SDP Solver of Optimal Power Flow
User's Manual

Ramtin Madani, Morteza Ashraphijuo and Javad Lavaei

1 Introduction

This solver aims to solve the optimal power flow (OPF) problem by means of the semidefinite programming (SDP) relaxation method. The solver is able to solve two problems: (i) an SDP relaxation of OPF leading to a lower bound on the optimal cost [1] and (ii) a penalized convex relaxation leading to an approximate (feasible) solution [2, 3]. The goal is to seek a global or near-global solution with a guaranteed global optimality degree for the nonconvex OPF problem. The solver rests upon a graph decomposition technique employed for three purposes: (i) to accelerate the computation, (ii) to detect the problematic lines of the network causing a nonzero duality gap for OPF, and (iii) to design a penalization term assisting with the recovery of a near-global solution for the cases where duality gap is nonzero.

Let $\mathbf{W}$ denote the SDP matrix to be found. The main complicating constraint of the SDP relaxation is the matrix inequality $\mathbf{W} \succeq \mathbf{0}$ [1]. The SDP relaxation is exact—leading to a global solution of OPF—if it has a rank-1 solution $\mathbf{W}^{\text{opt}}$. Given a power network, the solver first designs a tree decomposition for the power graph with a low maximal clique order. Each node of the tree decomposition is a group of buses of the network, which we refer to as a *bag*. Let $B_1, B_2, ..., B_k$ denote the bags of the tree decomposition. The computationally-expensive condition $\mathbf{W} \succeq \mathbf{0}$ in the SDP relaxation can be equivalently replaced by the cheaper conditions $\mathbf{W}\{B_i\} \succeq 0$ for $i = 1, ..., k$, where $\mathbf{W}\{B_i\}$ is a submatrix of $\mathbf{W}$ induced by the vertices in the bag B_i. For more details on the algorithm used to obtain a tree decomposition, please see [4]. As discussed in [2], the SDP relaxation is exact if the submatrices $\mathbf{W}\{B_1\}, \mathbf{W}\{B_2\}, ..., \mathbf{W}\{B_k\}$ are all rank-1 at optimality. OPF Solver first detects every **problematic bag** B_i—coresponding to a submatrix $\mathbf{W}\{B_i\}$'s with a rank higher than 1—and then identifies all lines of the network which appear in the problematic bags. Each such line is referred to as a **problematic line**.

To enforce the submatrices $\mathbf{W}\{B_1\}, \mathbf{W}\{B_2\}, ..., \mathbf{W}\{B_k\}$ to become rank-1 at optimality, OPF Solver allows a penalization of the total generating reactive power (or equivalently the reactive loss of the network) [3]. After solving this problem, the solver identifies all problematic lines of the network and then allows to further penalize the SDP relaxation via the total loss of apparent power flows over a set of user-defined or software-detected problematic lines. OPF Solver accepts a two-tier objective function (consisting of the cost function and two penalty terms) for its penalized SDP relaxation:

$$\sum_{k\in\mathcal{G}} f_k(P_{G_k}) + \epsilon_b \sum_{k\in\mathcal{G}} Q_{G_k} + \epsilon_l \sum_{(l,m)\in\mathcal{L}_0} |S_{lm} + S_{ml}| \tag{1}$$

where the sum represents the total active power generation cost, a penalized reactive loss of all lines, and a penalized apparent loss of the problematic lines (the set of such lines is denoted as $\mathcal{L}_0$). A customary SDP relaxation can be obtained by setting ϵ_b and ϵ_l to zero.

1.1 How to use this solver?

In order to run OPF Solver, you need to have CVX v2.1 installed on your computer. One of the three common SDP solvers of Mosek, SeDumi and SDPT3 should be called in CVX to numerically solve the SDP relaxation. SDPT3 can be used to find a solution with a high precision for a large-scale OPF problem. Mosek is usually noticeably faster than SDPT3 but the quality of the solution is lower. In order to obtain an accurate solution for a large-scale OPF via SDPT3, the files

```
sqlpcheckconvg.m
sqlpmain.m
```

should be tweaked in order to disable the stopping criteria of SDPT3. Please copy and replace these two files from the folder

```
...\Tweaked SDPT3 files
```

to the place where CVX is installed at

```
...\cvx\sdpt3\Solver
```

after getting a backup for yourself. If you have installed SDPT3 separately, you have to change the MATLAB path so that the SDP relaxation is handled by the SDPT3 inside CVX's folder (as opposed to another copy of the SDPT3).

OPF Solver uses Matpower data format [5]. Make sure that you either have the folder 'Test cases' or the folder of Matpower in the MATLAB path. To test whether the options of SDPT3 has been correctly manipulated, please run the script

```
clear settings;
settings.alpha = 1;
settings.line_prob = 'all';
settings.epL = 10000;
settings.tol_feas = 1.5 * 10^(-5);
results = OPF_Solver('case3012wp',settings);
```

You should observe two runs of SDP problems, each taking a few minutes, where the first run ends after exactly 120 iterations.

In order to solve an OPF problem with the input data stored in a file named 'case.m', the following command can be used:

```
results = OPF_Solver('case',settings);
```

where 'settings' is optional and described in Table 1. If you only need to solve a simple SDP relaxation of OPF without any penalization, then it suffices to run the following line:

```
results = OPF_Solver('case');
```

The content of 'results' is also described in Table 2.

Optional settings	**Description**	**Default values**
`settings.ebB`	The parameter ϵ_b for the penalty term '$\epsilon_b \sum Q_{G_k}$'.	0
`settings.ebL`	The parameter ϵ_l for the penalty term '$\epsilon_l \sum \lvert S_{lm} + S_{ml} \rvert$'.	0
`settings.prob_line`	List of lines whose apparent power loss should be penalized. To penalize all of the lines, set it to '`all`'.	None
`settings.alpha`	The parameter α in the objective '$\alpha \times$ Degree + Fill_In' of the greedy algorithm for tree decomposition.	0
`settings.solver`	The custom solver. '`sdpt3`', '`sedumi`' and '`Mosek`' can be chosen.	'`sdpt3`' for TW $\leq$ 24 and '`Mosek`' for TW $>$ 24.
`settings.tol_feas`	Feasibility tolerance for each constraint.	10^{-6}
`settings.tol_rank`	A bag with the second largest eigenvalue greater than this tolerance will be considered as a problematic bag.	10^{-6}

Table 1: Description of the optional settings of OPF Solver.

Results	**Description**
`results.sdp.W`	The optimal sparse matrix **W** obtained by solving the SDP
`results.sdp.Wbag`	The principal submatrices of **W** corresponding to the bags of tree decomposition
`results.sdp.cost`	The resulting power generating cost obtained by solving the SDP
`results.sdp.Sg`	Production levels of generators obtained from SDP
`results.sdp.Sb`	Production levels of buses obtained from SDP
`results.sdp.sf`	Line flows from initial nodes to terminal nodes obtained from SDP
`results.sdp.st`	Line flows from terminal nodes to initial nodes obtained from SDP
`results.rec.V`	The recovered voltage phasors
`results.rec.cost`	The power generating cost corresponding to the recovered voltages
`results.rec.Sg`	Production levels of generators corresponding to the recovered voltages
`results.rec.Sb`	Production levels at buses corresponding to the recovered voltages
`results.rec.sf`	Line flows from initial nodes to terminal nodes resulted from the recovered voltages
`results.rec.st`	Line flows from terminal nodes to initial nodes resulted from the recovered voltages
`results.tw`	Upper bound on the treewidth of the network
`results.bags`	The bags of the tree decomposition
`results.bags_prob`	List of those bags whose corresponding submatrices of **W** are not rank-1
`results.line_prob`	List of those lines that are the member of at least one problematic bag
`results.violations`	Violated OPF constraints
`results.feas_flag`	Is '1' if all constraints are satisfied for the recovered solution (with the preset accuracy) and is '0' otherwise

Table 2: Description of the outcome of OPF Solver

2 Case studies

OPF Solver is tested on several systems and the results are reported in Table 3. For each system, the following numbers are reported:

- TW: an upper bound on treewdith
- # prob. bags: number of problematic bags
- Lower bound: lower bound on the globally-optimal cost of OPF
- Upper Bound: upper bound on the global cost of OPF associated with the recovered solution
- Opt. degree: global optimality degree interpreted as the closeness of the recovered solution to the unknown global solution (in percentage)
- Com. time: the total computation time (in seconds) including those consumed towards tree decomposition, finding a $\mathbf{W}$, and recovering a solution (the simulations were run on a desktop computer with an Intel Core i7 quad-core 3.4 GHz CPU and 16 GB RAM).

Note that the permissible feasibility violation for the recovered solution is equal to 10^{-6} for each of the cases reported in Table 3, except for Polish 3012wp and Polish 3120sp for which the violation level is set as 1.5×10^{-5}.

Test cases	**TW**	**# prob. bags**	ϵ_b	ϵ_l	**Lower bound**	**Upper bound**	**Opt. degree**	**Com. time**
IEEE 14 bus	2	0	0	0	8081.53	8081.53	%100	≤ 5
IEEE 30 bus	3	1	0.1	0	576.89	576.89	%100	≤ 5
NE 39 bus	3	1	10	0	41862.08	41864.40	%99.994	≤ 5
IEEE 57 bus	5	0	0	0	41737.78	41737.78	%100	≤ 5
IEEE 118 bus	4	61	10	0	129654.61	129660.81	%99.995	≤ 5
IEEE 300 bus	6	7	0.1	100	719711.63	719725.10	%99.998	13.9
Polish 2383wp	23	651	3500	3000	1861510.42	1874322.65	%99.316	529
Polish 2736sp	23	1	1500	0	1307882.29	1308270.20	%99.970	701
Polish 2737sop	23	3	1000	0	777626.26	777664.02	%99.995	675
Polish 2746wop	23	1	1000	0	1208273.91	1208453.93	%99.985	801
Polish 2746wp	24	1	1000	0	1631772.83	1632384.87	%99.962	699
Polish 3012wp	24	605	0	10000	2587740.98	2608918.45	%99.188	814
Polish 3120sp	24	20	0	10000	2140765.92	2160800.42	%99.073	910

Table 3: Performance of OPF Solver on several case studies.

In what follows, the scripts used to generate the above results will be provided. As an example, the command ‘`results = OPF_Solver('case14');`’ solves the OPF problem for the IEEE 14 bus system. For some networks such as the 300-bus system, there are two ways to solve the problem. The first one assumes that the problematic lines are unknown and therefore the SDP relaxation is solved twice (to find the problematic lines in the first stage and solve a penalized relaxation based on that in the second stage). The second script assumes that the problematic lines are known and the SDP relaxation is then solved once. Please refer to [2] for more details.

IEEE 14 bus system:

```
results = OPF_Solver('case14');
```

IEEE 30 bus system:

```
clear settings;
settings.epB = 0.1;
results = OPF_Solver('case30',settings);
```

New England 39 bus system:

```
clear settings;
settings.epB = 10;
results = OPF_Solver('case39',settings);
```

IEEE 57 bus system:

```
results = OPF_Solver('case57');
```

IEEE 118 bus system:

```
clear settings;
settings.epB = 10;
results = OPF_Solver('case118',settings);
```

IEEE 300 bus system:

```
clear settings;
settings.epB = 0.1;
settings.epL = 100;
settings.line_prob = [38; 402];
results = OPF_Solver('case300',settings);
```

Penalized SDP with automatic problematic line detection:

```
clear settings;
settings.epB = 0.1;
settings.epL = 100;
results = OPF_Solver('case300',settings);
settings.line_prob = results.line_prob;
results = OPF_Solver('case300',settings);
```

Polish 2383 bus system winter 1999-2000 peak:

```
clear settings;
settings.epB = 3500;
settings.epL = 3000;
settings.line_prob = [100; 101; 102; 103; ...
    104; 130; 134; 819; 821];
results = OPF_Solver('case2383wp',settings);
```

Penalized SDP with automatic problematic line detection:

```
clear settings;
settings.epB = 3500;
settings.epL = 3000;
results = OPF_Solver('case2383wp',settings);
settings.line_prob = results.line_prob;
results = OPF_Solver('case2383wp',settings);
```

Polish 2736 bus system summer 2004 peak:

```
clear settings;
settings.epB = 1500;
results = OPF_Solver('case2736sp',settings);
```

Polish 2737 bus system summer 2004 off-peak:

```
clear settings;
settings.epB = 1000;
results = OPF_Solver('case2737sop',settings);
```

Polish 2746 bus system winter 2003-04 off-peak:

```
clear settings;
settings.epB = 1000;
results = OPF_Solver('case2746wop',settings);
```

Polish 2746 bus system winter 2003-04 evening peak:

```
clear settings;
settings.epB = 1000;
results = OPF_Solver('case2746wp',settings);
```

Polish 3012 bus system winter 2007-08 evening peak:

```
clear settings;
settings.alpha = 1;
settings.line_prob = 'all';
settings.epL = 10000;
settings.tol_feas = 1.5 * 10^(-5);
results = OPF_Solver('case3012wp',settings);
```

Polish 3120 bus system summer 2008 morning peak:

```
clear settings;
settings.alpha = -1.5;
settings.line_prob = 'all';
settings.epL = 10000;
settings.tol_feas = 1.5 * 10^(-5);
results = OPF_Solver('case3120sp',settings);
```

References

[1] J. Lavaei and S. Low, "Zero duality gap in optimal power flow problem," *IEEE Transactions on Power Systems*, 2012.

[2] R. Madani, M. Ashraphijuo, and J. Lavaei, "Promises of conic relaxation for contingency-constrained optimal power flow problem," in *Allerton*, 2014.

[3] R. Madani, S. Sojoudi, and J. Lavaei, "Convex relaxation for optimal power flow problem: mesh networks," in *IEEE Transactions on Power Systems*, 2014.

[4] H. L. Bodlaender and A. M. Koster, "Treewidth computations i. upper bounds," *Information and Computation*, vol. 208, no. 3, pp. 259–275, 2010.

[5] R. D. Zimmerman and C. E. Murillo-Sánchez, "Matpower 4.1 users manual," *Power Systems Engineering Research Center (PSERC)*, 2011.