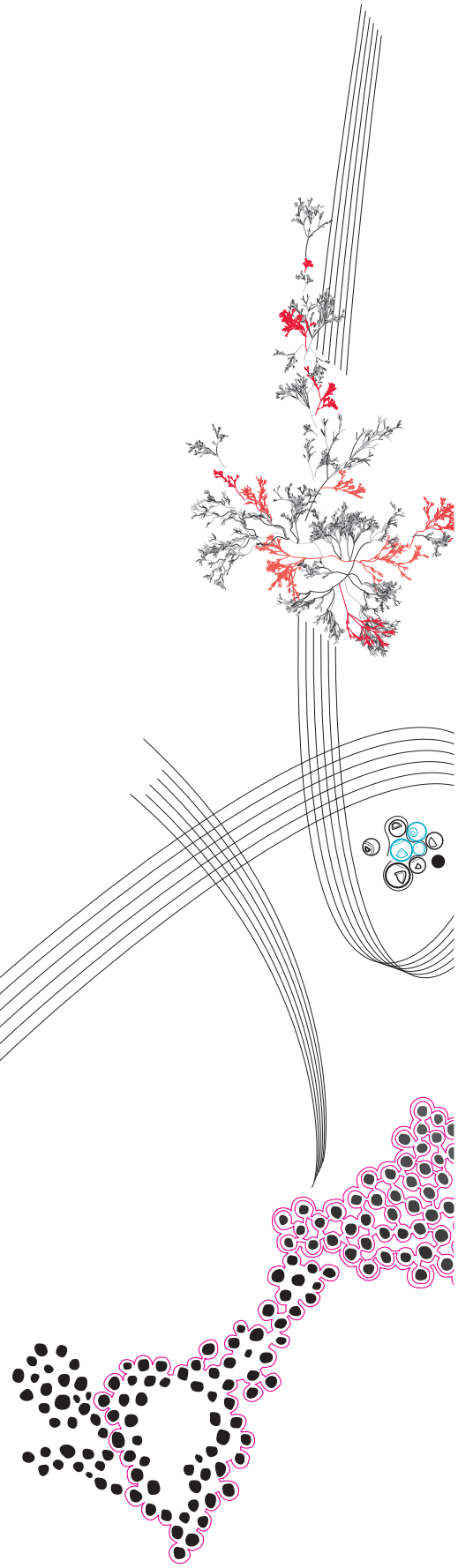

BSc Thesis Applied Mathematics

Nonlinear convex optimisation problems in the smart grid

Jarco Slager

Supervisors: Ir. M. H. H. Schoot Uiterkamp and Prof. Dr. J.L. Hurink

June, 2019

Department of Applied Mathematics
Faculty of Electrical Engineering,
Mathematics and Computer Science

UNIVERSITY OF TWENTE.

Nonlinear convex optimisatoin problems in the smart grid

Jarco Slager *

June, 2019

Abstract

The energy that is generated for our society is shifting from fossil fuels to renewable energy sources (i.e. wind and solar power). Due to the dependence on the sun and wind, power cannot always be generated when needed. To tackle this problem, smart grids have been introduced, in which we are able to store energy in and receive energy from smart appliances in the energy system. In order to distribute the energy efficiently, we formulate this distribution problem as a mathematical optimisation problem with the objective to minimise for example the power loss. These optimisation problems are nonlinear and convex and therefore solved via the primal-dual interior-point method (IPM). Even though this method is efficient, the algorithm will be run on embedded systems with low computational power. Therefore a gain in efficiency is desired.

We discuss two models that arise in the smart grid, and show that the problem structure can be exploited to improve the computational complexity of the IPM algorithm. For both models, the time consuming step in the algorithm has been reduced from a worst-case time complexity of $O(n^3)$ to $O(n)$. Numerical tests for one of the models confirms this gain in efficiency in practice.

Keywords: interior-point method, smart grid, resource allocation problem, EV charging problem

1 Introduction

Our society is moving towards an energy system that contains more and more renewable energy sources, for instance wind and solar power. In the conventional way of power production (e.g. generating power in coal-fired plants), the energy supply can be adjusted relatively adequately to the energy demand. However, the energy supply of renewable energy sources is not that smoothly adjustable to the energy demand. Solar power can only be generated when the sun is shining, and wind power only when there is wind to put wind turbines in motion. Hence, this change in power production impacts our energy distribution system [15].

Currently, our society operates with a centralised energy management system, which means that the power is generated in power plants. However, there is a shift towards a decentralised energy management system, in which the power is generated locally, at a consumer's home. This leads to the concept of a smart grid. In a smart grid, smart appliances such as electrical vehicles (EVs) are included in the energy system. Here, these appliances should be able to use/store energy when much power is being produced. On the other hand, when power is required by for example consumers, these appliances should be able

*Email: j.slager@student.utwente.nl

to provide (a part of) their stored energy back to the system. In this new energy system, the power should be distributed in a clever way, with the objective to minimise, for example, the power loss. To solve this problem, it is often formulated and solved as a mathematical optimisation problem.

The optimisation problems that arise in this context are resource allocation problems. In a resource allocation problem, limited resources must be allocated among several activities [16]. This is also a description of the so-called EV charging problem, in which energy has to be allocated to the battery of the EV, subject to some constraints. The optimisation problems that emerge in these situations are usually convex and nonlinear. Therefore, nonlinear optimisation methods are used to compute the solutions to these problems, for instance the barrier method or the primal-dual interior-point method (IPM) [1]. These existing methods have been proven to be fast and efficient. However, for the application in smart grids, the algorithms are required to run on embedded systems with low computational power. Therefore we desire any gain in efficiency. The primal-dual IPM is a general algorithm to solve nonlinear and convex optimisation problems, and therefore does not exploit any specific problem structure. Thus, in this paper we aim to analyse the most time-consuming steps in the primal-dual IPM algorithm and investigate whether the specific problem structure of the smart grid-related problems can be exploited to speed up these steps.

In this paper, we formulate two EV charging problems which are solved with the primal-dual IPM. Whereas in the standard primal-dual IPM algorithm a search direction is computed by solving a system of linear equations which has a worst-case time complexity of $O(n^3)$, we analytically derive an explicit formula for finding this search direction that takes only $O(n)$ time to evaluate for both of the EV charging problems. Numerical results for one of these problems confirm the analytic results, with a reduction in computation time of approximately $97,1\%$ compared to the original primal-dual IPM algorithm.

The remainder of this paper is organised as follows. In Section 2 we in particular discuss other research that tries to find efficient methods for solving resource allocation problems. In Section 3 we explain the mathematical models that we study in this paper. In Section 4 we discuss several aspects of the primal-dual IPM. Based on this, we use the problem structure of the models to derive an expression for the search direction that can be evaluated in $O(n)$ time in Section 5. In Section 6 we state the numerical results, on which we comment briefly in Section 7. Finally, we end with concluding remarks and future work in Section 8.

2 Related work

The fundamental work for this research is that of Van der Klauw [15]. This paper explains the two specific problems that arise in decentralised energy management (those of the EV-charging problems). It provides a recursive algorithm to solve resource allocation problems that have cumulative bounds (see the constraints of the CRA problem, Section 3.2). Since this is a general solution method, specific forms of objective functions (quadratic) have not been exploited for improving computational complexity.

Due to the nature of the EV-charging problem, the objective functions are quadratic and separable (this will be dealt with in Section 3). Research on resource allocation problems

with quadratic objective functions has been done, in which efficient algorithms were developed that have a linear worst-time complexity [2][13][14]. In the work of Robinson [13], they approach the problem by looking at the geometric interpretation that the optimal value x^* can be seen as an orthogonal projection of the origin onto an intersection of hyperplanes and hyperspaces that arise from the constraints. A different approach is taken by Stefanov [14], in which the author relies on the Karush-Kuhn-Tucker (KKT) conditions for optimality (see Section 4). Even though we restrict ourselves to the use of primal-dual IPMs, these papers show that there are other ways to solve the EV-charing problem.

The first IPM that was able to compete with the simplex method in linear programming was developped by Karmarkar in 1984 [8]. They can now also be used in nonlinear programming problems. Since Karmarkar's work in 1984, IPMs have been developed further, and their applications and performance have been growing ever since [17]. Since an IPM is an iterative algorithm in which the number of iterations needed to solve the problem depends very little on the problem dimension, it is in particular very suitable for large scale problems [4]. The most prevalent IPM is the primal-dual IPM. It is often more efficient than the barrier method (a different IPM), and regarding quadratic programming, its performance is superior to that of the barrier method, as stated in the book of Boyd and Vandenberghe [1]. For both the barrier IPM as the primal-dual IPM, an algorithm can be found in the book of Boyd and Vandenberghe.

Besides the models for the EV-charging problem stated by Van der Klauw [15], other models for this problem have been developed and solved by IPMs. The work in [5] tries to minimise the total cost of all EVs that charge and discharge during the day. Instead of optimising the situation for one EV, they optimise over all vehicles that arrive and depart during a single day. In this paper, the structure that might be present in the problem is not utilised, and the standard primal-dual IPM from Boyd and Vandenberghe [1] is used to solve the optimisation problem.

In general, the approach for modelling the allocation of energy from renewable energy sources to EVs in the smart grid is based on the goal of reducing energy loss in the system. Another view is taken in the research of Jin [7]. They look at a customer's perspective of the problem, creating a queueing model for the scheduling problem. This is a different approach than what is commonly done and thus requires a different solution method. They use Lyapunov optimisation, which is an optimisation technique used in queueing networks [11]. This is a completely different approach to deal with smart grid-related optimisation problems, but it does create additional opportunities for computations in the smart grid.

3 Background information - Model(s)

In this section, we discuss the optimisation problems that are studied in the paper of Van der Klauw [15]. Two problems are considered which are explained in Section 3.1 and Section 3.2, respectively. We first provide the context of the problem. Since we are dealing with a problem in electrical networks, we discuss quantities of energy. For the purpose of applications, the unit of the amount of energy is chosen to be kWh.

We consider the case in which an EV arrives at a certain point in time in an electrical system. The total time that the EV is in the system is divided into n intervals. For $i \in J = \{1, 2, \ldots, n\}$, let x_i denote the amount of energy to be provided to the EV in

interval i. We assume that, due to practical constraints of the EV, this quantity x_i can be at most u_i (e.g. the maximum charging rate in interval i).

Furthermore, we associate a cost function $f_i(x_i)$ with each interval i, which depends on the desired amount of energy in interval i, i.e. it depends on x_i. From practice, we can assume that the functions f_i are convex, for example linear or quadratic [15]. In this paper, we assume the cost functions to be of the form $f_i(x_i) = \frac{1}{2}x_i^2 - c_i x_i$ with $c_i \in \mathbb{R}$. Let $x \in \mathbb{R}^n$ and $c \in \mathbb{R}^n$ with components x_i and c_i respectively, for $i \in J$. Now, we are able to formulate the objective function

$$f(x) = \sum_{i=1}^{n} f_i(x_i) = \frac{1}{2}x^T x - c^T x. \tag{1}$$

We note that the result could be generalised to the case where $f_i(x_i) = \frac{1}{2}a_i x_i^2 - c_i x_i$, with $a_i > 0,\ i \in J$. However, to maintain rentability of this paper, we assume that $a_i = 1,\ i \in J$.

Now we will specify the details of two versions of the EV charging problem.

3.1 The SRA model

The first EV-charing model is the simple resource allocation (SRA) model. Suppose the battery of the EV has capacity C. Upon departure of the EV from the smart grid, i.e. after n intervals, the battery is required to be fully charged. In addition, no energy can be taken from the battery, i.e. x_i is non-negative for each interval i. The objective function for the SRA model is given by Equation (1). Thus, the SRA model corresponds to the following minimisation problem:

$$\begin{aligned} \min_x\ & f(x) = \frac{1}{2}x^T x - c^T x \\ \text{s.t.}\ & \sum_{i=1}^{n} x_i = C \\ & 0 \le x_i \le u_i,\ i \in J. \end{aligned} \tag{SRA}$$

3.2 The CRA model

The second model for the EV charging problem is the cumulative resource allocation (CRA) model. In general, the battery of the EV is able to store more than the required amount of energy for the next travel, which makes room for the EV to supply energy to the smart grid, commonly known as V2G (vehicle-to-grid) [9]. Therefore, the x_i's are allowed to attain negative values in this model. The sum of the x_i's over the first j intervals has to be at least B_j (the cumulative demand of energy up to interval j) and at most C_j (the storage capacity of the battery, minus the cumulative demand up to inverval j). The objective function remains as in Equation (1). This allows us to write the CRA model as the following minimsation problem:

$$\begin{aligned}
&\min_x f(x) = \frac{1}{2}x^T x - c^T x \\
&\text{s.t. } B_j \leq \sum_{i=1}^{j} x_i \leq C_j,\ j \in J \\
&\qquad l_i \leq x_i \leq u_i,\ i \in J,
\end{aligned}$$

where l_i is maximum amount of energy that can be delivered to the energy network in interval i. We note that we can shift every x_i such that we attain the form of $x_i \geq 0$. After finding a solution to this shifted problem, the inverse transformation should be applied to obtain the real solution. Thus, for the analysis we restrict ourselves to the transformed problem given in problem (CRA):

$$\begin{aligned}
&\min_x f(x) = \frac{1}{2}x^T x - c^T x \\
&\text{s.t. } B_j \leq \sum_{i=1}^{j} x_i \leq C_j,\ j \in J \\
&\qquad 0 \leq x_i \leq u_i,\ i \in J.
\end{aligned} \tag{CRA}$$

4 Methods

In this section, we will how the problems given in Section 3 can be solved. In Section 4.1 and 4.2 we state the KKT and the perturbed KKT conditions respectively, which are needed for the understanding of the primal-dual IPM given in Section 4.3.

4.1 The general KKT conditions

In order to solve the an optimisation problem with IPMs, the KKT conditions are exploited (see for example the barrier method [1]). The KKT conditions are necessary for optimal solutions in nonlinear programming and are based on properties of primal and dual problems. Consider the following general problem for some vector $x \in \mathbb{R}^n$, a general matrix $A \in \mathbb{R}^{p \times n}$ and convex functions $f : \mathbb{R}^n \to \mathbb{R}$ and $g_i(x) : \mathbb{R}^n \to \mathbb{R}$ with $i \in \{1, 2, \ldots, k\}$:

$$\begin{aligned}
&\min_x f(x) \\
&\text{s.t. } g_i(x) \leq 0,\ i \in \{1, 2, ..., k\} \\
&\qquad Ax = b.
\end{aligned} \tag{GP}$$

Suppose that (λ, v) are the dual variables that correspond to the dual problem of problem (GP), with $\lambda \in \mathbb{R}^k$ and $v \in \mathbb{R}^p$. Then the KKT conditions for problem (GP) are given by the four following points:

1. Stationarity: $\nabla f(x) + \sum_{i=1}^{k} \lambda_i \nabla g_i(x) + A^T v = 0$.
2. Complementary slackness: $\lambda_i g_i(x) = 0 \quad \forall i$.
3. Primal feasibility: $g_i(x) \leq 0$ and $Ax - b = 0 \quad \forall i$.

4. Dual feasibility: $\lambda_i \geq 0 \quad \forall i$.

These conditions are both necessary and sufficient for optimal primal and dual variables since the functions are convex [1]. More precisely, let x^*, λ^* and v^* be the primal and dual optimal variables, respectively. Then the four KKT conditions hold if and only if $x = x^*$, $\lambda = \lambda^*$ and $v = v^*$.

4.2 The KKT Conditions for Primal-Dual IPMs

The primal-dual IPM is used for solving the optimisation problems in the smart grid. In Section 4.1 we discussed the KKT conditions that are applied in the barrier method. For the primal-dual IPM are changed slightly. We consider the problem (GP), for which the perturbed KKT conditions are given as follows:

1. Stationarity: $\nabla f(x) + \sum_{i=1}^{k} \lambda_i \nabla g_i(x) + A^T v = 0$.
2. Complementary slackness: $\lambda_i g_i(x) = -\frac{1}{t} \quad \forall i$ and some $t > 0$.
3. Primal feasibility: $g_i(x) \leq 0$ and $Ax - b = 0 \quad \forall i$.
4. Dual feasibility: $\lambda_i \geq 0 \quad \forall i$.

The perturbed KKT conditions are very similar to those posed in Section 4.1, however the second condition, the complementary slackness, has been modified slightly. It now states that $\lambda_i g_i(x)$ is not necessarily equal to zero, but some (small) value $-\frac{1}{t}$. The minus sign comes from the fact that the primal-dual IPM maintains strict feasibility; this implies that for all i we have $g_i(x) < 0$ and $\lambda_i > 0$ and hence their product is negative. The primal-dual IPM considers the equations in the first three modified KKT conditions and tries to find a solution iteratively such that those hold. In each iteration, the value of t is increased and we note that as $t \to \infty$, the perturbed KKT conditions resemble the normal KKT conditions more and more. The strict feasibility of the problem is enforced by using a line search algorithm. More explanation on this topic is given in Section 4.3.

4.3 The Primal-Dual IPM

Now that the perturbed KKT conditions have been obtained in Section 4.2, we will set up the system that the primal-dual IPM tries to solve. First, we define the following notation:

$$g(x) = \begin{pmatrix} g_1(x) \\ g_2(x) \\ \vdots \\ g_k(x) \end{pmatrix}, \qquad Dg(x) = \begin{pmatrix} \nabla g_1(x)^T \\ \nabla g_2(x)^T \\ \vdots \\ \nabla g_k(x)^T \end{pmatrix}. \tag{2}$$

Using the notation in Equation (2), we can express the first three perturbed KKT conditions by defining $r_t(x, \lambda, v) = 0$, or

$$r_t(x, \lambda, v) = \begin{bmatrix} \nabla f(x) + Dg(x)^T \lambda + A^T v \\ -\mathrm{diag}(\lambda) g(x) - \frac{1}{t} \mathbb{1}_k \\ Ax - b \end{bmatrix} = \begin{bmatrix} r_{\text{dual}} \\ r_{\text{cent}} \\ r_{\text{prim}} \end{bmatrix} = 0, \tag{3}$$

with $t > 0$ and $\mathbb{1}_k = [1, 1, \cdots, 1]^T \in \mathbb{R}^k$ and we let $\mathbb{1} = \mathbb{1}_n$. We call r_t the residual vector, in which the three components denote the so-called dual, central and primal residual. In

the case that x, λ and v satisfy $r_t(x, \lambda, v) = 0$ (and $g_i(x) < 0$, $\lambda_i > 0$), then $x = x^*(t)$, $\lambda = \lambda^*(t)$ and $v = v^*(t)$ are optimal [1]. The strict inequalities for $g_i(x)$ and λ_i hold since the primal-dual IPM works with strictly feasible solutions. Since in general Equation (3) yields a nonlinear system that we would like to solve, we first apply a (local) linearisation as an approximation.

Suppose that $r_t(x, \lambda, v) = 0$, for t fixed, at the point (x, λ, v) such that $g_i(x) < 0$ and $\lambda_i > 0$ $\forall i$. Let $y = (x, \lambda, v)$ and $\Delta y = (\Delta x, \Delta\lambda, \Delta v)$ denote the current point and the Newton step (or the search/descent direction), respectively. Hence we obtain the following local linearisation of r_t at the point (x, λ, v):

$$r_t(y + \Delta y) \approx r_t(y) + Dr_t(y)\Delta y = 0.$$

This can be rewritten by using the variables x, λ and v and by bringing $r_t(x, \lambda, v)$ to the other side to arrive at

$$\begin{bmatrix} \nabla^2 f(x) + \sum_{i=1}^{k} \lambda_i \nabla^2 g_i(x) & Dg(x)^T & A^T \\ -\mathrm{diag}(\lambda) Dg(x) & -\mathrm{diag}(g(x)) & 0 \\ A & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \\ \Delta v \end{bmatrix} = - \begin{bmatrix} r_{\text{dual}} \\ r_{\text{cent}} \\ r_{\text{prim}} \end{bmatrix}. \tag{4}$$

The system in Equation (4) can be solved for the primal-dual search direction given by $(\Delta x_{\text{pd}}, \Delta\lambda_{\text{pd}}, \Delta v_{\text{pd}})$. This search direction is computed in each iteration of the primal-dual IPM, which has a worst-case time complexity $O(n^3)$ since a system of linear equations is solved. In iteration j, suppose we are at the point $(x^{(j)}, \lambda^{(j)}, v^{(j)})$. The algorithm updates the current point to the point $(x^{(j+1)}, \lambda^{(j+1)}, v^{(j+1)})$ in the direction given by the search direction. However, the search direction does not take the strict primal and dual feasibility into account. Therefore, this search direction might cause one of the constraints to be violated and hence a maximum step size has to be introduced that gives a restriction on the distance that can be moved into the search direction.

As an illustration, consider the case for λ. Starting with λ strictly feasible (as required in the primal-dual IPM), all the entries λ_i are strictly positive. In the case that for some $i \leq n$ we have $\Delta\lambda_i < 0$ and $|\Delta\lambda_i| \geq |\lambda_i|$, the λ_i will not be strictly feasible in the next iteration. Therefore, the step size s is introduced, which determines how far can be moved into the search direction by maintaining the strict feasibility for the problem. We note that $s \leq 1$ since we move at most one complete step into the search direction.

5 Analytic Results

In this section, we will set up the Equations (3) and (4) for the SRA and CRA problem in the Sections 5.1 and 5.2 respectively. In addition, we will use these equations to find explicit formulas for the search direction.

5.1 The SRA problem

We start with the SRA problem. Let $J = \{1, 2, \ldots, n\}$. We rewrite the SRA problem to match the form in Equations (3) and (4) as follows:

$$
\begin{aligned}
&f(x) = \frac{1}{2}x^T x - c^T x,\\
&A = \mathbb{1}^T,\\
&b = C,
\end{aligned}
$$

$$
g_i(x) = \begin{cases} x_i - u_i & \text{if } i \in J,\\ -x_i & \text{if } i \in \{n+1, n+2, \ldots, 2n\}. \end{cases}
$$

With this information, we can construct the residual $r_t(x, \lambda, v)$ and set it equal to zero (as in Equation (3))

$$
r_t(x, \lambda, v) = \begin{bmatrix} x - c + Dg^T\lambda + \mathbb{1}v \\ -\text{diag}(\lambda)g(x) - \frac{1}{t}\mathbb{1}_{2n} \\ \mathbb{1}^T x - C \end{bmatrix} = \begin{bmatrix} r_{\text{dual}} \\ r_{\text{cent}} \\ r_{\text{prim}} \end{bmatrix} = 0. \tag{5}
$$

Now that we have found the expression for the residuals, we will continue by setting up the system in Equation (4). We first derive expressions for $\nabla^2 f(x)$, $\nabla^2 g_i(x)$ and $Dg(x)$:

$$
\nabla^2 f(x) = I, \tag{6}
$$

$$
\nabla^2 g_i(x) = 0, \tag{7}
$$

$$
Dg(x) = \begin{bmatrix} I \\ -I \end{bmatrix}. \tag{8}
$$

Combining these results with the system of Equation (5), we obtain:

$$
\begin{bmatrix} I & Dg(x)^T & \mathbb{1} \\ -\text{diag}(\lambda)\begin{bmatrix} I \\ -I \end{bmatrix} & -\text{diag}(g(x)) & 0 \\ \mathbb{1}^T & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \\ \Delta v \end{bmatrix} = - \begin{bmatrix} r_{\text{dual}} \\ r_{\text{cent}} \\ r_{\text{prim}} \end{bmatrix}. \tag{9}
$$

The matrix in the linear system of Equation (9) has a rich structure, i.e. a structure that possibly can be exploited. Using row operations, we can derive an explicit solution for the primal-dual search direction Δy. For convenience, in most cases we write $Dg(x)$ instead of $\begin{bmatrix} I \\ -I \end{bmatrix}$. Moreover, we write g_i instead of $g_i(x)$. Finally, we introduce the following notation

$$
\begin{aligned}
&\Lambda_1 = \text{diag}([\lambda_1, \lambda_2, \cdots, \lambda_n]),\\
&\Lambda_2 = \text{diag}([\lambda_{n+1}, \lambda_{n+2}, \cdots, \lambda_{2n}]),\\
&G_1 = \text{diag}([g_1, g_2, \cdots, g_n]),\\
&G_2 = \text{diag}([g_{n+1}, g_{n+2}, \cdots, g_{2n}]),\\
&P = \begin{bmatrix} \Lambda_1 - G_1 & -\Lambda_1 \\ -\Lambda_2 & \Lambda_2 - G_2 \end{bmatrix},\\
&\rho = -n + \mathbb{1}^T Dg(x)^T P^{-1} \begin{bmatrix} \Lambda_1 \\ -\Lambda_2 \end{bmatrix} \mathbb{1}.
\end{aligned}
$$

More explanation on this notation can be found in Appendix A. Let r_{dual}, r_{cent} and r_{prim} be denoted by r_1, r_2 and r_3 respectively. After performing row operations on the system in Equation (9), we obtain a closed form expression for Δy:

$$\begin{bmatrix}\Delta x\\ \Delta\lambda\\ \Delta v\end{bmatrix} = \begin{bmatrix} -r_1 - Dg(x)^T P^{-1}(-r_2 - \begin{bmatrix}\Lambda_1\\ -\Lambda_2\end{bmatrix} r_1) - (\mathbb{1} - Dg(x)^T P^{-1} \begin{bmatrix}\Lambda_1\\ -\Lambda_2\end{bmatrix} \mathbb{1})\Delta v \\ P^{-1}(-r_2 - \begin{bmatrix}\Lambda_1\\ -\Lambda_2\end{bmatrix} r_1) - P^{-1} \begin{bmatrix}\Lambda_1\\ -\Lambda_2\end{bmatrix} \mathbb{1}\Delta v \\ (-r_3 + \mathbb{1}^T r_1 + \mathbb{1}^T Dg(x)^T P^{-1}(-r_2 - \begin{bmatrix}\Lambda_1\\ -\Lambda_2\end{bmatrix} r_1))/\rho \end{bmatrix}. \tag{10}$$

The derivation for this formula can be found in Appendix A. The term $(-r_3 + \mathbb{1}^T r_1 + \mathbb{1}^T Dg(x)^T P^{-1}(-r_2 - \begin{bmatrix}\Lambda_1\\ -\Lambda_2\end{bmatrix} r_1))/\rho$ turns out to be present in all three components Δx, $\Delta\lambda$ and Δv of the search direction. This allows us to compute this value and use it in all three expressions to save computational time. We note that P is a partitioned matrix, consisting of only diagonal matrices. Therefore, an analytic solution can be obtained for its inverse, which is given in Appendix A.

The computations that are time consuming in Equation (10) and the formula for ρ are multiplications of matrices with matrices or matrices with vectors. Since the matrices are either diagonal, consist of diagonal matrices or even consist of identity matrices, this search direction can be computed with a computational complexity of $O(n)$.

5.2 The CRA problem

In this section, we set up the Equations (3) and (4) given in Section 4.3 for the problem given in Section 3.2. We obtain the following form for our problem:

$$f(x) = \frac{1}{2}x^T x - c^T x$$

$$g_i(x) = \begin{cases} x_i - u_i & \text{if } i \in J \\ -x_i & \text{if } i \in \{n+1, n+2, \dots, 2n\} \\ \sum_{k=1}^{i-n} x_k - C_k & \text{if } i \in \{2n+1, n+2, \dots, 3n\} \\ -\sum_{k=1}^{i-2n} x_k + B_k & \text{if } i \in \{3n+1, 2n+2, \dots, 4n\}. \end{cases}$$

Now we can set up the residual function which we call $R_t(x, \lambda)$. Note that the dual variable v is not present, since this problem does not have equality constraints. We obtain the following equation:

$$R_t(x,\lambda) = \begin{bmatrix} Ix - c + Dg(x)^T\lambda \\ -\text{diag}(\lambda)g(x) - \frac{1}{t}\mathbb{1}_{4n}\end{bmatrix} = \begin{bmatrix}R_{\text{dual}}\\ R_{\text{cent}}\end{bmatrix} = 0.$$

The next step is solving this nonlinear system by taking a Newton step in the direction of $(\Delta x, \Delta\lambda)$. Since $f(x)$ is defined as in Section 5.1 and $g_i(x)$ is linear in x, the values of $\nabla^2 f(x)$ and $\nabla^2 g_i(x)$ given by Equations (6) and (7), respectively. However, the Jacobian

matrix $Dg(x)$ has a different form. To derive this form, we define the matrix $L \in \mathbb{R}^{n \times n}$ to be the lower triangular matrix such that $L_{ij} = 1$ for $i \geq j$ and $L_{ij} = 0$ for $i < j$. It can be verified that

$$Dg(x) = \begin{bmatrix} I \\ -I \\ L \\ -L \end{bmatrix},$$

and we note that $Dg(x)$ is a $4n \times n$ matrix.

We obtain the following system of equations after linearisation:

$$\begin{bmatrix} I & Dg(x)^T \\ -\text{diag}(\lambda)Dg(x) & -\text{diag}(g(x)) \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \end{bmatrix} = - \begin{bmatrix} R_{\text{dual}} \\ R_{\text{cent}} \end{bmatrix}. \tag{11}$$

In order to give a compact formula for the search direction $(\Delta x, \Delta \lambda)$, we introduce some notation. We define the following diagonal matrices:

$$\begin{aligned}
\Lambda_1 &= \text{diag}([\lambda_1, \lambda_2, \cdots, \lambda_n]) \\
\Lambda_2 &= \text{diag}([\lambda_{n+1}, \lambda_{n+2}, \cdots, \lambda_{2n}]) \\
\Lambda_3 &= \text{diag}([\lambda_{2n+1}, \lambda_{2n+2}, \cdots, \lambda_{3n}]) \\
\Lambda_4 &= \text{diag}([\lambda_{3n+1}, \lambda_{3n+2}, \cdots, \lambda_{4n}]) \\
G_1 &= \text{diag}([g_1, g_2, \cdots, g_n]) \\
G_2 &= \text{diag}([g_{n+1}, g_{n+2}, \cdots, g_{2n}]) \\
G_3 &= \text{diag}([g_{2n+1}, g_{2n+2}, \cdots, g_{3n}]) \\
G_4 &= \text{diag}([g_{3n+1}, g_{3n+2}, \cdots, g_{4n}]).
\end{aligned}$$

Lastly, we define the matrix N:

$$N = \begin{bmatrix} \Lambda_1 - G_1 & -\Lambda_1 & \Lambda_1 L^T & \Lambda_1 L^T \\ -\Lambda_2 & \Lambda_2 - G_2 & \Lambda_2 L^T & -\Lambda_2 L^T \\ -\Lambda_3 L & \Lambda_3 L & \Lambda_3 L L^T - G_3 & -\Lambda_3 L L^T \\ \Lambda_4 L & -\Lambda_4 L & -\Lambda_4 L L^T & \Lambda_4 L L^T - G_4 \end{bmatrix}. \tag{12}$$

With these additions to our notation, the expression for the search direction, i.e. the solution to the system in Equation (11), is as follows:

$$\begin{bmatrix} \Delta x \\ \Delta \lambda \end{bmatrix} = \begin{bmatrix} -R_{\text{dual}} - Dg(x)^T N^{-1}(-R_{\text{cent}} - \text{diag}(\lambda)Dg(x)R_{\text{dual}}) \\ N^{-1}(-R_{\text{cent}} - \text{diag}(\lambda)Dg(x)R_{\text{dual}}) \end{bmatrix}. \tag{13}$$

The derivation of Equation (13) can be found in Appendix B. Similar to the observation made in Section 5.1, we note that the expression for $N^{-1}(-R_{\text{cent}} - \text{diag}(\lambda)Dg(x)R_{\text{dual}})$ is present in the expression for both Δx and $\Delta \lambda$. This allows us to pre-compute this value and save computational time.

Similar to the case in Section 5.1, the matrix multiplications are the time consuming operations in Equation (13). Computing $N^{-1}w$ for any vector $w \in \mathbb{R}^n$ can be done in a complexity of $O(n)$ (see Appendix C). Therefore, the explicit formula for the search direction $(\Delta x, \Delta \lambda)$ has a worst-case complexity of $O(n)$.

6 Numerical results of the SRA problem

In this section, we focus on numerical experiments for the SRA problem. We first give a pseudo-code for the primal-dual IPM. Afterwards, we will investigate whether the formula for the search direction given in Equation (10) reduces the computation time of the algorithm in the case where the parameters originate from the real-world problem. Finally, we compare the computation time of the IPM with the explicit solution (the explicit method) for the search direction with the computation time of the IPM with the standard solver for the search direction (the standard method) when we increase the scale of the problem.

6.1 Pseudo-code for the primal-dual IPM

In this section, we provide and explain the pseudo-code for the primal-dual IPM. In Algorithm 1 , we state the pseudo-code:

Algorithm 1: Pseudo-code for the primal-dual IPM.

Start with a strict feasible point $x^{(0)}$ with $g_i(x^{(0)}) < 0$, $\lambda_i^{(0)} > 0 \; \forall i$ and $v^{(0)}$.
Initialise the surrogate duality gap $\eta^{(0)} = -g(x^{(0)})^T \lambda^{(0)}$. Set $\mu > 1$.
Then for each $j = 1, 2, \ldots$:
while $\eta^{(j)} > \epsilon_{gap}$ *or* $(||r_{prim}||_2^2 + ||r_{dual}||_2^2)^{\frac{1}{2}} > \epsilon_{feas}$ **do**
    Compute $t = \frac{\mu k}{\eta^{(j-1)}}$
    Determine the search direction $(\Delta x, \Delta\lambda, \Delta v)$
    Determine the step size s
    Update $x^{(j)} = x^{(j-1)} + s\Delta x$, $\lambda^{(j)} = \lambda^{(j-1)} + s\Delta\lambda$ and $v^{(j)} = v^{(j-1)} + s\Delta v$
    Determine $\eta^{(j)} = -g(x^{(j)})^T \lambda^{(j)}$.
end

Hence an interior-point should be computed for the initialisation. The surrogate duality gap η gives an indication of how far away the current solution is from the optimum. The factor μ ensures that the value of t increases such that the perturbed KKT conditions given in Section 4.2 will converge to the standard KKT conditions given in Section 4.1. The values $\epsilon_{\text{gap}} > 0$ and $\epsilon_{\text{feas}} > 0$ are the desired degree of accuracy with regard to the optimality and feasibility respectively. Note that the value of k is the number of inequality constraints, hence in our problem $k = 2n$.

The iteration procedure for the standard method and for the explicit method are almost identical. The only difference in the implementation is that for the standard method, the search direction $(\Delta x, \Delta\lambda, \Delta v)$ will be determined by solving the system of linear equations in Equation (4) by using a standard solver, whereas in the explicit method, the explicit formula for the search direction from Equation (10) will be utilised. Both of these methods have been implemented in Matlab, for which we conduct numerical experiments in Sections 6.2 and 6.3. The Matlab code is available and access can be requested from the author or supervisors.

6.2 Results with real-world data

In this section, we execute an experiment to compare the computation time of the two different methods for finding the search direction $(\Delta x, \Delta\lambda, \Delta v)$. We conduct 10,000 instances and measure the time that is needed for the IPM to solve them. More precisely, we compare the running time for the case where we solve the system of Equation (4) directly (thus via the standard method) and the case where we compute the search direction using

Equation (10) (via the explicit method).

The parameters that we use in this experiment follow from the real-world problem. Therefore, we assume that the number of time intervals is 100, i.e. $n = 100$. The capacity of the battery of an EV ranges from approximately 30 kWh [12] to 100 kWh. We therefore assume in this experiment that the capacity is 50, i.e. $C = 50$. Furthermore, the maximum charging rate does not change per time interval and is $u_i = 1$, $i = 1, 2, \ldots, n$. The vector c that appears in the objective function is chosen randomly according to a uniform distribution for each instance, where its entries lie in the interval $(-500, 500)$. Finally, we choose $\epsilon_{\text{gap}} = \epsilon_{\text{feas}} = 10^{-6}$ to ensure that we obtain a reasonable solution and that numerical problems do not occur.

The initialisation of the problem is as follows. The initial value for x is chosen such that the amount of energy is equally distributed over all intervals. The initial value of v is chosen to be zero. For all 10,000 instances, the same initial values of x and v are used, whereas the initial value of λ is chosen randomly according to a uniform distribution for each instance, where its entries lie in the interval $(0, 1)$. Hence each instance is different in the sense that the initial value for λ is chosen randomly (and feasible) and that the vector c is chosen randomly as well according to the distribution that has been stated in the last paragraph. The result of this experiment is given in Table 1:

TABLE 1: The results of 10,000 instances with time in seconds.

	Solving with the standard method	Solving with the explicit method
Average time per instance	0.08604	0.002474
Minimum time for an instance	0.05379	0.001558
Maximum time for an instance	0.44169	0.011204

Both methods yield the same optimal solution. From Table 1 we see that the algorithm with the explicit formula performs better in the sense that in all three categories (average, minimum and maximum time per for an instance) it has a lower computation time. The most significant category is the average time per instance. This has been reduced to $\frac{0.002474}{0.08604} \cdot 100\% \approx 2.88\%$ of the original computation time. Furthermore, we note that the highest running time of the explicit method is lower than the minimum running time of the standard method, which implies that the explicit method has been faster than the standard method for every instance. These results confirm the improvement in time complexity compared to the original method.

6.3 Results of scaling experiment

In this section, we investigate what the consequences are for the computation time when we increase the number of time intervals. We let $n \in \{2, 10, 100, 200, 400, 600, 800, 1000\}$ and we use the same parameters as in Section 6.2. We take the first value of n to be 2 such that a search direction can still be computed. In order to maintain feasibility of the problem, we let $C = 1$ for $n = 2$ and $n = 10$, for the other two problems C is defined as in Section 6.2. Furthermore, we reduce the number of instances from 10,000 to 10, since the running time for the standard method increases drastically as n grows large. Using the values of n between $n = 2$ to $n = 1000$, we can plot the average computation time per

instance of the standard method and the explicit method, which can be seen in Figure 1 and 2, respectively. Due to the steep increase in computation time, we do not consider values of n above 1000.

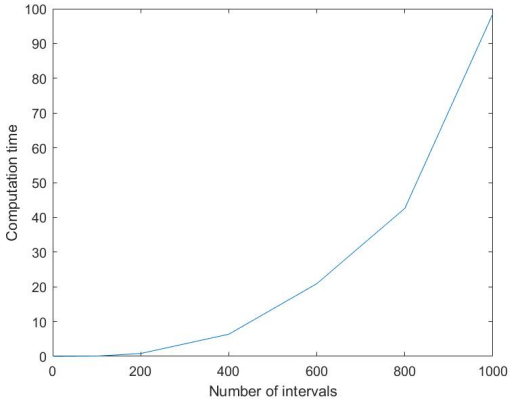

FIGURE 1: The average running time in seconds of the standard method for a different number of intervals.

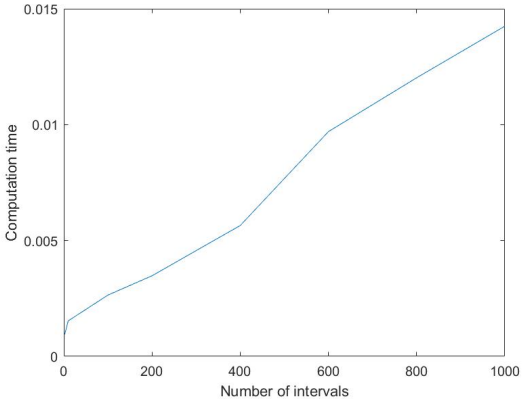

FIGURE 2: The average running time in seconds of the explicit method for a different number of intervals.

We note that in Figure 1, the graph of the computation time for the standard method becomes steeper and steeper as we increase n, which seems to behave as a quadratic or cubic function. This will be commented on in Section 7. In Figure 2 on the other hand, the computation time increases at approximately a constant rate. In Section 4.3 and 5.1 we stated that the computational complexity of the search direction of the standard method and the explicit method are $O(n^3)$ and $O(n)$, respectively. We conclude that the results of this experiment are in line with the analytical results that the computational complexity can be reduced to $O(n)$.

7 Discussion

The results of the scaling experiment in Section 6.3 show how the average computation time increases as n increases for both the standard method as the explicit method. We have not verified that the graphs in Figure 1 and 2 are in accordance with their respective computational complexities, $O(n^3)$ and $O(n)$. For example, it could be that the graph in Figure 1 does not behave like a cubic function due to the fact that the system of linear equations in Equation (4) is sparse and therefore, Matlab is able to find a solution in less that $O(n^3)$ time. Furthermore, we have conducted this experiment with only 10 instances due to the high computation time for the standard method. For further investigation whether these graphs match with their computational complexities, the number of instances should be increased and a curve fitting tool should be applied.

8 Conclusion

In the conclusion, we first state the analytic achievements that have been developed in Section 5. Secondly, we state what the results (see Section 6) and consequences are of those achievements. Finally, we conclude with ideas for future work that emerge from this

research.

For both EV charging problems that we studied in this paper, we derived formulas for the search direction that is computed in the primal-dual IPM. This has reduced the computational complexity of the time consuming step in the primal-dual IPM from $O(n^3)$ to $O(n)$.

In order to verify the gain in efficiency in practice, we conducted a numerical study for one of the EV charging problems. The results of the numerical experiments show that the computation time can be reduced 2.88% of the original computation time required when the search direction is found by a standard solver for systems of linear equations. Furthermore, we have looked into the behaviour of the computation time of the standard method and the explicit method as we increase the number of intervals that an EV is present in the electrical system. The results of these experiments are in accordance with the analytic results of the worst-case computation time.

From these results, we can conclude that the structure of the smart grid-related problems can be used to reduce the computational complexity of the primal-dual IPM. The consequences are that the solution method of these problems can be run on systems with lower computational power and that therefore the shift towards decentralised energy management is more feasible.

There are several ideas for further research in this field. The numerical results have given us an idea of the practical improvements that can be achieved when exploiting the problem structure. Further research should execute a similar experiment for the second EV charging problem, using the formula for the search direction given in this research. Furthermore, the analytic results for both of the EV charging problems can possibly be generalised for other resource allocation problems with this structure, for example where the constants in front of the quadratic terms in the objective function are arbitrary, strictly positive constants.

9 Acknowledgements

This paper has been written under the supervision of Ir. M. H. H. Schoot Uiterkamp and Prof. Dr. J.L. Hurink. The author would like to thank them for their guidance and advice.

References

[1] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[2] P. Brucker. An O(n) algorithm for quadratic knapsack problems. *Operations Research Letters*, 3(3):163–166, 1984.

[3] M.E.A. El-Mikkawy. On the inverse of a general tridiagonal matrix. *Applied Mathematics and Computation*, 150(3):669–679, 2004.

[4] J. Gondzio. Interior point methods 25 years later. *European Journal of Operational Research*, 218(3):587–601, 2012.

[5] Y. He, B. Venkatesh, and L. Guan. Optimal scheduling for charging and discharging of electric vehicles. *IEEE Transactions on Smart Grid*, 3(3):1095–1105, 2012.

[6] H.V. Henderson and S.R. Searle. On deriving the inverse of a sum of matrices. *SIAM Review*, 23(1):53–60, 1981.

[7] C. Jin, J. Tang, and P. Ghosh. Optimizing electric vehicle charging: A customer's perspective. *IEEE Transactions on Vehicular Technology*, 62(7):2919–2927, 2013.

[8] N. Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 302–311. ACM, 1984.

[9] W. Kempton and J. Tomić. Vehicle-to-grid power fundamentals: Calculating capacity and net revenue. *Journal of power sources*, 144(1):268–279, 2005.

[10] R.K. Mallik. The inverse of a tridiagonal matrix. *Linear Algebra and its Applications*, 325(1-3):109–139, 2001.

[11] M.J. Neely. Stochastic network optimization with application to communication and queueing systems. *Synthesis Lectures on Communication Networks*, 3(1):1–211, 2010.

[12] K. Qian, C. Zhou, M. Allan, and Y. Yuan. Modeling of load demand due to ev battery charging in distribution systems. *IEEE transactions on power systems*, 26(2):802–810, 2010.

[13] A.G. Robinson, N. Jiang, and C. S. Lerme. On the continuous quadratic knapsack problem. *Mathematical Programming*, 55(1):99–108, 1992.

[14] S.M. Stefanov. Convex quadratic minimization subject to a linear constraint and box constraints. *Applied Mathematics Research Express*, 2004(1):17–42, 2004.

[15] T. van der Klauw, M.E.T. Gerards, and J.L. Hurink. Resource allocation problems in decentralized energy management. *OR Spectrum*, 39(3):749–773, 2017.

[16] W.L. Winston. *Operations Research: Applications and Algorithms*. Brooks/Cole, 2004.

[17] M. Wright. The interior-point revolution in optimization: history, recent developments, and lasting consequences. *Bulletin of the American mathematical society*, 42(1):39–56, 2005.

A Derivation of explicit search direction for SRA

The system that we start with is the following (also given in Equation (9)):

$$\begin{bmatrix} I & Dg(x)^T & \mathbb{1} \\ -\mathrm{diag}(\lambda)\begin{bmatrix} I \\ -I \end{bmatrix} & -\mathrm{diag}(g(x)) & 0 \\ \mathbb{1}^T & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \\ \Delta v \end{bmatrix} = - \begin{bmatrix} r_{\mathrm{dual}} \\ r_{\mathrm{cent}} \\ r_{\mathrm{prim}} \end{bmatrix}$$

For simplicity, let r_{dual}, r_{cent} and r_{prim} be denoted by r_1, r_2 and r_3 respectively. Let

$$\begin{aligned} \Lambda_1 &= \mathrm{diag}([\lambda_1, \lambda_2, \cdots, \lambda_n]), \\ \Lambda_2 &= \mathrm{diag}([\lambda_{n+1}, \lambda_{n+2}, \cdots, \lambda_{2n}]), \\ G_1 &= \mathrm{diag}([g_1, g_2, \cdots, g_n]), \\ G_2 &= \mathrm{diag}([g_{n+1}, g_{n+2}, \cdots, g_{2n}]). \end{aligned}$$

With this notation and Gaussian elimination we make the following steps:

$$\begin{bmatrix} I & Dg(x)^T & \mathbb{1} & -r_1 \\ -\begin{bmatrix} \Lambda_1 \\ -\Lambda_2 \end{bmatrix} & -\mathrm{diag}(g(x)) & 0 & -r_2 \\ \mathbb{1}^T & 0 & 0 & -r_3 \end{bmatrix}$$

$$\sim \begin{bmatrix} I & Dg(x)^T & \mathbb{1} & -r_1 \\ 0 & -\mathrm{diag}(g(x)) + \begin{bmatrix} \Lambda_1 \\ -\Lambda_2 \end{bmatrix} Dg(x)^T & \begin{bmatrix} \Lambda_1 \\ -\Lambda_2 \end{bmatrix} \mathbb{1} & -r_2 - \begin{bmatrix} \Lambda_1 \\ -\Lambda_2 \end{bmatrix} r_1 \\ \mathbb{1}^T & 0 & 0 & -r_3 \end{bmatrix}$$

$$\sim \begin{bmatrix} I & Dg(x)^T & \mathbb{1} & -r_1 \\ 0 & \begin{bmatrix} \Lambda_1 & -\Lambda_1 \\ -\Lambda_2 & \Lambda_2 \end{bmatrix} - \mathrm{diag}(g(x)) & \begin{bmatrix} \Lambda_1 \\ -\Lambda_2 \end{bmatrix} \mathbb{1} & -r_2 - \begin{bmatrix} \Lambda_1 \\ -\Lambda_2 \end{bmatrix} r_1 \\ 0 & -\mathbb{1}^T Dg(x)^T & -n & -r_3 + \mathbb{1}^T r_1 \end{bmatrix}. \tag{14}$$

Note that in this last step we used that $-\mathbb{1}^T\mathbb{1} = -n$. Furthermore, we introduce the following matrix, which is equal to the entry in the second row in the second column of Equation (14):

$$P = \begin{bmatrix} \Lambda_1 - G_1 & -\Lambda_1 \\ -\Lambda_2 & \Lambda_2 - G_2 \end{bmatrix}.$$

We can multiply the second row in Equation (14) by P^{-1} to create a $2n \times 2n$ identity matrix on the second entry of this row. This identity matrix will be denoted as I_{2n} in order to avoid confusion with I, which denotes the $n \times n$ identity matrix. Furthermore, since P has matrices as entries (and is hence called a block or partitioned matrix), its inverse can be computed using the following formula:

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} = \begin{bmatrix} A^{-1} + A^{-1}B(D - CA^{-1}B)^{-1}CA^{-1} & A^{-1}B(D - CA^{-1}B)^{-1} \\ (D - CA^{-1}B)^{-1}CA^{-1} & (D - CA^{-1}B)^{-1} \end{bmatrix}.$$

In order to write down the inverse of P efficiently, we introduce the vector $\phi \in \mathbb{R}^n$ with entries

$$\phi_i = (\lambda_i - g_i)(\lambda_{i+n} - g_{i+n}) - \lambda_i\lambda_{i+n} = -\lambda_i g_{i+n} - \lambda_{i+n}g_i + g_i g_{i+n},$$

for $i \in J = \{1, 2, \ldots, n\}$. Furthermore, we introduce the four vectors $\psi^{(k)} \in \mathbb{R}^n$, $k = 1, 2, 3, 4$. For $i \in J$, the entries of these vectors are

$$\psi_i^{(1)} = \frac{(\lambda_{i+n} - g_{i+n})}{\phi_i} \tag{15}$$

$$\psi_i^{(2)} = \frac{\lambda_i}{\phi_i} \tag{16}$$

$$\psi_i^{(3)} = \frac{\lambda_{i+n}}{\phi_i} \tag{17}$$

$$\psi_i^{(4)} = \frac{(\lambda_i - g_i)}{\phi_i}. \tag{18}$$

It can be verified that P^{-1} can be written as

$$P^{-1} = \begin{bmatrix} \operatorname{diag}(\psi^{(1)}) & \operatorname{diag}(\psi^{(2)}) \\ \operatorname{diag}(\psi^{(3)}) & \operatorname{diag}(\psi^{(4)}) \end{bmatrix}.$$

This inverse exists if and only if the entries of the diagonal matrices are nonzero and if $\phi_i \neq 0 \ \forall i \in J$. Since $\lambda_i > 0$ and $g_i < 0 \ \forall i = 1, 2, \ldots, 2n$ (due to strict feasibility), we note that the numerators of the vectors in Equations (15) - (18) only have nonzero entries. For the same reason, $\phi_i > 0 \ \forall i = 1, 2, \ldots, n$. Hence the strict feasibility of the problem ensure that the inverse of P exists.

In the rest of the derivation of the explicit formula for the SRA problem, we maintain the use of P^{-1} for simplicity. We let I_{2n} denote the $2n \times 2n$ identity matrix. We continue with Equation (14):

$$\begin{bmatrix} I & Dg(x)^T & \mathbb{1} & -r_1 \\ 0 & I_{2n} & P^{-1}\begin{bmatrix} \Lambda_1 \\ -\Lambda_2 \end{bmatrix}\mathbb{1} & P^{-1}(-r_2 - \begin{bmatrix} \Lambda_1 \\ -\Lambda_2 \end{bmatrix} r_1) \\ 0 & -\mathbb{1}^T Dg(x)^T & -n & -r_3 + \mathbb{1}^T r_1 \end{bmatrix}$$

$$\sim \begin{bmatrix} I & 0 & \mathbb{1} - Dg(x)^T P^{-1}\begin{bmatrix} \Lambda_1 \\ -\Lambda_2 \end{bmatrix}\mathbb{1} & -r_1 - Dg(x)^T P^{-1}(-r_2 - \begin{bmatrix} \Lambda_1 \\ -\Lambda_2 \end{bmatrix} r_1) \\ 0 & I_{2n} & P^{-1}\begin{bmatrix} \Lambda_1 \\ -\Lambda_2 \end{bmatrix}\mathbb{1} & P^{-1}(-r_2 - \begin{bmatrix} \Lambda_1 \\ -\Lambda_2 \end{bmatrix} r_1) \\ 0 & -\mathbb{1}^T Dg(x)^T & -n & -r_3 + \mathbb{1}^T r_1 \end{bmatrix}$$

$$\sim \begin{bmatrix} I & 0 & \mathbb{1} - Dg(x)^T P^{-1} \begin{bmatrix} \Lambda_1 \\ -\Lambda_2 \end{bmatrix} \mathbb{1} & -r_1 - Dg(x)^T P^{-1}(-r_2 - \begin{bmatrix} \Lambda_1 \\ -\Lambda_2 \end{bmatrix} r_1) \\ 0 & I_{2n} & P^{-1} \begin{bmatrix} \Lambda_1 \\ -\Lambda_2 \end{bmatrix} \mathbb{1} & P^{-1}(-r_2 - \begin{bmatrix} \Lambda_1 \\ -\Lambda_2 \end{bmatrix} r_1) \\ 0 & 0 & -n + \mathbb{1}^T Dg(x)^T P^{-1} \begin{bmatrix} \Lambda_1 \\ -\Lambda_2 \end{bmatrix} \mathbb{1} & -r_3 + \mathbb{1}^T r_1 + \mathbb{1}^T Dg(x)^T P^{-1}(-r_2 - \begin{bmatrix} \Lambda_1 \\ -\Lambda_2 \end{bmatrix} r_1) \end{bmatrix}. \tag{19}$$

For simplicity, we define $\rho := -n + \mathbb{1}^T Dg(x)^T P^{-1} \begin{bmatrix} \Lambda_1 \\ -\Lambda_2 \end{bmatrix} \mathbb{1}$. If we divide the third row in Equation (19) by ρ, the last column of this row gives an expression for Δv:

$$\Delta v = (-r_3 + \mathbb{1}^T r_1 + \mathbb{1}^T Dg(x)^T P^{-1}(-r_2 - \begin{bmatrix} \Lambda_1 \\ -\Lambda_2 \end{bmatrix} r_1))/\rho. \tag{20}$$

With this notation, we conduct the final steps of the derivation

$$\begin{bmatrix} I & 0 & \mathbb{1} - Dg(x)^T P^{-1} \begin{bmatrix} \Lambda_1 \\ -\Lambda_2 \end{bmatrix} \mathbb{1} & -r_1 - Dg(x)^T P^{-1}(-r_2 - \begin{bmatrix} \Lambda_1 \\ -\Lambda_2 \end{bmatrix} r_1) \\ 0 & I_{2n} & P^{-1} \begin{bmatrix} \Lambda_1 \\ -\Lambda_2 \end{bmatrix} \mathbb{1} & P^{-1}(-r_2 - \begin{bmatrix} \Lambda_1 \\ -\Lambda_2 \end{bmatrix} r_1) \\ 0 & 0 & 1 & \Delta v \end{bmatrix}$$

$$\sim \begin{bmatrix} I & 0 & 0 & -r_1 - Dg(x)^T P^{-1}(-r_2 - \begin{bmatrix} \Lambda_1 \\ -\Lambda_2 \end{bmatrix} r_1) - (\mathbb{1} - Dg(x)^T P^{-1} \begin{bmatrix} \Lambda_1 \\ -\Lambda_2 \end{bmatrix} \mathbb{1}) \Delta v \\ 0 & I_{2n} & P^{-1} \begin{bmatrix} \Lambda_1 \\ -\Lambda_2 \end{bmatrix} \mathbb{1} & P^{-1}(-r_2 - \begin{bmatrix} \Lambda_1 \\ -\Lambda_2 \end{bmatrix} r_1) \\ 0 & 0 & 1 & \Delta v \end{bmatrix}$$

$$\sim \begin{bmatrix} I & 0 & 0 & -r_1 - Dg(x)^T P^{-1}(-r_2 - \begin{bmatrix} \Lambda_1 \\ -\Lambda_2 \end{bmatrix} r_1) - (\mathbb{1} - Dg(x)^T P^{-1} \begin{bmatrix} \Lambda_1 \\ -\Lambda_2 \end{bmatrix} \mathbb{1}) \Delta v \\ 0 & I_{2n} & 0 & P^{-1}(-r_2 - \begin{bmatrix} \Lambda_1 \\ -\Lambda_2 \end{bmatrix} r_1) - P^{-1} \begin{bmatrix} \Lambda_1 \\ -\Lambda_2 \end{bmatrix} \mathbb{1} \Delta v \\ 0 & 0 & 1 & \Delta v \end{bmatrix}.$$

This is equivalent to the expression in Equation (10) in Section 5.1, where Δv is given by Equation (20):

$$\begin{bmatrix} \Delta x \\ \Delta \lambda \\ \Delta v \end{bmatrix} = \begin{bmatrix} -r_1 - Dg(x)^T P^{-1}(-r_2 - \begin{bmatrix} \Lambda_1 \\ -\Lambda_2 \end{bmatrix} r_1) - (\mathbb{1} - Dg(x)^T P^{-1} \begin{bmatrix} \Lambda_1 \\ -\Lambda_2 \end{bmatrix} \mathbb{1}) \Delta v \\ P^{-1}(-r_2 - \begin{bmatrix} \Lambda_1 \\ -\Lambda_2 \end{bmatrix} r_1) - P^{-1} \begin{bmatrix} \Lambda_1 \\ -\Lambda_2 \end{bmatrix} \mathbb{1} \Delta v \\ (-r_3 + \mathbb{1}^T r_1 + \mathbb{1}^T Dg(x)^T P^{-1}(-r_2 - \begin{bmatrix} \Lambda_1 \\ -\Lambda_2 \end{bmatrix} r_1))/\rho \end{bmatrix}.$$

B Derivation of explicit search direction for CRA

In this appendix, we derive Equation (13) in Section 5.2. We let $R_{\text{dual}} = R_1$ and $R_{\text{cent}} = R_2$ and start with the following equation:

$$\begin{bmatrix} I & Dg(x)^T \\ -\text{diag}(\lambda)Dg(x) & -\text{diag}(g(x)) \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \end{bmatrix} = - \begin{bmatrix} R_{\text{dual}} \\ R_{\text{cent}} \end{bmatrix}$$

Similar to the approach in Appendix A, we use Gaussian elimination to solve this system. A first step yields

$$\begin{bmatrix} I & Dg(x)^T & -R_1 \\ 0 & -\text{diag}(g(x)) + \text{diag}(\lambda)Dg(x)Dg(x)^T & -R_2 - \text{diag}(\lambda)Dg(x)R_1 \end{bmatrix}. \tag{21}$$

For the next step, we will use the matrix N which is defined in Equation (12). We derive that $N = -\text{diag}(g(x)) + \text{diag}(\lambda)Dg(x)Dg(x)^T$:

$$\begin{aligned}
N &= \begin{bmatrix} \Lambda_1 - G_1 & -\Lambda_1 & \Lambda_1 L^T & \Lambda_1 L^T \\ -\Lambda_2 & \Lambda_2 - G_2 & \Lambda_2 L^T & -\Lambda_2 L^T \\ -\Lambda_3 L & \Lambda_3 L & \Lambda_3 L L^T - G_3 & -\Lambda_3 L L^T \\ \Lambda_4 L & -\Lambda_4 L & -\Lambda_4 L L^T & \Lambda_4 L L^T - G_4 \end{bmatrix} \\
&= \begin{bmatrix} \Lambda_1 & -\Lambda_1 & \Lambda_1 L^T & \Lambda_1 L^T \\ -\Lambda_2 & \Lambda_2 & \Lambda_2 L^T & -\Lambda_2 L^T \\ -\Lambda_3 L & \Lambda_3 L & \Lambda_3 L L^T & -\Lambda_3 L L^T \\ \Lambda_4 L & -\Lambda_4 L & -\Lambda_4 L L^T & \Lambda_4 L L^T \end{bmatrix} - \begin{bmatrix} G_1 & 0 & 0 & 0 \\ 0 & G_2 & 0 & 0 \\ 0 & 0 & G_3 & 0 \\ 0 & 0 & 0 & G_4 \end{bmatrix} \\
&= \begin{bmatrix} \Lambda_1 & 0 & 0 & 0 \\ 0 & \Lambda_2 & 0 & \\ 0 & 0 & \Lambda_3 & 0 \\ 0 & 0 & 0 & \Lambda_4 \end{bmatrix} \begin{bmatrix} I & -I & -L^T & L^T \\ -I & I & L^T & -L^T \\ -L & L & LL^T & -LL^T \\ L & -L & -LL^T & LL^T \end{bmatrix} - \begin{bmatrix} G_1 & 0 & 0 & 0 \\ 0 & G_2 & 0 & 0 \\ 0 & 0 & G_3 & 0 \\ 0 & 0 & 0 & G_4 \end{bmatrix} \\
&= \begin{bmatrix} \Lambda_1 & 0 & 0 & 0 \\ 0 & \Lambda_2 & 0 & \\ 0 & 0 & \Lambda_3 & 0 \\ 0 & 0 & 0 & \Lambda_4 \end{bmatrix} \begin{bmatrix} I \\ -I \\ -L \\ L \end{bmatrix} \begin{bmatrix} I & -I & -L^T & L^T \end{bmatrix} - \begin{bmatrix} G_1 & 0 & 0 & 0 \\ 0 & G_2 & 0 & 0 \\ 0 & 0 & G_3 & 0 \\ 0 & 0 & 0 & G_4 \end{bmatrix} \\
&= \text{diag}(\lambda)Dg(x)Dg(x)^T - \text{diag}(g(x)).
\end{aligned}$$

Now the system in Equation (21) can be written more compactly and we continue the process of Gaussian elimination:

$$\begin{bmatrix} I & Dg(x)^T & -R_1 \\ 0 & N & -R_2 - \text{diag}(\lambda)Dg(x)R_1 \end{bmatrix}$$

$$\sim \begin{bmatrix} I & Dg(x)^T & -R_1 \\ 0 & I & N^{-1}(-R_2 - \text{diag}(\lambda)Dg(x)R_1) \end{bmatrix}$$

$$\sim \begin{bmatrix} I & 0 & -R_1 - Dg(x)^T N^{-1}(-R_2 - \text{diag}(\lambda)Dg(x)R_1) \\ 0 & I & N^{-1}(-R_2 - \text{diag}(\lambda)Dg(x)R_1) \end{bmatrix}.$$

From this follows Equation (13) in Section 5.2:

$$\begin{bmatrix}\Delta x\\ \Delta\lambda\end{bmatrix} = \begin{bmatrix}-R_{\text{dual}} - Dg(x)^T N^{-1}(-R_{\text{cent}} - \text{diag}(\lambda)Dg(x)R_{\text{dual}})\\ N^{-1}(-R_{\text{cent}} - \text{diag}(\lambda)Dg(x)R_{\text{dual}})\end{bmatrix}.$$

C Computing $N^{-1}w$ for any vector w

In the explicit formula for the search direction of the CRA problem in the primal-dual IPM (see Equation (13)), we encountered the matrix N. In Appendix B we derived that

$$N = \text{diag}(\lambda)Dg(x)Dg(x)^T - \text{diag}(g(x)). \tag{22}$$

In order to find its inverse, we apply the Woodbury Matrix Identity [6], since we are dealing with a sum of matrices. The Woodbury Matrix Identity is given by

$$(A+UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1},$$

where A, U, C and V denote matrices of appropriate sizes with A and C invertible. We transform Equation (22) slightly to obtain the desired form

$$\text{diag}(\lambda)^{-1}N = -\text{diag}(\lambda)^{-1}\text{diag}(g(x)) + Dg(x)Dg(x)^T.$$

Our goal is to find N^{-1} and we do so by applying the Woodbury Matrix Identity, where $A = -\text{diag}(\lambda)^{-1}\text{diag}(g(x))$, $U = Dg(x)$, $C = I$ and $V = Dg(x)^T$. Let θ denote the vector with $\theta_i = \frac{\lambda_i}{g_i(x)}$ for $i = 1, 2, \ldots, 4n$. We obtain

$$\begin{aligned}(\text{diag}(\lambda)^{-1}N)^{-1} = &-\text{diag}(\theta) - \text{diag}(\theta)Dg(x)\\ &\cdot (I - Dg(x)^T\text{diag}(\theta)Dg(x))^{-1}Dg(x)^T\text{diag}(\theta).\end{aligned}$$

This equation contains another inverse, $(I - Dg(x)^T\text{diag}(\theta)Dg(x))^{-1}$, and in order to find this inverse we define the following matrices:

$$\begin{aligned}B &= I - Dg(x)^T\text{diag}(\theta)Dg(x),\\ \Omega^{(1)} &= \text{diag}([\frac{\lambda_1}{g_1(x)}, \frac{\lambda_2}{g_2(x)}, \ldots, \frac{\lambda_n}{g_n}]),\\ \Omega^{(2)} &= \text{diag}([\frac{\lambda_{n+1}}{g_{n+1}(x)}, \frac{\lambda_{n+2}}{g_{n+2}(x)}, \ldots, \frac{\lambda_{2n}}{g_{2n}}]),\\ \Omega^{(3)} &= \text{diag}([\frac{\lambda_{2n+1}}{g_{2n+1}(x)}, \frac{\lambda_{2n+2}}{g_{2n+2}(x)}, \ldots, \frac{\lambda_{3n}}{g_{3n}}]),\\ \Omega^{(4)} &= \text{diag}([\frac{\lambda_{3n+1}}{g_{3n+1}(x)}, \frac{\lambda_{3n+2}}{g_{3n+2}(x)}, \ldots, \frac{\lambda_{4n}}{g_{4n}}]).\end{aligned}$$

Since we are interested in the inverse of B, we expand the formula for B as follows:

$$
\begin{aligned}
B &= I - Dg(x)^T \mathrm{diag}(\theta) Dg(x) \\
&= I - \begin{bmatrix} I & -I & -L^T & L^T \end{bmatrix} \begin{bmatrix} \Omega^{(1)} & 0 & 0 & 0 \\ 0 & \Omega^{(2)} & 0 & 0 \\ 0 & 0 & \Omega^{(3)} & 0 \\ 0 & 0 & 0 & \Omega^{(4)} \end{bmatrix} \begin{bmatrix} I \\ -I \\ -L \\ L \end{bmatrix} \\
&= I - \begin{bmatrix} I & -I & -L^T & L^T \end{bmatrix} \begin{bmatrix} \Omega^{(1)} \\ -\Omega^{(2)} \\ -\Omega^{(3)} L \\ \Omega^{(4)} L \end{bmatrix} \\
&= I - (\Omega^{(1)} + \Omega^{(2)} + L^T \Omega^{(3)} L + L^T \Omega^{(4)} L) \\
&= I - \Omega^{(1)} - \Omega^{(2)} - L^T (\Omega^{(3)} + \Omega^{(4)}) L.
\end{aligned}
$$

It can be verified that

$$
L^{-1} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ -1 & 1 & 0 & \cdots & 0 \\ 0 & -1 & 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & -1 & 1 \end{bmatrix},
$$

i.e. the inverse of L is a lower bidiagonal matrix and we also note that $(L^T)^-1 = (L^{-1})^T$. Therefore, we write

$$
(L^T)^{-1} B L^{-1} = (L^T)^{-1} (I - \Omega^{(1)} - \Omega^{(2)}) L^{-1} - \Omega^{(3)} - \Omega^{(4)}. \tag{23}
$$

It can be verified that the right hand side of Equation (23) is a tridiagonal matrix. Several linear-time algorithms exist to solve systems involving such matrices [3][10]. Define $T = (L^T)^{-1}(I - \Omega^{(1)} - \Omega^{(2)})L^{-1} - \Omega^{(3)} - \Omega^{(4)}$. Then we derive

$$
\begin{aligned}
&(L^T)^{-1} B L^{-1} = T \\
&((L^T)^{-1} B L^{-1}) = T^{-1} \\
&L B^{-1} L^T = T^{-1} \\
&B^{-1} = L^{-1} T^{-1} (L^T)^{-1}.
\end{aligned}
$$

Finally, we can derive the inverse of N:

$$
\begin{aligned}
&(\mathrm{diag}(\lambda)^{-1} N)^{-1} = -\mathrm{diag}(\theta) - \mathrm{diag}(\theta) Dg(x) B^{-1} Dg(x)^T \mathrm{diag}(\theta) \\
&N^{-1} \mathrm{diag}(\lambda) = -\mathrm{diag}(\theta) - \mathrm{diag}(\theta) Dg(x) B^{-1} Dg(x)^T \mathrm{diag}(\theta).
\end{aligned}
$$

If we let γ denote the vector with entries $\gamma_i = \frac{1}{g_i(x)}$ for $i = 1, 2, \ldots, 4n$, we obtain

$$
N^{-1} = -\mathrm{diag}(\gamma) - \mathrm{diag}(\theta) Dg(x) L^{-1} T^{-1} (L^T)^{-1} Dg(x)^T \mathrm{diag}(\gamma). \tag{24}
$$

We note that for any vector $w \in \mathbb{R}^n$, the product $N^{-1}w$ can be computed with a complexity of $O(n)$ by using the approaches in [3] or [10]. This can be used in the computation of the search direction for the CRA problem given in Equation (13).