PURDUE UNIVERSITY · CS 59000-OPT
COMPUTATIONAL METHODS IN OPTIMIZATION

CS590OPT PROJECT REPORT

Nan Ding & Lin Yuan

May 28, 2012

1 Outline

Augmented Lagrangian Method (ALM) has gained great popularity in the recent decades in solving constrained optimization problems. The ALM is flexible in problem formulation, and has nice convergence properties as well. However, the nature of the ALM makes it hard to implement distributed computing. To overcome this disadvantage of the ALM, S. Boyd et al. [2] proposed the Alternating Direction Method of Multipliers (ADMM) as an alternative of the ALM. In this report, we are interested in comparing the Augmented Lagrangian Method and the Alternating Direction Method of Multipliers in solving the Thomson Problem.
In the ALM algorithm, the ρ-schedule is one of the important issues. Many nice heuristics has been proposed [4]. Interestingly, most papers that apply the ADMM algorithms including [2] seem to overlook the ρ-schedule of the ADMM. We propose a tentative ρ-schedule algorithm. The results seems to be slightly better than the one with ρ-fixed but no significant difference is observed.
We find that the ALM algorithm seems to work significantly better than the ADMM algorithm. We consider the reasons probably are: 1. The Thomson problem strongly correlates the individual terms in the objective function. By solving these terms separatively, the ADMM ignores the interactions between the terms and results in poor performance. 2. The Thomson problem is non-convex, and there seems to be little convergence guarantee for the ADMM on nonconvex optimization. 3. Better ρ-schedule scheme could be applied.
We also implemented the spherical graph embedding problem. But for the integrity of the report, we will mainly discuss the the Thomson problem.
The structure of the remaining report is as follows. Section 2 provides a basic gradient check of the Thomson problem as well as the spherical graph embedding problem. Section 3 mainly describes the ALM algorithm as well its ρ-schedule. Section 4 discusses the ADMM algorithms and provides a tentative ρ-schedule for the ADMM. The experiments are in Section 5. And a short conclusion is included in Section 6. All the implemented code, CMake file, as well as the scripts which produce all the experiment results can be downloadable through the following links:

- `http://www.cs.purdue.edu/homes/ding10/thompson.tar.bz2`.
- `http://web.ics.purdue.edu/~yuanl/download/CS590OPT/thompson.tar.bz2`.

2 Calculating and Checking Gradient

We first calculate the gradient and Jacobian of the objective function and constraint functions. And then we check the gradient for $f(\mathbf{x})$ and the augmented Lagrangian function $L_\rho(\mathbf{x},\mathbf{y})$ w.r.t $\mathbf{x}$.

$$\nabla L_{\rho,\mathbf{y}}(\mathbf{x}) = \nabla f(\mathbf{x}) - \nabla \mathbf{c}(\mathbf{x})^T \cdot (\mathbf{y} - \rho\, \mathbf{c}(\mathbf{x})) \tag{1}$$

2.1 Thomson problem

For the Thomson problem, we have: $\mathbf{x}_1 \dots \mathbf{x}_n \in \mathbb{R}^d, \mathbf{x} \triangleq \left(\mathbf{x}_1^T \dots \mathbf{x}_n^T\right)^T$. $\mathbf{c}(\mathbf{x}) \triangleq (c_1(\mathbf{x}) \dots c_n(\mathbf{x}))^T$.

$$\begin{array}{lll} \text{minimize}_{\mathbf{x}_1 \dots \mathbf{x}_n} & f(\mathbf{x}) = \sum_{i \neq j} \frac{1}{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2} & 1 \le i,j \le n \\ \text{subject to} & c_i(\mathbf{x}) \triangleq \|\mathbf{x}_i\|_2^2 - 1 = 0, & \forall i = 1 \dots n \end{array} \tag{2}$$

The gradient of $f(\mathbf{x})$ and Jacobian of $\mathbf{c}(\mathbf{x})$ are:

$$\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}_i} = \sum_{j\neq i} \frac{2(\mathbf{x}_j - \mathbf{x}_i)}{\|\mathbf{x}_j - \mathbf{x}_i\|_2^4} \tag{3}$$

$$\frac{\partial c_i(\mathbf{x})}{\partial \mathbf{x}_i} = 2\mathbf{x}_i \tag{4}$$

$$\frac{\partial c_i(\mathbf{x})}{\partial \mathbf{x}_j} = 0 \quad \text{for } j \neq i \tag{5}$$

2.2 Spherical graph embedding problem

For the spherical graph embedding problem, we have: $\mathbf{x}_1 \dots \mathbf{x}_n \in \mathbb{R}^d, \boldsymbol{X} \triangleq (\mathbf{x}_1 \dots \mathbf{x}_n)^T$. $\mathbf{c}(\boldsymbol{X}) \triangleq (c_1(\boldsymbol{X}) \dots c_n(\boldsymbol{X}))^T$. $\boldsymbol{L}$ is a Laplacian matrix of the adjacency matrix of a graph G.

$$\begin{array}{ll} \text{minimize}_{\boldsymbol{X}} & f(\boldsymbol{X}) = \text{trace}\left(\boldsymbol{X}^T \boldsymbol{L} \boldsymbol{X}\right) \\ \text{subject to} & c_i(\mathbf{x}) \triangleq \|\mathbf{x}_i\|_2^2 - 1 = 0, \quad \forall i = 1 \dots n \end{array} \tag{6}$$

In addition, we might also want $\tilde{\mathbf{c}}(\boldsymbol{X}) \triangleq \boldsymbol{X}^T \mathrm{e}_{n\times 1} = 0$.
The gradient of $f(\boldsymbol{X})$ and Jacobian of $\mathbf{c}(\boldsymbol{X}), \tilde{\mathbf{c}}(\boldsymbol{X})$ are:

$$\frac{\partial f(\boldsymbol{X})}{\partial \mathbf{x}_i} = \sum_{j\neq i} \boldsymbol{L}_{i,j}\mathbf{x}_j + 2\boldsymbol{L}_{i,i}\mathbf{x}_i \tag{7}$$

$$\frac{\partial c_i(\mathbf{x})}{\partial \mathbf{x}_i} = 2\mathbf{x}_i \quad i = 1 \dots n \tag{8}$$

$$\frac{\partial c_i(\mathbf{x})}{\partial \mathbf{x}_j} = 0 \quad i, j = 1 \dots n, i \neq j \tag{9}$$

$$\frac{\partial \tilde{c}_k(\mathbf{x})}{\partial \mathbf{x}_i} = \mathrm{e}_k \quad \text{for } i = 1 \dots n, k = 1 \dots d \tag{10}$$

where e_k is a d-dimensional vector that has 1 on the kth dimension and 0 otherwise.
In order to check the gradient ∇f and ∇L, we coded the above functions in MATLAB. We used the `gradientcheck` function in the `Poblano` toolbox. Please see `checkgradients.m` for the code. We checked the gradient with randomly initialized points on a 3d sphere. For the graph problem, we used a social network of 1200+ nodes from the AddHealth dataset [1], and then use snowball sampling to sample a sub-graph of the desired size to generate the Laplacian matrix $\boldsymbol{L}$.
The result for checking the gradients are plotted in Figure 1. It can be noted that for the Thomson problem, the finite difference method cannot yield a desirable gradient checking result when the number of particles is greater than 8. This indicates that the objective function becomes not very smooth as the number of points goes up. This observation also coincides with the difficulty we faced when tuning the parameter ρ_k.

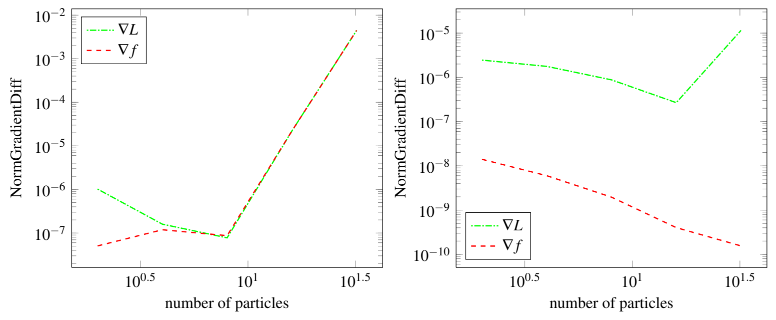

Figure 1: Left: Thomson problem, $d = 3$; Right: Spherical graph embedding problem, $d = 3$

3 Introduction to Methods of Multipliers

Through out this project, we try to address the constraint optimization problem

$$\begin{array}{ll} \text{minimize}_{\mathbf{x}} & f(\mathbf{x}) \\ \text{subject to} & \mathbf{c}(\mathbf{x}) = 0 \end{array} \tag{11}$$

where f is continuously differentiable. We also assume there are m equality constraints, and use $\mathbf{y}_1 \dots \mathbf{y}_m$ as their Lagrangian multipliers.

3.1 Lagrangian Method

Lagrangian methods work by alternating between minimizing the primal given dual and maximizing the dual given primal. The convergence in both dual and primal space require very strong assumptions[2]. However, it allows for decomposition of the primal variables, which may potentially lead to parallel optimization algorithms.
By introducing the Lagrangian multiplier $\mathbf{y}$, we can write the Lagrangian function as

$$L(\mathbf{x},\mathbf{y}) = f(\mathbf{x}) - \mathbf{y}^T \mathbf{c}(\mathbf{x}). \tag{12}$$

The normal Lagrangian method solves the constrained minimization problem by

$$\max_{\mathbf{y}} \min_{\mathbf{x}} L(\mathbf{x},\mathbf{y}) \tag{13}$$

which yields the following two step update,

- $\mathbf{x}^{k+1} = \text{argmin}_{\mathbf{x}} L(\mathbf{x},\mathbf{y}^k)$

- $y_i^{k+1} = y_i^k - \alpha^k \mathbf{c}(\mathbf{x})$

3.2 Augmented Lagrangian Method

In order to make the algorithm have better behaviors and easier to converge, one add penalized constraint terms to form the augmented Lagrangian. This term, however, will make the optimization non-decomposable.

$$L_\rho(\mathbf{x},\mathbf{y}) = f(\mathbf{x}) - \mathbf{y}^T \mathbf{c}(\mathbf{x}) + \frac{\rho}{2} \| \mathbf{c}(\mathbf{x}) \|^2. \tag{14}$$

Similar to Lagrangian methods, we obtain

$$\max_{\mathbf{y}} \min_{\mathbf{x}} L_{\rho_k}(\mathbf{x},\mathbf{y}) \tag{15}$$

which yields the following two step update,

$$\mathbf{x}^{k+1} = \underset{\mathbf{x}}{\text{argmin}} L_{\rho_k}(\mathbf{x},\mathbf{y}^k) \tag{16}$$

$$\mathbf{y}_i^{k+1} = \mathbf{y}_i^k - \rho_k \mathbf{c}\left(\mathbf{x}^k\right) \tag{17}$$

An important observation is that in order for the ALM to work well, the schedule of the penalizing factor ρ_k is very important. We follow [3, Algorithm 17.4] as our framework for implementing the ALM, and borrow expertise from [4, Algorithm 1]. See Algorithm 1.

3.2.1 Implementation

We used the libLBFGS package as our subroutine for solving (16). The parameters we used for the ALM method are: $\rho_0 = 10$, $\alpha = 0.5$, $\omega_* = 10^{-8}$, $\eta_* = 10^{-8}$, $\varepsilon_* = 10^{-16}$. We then tune the parameters τ, β and compare the experiment results. It turns out that a smaller β helps in obtaining better dual feasibility (better fit for the objective function). However, the parameter τ is important for obtaining a good schedule for ρ_k. On one hand, ρ_k has to grow larger to satisfy the primal feasibility. On the other hand, a large ρ_k will result in a line search error which will be described in detail in the following section. In Algorithm 1, we used the L_2 norm for convergence test, to be consistent with the libLBFGS. However, in reality, for the Thomson problem, lots of times the ALM will stop due to that ε_* is satisfied instead of ω_*.

Algorithm 1 Augmented Lagrangian Method

```
INPUT: Choose initial point $\mathbf{x}_0$, initial multipliers $\mathbf{y}_0$. Choose convergence parameter $\eta_*, \omega_*, \varepsilon_*$, choose $\rho_0 > 0$, $\tau > 1$, $0 < \alpha < 1$, $\beta > 0$.
INPUT: Objective function $f(\mathbf{x})$, constraints $\mathbf{c}(\mathbf{x})$.
OUTPUT: A stationary point $x$ that satisfies $\nabla L_{\rho_k}(\mathbf{x}, \mathbf{y}) \approx 0$.
$\eta_0 = 1/\rho_0^{\alpha}$.
$\omega_0 = 1/\rho_0$.
for $k = 0, 1, 2, \dots$ do
    Use LBFGS to find an approximate solution $\mathbf{x}^{k+1}$ to (16), that satisfies $\frac{\|\nabla L_{\rho_k, \mathbf{y}}(\mathbf{x})\|_2}{\|x^k\|_2} \leq \max(\omega_k, \omega_*)$.
    if $\|\mathbf{c}(\mathbf{x}^{k+1})\|_2 \leq \max(\eta_k, \eta_*)$ then
        if $\|\mathbf{c}(\mathbf{x}^{k+1})\|_2 \leq \eta_*$ and $\|\nabla L_{\rho_k}(\mathbf{x}, \mathbf{y}^k)\|_2 \leq \omega_*$ then
            Stop with approximate solution $\mathbf{x}^{k+1}$.
        end if
        if $\|\mathbf{c}(\mathbf{x}^{k+1})\|_2 \leq \eta_*$ and $\|f(\mathbf{x}^k) - f(\mathbf{x}^{k+1})\|_2 \leq \varepsilon_*$ then
            Stop with approximate solution $\mathbf{x}^{k+1}$.
        end if
        $\mathbf{y}^{k+1} = \mathbf{y}^k - \rho_k \mathbf{c}(\mathbf{x}^{k+1})$
        $\rho_{k+1} = \rho_k$
        $\eta_{k+1} = \eta_k / \rho_{k+1}^{\beta}$
        $\omega_{k+1} = \omega_k / \tau$.
    else
        $\mathbf{y}^{k+1} = \mathbf{y}^k$
        $\rho_{k+1} = \tau \cdot \rho_k$
        $\eta_{k+1} = 1/\rho_{k+1}^{\alpha}$.
        $\omega_{k+1} = 1/\rho_{k+1}$.
    end if
end for
```

3.2.2 ρ schedule for the ALM

There has been numerous literatures talking about the schedule of ρ_k for the ALM. Due to the lack of accessibility to professional software packages such as `LANCELOT` and `MINOS`, we turn our attention to a freely available software package `TANGO` and its relevant literature. Birgin and Martínez [5] introduced a specific schedule for ρ_k that may increase or decrease ρ_k in various different situations. Moreover, the subroutine for solving (16) is specialized and is guaranteed to achieve the specified tolerance level ω_k. In our case, the libLBFGS package which uses a More Thuente line search often stops due to a failure that the uncertainty region of the curvature condition drops below the machine precision. Going back to Algorithm 1, we find that our ρ_k is always increasing. We tried the following scheme for fixing the line search error, but couldn't find a satisfying solution:

- We ported the C version the ALM code to MATLAB, and used different optimization procedures to replace LBFGS. Using the conjugate gradient method, we found the same line search error in `Poblano toolbox`. When using the `fminunc` function, it returned with the fact that `ftol` is satisfied.
- For Thomson problem, we used different values of τ to hand tune the result for 100 points. And found out that when we increase τ carefully from 1.2 to 10, there are some certain value that makes ω_* satisfied. However, we cannot find a clear relation ship between τ and the number of particles.

4 Alternating Direction Method of Multipliers

The main purpose of applying the Alternating Direction Method of Multipliers (ADMM) [2] is to make the constrained optimization problems distributable by introducing some extra variables. For the purpose of solving the problems in this project, we are particularly interested in the following *Consensus* problem:

$$\begin{aligned} \min \ & \sum_j f_j(\mathbf{x}_j) \\ \text{s.t.} \ & \mathbf{x}_1 = \ldots = \mathbf{x}_j = \ldots = \mathbf{x}_n = \mathbf{z}. \end{aligned} \tag{18}$$

with the Lagrangian function,

$$L(\mathbf{x},\mathbf{z},\mathbf{y}) = \sum_j f_j(\mathbf{x}_j) + \sum_j \mathbf{y}_j^T(\mathbf{x}_j - \mathbf{z}) \tag{19}$$

The primal feasibility of the problem is by taking gradient of $L(\mathbf{x},\mathbf{z},\mathbf{y})$ with respect to $\mathbf{y}$,

$$\nabla_{\mathbf{y}} L(\mathbf{x},\mathbf{z},\mathbf{y}) = 0 \Longrightarrow \mathbf{x}_j = \mathbf{z} \quad \forall j \tag{20}$$

The dual feasibility of the problem is by taking gradient of $L(\mathbf{x},\mathbf{z},\mathbf{y})$ with respect to $\mathbf{x}$ and $\mathbf{z}$,

$$\nabla_{\mathbf{x}_j} L(\mathbf{x},\mathbf{z},\mathbf{y}) = 0 \Longrightarrow \nabla_{\mathbf{x}_j} f_j(\mathbf{x}_j) + \mathbf{y}_j = 0 \quad \forall j \tag{21}$$

$$\nabla_{\mathbf{z}} L(\mathbf{x},\mathbf{z},\mathbf{y}) = 0 \Longrightarrow \sum_j \mathbf{y}_j = 0 \tag{22}$$

Using the ALM with parameter ρ, we can formulate the above problem into,

$$L_\rho(\mathbf{x},\mathbf{z},\mathbf{y}) = \sum_j f_j(\mathbf{x}_j) + \sum_j \mathbf{y}_j^T(\mathbf{x}_j - \mathbf{z}) + \sum_j \frac{\rho}{2} \| \mathbf{x}_j - \mathbf{z} \|_2^2 \tag{23}$$

Based on (23), we design the following three-step algorithm to optimize $\mathbf{z}$, $\mathbf{x}$ and $\mathbf{y}$ iteratively,

- Update $\mathbf{z}$ ($\mathbf{z}$-step):
  - $\mathbf{z}^{k+1} = \frac{1}{n}\sum_j \left(\mathbf{x}_j^k + \frac{1}{\rho}\mathbf{y}_j^k \right)$
- Update $\mathbf{x}$ ($\mathbf{x}$-step):
  - $\mathbf{x}_j^{k+1} = \text{argmin}_{\mathbf{x}_j} \left(f_j(\mathbf{x}_j) + \left\langle \mathbf{y}_j^k, \mathbf{x}_j - \mathbf{z}^{k+1} \right\rangle + \frac{\rho}{2} \| \mathbf{x}_j - \mathbf{z}^{k+1} \|_2^2 \right)$
- Update $\mathbf{y}$ ($\mathbf{y}$-step):
  - $\mathbf{y}_j^{k+1} = \mathbf{y}_j^k + \rho(\mathbf{x}_j^{k+1} - \mathbf{z}^{k+1})$.

One important property of the above algorithm is that after $\mathbf{x}$-step, $(\mathbf{x}^{k+1}, \mathbf{z}^{k+1}, \mathbf{y}^k)$ satisfy,

$$\nabla_{\mathbf{x}_j} f_j(\mathbf{x}_j^{k+1}) + \mathbf{y}_j^k + \rho(\mathbf{x}_j^{k+1} - \mathbf{z}^{k+1}) = 0$$

Because that $\mathbf{y}_j^{k+1} = \mathbf{y}_j^k + \rho(\mathbf{x}_j^{k+1} - \mathbf{z}^{k+1})$, we have,

$$\nabla_{\mathbf{x}_j} f_j(\mathbf{x}_j^{k+1}) + \mathbf{y}_j^{k+1} = 0$$

Therefore, $(\mathbf{x}^{k+1}, \mathbf{z}^{k+1}, \mathbf{y}^{k+1})$ always satisfy (21). However, $(\mathbf{x}^{k+1}, \mathbf{z}^{k+1}, \mathbf{y}^{k+1})$ may not satisfy (20) and (22) unless the following tol_1^{k+1} and tol_2^{k+1} goes to zero,

$$\begin{aligned} tol_1^{k+1} &\triangleq \sum_j \|\nabla_{\mathbf{y}_j} L_\rho(\mathbf{x}^{k+1}, \mathbf{z}^{k+1}, \mathbf{y}^{k+1})\|_2 \\ &= \sum_j \| \mathbf{x}_j^{k+1} - \mathbf{z}^{k+1} \|_2 \\ &= \sum_j \| \mathbf{y}_j^{k+1} - \mathbf{y}_j^k \|_2 \\ tol_2^{k+1} &\triangleq \|\nabla_{\mathbf{z}} L_\rho(\mathbf{x}^{k+1}, \mathbf{z}^{k+1}, \mathbf{y}^{k+1})\|_2 \\ &= \|\rho \sum_j \left(\mathbf{z}^{k+1} - \mathbf{x}_j^{k+1} \right) + \sum_j \mathbf{y}_j^{k+1} \|_2 \\ &= n\rho \| \mathbf{z}^{k+2} - \mathbf{z}^{k+1} \|_2 \end{aligned}$$

Obviously, if tol_1^{k+1} goes to 0, then $\mathbf{y}^{k+1}$ converges and $\mathbf{x}_j^{k+1} \to \mathbf{z}^{k+1}$. If tol_2^{k+1} goes to 0, then $\mathbf{z}^{k+1}$ converges and $\sum_j \mathbf{y}_j^{k+1} \to 0$. Therefore, (20) and (22) satisfy.

4.1 Parallelization Using Intel Threading Building Blocks

Inside each $\mathbf{x}$-step (similar for $\mathbf{z}$-step, as well as $\mathbf{y}$-step), the variable pair $\mathbf{x}_i$ that got updated is independent of the other variables that got updated in the $\mathbf{x}$-step. Therefore, inside each $\mathbf{x}$-step (again similar for $\mathbf{z}$-step, as well as $\mathbf{y}$-step), the updates can be parallelized.
There are a lot of ways to do parallel computing. In this project, we utilize the Intel Threading Building Blocks (IntelTBB) [1] . The IntelTBB is a convenient multi-thread computational tool to do parallelization in multi-core computers, which we consider are very commonly used now.
We mainly use the `parallel_for` to update the variables $\mathbf{x}$. Meanwhile, in order to compute tol_1 and tol_2, we use the `parallel_reduce` to update the variables $\mathbf{y}$ and $\mathbf{z}$.
To use the IntelTBB, simply download it from its website, and set the environment variables by:

- `source bin/tbbvars.sh [your computer architecture]`

More details and user manuals are available online.

4.2 ρ Schedule for the ADMM

Unfortunately, in many literatures [2] that use the ADMM algorithm, people seem to overlook one important issue in the ALM algorithm: the schedule of ρ. In most of the existing ADMM algorithms, ρ are always fixed and the setting of the ρ are not emphasized. In our experiment, we find that the setting of ρ is very important for the performance of the algorithm.
There are a couple of differences between the ADMM and the ALM algorithm in designing the ρ schedule. In the ALM the dual feasibility is always satisfied, and therefore ρ is only adjusted based on the convergence the primal feasibility. In addition, the update of $\mathbf{x}$ based on (16) is sometimes hard to be optimized within a small tolerance. Some nice heuristics, e.g. [4], will apply.
On the other hand, in the ADMM, the update of $\mathbf{x}$ is very simple since they are decomposed into small problems. The difficulty mainly comes from the fact that both primal and dual feasibility are not satisfied. We propose the following scheme of ρ scheduling as a tentative extension of Algorithm 1. We will later combine with the experiments to discuss why we think the following extension might be reasonable.

4.3 Thomson Problem

Based on the above discussion, we formulated the Thomson problem as

$$\begin{aligned} \min \quad & \frac{2}{n(n-1)} \sum_{i \neq j} \frac{1}{\| \mathbf{x}_{ij} - \mathbf{x}_{ji} \|_2^2} + \sum_i \delta(\| \mathbf{x}_{ii} \|^2 = 1) \\ \text{s.t.} \quad & \mathbf{x}_{i1} = \ldots = \mathbf{x}_{ii} = \ldots = \mathbf{x}_{in} = \mathbf{z}_i, \ \forall i. \end{aligned} \tag{24}$$

with the Lagrangian function,

$$L(\mathbf{x}, \mathbf{z}, \mathbf{y}) = \underbrace{\frac{2}{n(n-1)} \sum_{i \neq j} \frac{1}{\| \mathbf{x}_{ij} - \mathbf{x}_{ji} \|_2^2} + \sum_i \delta(\| \mathbf{x}_{ii} \|^2 = 1)}_{f(\mathbf{x})} + \sum_i \sum_j \mathbf{y}_{ij}^T (\mathbf{x}_{ij} - \mathbf{z}_i) \tag{25}$$

where n is the number of charged particles on the sphere.
The primal feasibility of the problem is by taking gradient of $L(\mathbf{x}, \mathbf{z}, \mathbf{y})$ with respect to $\mathbf{y}$,

$$\mathbf{x}_{ij} = \mathbf{z}_i \quad \forall i, j \tag{26}$$

The dual feasibility of the problem is by taking gradient of $L(\mathbf{x}, \mathbf{z}, \mathbf{y})$ with respect to $\mathbf{x}$ and $\mathbf{z}$,

$$\nabla_{\mathbf{x}_{ij}} f(\mathbf{x}) + \mathbf{y}_{ij} = 0 \quad \forall i, j \tag{27}$$

$$\sum_j \mathbf{y}_{ij} = 0 \quad \forall i \tag{28}$$

[1] `http://threadingbuildingblocks.org/`

Algorithm 2 Alternating Direction Method of Multipliers

```
INPUT: Choose initial point $\mathbf{x}_0$, $\mathbf{z}_0$, $\mathbf{y}_0$. Choose convergence parameter $\eta_*$, choose $\rho_0 > 0$, $\tau \geq 1$, $0 < \alpha < 1$, $\beta > 0$ and $\eta_0 = 1$.
INPUT: Objective function $f(\mathbf{x})$, constraints $\mathbf{c}(\mathbf{x})$.
OUTPUT: A stationary point $(\mathbf{x}, \mathbf{z}, \mathbf{y})$ that satisfies $\nabla L_{\rho_k}(\mathbf{x}, \mathbf{z}, \mathbf{y}) \approx 0$.
for $k = 0, 1, 2, \ldots$ do
  Use $\mathbf{z}$-step to update $\mathbf{z}$
  Use $\mathbf{x}$-step to update $\mathbf{x}$
  Use $\mathbf{y}$-step to update $\mathbf{y}$
  if $tol_1 \leq \eta_k$ and $tol_2 \leq \eta_k$ then
    if $tol_1 \leq \eta_*$ and $tol_2 \leq \eta_*$ then
      Stop with approximate solution $(\mathbf{x}^{k+1}, \mathbf{z}^{k+1}, \mathbf{y}^{k+1})$.
    end if
    $\rho_{k+1} = \rho_k$
    $\eta_{k+1} = \eta_k / \rho_{k+1}^{\beta}$
  end if
  if $tol_1 \geq \eta_k$ and $tol_2 \leq \eta_k$ then
    $\rho_{k+1} = \rho_k \cdot \tau$
    $\eta_{k+1} = 1/\rho_{k+1}^{\alpha}$.
  end if
  if $tol_1 \leq \eta_k$ and $tol_2 \geq \eta_k$ then
    $\rho_{k+1} = \rho_k / \tau$
    $\eta_{k+1} = 1/\rho_{k+1}^{\alpha}$.
  end if
  if $tol_1 \geq \eta_k$ and $tol_2 \geq \eta_k$ then
    $\rho_{k+1} = \rho_k$
    $\eta_{k+1} = 1/\rho_{k+1}^{\alpha}$.
  end if
end for
```

Using the ALM with parameter ρ, we can formulate the above problem into,

$$L_\rho(\mathbf{x}, \mathbf{z}, \mathbf{y}) = \frac{2}{n(n-1)} \sum_{i \neq j} \frac{1}{\| \mathbf{x}_{ij} - \mathbf{x}_{ji} \|_2^2} + \sum_i \delta(\| \mathbf{x}_{ii} \|^2 = 1) + \sum_i \sum_j \mathbf{y}_{ij}^T (\mathbf{x}_{ij} - \mathbf{z}_i) + \sum_i \sum_j \frac{\rho}{2} \| \mathbf{x}_{ij} - \mathbf{z}_i \|_2^2 \quad (29)$$

Based on (29), we design the following three-step algorithm to optimize $\mathbf{z}$, $\mathbf{x}$ and $\mathbf{y}$ iteratively,

- Update $\mathbf{z}$ ($\mathbf{z}$-step):
  - $\mathbf{z}_i^{k+1} = \frac{1}{n} \sum_j \left(\mathbf{x}_{ij}^k + \frac{1}{\rho} \mathbf{y}_{ij}^k \right)$
- Update $\mathbf{x}$ ($\mathbf{x}$-step):
  - Update $\mathbf{x}_{ij}$, for $i \neq j$:

$$\mathbf{x}_{ij}^{k+1}, \mathbf{x}_{ji}^{k+1} = \underset{\mathbf{x}_i, \mathbf{x}_j}{\operatorname{argmin}} \left(\frac{2}{n(n-1) \| \mathbf{x}_i - \mathbf{x}_j \|_2^2} + \left\langle \mathbf{y}_{ij}^k, \mathbf{x}_i - \mathbf{z}_i^{k+1} \right\rangle + \frac{\rho}{2} \| \mathbf{x}_i - \mathbf{z}_i^{k+1} \|_2^2 + \left\langle \mathbf{y}_{ji}^k, \mathbf{x}_j - \mathbf{z}_j^k \right\rangle + \frac{\rho}{2} \| \mathbf{x}_j - \mathbf{z}_j^k \|_2^2 \right)$$

  - Update $\mathbf{x}_{ii}$:

$$\begin{aligned} \mathbf{x}_{ii}^{k+1} =& \underset{\mathbf{x}_i}{\operatorname{argmin}} \left(\delta(\| \mathbf{x}_i \|^2 = 1) + \left\langle \mathbf{y}_{ii}^k, \mathbf{x}_i - \mathbf{z}_i^{k+1} \right\rangle + \frac{\rho}{2} \| \mathbf{x}_i - \mathbf{z}_i^{k+1} \|_2^2 \right) \\ =& \prod_{\| \cdot \|^2 = 1} (\mathbf{z}_i^{k+1} - \frac{1}{\rho} \mathbf{y}_{ii}^k) \\ =& (\mathbf{z}_i^{k+1} - \frac{1}{\rho} \mathbf{y}_{ii}^k) / \| \mathbf{z}_i^{k+1} - \frac{1}{\rho} \mathbf{y}_{ii}^k \|_2 \end{aligned}$$

- Update $\mathbf{y}$ ($\mathbf{y}$-step):
  - $\mathbf{y}_{ij}^{k+1} = \mathbf{y}_{ij}^{k} + \rho(\mathbf{x}_{ij}^{k+1} - \mathbf{z}_{i}^{k+1})$.

5 Experiments

5.1 Experiments on the ALM

In the first part of the experiments, we try to address the issue of:

- How the parameters β, α, τ affect the schedule of ρ in Algorithm 1.
- What's the relationship between the augmented Lagrangian and the original objective function.
- How does the difficulty of the Thomson problem change as the number of particles goes up.

During the experiment, we find that α doesn't affect the ALM much due to the rounding error in the line search routine discussed in Section 3.2.2: the ω_k cannot be always satisfied. Therefore, the subroutine doesn't exactly go as the algorithm expects. This is a major drawback of our implementation, which may affect the convergence property of Algorithm 1. Therefore, we only vary β and τ for the experiment. We selected different β's from $\{0.2, 0.5, 0.9, 1.5, 2.0\}$, and different τ's from $\{1.5, 5.0, 10.0, 20.0\}$. We solved the Thomson problem for a scale of $n = \{2, 4, 8, 16, \ldots 1024\}$ particles in 3 dimension. For the convenience of plotting, we re-scaled $f(\mathbf{x})$ with a factor of $\frac{2}{n(n-1)}$, and $\|\mathbf{c}(\mathbf{x})\|_2$ with a factor of $\frac{1}{\sqrt{n}}$. We have made the following observations according to the results of our experiment:

- From $\|\mathbf{c}(\mathbf{x})\|_2$ of Figure2, we can see that the larger β is, the more aggressive Algorithm 1 tries to satisfy the constraints.
- From $\|\mathbf{c}(\mathbf{x})\|_2$ of Figure4, we can see that the larger τ is, the more aggressive Algorithm 1 tries to satisfy the constraints.
- From $\|\nabla L(\mathbf{x})\|_2$ of Figure5, we can see that as the number of particles grow larger, it becomes harder to satisfy the dual feasibility.
- However, we cannot see obvious relationship between the gradient of $g(\mathbf{x})$ and $\nabla L(\mathbf{x})$, different parameters tend to get the same solution, but not being able to satisfy $g(\mathbf{x}) \approx 0$ very well. We tend to think this is the major issue of Algorithm 1 due to the line search issue.

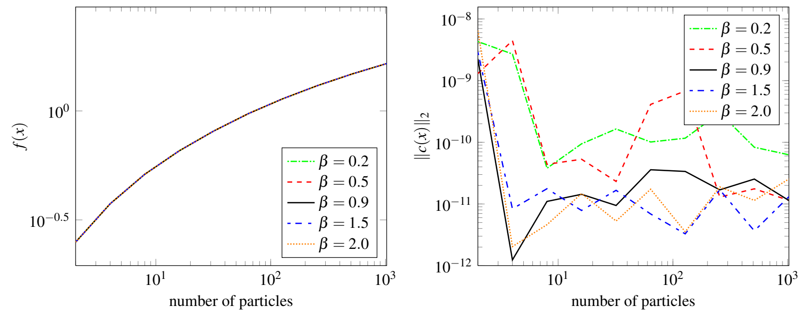

Figure 2: Left: objective value, Right: constraint, for different β's, when $\tau = 5.0$.

5.2 Experiments on the ADMM

In the second part of the experiment, we will apply the ADMM algorithm on the Thomson problem. Our experiments are designed to answer the following questions:

- How does the ρ setting affect the performance?
- Does ρ schedule improve the performance?

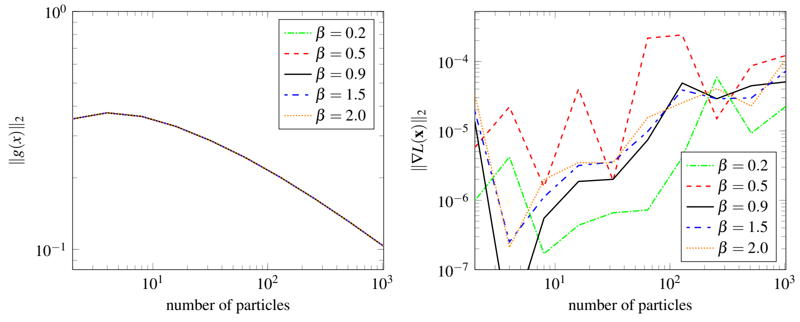

Figure 3: Left: gradient of objective, Right: gradient of augmented Lagrangian, for different β's, when $\tau = 5.0$.

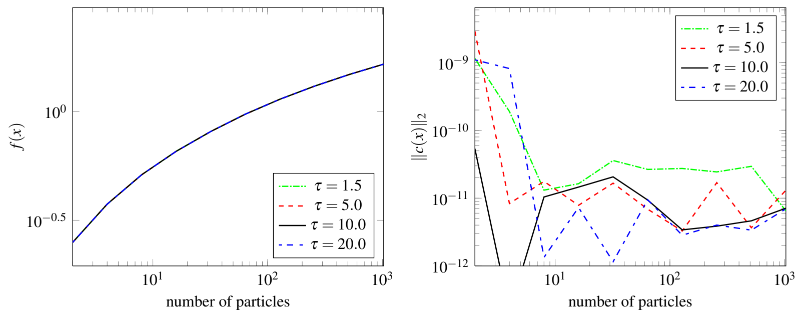

Figure 4: Left: objective value, Right: constraint, for different τ's, when $\beta = 1.5$.

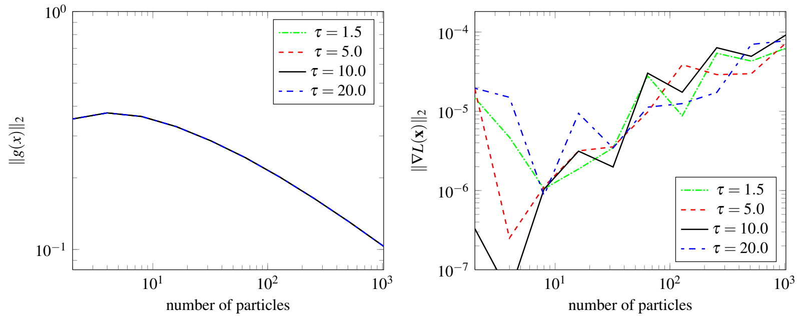

Figure 5: Left: gradient of objective, Right: gradient of augmented Lagrangian, for different τ's, when $\beta = 1.5$.

- What is the speedup using multi-core machines?
- Is the ADMM faster than the ALM because of parallelization in the Thomson problem?

5.2.1 How does the ρ setting affect the performance?

To answer the first question, we turn off the ρ-schedule by setting τ to be equal to 1.0. We plot $f(x)$, $\|c(x)\|_2$ as well as tol_1 and tol_2 using different values of ρ from 0.1 to 1000. We test the Thomson problem of $\{2,4,8,\ldots,512,1024\}$ particles. The algorithm stops if either it converges with $\eta = 1e-6$ or a maximum of 5000 iterations are run. All the experiments are done on the Rossmann Cluster. We report the results in Figure 6.

First of all, we see that ρ which is smaller than 1, e.g. 0.1, seems to work badly. It cannot find a good objective value and cannot satisfy the constraint. Secondly, large ρ's can always satisfy the constraint, but they brings poor objective values. Thirdly, as ρ gets larger, tol_1 usually goes lower, but tol_2 gets higher.

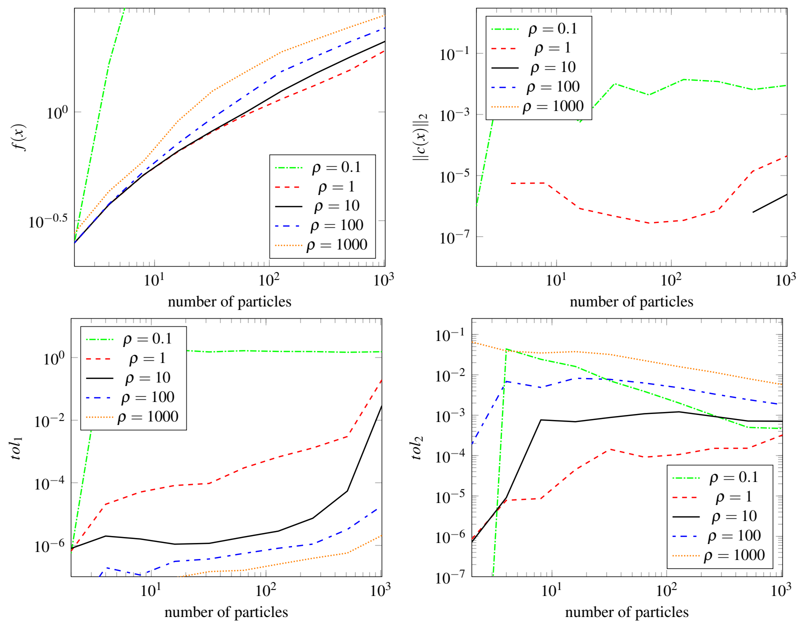

Figure 6: $f(x)$,$\|c(x)\|_2$, tol_1, tol_2, using different ρ's.

5.2.2 Does ρ schedule improve the performance?

Based on the results by differentρ setting, it seems to be reasonable to schedule the ρ variables as following

- if $tol_1^k > \eta_k$ and $tol_2^k < \eta_k$, we should increase ρ;
- if $tol_1^k < \eta_k$ and $tol_2^k > \eta_k$, we should decrease ρ;
- otherwise, we just keep ρ fixed.

Our algorithm is designed using the above ρ-scheduling. We initialize $\rho = 10$, and test the ρ-scheduling using different parameters $\tau \in \{1.00, 1.01, 1.1, 2, 10\}$ and $\alpha \in \{0.1, 0.5, 0.9\}$ and $\beta \in \{0.01, 0.1, 1, 2\}$. We test the Thomson problem of $\{2,4,8,\ldots,512,1024\}$ particles. The algorithm stops if either it converges with $\eta = 1e-6$ or a maximum of 5000 iterations are run. All the experiments are done on the Rossmann Cluster.

The sweep of these three parameters leads to a total number of $5 \times 3 \times 4$ configurations. In order to show our results, we first choose the *lucky* settings which minimize the averaged objective value of the problems of $\{2,4,8,\ldots,512,1024\}$ particles. The lucky setting turns out to be $\tau = 1.1$, $\alpha = 0.5$ and $\beta = 2$. We then plot the curves of different τ, α and β respectively with the other two parameter fixed to be the lucky settings. We report the results in Figure 7, Figure 8 and Figure 9.

Firstly, we find that τ cannot be set to be very large for the ADMM. The reason is that the ADMM usually require a lot of steps to converge, and large τ may make the ρ grows/decreases too quickly in the early stage of the algorithm. Secondly, it appears that large α or small β also leads to poor performances. If β is too slow, then the η_k decrease too slowly and the ρ-scheduling is inactive in the most of time. If α is too large, then η_k might be too small which makes ρ change too rapidly. But overall, it seems that the ρ-schedule does not seem to lead to significant difference of the performance.

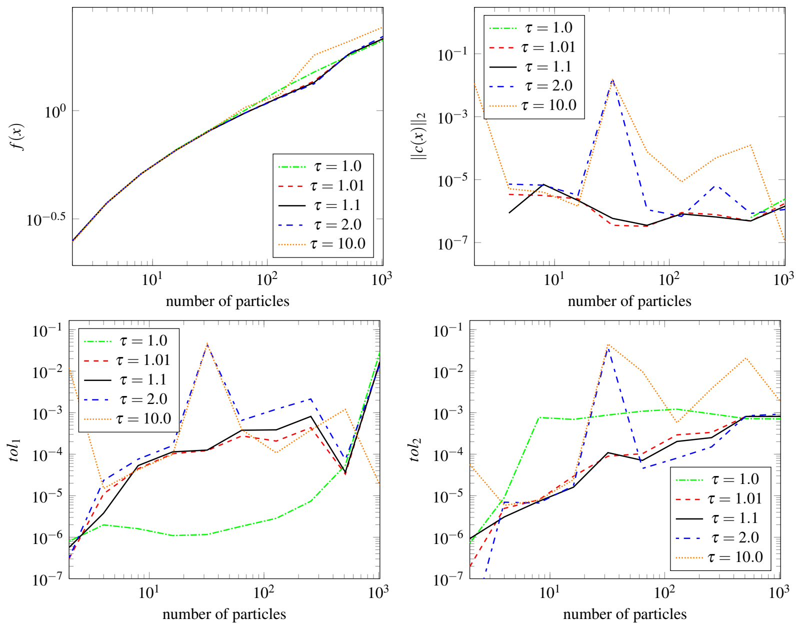

Figure 7: $f(x)$,$\|c(x)\|_2$, tol_1, tol_2, using different τ's.

5.2.3 What is the speedup using multi-core machines?

To see the parallelization performance of the algorithm, we test the code on the problem with $\{64, 128, 256, 512, 1024\}$ particles. We run the code using 1-core, 2-core, 3-core, and 4-core using the Rossmann Cluster. It turns out that the parallel speedup is close to be linear with the rate at about 0.75.

5.2.4 Is the ADMM better than the ALM because of parallelization in the Thomson problem?

In the Thomson problem, the answer is no. We take the 1024 particle problem as an example. Using the lucky setting, for the ADMM with 4-core, it runs 5000 iterations in 1953 seconds, and reaches $f(x) = 2.15$ while $\|c(x)\|_2 \simeq 1e-6$. In the ALM, it runs 24 iterations in 2806 seconds and reaches $f(x) = 1.65$ while $\|c(x)\| \simeq 1e-8$. The result of the ALM appears to be significantly better than that of the ADMM. We plot the change of $f(x)$ during CPUtime between the ALM and the ADMM.
In each iteration, the ADMM performs significantly faster than the ALM because each decomposed problem is very easy to solve. However, the ADMM algorithm appears to converge extremely slowly to high accuracy. [2] In the Thomson problem, each pair $(\mathbf{x}_{ij}, \mathbf{x}_{ji})$ is strongly correlated to all the other pairs which are dependent on either particle i or j. By decomposing the problem, the ADMM more or less ignores the interactions between the particles. The ignorance results in the overall problem being poorly solved by the ADMM, especially when the number of particles is large. Furthermore, the Thomson problem is a nonconvex optimization problem. Although there is convergence guarantee (to a certain KKT point) for the ALM algorithm, there seems to be little convergence guarantee for the ADMM algorithm on nonconvex optimization problems. We designed a tentative ρ-schedule scheme but it does not show significant better performance. Better ρ-schedule for the ADMM may as well improve its performance.

[2]Later we find a similar comment from Page 17 [2].

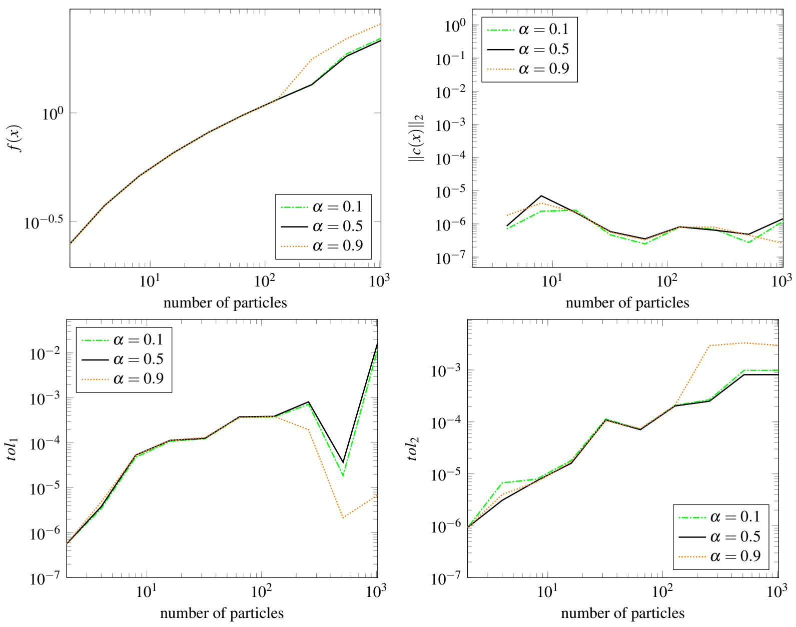

Figure 8: $f(x)$,$\|c(x)\|_2$, tol_1, tol_2, using different α's.

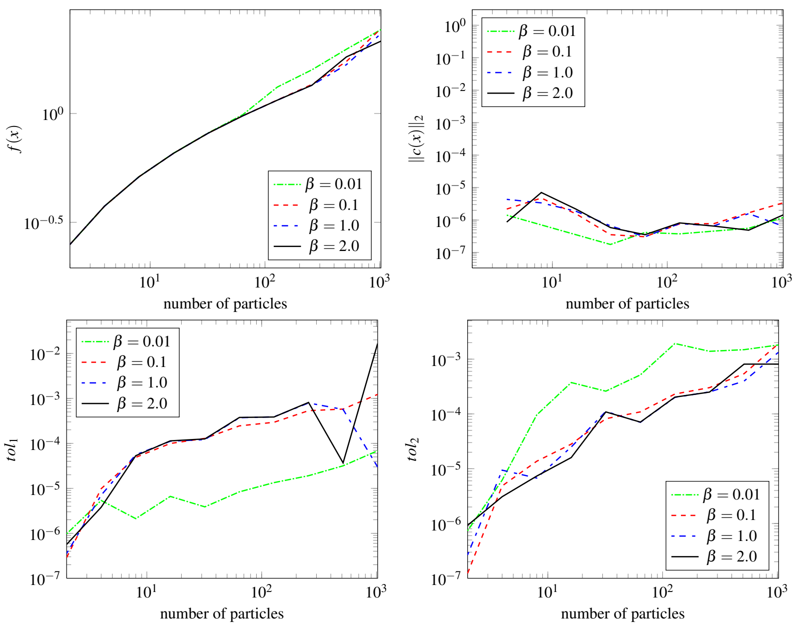

Figure 9: $f(x)$,$\|c(x)\|_2$, tol_1, tol_2, using different β's.

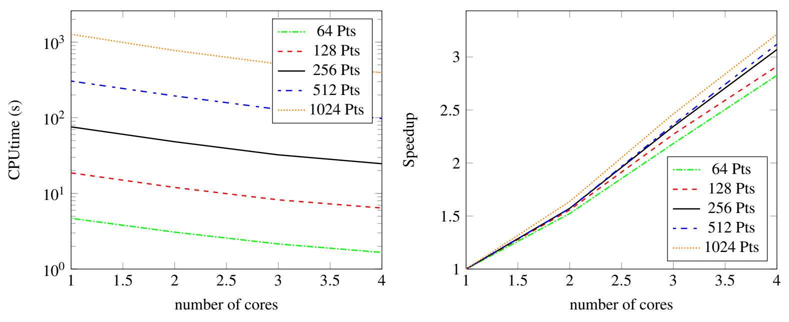

Figure 10: Left: CPU time, Right: Speedup, using different number of cores after 1000 iterations.

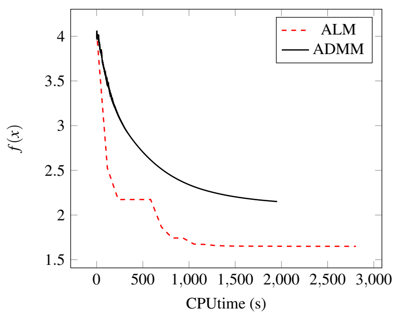

Figure 11: Comparison of the $f(x)$ between the ALM and the ADMM on the 1024-particle Thomson problem.

5.3 Experiments on spherical graph embedding problem

We have programmed the ALM for solving the graph embedding problem, but to maintain the integrity of this report, we only report briefly 1 sample result. We snowball-sampled the social network in [1] to 1024 nodes. It is an undirected unit weight graph. ALM converged with around 10 minutes with a result illustrated in Figure 12.

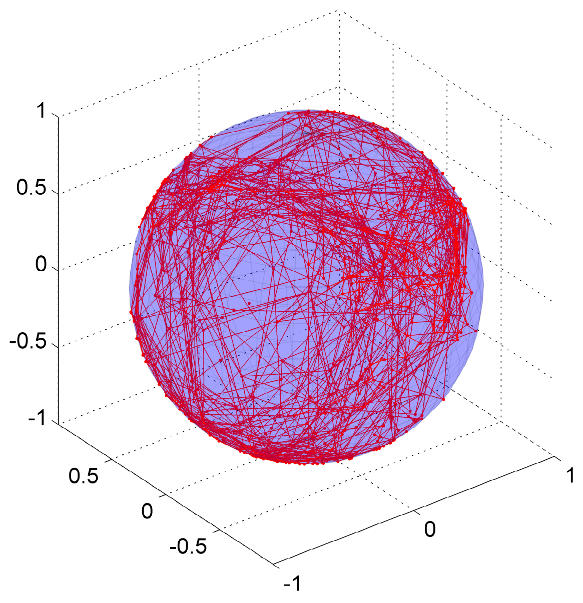

Figure 12: Embedding a 1024 node graph onto a 3d sphere

6 Conclusion

We have investigated augmented Lagrangian methods (ALM) and alternating direction method of multipliers (ADMM) for solving spherical constrained optimization problems. By implementing these methods by ourselves, we significantly improved our understanding of these methods. ALM is well suited for large-scale constraint optimization as evidenced by the successful deployment of the commercial softwares such as `LANCELOT` and `MINOS`, while the schedule of ρ_k and special techniques in treating a sparse Jacobian of $\mathbf{c}(\mathbf{x})$ is essential to the efficiency of these commercial systems. Given that we could not find a perfect schedule of ρ_k and our subroutine for the intermediate update is problematic, ALM still gives reasonable result for the Thomson problem. ADMM enables a decomposable objective function for parallel optimization and yields shorter unit-iteration time. However, the ADMM appears to converge very slowly to high accuracy. The reason may be three folds: one, the decomposed sub-problem ignores the inherent interaction between particles on the sphere; two, there is no convergence guarantee for the ADMM on the nonconvex optimization problems; three, there lacks investigation of the scheduling of ρ in the ADMM literature, which is worthwhile to look into as a future research topic.

References

[1] K. Harris. The national longitudinal study of adolescent health (addhealth), waves i & ii, 1994–1996; wave iii, 2001–2002, 2008.

[2] 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*, 2011.

[3] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, 2006.

[4] M. P. Friedlander. *A Globally Convergent Lineraly Constrained Lagrangian Method For Nonlinear Optimization*. PhD thesis, Stanford University, 2002.

[5] E. G. Birgin and J. M. Martínez. Improving ultimate convergence of an augmented lagrangian method. *Optimization Methods Software*, 23(2):177–195, April 2008. ISSN 1055-6788.