼록성 (strict convexity)를 지니므로 $\lambda/n = 0.1$, 0.3인 모의실험에서 BCDR 알고리즘과 BCD 알고리즘은 동일한 해로 수렴하였음을 알 수 있다.

본 연구에서 제안하는 BCDR 알고리즘은 병렬화에 대한 이점이 존재하여 이를 활용할 경우에 알고리즘의 효율을 크게 증가 시킬 수 있다. 이해를 돕기 위하여, AR2 및 스케일프리 네트워크 구조에 대한 평균 시간의 비교 그래프를 Figure 4.2과 Figure 4.3에 각각 제시하였다. AR2 및 스케일프리 네트워크 구조 하에서 $p = 500$인 경우, BCD 알고리즘이 CPU-BCDR 및 GPU-BCDR 알고리즘 보다 빠르게 계산이 이루어지지만 $p = 1000$인 경우 BCD알고리즘과 GPU-BCDR 알고리즘의 계산 속도는 비슷하게 나타나며, $p > 1000$인 경우에는 항상 GPU-BCDR 알고리즘의 계산 속도가 BCD 알고리즘보다 약 1.7 $\sim$ 3.3 배 빨라지는 것을 확인할 수 있었다 (Table 4.1, 4.2 참조).

차원이 상대적으로 낮은 경우 ($p = 500$), GPU-BCDR 알고리즘의 효율이 저하되는 원인은 GPU를 활용한 병렬 연산의 경우 데이터를 호스트 (host) 메모리에서 디바이스 (device) 메모리로, 디바이스 메모리에서 호스트 메모리로 전송하는 메모리 간의 통신 시간이 포함되기 때문이다. 차원이 큰 경우에는 전체 계산 시간에서 데이터의 전송 시간의 비율이 상대적으로 낮아지게 되어 GPU 병렬 연산으로 인한 효율이 높아진다.

마지막으로 AR2 네트워크와 스케일프리 네트워크의 구조에 따른 수렴 속도를 비교하면 $p = 1000$, $\lambda/n = 0.1$인 경우를 제외하면 스케일프리 네트워크 구조에서의 수렴 속도가 AR2 네트워크에서의 수렴 속도 보다 약 1.1 $\sim$ 5.3 배 정도 느리게 나타남을 알 수 있다. 위와 같이 수렴 속도의 차이가 발생한 원인은 본 연구에서 참 정밀 행렬의 생성 과정에서 AR2 네트워크의 경우, 고정된 값을 이용하여 상대적으로 낮은 행렬의 조건 수 (condition number)를 지니도록 생성되었으나, 스케일프리 네트워크의 경우, 연결선을 먼저 생성한 뒤에 최소 고유값이 0보다 큰 조건만 만족하면 생성되도록 하여 AR2 네트워크의 조건 수 보다 상대적으로 높게 생성되었다. 자세히 살펴 보면, AR2의 참 정밀 행렬의 조건수는 모든 차원에 대하여 약 37 정도로 계산 되었으나, 스케일프리 네트워크의 경우, $p = 500, 1000, 2000, 3000$에 대하여 각각 $429, 58, 106, 64$로 계산 되어 $p = 500$, 2000인 경우에 AR2와 비교하여 상대적으로 더 많은

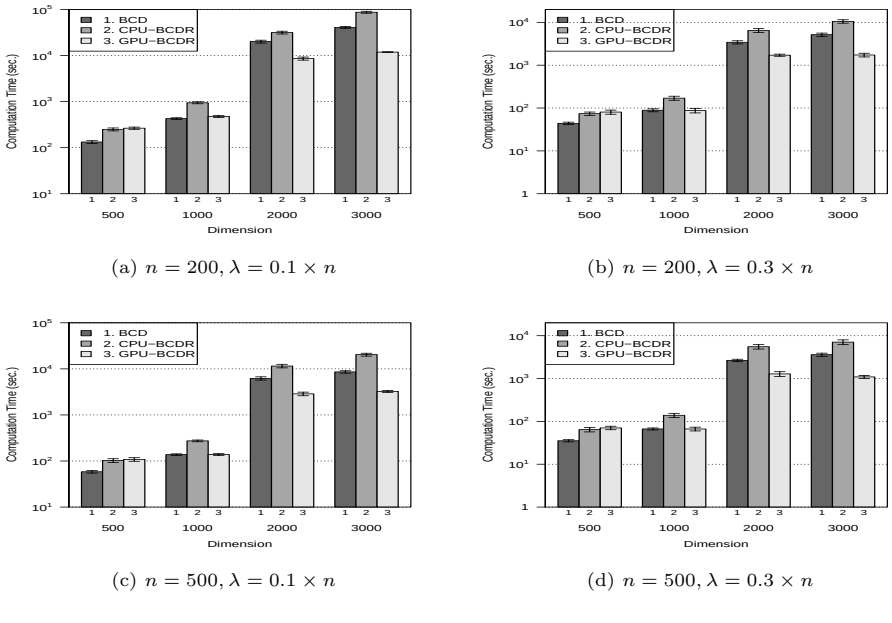
수렴 시간이 필요함을 알 수 있었다.



(a) $n = 200, \lambda = 0.1 \times n$ (b) $n = 200, \lambda = 0.3 \times n$

(c) $n = 500, \lambda = 0.1 \times n$ (d) $n = 500, \lambda = 0.3 \times n$

Figure 4.2 Averages of computation times for AR(2) network.
Vertical lines denote 95% confidence intervals of mean of computations times



(a) $n = 200, \lambda = 0.1 \times n$ (b) $n = 200, \lambda = 0.3 \times n$

(c) $n = 500, \lambda = 0.1 \times n$ (d) $n = 500, \lambda = 0.3 \times n$

Figure 4.3 Averages of computation times for scale-free network.
Vertical lines denote 95% confidence intervals of mean of computations times

Table 4.1 Averages of computation times, number of iterations, and converged objective function values for AR2 network. Numbers in parenthesis denote standard errors.

λ/n	n	p	Computation time(sec.)			Iteration			$L(\Omega)$		
			BCD	BCDR(CPU)	BCDR(GPU)	BCD	BCDR(CPU)	BCDR(GPU)	BCD	BCDR(CPU)	BCDR(GPU)
0.1	200	500	40.1	73.8	78.35	119	210.9	211	4020.49	4020.01	4020
			(0.27)	(0.47)	(0.62)	(0.73)	(1.26)	(1.69)	(36.13)	(36.12)	(36.16)
		1000	598.56	1083.47	545.49	229.4	388.3	387.4	11614.18	11611.37	11611.12
			(3.28)	(4.76)	(2.70)	(1.26)	(1.80)	(1.94)	(76.15)	(76.22)	(76.13)
		2000	7872.78	13598.8	3629.88	381.7	625.9	627.9	30574.19	30562.92	30563.69
			(69.62)	(60.07)	(11.00)	(3.30)	(2.61)	(1.78)	(81.46)	(81.70)	(81.71)
		3000	35309.4	62323.98	10656.02	484	787.2	784.3	52243.07	52218.43	52216.76
			(1091.73)	(2852.91)	(45.62)	(2.08)	(2.74)	(3.48)	(170.61)	(169.95)	(170.17)
	500	500	11.85	23.51	24.95	34.3	66.9	66.5	1938.35	1938.32	1938.31
			(0.09)	(0.26)	(0.21)	(0.21)	(0.74)	(0.58)	(9.96)	(9.96)	(9.97)
		1000	124.24	248.89	128.56	46.6	88.2	89.7	4687.09	4686.89	4686.97
			(0.88)	(1.62)	(0.91)	(0.34)	(0.53)	(0.65)	(14.48)	(14.48)	(14.46)
		2000	1496.95	2907.65	767.51	71.6	132.8	131.9	12203.35	12202.57	12202.63
			(8.19)	(17.21)	(3.58)	(0.37)	(0.81)	(0.59)	(36.82)	(36.76)	(36.77)
		3000	7064.28	13688.18	2293.19	91.4	168.7	168	21556.79	21554.57	21554.89
			(27.16)	(64.88)	(10.15)	(0.34)	(0.76)	(0.75)	(40.92)	(40.67)	(40.76)
0.3	200	500	9.25	18.88	19.9	26.6	53.3	52.9	1275.94	1275.93	1275.93
			(0.10)	(0.13)	(0.24)	(0.31)	(0.37)	(0.66)	(4.15)	(4.15)	(4.15)
		1000	83.07	172.03	88.03	30.9	60.7	61.7	2701.55	2701.51	2701.52
			(0.48)	(2.03)	(1.10)	(0.18)	(0.70)	(0.73)	(10.61)	(10.62)	(10.62)
		2000	786.43	1564.38	418.19	37.2	71.1	71.5	5781.81	5781.71	5781.7
			(10.50)	(9.53)	(3.48)	(0.51)	(0.43)	(0.58)	(8.32)	(8.32)	(8.33)
		3000	3184.96	6432.07	1084.23	41.4	79.8	78.9	9052.17	9052.05	9051.99
			(127.90)	(303.68)	(10.22)	(0.27)	(0.55)	(0.77)	(21.77)	(21.77)	(21.77)
	500	500	7.66	15.7	17.37	21.7	44.3	46	1220.74	1220.72	1220.73
			(0.08)	(0.13)	(0.22)	(0.21)	(0.45)	(0.58)	(2.53)	(2.53)	(2.53)
		1000	60.78	135.94	67.74	22.2	47.5	46.8	2451.93	2451.92	2451.93
			(0.34)	(1.46)	(0.57)	(0.13)	(0.54)	(0.42)	(3.03)	(3.03)	(3.03)
		2000	496.07	1096.56	291.19	23	49.4	49.4	4941.36	4941.34	4941.34
			(3.13)	(8.31)	(3.23)	(0.15)	(0.37)	(0.56)	(6.14)	(6.14)	(6.14)
		3000	1853.09	4062.15	691.82	23.2	49.3	50	7459.55	7459.56	7459.56
			(11.08)	(22.69)	(6.07)	(0.13)	(0.26)	(0.45)	(7.12)	(7.12)	(7.12)

Table 4.2 Averages of computation times, number of iterations, and converged objective function values for scale-free network. Numbers in parenthesis denote standard errors.

λ/n	n	p	Computation time(sec.)			Iteration			$L(\Omega)$		
			BCD	BCDR(CPU)	BCDR(GPU)	BCD	BCDR(CPU)	BCDR(GPU)	BCD	BCDR(CPU)	BCDR(GPU)
0.1	200	500	132.63	249.51	264.63	399.5	717.3	712.5	1212.57	1212.53	1212.53
			(4.56)	(9.18)	(7.50)	(13.58)	(26.20)	(20.20)	(24.19)	(24.19)	(24.18)
		1000	426.59	942.28	477.16	163.5	337.4	338.3	3712.19	3712.2	3712.18
			(9.60)	(23.40)	(12.58)	(3.75)	(8.44)	(8.95)	(37.42)	(37.42)	(37.42)
		2000	20048.82	31479.29	8593.72	964.2	1450.6	1493.6	10053.1	10052.42	10052.61
			(741.36)	(1107.25)	(337.35)	(33.69)	(50.37)	(58.41)	(54.73)	(54.75)	(54.76)
		3000	40764.97	87073.98	11853.47	494.6	879.7	872.8	17663.94	17663.48	17663.41
			(942.09)	(2155.78)	(114.90)	(11.43)	(12.44)	(8.16)	(127.74)	(127.65)	(127.76)
	500	500	58.41	102.59	108.34	175.4	292.8	290.9	459.31	459.3	459.3
			(1.91)	(4.79)	(4.94)	(5.77)	(13.70)	(13.32)	(3.67)	(3.67)	(3.67)
		1000	137.52	273.54	137.71	51.9	97.2	96.7	1224.39	1224.39	1224.39
			(2.74)	(6.07)	(3.40)	(1.08)	(2.17)	(2.41)	(11.69)	(11.69)	(11.69)
		2000	6211.97	11529.5	2868.9	272.7	473.9	494.4	3178.78	3178.73	3178.76
			(256.07)	(520.89)	(130.81)	(11.05)	(21.95)	(22.50)	(13.20)	(13.22)	(13.19)
		3000	8592.14	20379.28	3241.93	109.3	244.3	237.7	5678.32	5678.28	5678.28
			(260.56)	(737.45)	(86.24)	(3.39)	(4.09)	(6.36)	(19.87)	(19.86)	(19.86)
0.3	200	500	43.9	73.91	80.54	131.7	211.7	216.1	405.08	405.08	405.08
			(1.48)	(3.42)	(4.54)	(4.54)	(9.80)	(12.25)	(2.09)	(2.09)	(2.09)
		1000	88.82	170.21	87.08	33.2	60.1	60.9	920.6	920.6	920.6
			(3.58)	(9.36)	(5.17)	(1.38)	(3.39)	(3.66)	(2.32)	(2.32)	(2.32)
		2000	3425.34	6454.55	1712.53	163.8	292.4	297.2	1984.61	1984.59	1984.59
			(147.29)	(342.92)	(49.79)	(6.13)	(13.38)	(8.88)	(4.27)	(4.27)	(4.27)
		3000	5152.94	10525.56	1730.45	63.5	121	126.5	3236.67	3236.67	3236.67
			(233.07)	(495.10)	(87.32)	(2.72)	(5.54)	(6.44)	(7.68)	(7.68)	(7.68)
	500	500	35.3	64.23	70.43	105.3	183.4	188.8	357.62	357.62	357.62
			(1.06)	(3.78)	(3.34)	(3.22)	(10.80)	(9.00)	(0.47)	(0.47)	(0.47)
		1000	66.67	138.77	66.53	24.5	48.7	46.2	759.76	759.76	759.76
			(1.83)	(7.76)	(3.40)	(0.70)	(2.80)	(2.42)	(0.60)	(0.60)	(0.60)
		2000	2650.19	5533.49	1282.82	116.4	226.3	220.5	1479.17	1479.17	1479.17
			(87.57)	(334.24)	(82.05)	(4.05)	(13.60)	(14.20)	(1.45)	(1.45)	(1.45)
		3000	3604.58	7085.31	1092.35	44.3	81.1	79.4	2258.53	2258.53	2258.53
			(156.65)	(459.26)	(46.02)	(1.63)	(3.02)	(3.38)	(1.47)	(1.47)	(1.47)

5. 결론

본 논문에서는 최근 발표된 네트워크 추정 방법인 유사가능도 기반의 CONCORD 방법에 대하여 GPU 병렬 연산과 랜덤 치환 기반의 갱신 규칙을 적용한 BCDR 알고리즘을 제안하였다. 모의실험을 통하여 제안하는 알고리즘이 변수의 수가 많은 고차원 자료에 대하여 계산 시간을 상당히 감소 시킴을 확인할 수 있었다. GPU를 이용한 병렬 연산은 알고리즘의 계산 효율을 향상 시키는데 있어서 유용한 도구이나 본 연구의 모의실험에서 나타난 것처럼 메모리 간의 데이터 전송으로 인하여 차원이 낮은 경우에는 오히려 효율이 저하 될 수 있다. 따라서 GPU의 병렬 연산은 메모리 간의 통신과 병렬 연산에 대한 고려가 필수적이다. GPU의 병렬 연산은 새로운 GPU의 연산 능력의 향상과 저장 장치의 용량 증가에 따라 연산의 효율성이 점차 증가할 것이며, 앞으로 다양한 통계 계산 분야에 활용되어질 것이라 기대한다.

References

Banerjee, O., Ghaoui, L. E. and d'Aspremont, A. (2008). Model selection through sparse maximum likelihood estimation for multivariate Gaussian or binary data. *Journal of Machine Learning Research*, **9**, 485-516.

Barabasi, A. and Albert, R. (1999). Emergence of scaling in random networks. *Science*, **286**, 509-512.

Boyd, S. and Vandenberghe, L. (2004). *Convex optimization*, Cambridge University Press, New York.

Cai, T., Liu, W. D. and Luo, X. (2011). A constrained ℓ_1 minimization approach to sparse precision matrix estimation. *Journal of the American Statistical Association*, **106**, 594-607.

Candés, E. J. and Tao, T. (2007). The Dantzig selector: Statistical estimation when p is much larger than n. *Annals of Statistics*, **35**, 2313-2351.

Candés, E. J. and Plan, Y. (2011). A probabilistic and RIPless theory of compressed sensing. *Information Theory, IEEE Transactions*, **57**, 7235-7254.

Dong, H., Luo, L., Hong, S., Siu, H., Xiao, Y., Jin, L., Chen, R. and Xiong, M. (2010). Integrated analysis of mutations, miRNA and mRNA expression in glioblastoma. *BMC Systems Biology*, **4**, 1-20.

Drton, M. and Perlman, M. D. (2004). Model selection for Gaussian concentration graphs. *Biometrika*, **91**, 591-602.

Friedman, J., Hastie, T. and Tibshirani, R. (2008). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, **9**, 432-441.

Fu, W. (1998). Penalized regressions: The bridge vs the lasso. *Journal of Computational and Graphical Statistics*, **7**, 397-416.

Khare, K., Oh, S.-Y. and Rajaratnam, B. (2015). A convex pseudolikelihood framework for high dimensional partial correlation estimation with convergence guarantees. *Journal of the Royal Statistical Society B*, **77**, 803-825.

Kwon, S., Han, S. and Lee, S. (2013). A small review and further studies on the LASSO. *Journal of the Korean Data & Information Science Society*, **24**, 1077-1088.

Lauritzen, S. (1996). *Graphical Models*. Oxford Unversity Press Inc., New York.

Meinshausen, N. and Bühlmann, P. (2006). High-dimensional graph and variable selection with the lasso. *Annals of Statistics*, **34**, 1436-1462.

Nesterov, Y. (2012). Efficiency of coordinate descent methods on huge-scale optimizatioin problems. *SIAM Journal on Optimization*, **22**, 341-362.

Pang, H., Liu, H. and Vanderbei, R. (2014). The FASTCLIME package for linear programming and large-scale precision matrix estimation in R. *Journal of Machine Learning Research*, **15**, 489-493.

Peng, J., Wang, P., Zhou, N. and Zhu, J. (2009). Partial correlation estimation by Joint sparse regression models. *Journal of the American Statistical Association*, **104**, 735-746.

Shalev-Shwartz, S. and Tewari, A. (2011). Stochastic Methods for ℓ_1-regularized loss minimization. *Journal of Machine Learning Research*, **12**, 1865-1892.

Tang, H., Xiao, G., Behrens, C., Schiller, J., Allen, J., Chow, C. W., Suraokar, M., Corvalan, A., Mao, J., White, M. A., Wistuba, I. I., Minna, J. D. and Xie, Y. (2013). A 12-gene set predicts survival benefits from adjuvant chemotherapy in non-small cell lung cancer patients. *Clinical Cancer Research*. **19**, 1577-1586.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society B*, **58**, 267-288.

Vandenberghe, L., Boyd, S. and Wu, S. P. (1998). Determinant maximization with linear matrix inequality constraints. *SIAM Journal on Matrix Analysis and Applications*, **19**, 499-533.

Witten, D., Friedman, J. and Simon, N. (2011). New insights and faster computations for the graphical lasso. *Journal of Computational and Graphical Statistics*, **20**, 892-900.

Yu, D. and Lim, J. (2013). Introduction to general purpose GPU computing. *Journal of the Korean Data & Information Science Society*, **24**, 1043-1061.

Yuan, M. and Lin, Y. (2007). Model selection and estimation in the Gaussian graphical model. *Biometrika*, **94**, 19-35.

Journal of the Korean Data & Information Science Society
2016, **27**(2), 381–394

http://dx.doi.org/10.7465/jkdi.2016.27.2.381
한국데이터정보과학회지

BCDR algorithm for network estimation based on pseudo-likelihood with parallelization using GPU[†]

Byungsoo Kim[1] · Donghyeon Yu[2]

[1]Department of Statistics, Yeungnam University
[2]Department of Statistics, Keimyung University

Received 25 February 2016, revised 22 March 2016, accepted 23 March 2016

Abstract

Graphical model represents conditional dependencies between variables as a graph with nodes and edges. It is widely used in various fields including physics, economics, and biology to describe complex association. Conditional dependencies can be estimated from a inverse covariance matrix, where zero off-diagonal elements denote conditional independence of corresponding variables. This paper proposes a efficient BCDR (block coordinate descent with random permutation) algorithm using graphics processing units and random permutation for the CONCORD (convex correlation selection method) based on the BCD (block coordinate descent) algorithm, which estimates a inverse covariance matrix based on pseudo-likelihood. We conduct numerical studies for two network structures to demonstrate the efficiency of the proposed algorithm for the CONCORD in terms of computation times.

Keywords: BCD algorithm, graphical model, graphics processing unit, pseudo-likelihood, random permutation.

[†] This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT and Future Planning (NRF-2015R1C1A1A02036312).
[1] Assistant professor, Department of Statistics, Yeungnam University, Kyungsan 38541, Korea.
[2] Corresponding author: Assistant professor, Department of Statistics, Keimyung University, Daegu 42601, Korea. E-mail: dyu3@kmu.ac.kr