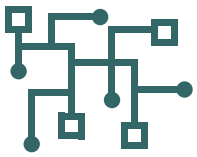

ISE

Industrial and Systems Engineering

A Low-Rank Coordinate-Descent Algorithm for Semidefinite Programming Relaxations of Optimal Power Flow

Jakub Mareček[1] and Martin Takáč[1]

[1]IBM Research Ireland [2]Lehigh University

ISE Technical Report 15T-006

LEHIGH UNIVERSITY

A Low-Rank Coordinate-Descent Algorithm for Semidefinite Programming Relaxations of Optimal Power Flow

Jakub Mareček and Martin Takáč

***Abstract*—A novel rank-constrained re-formulation of alternating-current optimal power flow problem makes it possible to derive novel semidefinite programming (SDP) relaxations. For those, we develop a solver, which is often as fast as Matpower's interior point method, within the same accuracy.**

***Index Terms*—Power system analysis computing, Optimization, Numerical Analysis (Mathematical programming)**

I. Introduction

Alternating-current optimal power flow problem (ACOPF) is one of the best known non-linear optimisation problems [28]. Due to its non-convexity, even deciding feasibility is NP-Hard [14]. Still, there has been much recent progress. Lavaei and Low [13] have shown that a semidefinite programming (SDP) relaxation produces exact solutions under some conditions. Ghaddar et al. [8] have studied an asymptotically convergent hierarchy of SDP relaxations. Unfortunately, the run-time of the best-known solvers for the SDP relaxations remains several orders of magnitude higher than that of commonly used methods without global convergence guarantees, e.g. [30].

Following a brief overview of our notation, we introduce a novel rank-constrained reformulation of the ACOPF problem, where all constraints, except for the rank constraint, are coordinate-wise. Next, we present a parallel coordinate-descent algorithm, which solves a sequence of convex relaxations with coordinate-wise constraints, using a novel closed-form step considering roots of cubic polynomials. Under certain assumptions, the algorithm converges to the global optimum. The proofs of convergence rely on the work of Burer and Moteiro [5], Grippo et al. [9], and our earlier work [26, 17]. Finally, we show that our method reaches precision comparable to the default settings of Matpower [30], the commonly used interior-point method without global convergence guarantees, within comparable times.

Jakub is with IBM Research – Ireland, B3 F14, IBM Campus Damastown, Mulhuddart, Dublin 15. email: jakub.marecek@ie.ibm.com Martin is with Leghigh University, Bethlehem, PA email: Takac.MT@gmail.com

II. The Problem

As in much recent work [1, 13, 19, 11, 8], we start with the rectangular power-voltage formulation of ACOPF over a network of buses N, connected by branches $L \subseteq N \times N$, modeled as Π-equivalent circuits. The input comprises:

- $G \subseteq N$, which are the generators
- $P_k^d + jQ_k^d$, which are the active and reactive load at each bus $k \in N$,
- $y \in \mathbb{C}^{|N|\times|N|}$, which is the network admittance matrix capturing the value of the shunt element $\bar{b}_{lm}$ and series admittance $g_{lm} + jb_{lm}$ at branch $(l,m) \in L$,
- $P_k^{\min}$, $P_k^{\max}$, $Q_k^{\min}$ and $Q_k^{\max}$, which are the limits on active and reactive generation capacity at bus k, where $P_k^{\min} = P_k^{\max} = Q_k^{\min} = Q_k^{\max} = 0$ for all $k \in N \setminus G$,
- $V_k^{\min}$ and $V_k^{\max}$, which are the limits on the absolute value of the voltage at a given bus k,
- $S_{lm}^{\max}$, which is the limit on the absolute value of the apparent power of a branch $(l,m) \in L$.

The variables are:

- $P_k^g + jQ_k^g$, which is the power generated at bus $k \in G$,
- $V_k = \Re V_k + j\Im V_k$, which is the voltage at each bus $k \in N$,

with the apparent power flow $S_{lm} = P_{lm} + jQ_{lm}$ on the line $(l,m) \in E$ and power represented in terms of voltages:

$$\begin{aligned} P_k^g =& P_k^d + \Re V_k \sum_{i=1}^{n} (\Re y_{ik} \Re V_i - \Im y_{ik} \Im V_i) \\ &+ \Im V_k \sum_{i=1}^{n} (\Im y_{ik} \Re V_i + \Re y_{ik} \Im V_i), \qquad (1) \\ Q_k^g =& Q_k^d + \Re V_k \sum_{i=1}^{n} (-\Im y_{ik} \Re V_i - \Re y_{ik} \Im V_i) \\ &+ \Im V_k \sum_{i=1}^{n} (\Re y_{ik} \Re V_i - \Im y_{ik} \Im V_i). \qquad (2) \end{aligned}$$

The power-flow equations are hence:

$$\begin{aligned} P_{lm} &= b_{lm}(\Re V_l \Im V_m - \Re V_m \Im V_l) \quad (3)\\ &\quad + g_{lm}(\Re {V_l}^2 + \Im {V_m}^2 - \Im V_l, \Im V_m - \Re V_l \Re V_m),\\ Q_{lm} &= b_{lm}(\Re V_l \Im V_m - \Im V_l \Im V_m - \Re {V_l}^2 - \Im {V_l}^2)\\ &\quad + g_{lm}(\Re V_l \Im V_m - \Re V_m \Im V_l - \Re V_m \Im V_l)\\ &\quad - \frac{\bar{b}_{lm}}{2}(\Re {V_l}^2 + \Im {V_l}^2). \quad (4) \end{aligned}$$

Further, let e_k be the k^{th} standard basis vector in $\mathbb{R}^{|N|}$. To model constraints (1)-(4), the following matrices are defined:

$$\begin{aligned} y_k &= e_k e_k^T y,\\ y_{lm} &= (j\frac{\bar{b}_{lm}}{2} + g_{lm} + jb_{lm}) e_l e_l^T - (g_{lm} + jb_{lm}) e_l e_m^T,\\ Y_k &= \frac{1}{2}\begin{bmatrix} \Re(y_k + y_k^T) & \Im(y_k^T - y_k) \\ \Im(y_k - y_k^T)) & \Re(y_k + y_k^T) \end{bmatrix},\\ \bar{Y}_k &= -\frac{1}{2}\begin{bmatrix} \Im(y_k + y_k^T) & \Re(y_k - y_k^T) \\ \Re(y_k^T - y_k) & \Im(y_k + y_k^T) \end{bmatrix},\\ M_k &= \begin{bmatrix} e_k e_k^T & 0 \\ 0 & e_k e_k^T \end{bmatrix},\\ Y_{lm} &= \frac{1}{2}\begin{bmatrix} \Re(y_{lm} + y_{lm}^T) & \Im(y_{lm}^T - y_{lm}) \\ \Im(y_{lm} - y_{lm}^T) & \Re(y_{lm} + y_{lm}^T) \end{bmatrix},\\ \bar{Y}_{lm} &= \frac{-1}{2}\begin{bmatrix} \Re\{Y_{lm} + Y_{lm}^T\} & \Im\{Y_{lm} - Y_{lm}^T\} \\ \Im\{Y_{lm}^T - Y_{lm}\} & \Re\{Y_{lm} + Y_{lm}^T\} \end{bmatrix}. \end{aligned}$$

in line with much recent work [13, 8]. Using those, we can rewrite ACOPF:

$$\begin{aligned} \min \sum_{k\in G} &\left(c_k^2 (P_k^g)^2 + c_k^1 P_k^g + c_k^0\right) && \text{[ACOPF]}\\ \text{s.t. } & P_k^{\min} \le P_k^g \le P_k^{\max} && \forall k \in N\\ & Q_k^{\min} \le Q_k^g \le Q_k^{\max} && \forall k \in N\\ & (V_k^{\min})^2 \le \Re {V_k}^2 + \Im {V_k}^2 \le (V_k^{\max})^2 && \forall k \in N\\ & P_{lm}^2 + Q_{lm}^2 \le S_{lm}^{\max} && \forall (l,m) \in L\\ & (1) - (4) \end{aligned}$$

as a real-valued polynomial optimization problem of degree 4:

$$\begin{aligned} \min \sum_{k\in G} & f_k(x) && \text{[PP4]}\\ \text{s.t. } & P_k^{\min} \le \text{tr}(Y_k x x^T) + P_k^d \le P_k^{\max} && (5)\\ & Q_k^{\min} \le \text{tr}(\bar{Y}_k x x^T) + Q_k^d \le Q_k^{\max} && (6)\\ & (V_k^{\min})^2 \le \text{tr}(M_k x x^T) \le (V_k^{\max})^2 && (7)\\ & (\text{tr}(Y_{lm} x x^T))^2 + (\text{tr}(\bar{Y}_{lm} x x^T))^2 \le (S_{lm}^{\max})^2, && (8) \end{aligned}$$

where variable $x \in \mathbf{R}^{2|N|}$ comprises $\Re V_k \in \mathbf{R}^{|N|}$ and $\Im V_k \in \mathbf{R}^{|N|}$ stacked and $f_k(W)$ is

$$\left(c_k^2 (P_k^d + \text{tr}(Y_k x x^T))^2 + c_k^1 (P_k^d + \text{tr}(Y_k x x^T)) + c_k^0\right).$$

Henceforth, we use n to denote $2|N|$, i.e., the dimension of the real-valued problem [PP4].

Further, the problem can be lifted to obtain a rank-constrained problem in $W = xx^T \in \mathbf{R}^{n\times n}$:

$$\begin{aligned} \min \sum_{k\in G} & f_k(W) && \text{[R1]}\\ \text{s.t. } & P_k^{\min} \le \text{tr}(Y_k W) + P_k^d \le P_k^{\max} && (9)\\ & Q_k^{\min} \le \text{tr}(\bar{Y}_k W) + Q_k^d \le Q_k^{\max} && (10)\\ & (V_k^{\min})^2 \le \text{tr}(M_k W) \le (V_k^{\max})^2 && (11)\\ & (\text{tr}(Y_{lm} W))^2 + (\text{tr}(\bar{Y}_{lm} W))^2 \le (S_{lm}^{\max})^2 && (12)\\ & W \succeq 0, \quad \text{rank}(W) = 1, && (13) \end{aligned}$$

which is still NP-Hard, but where can one drop the rank constraint to obtain a strong and efficiently solvable SDP relaxation:

$$\begin{aligned} \min \sum_{k\in G} & f_k(W) && \text{[LL]}\\ \text{s.t. } & (9-12), \quad W \succeq 0, \end{aligned}$$

as suggested by [1]. Lavaei and Low [13] studied the relaxation as Optimization 3. It was shown to be equivalent to a first-level $[OP_4 - H_1]$ of a certain hierarchy of relaxations by Ghaddar et al. [8]. We note that for traditional solvers [19, 13, 8], Optimization 4 of [13] is much easier to solve than [LL].

III. The Reformulation

The first contribution of this paper is a generalisation and reformulation of [R1]:

$$\begin{aligned} \min \quad & F(W) := \sum_{k\in G} f_k(W) && \text{[R}r\text{BC]}\\ \text{s.t. } & t_k = \text{tr}(Y_k W) && (14)\\ P_k^{\min} - P_k^d \le & t_k \le P_k^{\max} - P_k^d && (15)\\ & g_k = \text{tr}(\bar{Y}_k W) && (16)\\ Q_k^{\min} - g_k \le & g_k \le Q_k^{\max} - g_k && (17)\\ & h_k = \text{tr}(M_k W) && (18)\\ (V_k^{\min})^2 \le & h_k \le (V_k^{\max})^2 && (19)\\ & u_{lm} = \text{tr}(Y_{lm} W) && (20)\\ & v_{lm} = \text{tr}(\bar{Y}_{lm} W) && (21)\\ & z_{lm} = (u_{lm})^2 + (v_{lm})^2 && (22)\\ & z_{lm} \le (S_{lm}^{\max})^2 && (23)\\ & W \succeq 0, \text{rank}(W) \le r. && (24) \end{aligned}$$

There, we still have:

Proposition 1. *[R1BC] is equivalent to [PP4].*

Even in the special case of $r = 1$, however, we have lifted the problem to a higher dimension by adding t, g, and h variables, which are box-constrained functions of W.

Subsequently, we make four observations to motivate our approach to solving the [RrBC]:

O_1: constraints (14–21) and (23) are box constraints,
O_2: using elementary liner algebra:

$$\{W \in \mathcal{S}^n \text{ s.t. } W \succeq 0, \quad \text{rank}(W) \leq r\} = \{RR^T \text{ s.t. } R \in \mathbf{R}^{n\times r}\},$$

where $\mathcal{S}^n$ is the set of symmetric $n \times n$ matrices.
O_3: if rank$(W^*) > 1$ for the optimum W^* of [LL], there are no known methods for extracting the global optimum of [R1] from W, except for [8].
O_4: zero duality gap at any SDP relaxation in the hierarchy of [8] does not guarantee the solution of the SDP relaxation is exact for [PP4], c.f. [10].

Note that Lavaei and Low [13] restate the condition in Observation O_3 in terms of ranks using a related relaxation (Optimization 3), where the rank has to be strictly larger than 2.

IV. The Algorithm

Broadly speaking, we use the well-established Augmented Lagrangian approach [25], with a low-rank twist [5], and a parallel coordinate descent with a closed-form step. Considering Observation O_2, we replace variable $W \in \mathbf{R}^{n\times n}$ by $RR^T \in \mathbf{R}^{n\times n}$ for $R \in \mathbf{R}^{n\times r}$ to obtain the following augmented Lagrangian function:

$$\begin{aligned}
&\mathcal{L}(R,t,h,g,u,v,z,\lambda^t,\lambda^g,\lambda^h,\lambda^u,\lambda^v,\lambda^z) := \\
&\sum_{k\in G} f_k(RR^T) \\
&- \sum_k \lambda^t_k(t_k - \text{tr}(Y_k RR^T)) + \tfrac{\mu}{2}\sum_k (t_k - \text{tr}(Y_k RR^T))^2 \\
&- \sum_k \lambda^g_k(g_k - \text{tr}(\bar{Y}_k RR^T)) + \tfrac{\mu}{2}\sum_k (g_k - \text{tr}(\bar{Y}_k RR^T))^2 \\
&- \sum_k \lambda^h_k(h_k - \text{tr}(M_k RR^T)) + \tfrac{\mu}{2}\sum_k (h_k - \text{tr}(M_k RR^T))^2 \\
&- \sum_{(l,m)} \lambda^u_{(l,m)}(u_{(l,m)} - \text{tr}(Y_{(l,m)} RR^T)) \\
&+ \tfrac{\mu}{2}\sum_{(l,m)} (u_{(l,m)} - \text{tr}(Y_{(l,m)} RR^T))^2 \\
&- \sum_{(l,m)} \lambda^v_{(l,m)}(v_{(l,m)} - \text{tr}(\bar{Y}_{(l,m)} RR^T)) \\
&+ \tfrac{\mu}{2}\sum_{(l,m)} (v_{(l,m)} - \text{tr}(\bar{Y}_{(l,m)} RR^T))^2 \\
&- \sum_{(l,m)} \lambda^z_{(l,m)}(z_{(l,m)} - u^2_{(l,m)} - v^2_{(l,m)}) \\
&+ \tfrac{\mu}{2}\sum_{(l,m)} (z_{(l,m)} - u^2_{(l,m)} - v^2_{(l,m)})^2 + \nu\mathcal{R}.
\end{aligned}$$

Note that constants $\mu, \nu > 0$ pre-multiply regularisers, where $\mathcal{R}$ can often be 0 in practice, although not in our analysis, where we require $\mathcal{R} = \det(R^T R)$, which promotes low-rank solutions.

As suggested in Algorithm Schema 1, we increase the rank $r = 1, 2, \ldots$ allowed in W in an outer loop of the algorithm. In an inner loop, we use coordinate descent method to find an approximate minimizer of

Algorithm Schema 1: A Low-Rank Coordinate Descent Algorithm

```
1:  for r = 1, 2, ... do
2:    choose R ∈ R^{m×r}
3:    compute corresponding values of t, h, g, u, v, z
4:    project t, h, g, u, v, z onto the box constraints
5:    for k = 0, 1, 2, ... do
6:      in parallel, minimize L in t, g, h, u, v, z,
        coordinate-wise,
7:      in parallel, minimize L in R,
        coordinate-wise
8:      update Lagrange multipliers
        λ^t, λ^g, λ^h, λ^u, λ^v, λ^z
9:      update μ
10:     terminate, if criteria are met
11:   end for
12: end for
```

$\mathcal{L}(R,t,h,g,u,v,z,\lambda^t,\lambda^g,\lambda^h,\lambda^u,\lambda^v,\lambda^z)$, and denote the k-th iterate $R^{(k)}$. Note that variables t, h, g, z have simple box constraints, which have to be considered outside of $\mathcal{L}$.

In particular:

a) The Outer Loop: The outer loop (Lines 1-12) is known as the "low-rank method" [5]. As suggested by Observation O_3, in the case of [LL], one may want to perform only two iterations $r = 1, 2$. In the second iteration of the outer loop, one should like to test, whether the numerical rank of the iterate R_k in the inner iteration k has numerical rank 1. If this is the case, one can conclude the solution obtained for $r = 1$ is exact. This test, sometimes known as "flat extension", has been studied both in terms of numerical implementations and applicability by Burer and Choi [4].

b) The Inner Loop: The main computational expense of the proposed algorithm is to find an approximate minimum of $\mathcal{L}(R,t,h,g,u,v,z,\lambda^t,\lambda^g,\lambda^h,\lambda^u,\lambda^v,\lambda^z)$ with respect to R, t, h, g, u, v, z within the inner loop (Lines 6–7). Note that $\mathcal{L}$ as a function of R is – in general – non-convex. The inner loop employs a simple iterative optimization strategy, known as the coordinate descent. There, two subsequent iterates differ only in a single block of coordinates. In the very common special case, used here, we consider single coordinates, in a cyclical fashion. This algorithm has been used widely at least since 1950s. Recent theoretical guarantees of random coordinate descent algorithm are due to Nesterov [23] and the present authors [26, 17]. See the survey of Wright [29] for more details.

c) The Closed-Form Step: An important ingredient in the coordinate descent is a novel closed-form

step. Nevertheless, if we update only one scalar of R at a time, and fix all other scalars, the minimisation problem turns out to be the minimisation of an 4-order polynomial. In order to find the minimum of a polynomial $ax^4+bx^3+cx^2+dx+0$, we need to find a real root of the polynomial $4ax^3+3bx^2+2cx+d=0$. This polynomial has at most 3 real roots and can be found using closed form formulae due to Cardano. Whenever you fix the values across all coordinates except one, finding the best possible update for the one given coordinate requires either the minimisation of a quadratic convex function with respect to simple box constraints (for variables t, g, h, z) or minimisation of a polynomial function of degree 4 with no constraints (for variables R, u, v), either of which can be done by checking the objective value at each out of 2 (for variables t, g, h, z) or 3 (for variables R, u, v) stationary points and choosing the best one.

d) The Parallelisation: For instances large enough, one can easily exploit parallelism. Notice that minimization of coordinates of t, g, h, u, v, z can be carried out in parallel without any locks, as there are no dependences. One can also update coordinates of R in parallel, although some degradation of the speed-up thus obtainable is likely, as there can be some dependence in the updates. The degradation is hard to bound. Most analyses, c.f. [29], hence focus on the uniformly random choice of (blocks of) coordinates, although there are [17] exceptions.

e) Sufficient Conditions for Termination of the Inner Loop: For both our analysis and in our computational testing, we use a "target infeasibility" stopping criterion for the inner loop, considering squared error:

$$\begin{aligned} T_k = &\sum_k (t_k - \text{tr}(Y_k R^{(k)}(R^{(k)})^T))^2 + \\ &\sum_k (g_k - \text{tr}(\bar{Y}_k R^{(k)}(R^{(k)})^T))^2 + \\ &\sum_k (h_k - \text{tr}(M_k R^{(k)}(R^{(k)})^T))^2 + \\ &\sum_{(l,m)} (u_{(l,m)} - \text{tr}(Y_{(l,m)} R^{(k)}(R^{(k)})^T))^2 + \\ &\sum_{(l,m)} (v_{(l,m)} - \bar{Y}_{(l,m)} R^{(k)}(R^{(k)})^T))^2 + \\ &\sum_{(l,m)} (z_{(l,m)} - u^2_{(l,m)} - v^2_{(l,m)})^2. \end{aligned} \tag{25}$$

We choose the threshold to match the accuracy in terms of squared error obtained by Matpower using default settings on the same instance. $T_k \leq 0.00001$ is often sufficient.

f) The Initialisation: In our analysis, we assume that the instance is feasible. This is difficult to circumvent, considering Lehmann et al. [14] have shown it is NP-Hard to test whether an instance of ACOPF is feasible. In our numerical experiments, however, we choose R such that each elements is independently identically distributed, uniformly over [0, 1], which need not be feasible for the instance of ACOPF. Subsequently, we compute t, h, g, u, v, z to match the R, projecting the values onto the intervals given by the box-constraints. Although one may improve upon this simplistic choice by a variety of heuristics, it still performs well in practice.

g) The Choice of μ_k, ν_k: The choice of μ_k, ν_k may affect the performance of the algorithm. In order to prove convergence, one requires $\lim_{k\to\infty} \nu_k \to 0$, but computationally, we consider $\mathcal{R} = 0$, which obliterates the need for varying ν_k and computation of the gradient,

$$\nabla \det(R^T R) = R \,\text{cofactor}(R^T R),$$

as suggested by [5]. In order to prove convergence, any choice of $\mu_k > 0$ is sufficient. Computationally, we use $\mu_k = 0.0001$ throughout the reported experiments, although we have alos experimented with decreasing μ_k geometrically, i.e. $\mu_k = c \cdot \mu_{k-1}$, The fact one does not have to painstakingly tune the parameters is certainly a relief.

V. An Analysis

The analysis needs to distinguish between optima of the semidefinite programming problem [LL], which is convex, and stationary points, local optima, and global optima of [RrBC], the non-convex rank-constrained problem. Let us illustrate this difference with an example:

Example 2. Consider the following simple rank-constrained problem:

$$\max_{W \in \mathcal{S}^2_+, \text{rank}(W)=1} \text{tr}(\text{diag}(3,1)W), \tag{26}$$

subject to $\text{tr}(IW) = 1$ and

$$W^* := \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} = (1,0)(1,0)^T,$$
$$\tilde{W} := \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} = (0,1)(0,1)^T =: \tilde{R}\tilde{R}^T.$$

W^* is the unique optimal solution of (26), as well as the optimal solution of the SDP relaxation, where the rank constraint is dropped. There exists a Lagrange multiplier such that $\tilde{W}$ is a stationary point, though. Let us define a Lagrange function $\mathcal{L}(R, \lambda) = \text{tr}(\text{diag}(3,1)RR^T) + \lambda(\text{tr}(IRR^T) - 1)$. Then $\nabla_R \mathcal{L}(RR^T, \lambda) = 2\,\text{diag}(3,1)R + 2\lambda R$. If we plug in $\tilde{R}$ and $\tilde{\lambda} = -1$ we obtain $\nabla_R \mathcal{L}(\tilde{R}\tilde{R}^T, \tilde{\lambda}) = 2\,\text{diag}(3,1)(0,1)^T + 2\tilde{\lambda}(0,1)^T = (0,0)^T$. Hence $(\tilde{R}, \tilde{\lambda})$ is a stationary point of a Lagrange function. However, one can follow the proof of Proposition 5

to show that $[\tilde{R}, 0]$ is not a local optimum solution of SDP relaxation.

Indeed, in generic non-convex quadratic programming, the test whether a stationary point solution is a local optimum [22] is NP-Hard. For ACOPF, there are a number of sufficient conditions known, e.g. [20]. Under very strong assumptions, motivated by Observation O_3, we provide necessary and sufficient conditions, based on [9]. We use $\otimes$ for Kronecker product and I_r for the identity matrix in $\mathbf{R}^{r\times r}$.

Proposition 3. *Consider the SDP relaxation obtained from [RrBC] by dropping the rank constraint and write it down as:* $\min tr(QW)$ *such that* $tr(A_iW) = b_i$ *for constraints* $i = 1, \ldots, m$ *in variable* $W \succeq 0, W \in \mathcal{S}^n$. *If there exists an optimum* W^* *with rank* r *for the SDP relaxation, for any point* $R \in \mathbf{R}^{n\times r}$, RR^T *is a global minimiser of [RrBC] if and only if there exists a* $\lambda^* \in \mathbf{R}^m$ *such that:*

$$\left[\left(Q + \sum_{i=1}^{m} \lambda_i^* A_i\right) \otimes I_r\right] R = 0 \tag{27}$$

$$Q + \sum_{i=1}^{m} \lambda_i^* A_i \succeq 0$$

$$R^T(A_i \otimes I_r)R = b_i \quad \forall i = 1, \ldots, m.$$

Proof. The proof is based on Proposition 3 of Grippo et al. [9], which requires the existence of rank r optimum and strong duality of the SDP relaxation. The existence of an optimum of [LL] with rank r is assumed. One can use Theorem 1 [10] and Theorem 1 of [8] to show that a ball constraint is sufficient for strong duality in the SDP relaxation, c.f. Observation O_4. □

Next, let us consider the convergence:

Proposition 4. *There exists an instance-specific constant* r', *such that for every* $r \geq r'$, *whenever Algorithm 1 with* $\mathcal{R} = \det(R^T R)$ *and* $\lim_{k\to\infty} \nu_k \to 0$ *produces solution with* $\lim_{k\to\infty} T_k \to 0$ *and a local optimum* $R^{(k)}$ *is generated within the inner loop (5–11),* $R^{(k)}(R^{(k)})^T$ *is an optimal solution to [LL]. Moreover,* r' *depends on the number of constraints* m *in the optimisation problem and is* $\mathcal{O}(\sqrt{m})$.

Proof. The proof follows from Theorem 3.3 of Burer and Monteiro [5]. One can rephrase Theorem 3.3 to show that if $\{R^{(k)}\} \in \mathbf{R}^{n\times r}$ is a bounded sequence such that:

C_1: $\lim_{k\to\infty} T_k = 0$
C_2: $\lim_{k\to\infty} \nabla\mathcal{L}(R^{(k)}) = 0$
C_3: $\liminf_{k\to\infty} \nabla^2\mathcal{L}(R^{(k)})(H^k, H^k) \geq 0$ for all bounded sequences $\{H^k\}$, $H^k \in \mathbf{R}^{n\times r}$
C_4: $\text{rank}(R^{(k)}) < r$ for all k

every accumulation point of $R^{(k)}(R^{(k)})^T$ is an optimal solution of [LL]. Let us show that these four conditions are satisfied, in turn. Condition C_1, which effectively says that $W = R^{(k)}(R^{(k)})^T$ should be feasible with respect to constraints (9–12), is affected by the termination criteria of the inner loop, albeit only approximately for a finite k and finite machine precision. Conditions C_2 and C_3 follow from our assumption that $R^{(k)}$ is a local optimum. The satisfaction of Condition C_4 can be shown in two steps: First, there exists an r', such that for every feasible semidefinite programming problem with m constraints, there exists an optimal solution with a rank bounded from above by r'. This r' is $\mathcal{O}(\sqrt{m})$. This follows from Theorem 1.2 of Barvinok [2], as explained by Pataki [24]. Second, the $\mathcal{R} = \det(R^T R)$ regularisation forces the lower-rank optimum to be chosen, should there be multiple optima with different ranks. This can be seen easily by contradiction. Finally, one can remove the requirement on the sequence to be bounded by the arguments of Burer and Monteiro [5], as per Theorem 5.4. □

Further,

Proposition 5. *Consider an instance of ACOPF such that there exists an optimum solution* W^* *of [LL] with rank 1 and* $\forall k : c_k^2 \geq 0$. *Let* $R^{(k)}$ *be a iterate produced by the Algorithm 1 when run with* $r = 1$ *and moreover it is such that* $T_k = 0$. *Then if* RR^T *(where* $R = [R^{(k)}, \mathbf{0}]$*) is a local optimum of [R2BC], then* $R^{(k)}(R^{(k)})^T$ *is a global optimum solution of [R1BC] and [PP4].*

Proof. We will prove this proposition by contradiction. For the sake of contradiction, we assume that R is a local optimum and $R^{(k)}$ is not a global optimum. Therefore, we know that objective function of [R1BC] for $W_1 = W^*$ is smaller than for $W_2 = R^{(k)}(R^{(k)})^T$ and both W_1 and W_2 are feasible. By the assumption on optimum solution W^* of [LL], we know that W^* can be written as $W^* = ww^T$. Now, it is easy to observe that if we define $\tilde{R}^\lambda = [(1-\sqrt{\lambda})R^{(k)}, \sqrt{\lambda}w^*]$ then for any $\lambda \in [0, 1]$ this vector is feasible for [R2BC]. Moreover, for $\lambda = 0$ we have $\tilde{R}^0 = R$. Because the objective function F of [RrBC] is convex in W with $c_k^2 \geq 0$, we have that

$$F(\tilde{R}^\lambda(\tilde{R}^\lambda)^T) \leq (1-\lambda)F(W_2) + \lambda F(W_1) < F(W_2)$$

and hence for all $\lambda \in (0, 1]$ we have that $F(\tilde{R}^\lambda(\tilde{R}^\lambda)^T) < F(W_2)$, which is a contradiction with the assumption that R is a local optimum. □

Overall,

Remark 6. The first iteration of the outer loop of Algorithm 1 may produce a global optimum to [R1] and [RrBC] and [PP4], as suggested in Propositions 1–5. The second and subsequent iterations of the outer loop of Algorithm 1 may find the optimum of [LL], the semidefinite programming problem, as suggested

in Proposition 4, but for $\mathcal{R} = 0$, they are guaranteed not to find the global optimum of [R1] nor [RrBC] nor [PP4].

One should also like contrast the ability to extract low-rank solutions with other methods:

Remark 7. Whenever there are two or more optima of [LL] with two or more distinct ranks, the maximum rank solutions are in the relative interior (of the optimum face of) the feasible set, as per Lemma 1.4 in [12]. Primal-dual interior-point algorithms for semidefinite programming, such as SeDuMi [27], in such a case return a solution with maximum rank.

VI. Numerical Experiments

We implemented Algorithm Schema 1 in C with GSL and OpenMP and tested it on a collection [30] of well-known instances. In Table I, we compare the run-time of our method with the run-time of Matpower and two leading solvers, which use elaborate tree-width decompositions. In order to obtain the numbers, we ran Matpower first using default settings, record the accuracy with respect to squared error T_k (25), ran our solver up to the same accuracy, and record the time and the objective function. For `sdp_pf`, which is developed by Mohlzahn et al. [19] and shipped with Matpower 5 [30], we used default settings and SeDuMi [27]. For `OPF_Solver`, which is developed by Lavaei et al. [16], we used per-instance settings of epB, epL, and line_prob, as suggested by the authors, and SDPT3. Our implementation of Algorithm Schema 1, using default settings, including $\mathcal{R} = 0$, seems to perform well in this comparison.

VII. Conclusions

Our approach seems to bring the use of SDP relaxations of ACOPF closer to the engineering practice. Although a number of authors [3, 18, 21, 11] have recently explored elaborately constructed linear programming and second-order cone programming relaxations, Algorithm Schema 1 suggests that there are simple first-order algorithms, which can solve SDP relaxations of ACOPF within comparable times.

Further major advantage of first-order over second-order methods is their parallelisability. This paper presents a parallel version, but a distributed variant, which would allow for $\mathcal{C}$ agents to perform the iterations, is easy to envision [17]. The $\mathcal{C}$ agents may represent companies, each of whom owns some of the generators and does not want to expose the details, such as cost functions. If $\mathcal{C}$ agents are $\mathcal{C}$ computers, a considerable further speed-up can be obtained.

One can also try to combine first- and second-order methods. We have developed an extension, which allows for tap-changing and phase-shifting transformers, paralllel lines, and multiple generators at one bus, among others, which converges across all of Polish instances. There, we are currently using Smale's $\alpha-\beta$ theory [6] to stop the computation at the point z_0, where we know, based on the analysis of the Lagrangian and its derivatives, that a Newton method or a similar algorithm with quadratic speed of convergence will generate sequence z_i to the correct optimum z^*, i.e.

$$|z_i - z^*| \leq (1/2)^{2^i - i}(z_0 - z^*). \tag{28}$$

This should be seen as convergence-preserving means of auto-tuning of the switch to a second-order method for convex functions. It improves the performance considerably, but introduces much complexity, which is beyond the scope of this paper.

Further extensions include the following regularisation: In [R1], one could drop the rank-one constraint and modify the objective function to penalise the solutions with large ranks, e.g., by adding the term $\lambda\|W\|^*$ to the objective function, where $\|\cdot\|_*$ is a nuclear norm [7] and $\lambda > 0$ is a parameter. Alternatively, one can replace the rank constraint by the requirement that the nuclear norm of the matrix should be small, i.e. $\|W\|_* \leq \lambda$. However, both approaches require a search for a suitable parameter λ such that the optimal solution has indeed rank 1, moreover, the penalised alternative may not produce an optimal solution of [PP4].

Finally, the question as to whether the method could generalise to the higher-order relaxations of Ghaddar et al. [8] remains open. First steps [15] have been taken, but much work remains to be done.

References

[1] X. Bai, H. Wei, K. Fujisawa, and Y. Wang. Semidefinite programming for optimal power flow problems. *International Journal of Electrical Power & Energy Systems*, 30(6):383–392, 2008.

[2] A. Barvinok. A remark on the rank of positive semidefinite matrices subject to affine constraints. *Discrete & Computational Geometry*, 25(1):23–31, 2001.

[3] D. Bienstock and G. Munoz. LP approximations to mixed-integer polynomial optimization problems. *CoRR*, abs/1501.00288, 2015.

[4] S. Burer and C. Choi. Computational enhancements in low-rank semidefinite programming. *Optimisation Methods and Software*, 21(3):493–512, 2006.

[5] S. Burer and R. D. Monteiro. Local minima and convergence in low-rank semidefinite programming. *Mathematical Programming*, 103(3):427–444, 2005.

TABLE I: The results of our numerical experiments. Dash (–) indicates an error.

Instance		MATPOWER		`sdp_pf`		`OPF_Solver`		Algorithm Schema 1	
Name	Ref.	Obj.	Time [s]	Obj.	Time [s]	Obj.	Time [s]	Obj.	Time [s]
case6ww	[28]	3.144+03	0.114	3.144e+03	0.74	3.144e+03	2.939	3.144e+03	0.260
case14	[30]	8.082e+03	0.201	8.082e+03	0.84	–	–	8.082e+03	0.031
case30	[30]	5.769e+02	0.788	5.769e+02	2.70	5.765e+02	6.928	5.769e+02	0.074
case39	[30]	4.189e+04	0.399	4.189e+04	3.26	4.202e+04	7.004	4.186e+04	0.885
case57	[30]	4.174e+04	0.674	4.174e+04	2.69	–	–	4.174e+04	0.857
case118	[30]	1.297e+05	1.665	1.297e+05	6.57	–	–	1.297e+05	1.967
case300	[30]	7.197e+05	2.410	–	17.68	–	–	7.197e+05	90.103

[6] P. Chen. Approximate zeros of quadratically convergent algorithms. *Mathematics of Computation*, 63(207):247–270, 1994.

[7] M. Fazel. *Matrix rank minimization with applications*. PhD thesis, Stanford University, 2002.

[8] B. Ghaddar, J. Mareček, and M. Mevissen. Optimal power flow as a polynomial optimization problem. *IEEE Trans. Power Syst.*, page to appear, 2015.

[9] L. Grippo, L. Palagi, and V. Piccialli. Necessary and sufficient global optimality conditions for nlp reformulations of linear sdp problems. *Journal of Global Optimization*, 44(3):339–348, 2009.

[10] C. Josz and D. Henrion. Strong duality in Lasserre's hierarchy for polynomial optimization. *Optimization Letters*, pages 1–8, 2015.

[11] B. Kocuk, S. Dey, and X. Sun. Inexactness of SDP relaxation and valid inequalities for optimal power flow. *IEEE Trans. Power Syst.*, PP(99):1–10, 2015.

[12] M. Laurent. Sums of squares, moment matrices and optimization over polynomials. In *Emerging applications of algebraic geometry*, pages 157–270. Springer, 2009.

[13] J. Lavaei and S. Low. Zero duality gap in optimal power flow problem. *IEEE T. Power Syst.*, 27(1):92–107, 2012.

[14] K. Lehmann, A. Grastien, and P. Van Hentenryck. AC-feasibility on tree networks is NP-hard. *IEEE Trans. Power Syst.*, (to appear), 2015.

[15] W.-J. Ma. *Control, Learning, and Optimization for Smart Power Grids*. PhD thesis, The University of Notre Dame, 2015.

[16] R. Madani, M. Ashraphijuo, and J. Lavaei. Promises of conic relaxation for contingency-constrained optimal power flow problem. *IEEE Trans. Power Syst.*, PP(99):1–11, 2015.

[17] J. Mareček, P. Richtárik, and M. Takáč. Distributed block coordinate descent for minimizing partially separable functions. In *Recent Developments in Numerical Analysis and Optimization*. Springer, 2015.

[18] D. Molzahn and I. Hiskens. Sparsity-exploiting moment-based relaxations of the optimal power flow problem. *IEEE Trans. Power Syst.*, (too appear), 2014.

[19] D. Molzahn, J. Holzer, B. Lesieutre, and C. DeMarco. Implementation of a large-scale optimal power flow solver based on semidefinite programming. *IEEE T. Power Syst.*, 28(4):3987–3998, 2013.

[20] D. Molzahn, B. Lesieutre, and C. DeMarco. A sufficient condition for global optimality of solutions to the optimal power flow problem. *IEEE Trans. Power Syst.*, 29(2):978–979, March 2014.

[21] D. K. Molzahn and I. A. Hiskens. Mixed SDP/SOCP moment relaxations of the optimal power flow problem. In *PowerTech Eindhoven 2015*.

[22] K. G. Murty and S. N. Kabadi. Some np-complete problems in quadratic and nonlinear programming. *Mathematical programming*, 39(2):117–129, 1987.

[23] Y. Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM J. Optimiz.*, 22(2):341–362, 2012.

[24] G. Pataki. On the rank of extreme matrices in semidefinite programs and the multiplicity of optimal eigenvalues. *Mathematics of operations research*, 23(2):339–358, 1998.

[25] M. J. Powell. A fast algorithm for nonlinearly constrained optimization calculations. In *Numerical analysis*, pages 144–157. Springer, 1978.

[26] P. Richtárik and M. Takáč. Parallel coordinate descent methods for big data optimization. *Mathematical Programming*, pages 1–52, 2015.

[27] J. F. Sturm. Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones. *Optimization methods and software*, 11(1-4):625–653, 1999.

[28] A. Wood and B. Wollenberg. *Power Generation, Operation, and Control*. A Wiley-Interscience publication. Wiley, 1996.

[29] S. Wright. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, 2015.

[30] R. D. Zimmerman, C. E. Murillo-Sánchez, and R. J. Thomas. Matpower: Steady-state operations, planning and analysis tools for power systems research and education. *IEEE T. Power Syst.*, 26(1):12–19, 2011.