Project Assignment 2, Solving Large-Scale Optimization Problems with ADMM

Tuomas Porkamaa, Anton Saukkonen
Aalto University School of Science, Department of Mathematics and Systems Analysis
Aalto University School of Science, Department of Mathematics and Systems Analysis
{*tuomas.porkamaa, anton.saukkonen*}*@aalto.fi*

1 Background

The aim of this project is to implement Alternating Direction Method of Multipliers (ADMM) algorithm to solve large-scale linear optimization problems with decomposable structure. Specifically, the considered linear problems are randomly generated instances of stochastic capacity expansion problems. In the next sections, we will present the general form of ADMM algorithm and some of its basic properties, following by the detailed guidance how to apply ADMM to the problem at hand. The final chapter collects the main results of the experiments.

1.1 ADMM

The ADMM algorithm can be considered as a distributed version of augmented Lagrangian method of multipliers (ALMM), a method that yields very good convergence results using finite penalty terms and without strict demands on objective function. Constraining ourselves on problem formulations with only equality constraints, the ALMM solves problems formed as

$$\min_x \{f(x) : h(x) = 0\} \tag{1}$$

We seek the solution for primal problem via amounting to strong duality and iteratively solving unconstrained augmented Lagrangian subproblems by following a strategy derived from dual ascent (DA) method to update variables in primal and dual space given as

$$\begin{aligned} x^{k+1} &= \operatorname*{argmin}_x L_\rho(x, v^k) \\ v^{k+1} &= v^k + \rho h(x^{k+1}) \end{aligned} \tag{2}$$

where L_ρ is an augmented Lagrangian of (1), $L_\rho(x, v) = f(x) + v^T h(x) + \frac{\rho}{2} \sum_i h_i(x)^2$, $\rho > 0$ is a penalty term and x and v are primal and dual variables, respectively. The difference between ALMM and DA is that DA updates the primal variables by minimizing the unaugmented Lagrangian L_0. Both methods solve the corresponding dual problem $\max_v \{\inf_x L_\rho(x, v)\}$ by gradient ascent, but in ALMM fixed step size ρ is used, because it retains the optimality conditions in dual space between iterations. Overall, the additional quadratic penalty parameter allow ALMM to converge under more general conditions compared to DA.

The ADMM is essentially similar to ALMM, but where the corresponding variable update steps (2) are modified to support decentralized optimization. Furthermore, the decentralized methods could be applied if the problem formulation yield a specific decomposable structure, which basically

means that we could divide the set of primal variables into separate blocks (x, y), and hence the iterative updates could be calculated independently. This decomposability is especially a beneficial feature when we are dealing with large problems when the memory resources and computing time are in a crucial role, because it allows the optimization algorithm to be executed on multiple computing units or processor cores.

The basic ADMM algorithm is suited to solve problems in the form

$$\min_{x,y} \{f(x) + g(y) : Ax + By - c = 0\} \tag{3}$$

where the objective and constraints are now separated with respect to primal variables x and y. the properties of objectives f and g that imply the convergence of ADMM are discussed in chapter 1.2, but in practice, their assumptions are flexible enough for most applications. There also exist several application-dependent tricks to transform the original problem constraints to support the ADMM-form.

Even though the ADMM-format yields separable objective functions and constraints, because of the quadratic penalty term in L_ρ, it is impossible to divide the corresponding augmented Lagrangian to separable components $L_\rho(x, v)$ and $L_\rho(y, v)$, solve them independently on separate branches, and collect the results to master branch. However, it is sorta possible to recover the separability by relying on coordinate descent approach and solve the augmented Lagrangian for x with fixed y, and vice versa, in a blockwise order and hope for convergence. This procedure yields the following ADMM update equations for separated primals and dual variables.

$$\begin{aligned} x^{k+1} &= \operatorname*{argmin}_x L_\rho(x, y^k, v^k) \\ y^{k+1} &= \operatorname*{argmin}_y L_\rho(x^{k+1}, y, v^k) \\ v^{k+1} &= v^k + \rho(Ax^{k+1} + By^{k+1} - c) \end{aligned} \tag{4}$$

It is also worth of noting that it is often possible to separate the primal variables further to component level, when the primal objective becomes $f(x) = \sum_i f_i(x_i)$, as in the case of the given project assignment. Other considerable detail is related to the primal problem formulation (3), where generally only equality constraints could be used. There are several tricks to evade this restriction, where the most practically simple approach is to simply utilize constrained solver for primal variable updates. This was the approach we used in this project assignment to handle linear inequality constraints. Other relatively straightforward methods include the classic introduction of slack variables s, where we replace $Ax - b \leq 0$ by $Ax - b + s = 0$ and add s to the pool of optimization variables. For feasibility, we would like $s \geq 0$, but this would include another inequality to the constraints. To evade this, we could introduce a penalty function

$$g_2(s) = \begin{cases} 0, & \text{if } s \geq 0 \\ \infty, & \text{otherwise} \end{cases} \tag{5}$$

and reformulate problem statement in ADMM form as $\min_{x,s} \{f(x) + g_2(s) : Ax + s - c = 0\}$. This brings up another benefit of ADMM, where objective functions are allowed to take the value ∞

as it will be explained in following chapters. Similar approach could be used, for example, in any constrained convex optimization problems, where the indicator function is specified over convex set C.

1.2 Convergence

In this section we discuss about the convergence properties of ADMM by considering the general form of problem formulation and restricting ourselves to objective functions that have following properties:

Assumption 1: Functions $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ and $g : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ are closed, proper, and convex.

The function equivalently satisfies the first assumption if its epigraph $epi(f) = \{(x, t) \in \mathbb{R}^n \times \mathbb{R} : f(x) \leq t\}$ is nonempty, closed and convex. This assumption provides that there indeed exist primal variables x^{k+1} and y^{k+1} in (4) that minimize the corresponding augmented Lagrangian subproblems. Overall, this assumption is very general since it doesn't require differentiability or bounding on positive values for objective functions.

Assumption 2: The unaugmented Lagrangian L_0 has a saddle point.

The algebraic expression for this assumption is that there exist (x^*, y^*, v^*) such that $L_0(x^*, y^*, v) \leq L_0(x^*, y^*, v^*) \leq L_0(x, y, v^*)$ for all x, y and v. This implies that strong duality holds between primal and dual problems, alias our primal and dual problems have solutions (x^*, y^*) and v^*, and the corresponding optimal values are matching.

If assumptions 1 and 2 are satisfied, several basic convergence properties follows. First, the primal residuals $r^k = Ax^k + By^k - c$ convergence to approach feasibility, i.e., $\lim_{k\to\infty} r^k = 0$. Second, the primal objective converges to the optimal value, i.e., $\lim_{k\to\infty} f(x^k)+g(y^k) = f(x^*)+g(y^*)$. Lastly, the dual variables converge to the optimal point, i.e., $\lim_{k\to\infty} v^k = v^*$. It is also noteworthy to observe that assumptions doesn't imply the convergence of primal variables to their corresponding optimal points, although this property can be achieved by more strict assumptions. Typical feature of ADMM is also that the convergence to high accuracy might take a longer time compared to other methods such as ALMM. In terms of practicality of ADMM algorithm, this doesn't however cause notable issues since for many practical problems, the convergence tolerance ϵ can be set as such that the ADMM algorithm can terminate in reasonable time.

Beside the analytical results, it has been often noted that dynamical penalty parameters ρ^k could improve the convergence in practice. The dynamical ρ^k usually smooths the convergence, since the fixed small or large ρ's could cause opposite effects in primal and dual residuals (defined in the next section) that are primarily used to indicate for the convergence and terminate the algorithm.

1.3 Stopping conditions

The stopping condition of ADMM can be derived from the problem-specific dual and primal residuals $r^{k+1} = Ax^{k+1} + By^{k+1} - c$ and $s^{k+1} = \rho A^T B(y^{k+1} - y^k)$ at iteration $k+1$. This follows from

the fact that when $r^k \to 0$ and $s^k \to 0$ simultaniously $f(x^k) + g(y^k) - p^* \to 0$, where the last mentioned term measures the objective suboptimality at the current point, when $p^* = f(x^*) + g(y^*)$ is the optimal value of the objective. In more rigours terms, we can demonstrate the relation of primal and dual residuals and objective suboptimality via following inequality.

$$f(x^k) + g(y^k) - p^* \leq (-v^k)^T r^k + (x^k - x^*)^T s^k \leq (-v^k)^T r^k + d\left\|s^k\right\|_2 \leq \left\|v^k\right\|_2 \left\|r^k\right\|_2 + d\left\|s^k\right\|_2 \quad (6)$$

The expression after the second inequality introduces a parameter d that is used to approximate unknown distance between primal variable x and its optimal value x^*, say $\left\|x^k - x^*\right\|_2 \leq d$. By comparing the left and right hand side, we can see that shrinkage in residuals imply shrinkage in suboptimality, and therefore suitable stopping criterions can be derived based on residuals by introducing feasibility tolerances ϵ^{pri} and ϵ^{dual} for which $\left\|r^k\right\|_2 \leq \epsilon^{pri}$ and $\left\|s^k\right\|_2 \leq \epsilon^{dual}$ yields a suitable stopping condition. For linearly constrained problems $Ax + By = c$, one can define feasibly tolerances by introducing an alternative pair of absolute and relative tolerances $\epsilon^{abs} > 0$ and $\epsilon^{rel} > 0$. The relation of these four tolerance parameters is then expressed as

$$\begin{aligned} \epsilon^{pri} &= \sqrt{p}\,\epsilon^{abs} + \epsilon^{rel} \max\left\{\left\|Ax^k\right\|_2, \left\|By^k\right\|_2, \left\|c\right\|_2\right\} \\ \epsilon^{dual} &= \sqrt{n}\,\epsilon^{abs} + \epsilon^{rel} \left\|A^T y^k\right\|_2 \end{aligned} \quad (7)$$

where absolute and relative tolerances are some small fixed numbers, for instance $\epsilon^{rel} \in [10^{-3}, 10^{-4}$ and choice of ϵ^{abs} depends on the scaling of of problem variables.

It is worth of noting that in the stopping condition given above, it holds in general that $\epsilon^{pri} \neq \epsilon^{dual}$ and $\left\|r^k\right\|_2 \neq \left\|s^k\right\|_2$. To prevent the fuzz of multiple treshod parameters, the distance from convergence can also be defined as $\left\|r^k\right\|_2^2 + \left\|s^k\right\|_2^2$. This expression will also be used as sopping criteria in the project assignment.

1.4 ADMM pseudocode

Based on the discussions in previous chapters, the pseudocode of ADMM can be formulated as follows:

Algorithm 1 ADMM pseudocode

Require: tolerances $\epsilon^{pri}, \epsilon^{dual} > 0$, initial solutions x^0, y^0, v^0, iteration counter $k = 0$, penalty parameter $\rho > 0$.

1: **while** $\left\|Ax^{k+1} + By^{k+1} - c\right\| > \epsilon^{pri}$ and $\left\|\rho A^T B(y^{k+1} - y^k)\right\| > \epsilon^{dual}$ **do**
2: $\quad x^{k+1} = \operatorname{argmin} L_\rho(x, y^k, v^k)$
3: $\quad y^{k+1} = \operatorname{argmin} L_\rho(x^{k+1}, y, v^k)$
4: $\quad v^{k+1} = v^k + \rho(Ax^{k+1} + By^{k+1} - c)$
5: $\quad k = k + 1$
6: **end while**
7: **return** (x^k, y^k)

For initial solutions, usually setting $x^0 = y^0 = v^0 = 0$ is enough. The general stopping conditions based on primal and dual residuals in line 1 are those presented in section 1.3 and the ADMM update equations (4) are given in lines 2-4.

2 Application

2.1 Problem statement

Stochastic capacity expansion problem is general form of optimization problem where one seeks to find optimal solution to satisfy supply and demand balance between set of customers and set of suppliers with respect to the operational costs. In the particular case, stochastic problem is reformulated to the equivalent deterministic problem of the form:

$$\min_{x,y_s,u_s} \left\{ \sum_{i\in I} C_i x_i + \sum_{s\in S} P_s \left(\sum_{i\in I}\sum_{j\in J} F_{ij} y_{ijs} + \sum_{j\in J} Q_j u_{js} \right) : (x, y_s, u_s) \in K_s \forall s \in S \right\} \tag{8}$$

where K_s is set of constraints provided in the project description (for simplicity reasons this will not be relisted here), S is the set of possible independent scenarios and P_s is the probability of occurrence for the scenario s. Observe that the problem is linear and convex.
It is important to note that large-scale stochastic problem of this form is near to impossible to solve with not only analytical, but also numerical approaches. Therefore, its equivalent deterministic version must be considered. However, even in this case due to the large amount of parameters, analytical solutions are often unreachable and instead weaker numerical methods are employed. In this case, the obtained solution is approximate, however, very often sufficient for practical purposes. Nevertheless, large-scale of the problem and particularly, existence of variety of scenarios, large cardinalities of supplier-customers sets impose challenges even to numerical methods. Namely, computational resources start playing a part. Thus, it is necessary to choose a method which effectively tackles aforementioned challenges.

2.2 Employing ADMM to solve SCEP

To start with, it is important to mention that ADMM itself is not superior method in terms of desired algorithmic properties. Indeed, there exist methods which guarantee better asymptotic convergence, however, the essense of almost all numerical optimization algorithms is that they are inherently sequential and their steps are interdependent, which implies that in order to obtain desired extrema point somewhat sequentially strict order of steps must be maintained, regardless of the structure of problem in question. This property forbids any potential room for parallelism. On the other hand, the advantage of ADMM is that it by construction allows explicit distributed computation, with the only requirement for the problem to be decomposable into independent sub-problems. These sub-problems can then be solved independently of each other and the sub-solutions could be used to construct the solution of the original problem. Observe that problem described in the section 2.1 possess number (potentially very large) of **independent** scenarios, each of which could be effectively treated as an optimization problem on its own. Therefore, using ADMM we can solve sub-problem for every $s \in S$ in parallel, significantly cutting computational time. Moreover, this strategy will also allow to scale the solution: as cardinality of S grows, the algorithm will be able to stay within acceptable bounds in terms of time thus allowing for tackling potentially very large-scale sets of scenarios.

2.3 Derivation of steps (Properly reference pseudocode)

In this section we provide detailed derivation of the augmented Lagrangian, the update steps and residuals for checking of stopping criteria.
We start from the original problem described in 2.1:

$$\min_{x,y,u} \left\{ \sum_{i\in I} C_i x_i + \sum_{s\in S} P_s \left(\sum_{i\in I}\sum_{j\in J} F_{ij} y_{ijs} + \sum_{j\in J} Q_j u_{js} \right) : (x, y_s, u_s) \in K_s \forall s \in S \right\} \tag{9}$$

Observe that terms, dependent on objective variables (x, y_s, u_s), are independent of each other, that is: $\sum_{i\in I} C_i x_i$, $\sum_{i\in I}\sum_{j\in J} F_{ij} y_{ijs}$ and $\sum_{j\in J} Q_j u_{js}$ could be minimized independently. However, the term $\sum_{i\in I} C_i x_i$ is independent of s. In order to be able to exploit optimization for each s in parallel, we create this dependency by introducing scenario-dependent copy variables x_s for every $s \in S$. In addition, we set nonanticipativity constraint $x_s = z$ for every $s \in S, z \in \mathbb{R}^{n_x}$. The constraint insures that x_s are the same for all $s \in S$. For simplicity, lets denote all constraints as $(x_s, y_s, u_s, z) \in K_s, \forall s \in S$. Now, we rewrite the problem 9 as

$$\min_{x,y,u,z} \left\{ \sum_{s\in S} P_s \left(\sum_{i\in I} C_i x_{is} + \sum_{i\in I}\sum_{j\in J} F_{ij} y_{ijs} + \sum_{j\in J} Q_j u_{js} \right) : (x_s, y_s, u_s, z) \in K_s, \forall s \in S \right\} \tag{10}$$

Observe that 10 has both: the independence among decision variables and simultaneously scenario dependency of each decision variable. Thus, the problem in this form can be solved with ADMM. Using 10, we proceed to derive **augmented Lagrangian dual function**. It yields:

$$\min_{x,y,u,z} \sum_{s\in S} \left[P_s \left(\sum_{i\in I} C_i x_{is} + \sum_{i\in I}\sum_{j\in J} F_{ij} y_{ijs} + \sum_{j\in J} Q_j u_{js} \right) + \mu_s^T (x_s - z) + P_s \frac{\rho}{2} \|x_s - z\|_2^2 \right] \tag{11}$$

s.t. $(x_s, y_s, u_s, z) \in K_s, \forall s \in S$
By setting $v_s = \frac{\mu_s}{P_s}$ and denoting

$$L_s^\rho(x_s, y_s, u_s, z, v_s) = \sum_{i\in I} C_i x_{is} + \sum_{i\in I}\sum_{j\in J} F_{ij} y_{ijs} + \sum_{j\in J} Q_j u_{js} + v_s^T (x_s - z) + \frac{\rho}{2} \|x_s - z\|_2^2 \tag{12}$$

we state the objective function as:

$$\phi(v) = \min_{x,y,u,z} \sum_{s\in S} P_s L_s^\rho(x_s, y_s, u_s, z, v_s) \tag{13}$$

s.t. $(x_s, y_s, u_s, z) \in K_s, \forall s \in S$

Next, we derive the update directions. Since $z \in \mathbb{R}^{n_x}$, from 12 we observe that $\min \phi(v) = -\infty$

unless

$$v_s^T z = 0 \quad \forall s \in S \tag{14}$$

Therefore, 12 reduces to:

$$L_s^\rho(x_s, y_s, u_s, z, v_s) = \sum_{i \in I} (C_i + v_{is})\, x_{is} + \sum_{i \in I} \sum_{j \in J} F_{ij} y_{ijs} + \sum_{j \in J} Q_j u_{js} + \frac{\rho}{2} \|x_s - z\|_2^2 \tag{15}$$

From 15, we deduce **primal step update** as:

$$(x_s^{k+1}, y_s^{k+1}, u_s^{k+1}) = \underset{(x_s, y_s, u_s) \in K_s}{\operatorname{argmin}} L_s^\rho(x_s, y_s, u_s, z^k, v_s^k) \tag{16}$$

Observe that primal update step is a function of k and independent of iteration over $s \in S$. Thus, it can be done in parallel for every $s \in S$. Observe that 16 corresponds to the line 2 in algorithm 1 and essentially requires to solve quadratic problem with linear constraints. Next we derive z-update. Using $(x_s^{k+1}, y_s^{k+1}, u_s^{k+1})$ it yields:

$$z^{k+1} = \underset{z}{\operatorname{argmin}} \sum_{s \in S} P_s L_s^\rho(x_s^{k+1}, y_s^{k+1}, u_s^{k+1}, z, v_s^k) \tag{17}$$

Differentiating 17 wrt. z yields:

$$\sum_{s \in S} P_s \rho (x_s^{k+1} - z) = 0 \tag{18}$$

Using the facts that ρ is constant and measure of probability space equals to 1 we obtain **z-update step**:

$$z^{k+1} = \sum_{s \in S} P_s x_s^{k+1} \tag{19}$$

The z-update corresponds to the line 3 in algorithm 1 and since it involves sum over $s \in S$ requires the results from all scenarios be ready. In principle, parallelization is also possible here however, it unlikely yields significant speed up since the update is only linear in s. Finally, referencing the ADMM outline and using (x_s^{k+1}, z^{k+1}) we state the **dual update**:

$$v_s^{k+1} = v_s^k + \rho(x_s^{k+1} - z^{k+1}) \tag{20}$$

Observe that 20 resembles classical Gradient Descent with step size ρ and corresponds to the line 4 in algorithm 1. In addition, similarly with primal update, note that dual update is also function of k constant in s. Therefore, the dual updates could be also computed in parallel for every $s \in S$. Finally, the condition on stopping criteria is derived from primal and dual residual square norms, respectively $P_s \left\|x_s^{k+1} - z^{k+1}\right\|_2^2$ and $P_s \left\|z^{k+1} - z^k\right\|_2^2$. Summing two terms over all $s \in S$ and using the constraint $x_s = z$ yields stopping criteria:

$$\sum_{s \in S} P_s \left[\left\|x_s^{k+1} - z^{k+1}\right\|_2^2 + \left\|z^{k+1} - z^k\right\|_2^2 \right] = \sum_{s \in S} P_s \left\|x_s^{k+1} - z^k\right\|_2^2 < \epsilon \tag{21}$$

Where $0 < \epsilon < 1$ is predefined similarly to the description in section 1.3. Finally, observe that

$$\sum_{s\in S} P_s \left\|x_s^{k+1} - z^k\right\|_2^2 \leq \left(\sum_{s\in S} P_s \left\|x_s^{k+1} - z^k\right\|_2\right)^2 \leq 2^{|S|} \sum_{s\in S} P_s \left\|x_s^{k+1} - z^k\right\|_2^2 < 2^{|S|}\epsilon \tag{22}$$

which implies that

$$\sum_{s\in S} P_s \left\|x_s^{k+1} - z^k\right\|_2 < \sqrt{2^{|S|}\epsilon} = \tilde{\epsilon} \tag{23}$$

Which shows that we could equivalently use stopping criteria 23. Observe that due to absence of square terms in residual norms it results in less amount of computations needed and higher numerical stability. Finally, in order to make convergence smoother we multiply left hand side by ρ and fix $\tilde{\epsilon}$ appropriately. Observe that 23 is related to the line 1 in algorithm 1.

2.4 Technical set-up

Concerning the technical details related to the computational hardware and software, two alternatives were considered. Namely, standard desktop computer on maxOS BigSur with 8 M1 cores and 16GB RAM and and Aalto CS Jupyterlab server with Julia kernel 1.6.2. The latter was chosen due to efficiency and standardization reasons. Benchmarks were obtained with **@time** macro.

3 Discussion and conclusions

3.1 Full-scale model results: ADMM vs Ipopt

The results of structural comparison between ADMM and full-scale model solved with Ipopt are presented in the Tables 1, 2 and 3. Note that results of ADMM have numerical value of ρ in the first column, while the results of Ipopt solver contains * at the respective place.
On a very general level, two key points are observable. Firstly, note that for the small-scale problem, the Ipopt on average performs better than ADMM (Table 1). The result is expected, since invocation of parallelism always incur overhead required for creation and coordination of threads. However, already for problem of medium-scale, slight advantage in computational speed of ADMM is observable and for large-scale problem it becomes even more evident (note Tables 2, 3). In the attached notebook with implementation of experiments, problem of even larger scale was tested. The results are impressive, for 130 scenarios ADMM outperforms single-threaded optimization algorithm by a factor of 2. We omit presentation of them here for the sake of compactness, but they could be found in the attached notebook. If we would proceed to increase the scale of the problem, we would observe even greater asymptotic difference in the performance of both algorithms and thus, we conclude that ADMM indeed could benefit practical computational complexity of the optimization, provided the problem of reasonable size. For the small scale solution, optimized single-threaded algorithms such as the one used in Ipopt solver are easier and more efficient to use. Secondly, it it instructive to analyze complexity of algorithms in terms of number of iterations taken to converge. In case of Ipopt solver, doubling the number of scenarios almost linearly increases the number of iterations taken to get to the optimum, that is **343** vs. **534** (Tables 1 and 3). On the

other hand, double enlargement in the number of scenarios for ADMM nearly haven't yield any change of this metric, however, the time per iteration increased. This phenomena demonstrates the presence of parallelism: the amount of thread work per iteration of the loop over k grows as cardinality of S grows, however, as the loop over $s \in S$ is split between several threads, this mainly results in the increase of time per iteration rather than in increase in the number of iterations. For the problems with 130 scenarios the situation is slightly different, but not dramatically. Namely, the number of iterations starts to show explicit dependency on the penalty parameter ρ. Practically, the larger values of penalty term incur greater amount of steps taken to converge, since the initial value of objective function is larger. However, this only happens for the subset of large $\rho \geq 50$. For smaller values of ρ the number of iterations is intact with previous observation.

Furthermore, observe that for the small values of ρ solution obtained by ADMM exhibits slight numerical inaccuracy. This could be justified by empirical observation that smaller values of penalty imply faster convergence, however, it also enforces to consider optimal solution vectors closer to the boundary of the acceptable region defined by residual norm difference. Observe that for all problems, residuals up to value of $\rho = 20$ are of relatively large order of magnitude and they start to be negligibly small only for $\rho \geq 20$. In its turn, notice that solution produced by Ipopt is always very close to the analytical optimum. Thus, we conclude that either careful fine-tuning of parameter ρ is required or tolerance used to check stopping criteria has to be tightened in order to improve accuracy of ADMM.

Lastly, we also inspected the convergence behaviour of ADMM using a larger scale of penalty parameters. The results are given in figures 1, 2 and 3, where ρ varies from 0.5 to 100 with a step size of 5, despite the first four values that we hardcoded into $[0.5, 0.7, 0.9, 1]$. The convergence performance was measured in terms of total computation time and the number of iterations, and for all scenario cardinals, the sweet spot for ρ is inside interval $[0.7, 20]$. Overall, the performance differences are rather negligible, usually $\pm$ one iteration, which gives an experimental proof that in practical situations ADMM is quite robust against different tunings of penalty parameter.

Table 1: Results of small instance problem

ρ	Time(T/I)	Iterations	Solution	Residual
0.5	33/5.5	6	1498.93	0.08
0.7	34/6.8	5	1499.64	0.09
0.9	34/6.8	5	1499.91	0.07
1	35/7	5	1499.91	0.06
5	32/6.4	5	1500.00	0.04
7	34/6.8	5	1500.00	0.04
10	36/7.2	5	1500.00	0.05
20	29/4.3	6	1500.00	4.05e-6
50	35/5.8	6	1500.00	5.56e-6
70	35/5.8	6	1500.00	4.53e-6
100	30/5	6	1500.00	3.89e-6
*	29/0.08	343	1500.00	*

Table 2: Results of medium instance problem

ρ	Time(T/I)	Iterations	Solution	Residual
0.5	50/7.14	7	1499.51	0.08
0.7	53/8.83	6	1499.91	0.04
0.9	52/10.4	5	1499.90	0.05
1	53/10.6	5	1499.89	0.05
5	50/10	5	1500.00	0.05
7	51/10.2	5	1500.00	0.05
10	53/10.6	5	1500.00	0.04
20	53/10.6	5	1500.00	0.08
50	56/9.33	6	1500.00	5.35e-6
70	52/8.6	6	1500.00	6.24e-6
100	52/8.6	6	1500.00	4.07e-6
*	58/0.13	458	1500.00	*

Table 3: Results of large instance problem

ρ	Time(T/I)	Iterations	Solution	Residual
0.5	66/9.42	7	1499.50	0.08
0.7	68/11.33	6	1499.93	0.05
0.9	68/13.6	5	1499.91	0.06
1	67/13.4	5	1499.91	0.06
5	69/13.8	5	1500.00	0.04
7	68/13.6	5	1500.00	0.04
10	70/14	5	1500.00	0.04
20	69/13.8	5	1500.00	0.09
50	70/11.6	6	1500.00	3.66e-6
70	67/11.2	6	1500.00	5.20e-6
100	68/11.3	6	1500.00	4.01e-6
*	106/0.20	534	1500.00	*

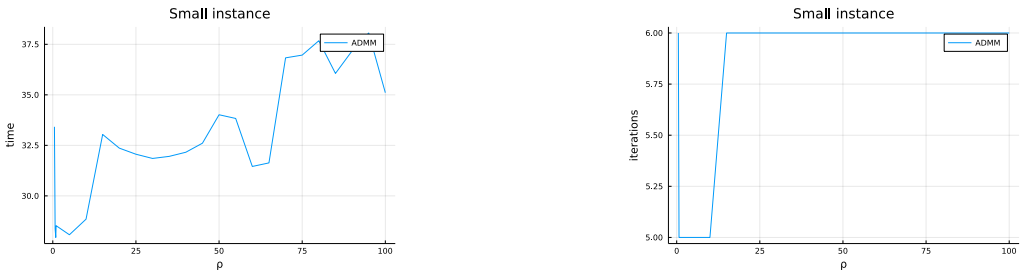

Figure 1: Computation time and iterations until convergence, 50 scenarios.

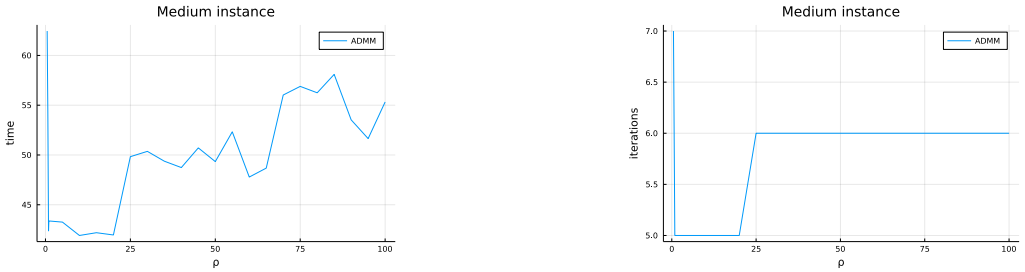

Figure 2: Computation time and iterations until convergence, 75 scenarios.

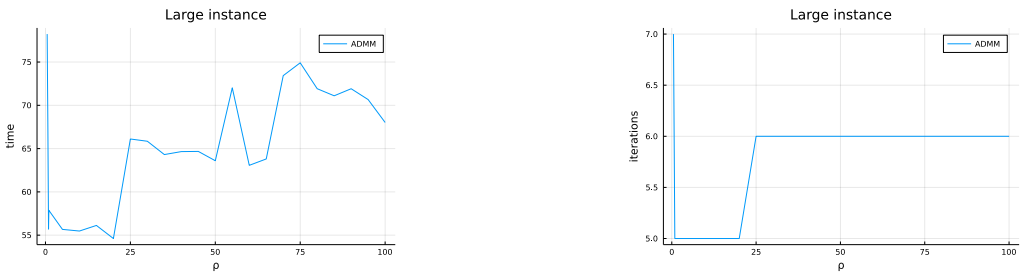

Figure 3: Computation time and iterations until convergence, 100 scenarios.