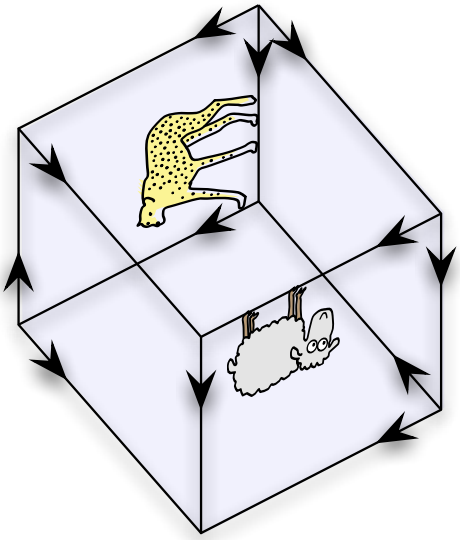
Linear and Quadratic Programming by Unique Sink Orientations

Diploma thesis in Mathematics

ETH Zürich



Martin Jaggi
Bergweidstrasse 8
CH-9200 Gossau
martin@*m8j.net*

supervised by Dr. Bernd Gärtner
gaertner@*inf.ethz.ch*
(Prof. Emo Welzl, Institute of Theoretical Computer Science)

September 2006

Abstract *Linear programming* and *convex quadratic programming* are two of the most frequent classes of optimisation problems that occur in practice. It has just recently been shown that every linear program (LP) in n variables and m equality constraints in a natural way defines a *unique sink orientation* of the edges of the n dimensional cube, with the sink of the cube telling us an optimal solution to the LP, or a certificate for infeasibility or unboundedness. In this thesis this method is generalised to *convex quadratic programming*, resulting in the fastest known deterministic algorithm for convex quadratic programming in the RAM model, for the central case $n = 2m$. Furthermore, a perturbation method is presented to speed up the algorithm for LP. The connection between the new approach for LP and symmetric *P-matrix linear complementarity problems* (PLCP) is studied, and a *regularisation* method is suggested for LP. Finally a first implementation of the algorithm is realised.

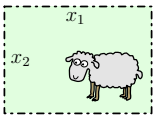
Zusammenfassung *Lineare Programmierung* und *konvexe quadratische Programmierung* sind zwei der häufigsten Klassen von Optimierungsproblemen die in der Praxis auftreten. Kürzlich wurde gezeigt, dass jedes lineare Programm (LP) in n Variablen und m Gleichungen in einer natürlichen Weise eine *Eindeutige-Senke-Orientierung* der Kanten des n-dimensionalen Würfels definiert, wobei die Senke dieser Orientierung eine optimale Lösung des LP liefert, oder ansonsten ein Zertifikat für Unlösbarkeit oder Unbeschränktheit. In dieser Arbeit wird diese Methode verallgemeinert für *konvexe quadratische Programmierung*, und ergibt den schnellsten bekannten deterministischen Algorithmus für konvexe quadratische Programmierung im RAM-Modell, für den 'zentralen' Fall $n = 2m$. Zudem wird eine Perturbations-Methode präsentiert, um den Algorithmus für LP weiter zu beschleunigen. Die Verbindung des neuen Ansatzes für LP mit dem symmetrischen P-Matrix *Linearen Komplementaritätsproblem* wird untersucht, und es wird eine *Regularisierungs-Methode* für LP vorgeschlagen. Zuletzt wird eine erste Implementierung des Algorithmus realisiert.

Keywords Convex Optimisation, Linear Programming, Quadratic Programming, Unique Sink Orientations, PLCP, Regularisation

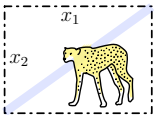
Dank Herzlichen Dank an Bernd Gärtner für die kompetente Betreuung, die vielen interessanten Ideen und Anregungen, an Leo Rüst für die witzigen gemeinsamen Experimente mit LP und PLCP, an Nadja für ihre Geduld, und an meine Eltern dafür dass sie mir das Studium ermöglicht haben.

Contents

An Easy Introduction This thesis provides a new solving method for *convex quadratic optimisation* problems. If this sounds a bit weird to you, you may consider the following situation: Given a sheep and a cheetah not harming each other, and L meters of fencing, we want to fence in a rectangular meadow suiting best for our two animals. For the sheep, our aim is to maximise the area of grass available. But for the cheetah, for security reasons we would rather like to minimise the longest straight line in the meadow, so that the cheetah may not gain too much speed to eventually jump over the fence and run away. If we denote the length and width of the meadow by x_1 and x_2, we can formulate our two problems as follows:



$$\begin{aligned} \text{maximise} \quad & x_1\, x_2 \\ \text{such that} \quad & 2x_1 + 2x_2 = L \\ & x_1, x_2 \geqslant 0\,, \end{aligned}$$



$$\begin{aligned} \text{minimise} \quad & x_1^2 + x_2^2 \\ \text{such that} \quad & 2x_1 + 2x_2 = L \\ & x_1, x_2 \geqslant 0\,. \end{aligned}$$

Optimisation problems of this form are called *quadratic programs*, since the functions $x_1\, x_2$ and $x_1^2 + x_2^2$ are quadratic. In the above slightly artificial case there are several easy approaches that solve the two problems directly (which by the way have the same solution), but in practice, there may be much more than just one constraint that has to be satisfied, so more sophisticated solution methods are needed. Furthermore we note that the second of the two above problems is in fact a *convex quadratic program*, whereas the first one is not, since the function $x_1\, x_2$ is not convex.

The new solution method for convex quadratic programs that is explained in this thesis works by jumping around—not on a meadow, but—on the corners of the n-dimensional cube, where n is the number of variables we have in our quadratic program.

Introduction Convex quadratic programming is a case of non-linear optimisation that has numerous applications in natural sciences, engineering, mathematics and economics, as for example in regression analysis, signal processing, image recognition, portfolio optimisation, control theory, filter design, machine learning, operations research and geometry.

In the last decades this class of optimisation problems has received more attention as several known concepts from linear programming have successfully been translated to convex quadratic programming (Simplex type methods, ellipsoid method, interior point methods) and given rise to improved solution algorithms. In this thesis another interesting concept which has just recently proved to be useful for linear programming [3], the concept of *unique sink orientations of the n-cube*, is translated to convex quadratic programming, and gives a new algorithm that is very fast in the RAM[1] model. The research for faster algorithms for LP as well as for convex QP is also motivated by the over thirty year old unsolved problem whether LP admits a polynomial time algorithm in the RAM model (also called a *strongly* polynomial algorithm).

In Section 2 of this thesis, a perturbation method for a linear program is studied as a method to further speed up the solving algorithm by unique sink orientations. We achieve a linear speed-up compared to the original algorithm.

Section 3 provides deeper insight into the properties of the method and the resulting cube-orientations, and gives a new proof that the LP solution obtained by the USO approach is indeed the shortest optimal solution to the LP. We also mention the connection to a problem very closely related to LP and QP, the problem of symmetric *P-matrix linear complementarity problems*. Furthermore, a regularisation method for LP is proposed.

In Section 4, a first implementation of the algorithm for linear programming by unique sink orientations is realised to further study the behaviour of the new algorithm and the properties of the resulting cube-orientations.

[1] *Random Access Machine* model, complexity is measured in number of elementary operations on e.g. integers or real numbers, in contrast to the *Turing Machine* model.

1 Quadratic Programming by Unique Sink Orientations

In this section, the reduction of linear programming to unique sink orientations from [3] is generalised to convex quadratic programming.

1.1 Introduction

Quadratic programming is the problem of minimising a quadratic function subject to linear (in)equality constraints. Here we consider convex quadratic programs of the form

$$(QP) \quad \begin{array}{ll} \min & \frac{1}{2}x^T Q x - c^T x \\ \text{s.t.} & Ax = b \\ & x \geqslant 0\,, \end{array} \tag{1.1}$$

with $Q \in \mathbb{R}^{n\times n}$ a positive semi-definite matrix, $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$. Note that any convex quadratic program can be converted to this form by introducing additional (slack) variables. In addition, w.l.o.g., we may assume that Q is symmetric (otherwise set $Q' := \frac{1}{2}(Q + Q^T)$).

To solve the above quadratic program means to compute an optimal solution x^* if the problem is feasible and bounded, and to report infeasibility or unboundedness otherwise.

In the following we will prove that the problem of solving a convex quadratic program in n non-negative variables and m equality constraints in a natural way defines a *unique sink orientation* (USO) of the n-dimensional cube, with the unique sink of the USO corresponding to the solution of the QP. The method is very different from simplex-type methods in many aspects: We obtain a *canonical solution* for every convex QP, with the solution being independent of the initial conditions or the internal rule being used. It's also remarkable that our procedure obtains this canonical solution also in the unbounded or even infeasible case, so there is no need for any preprocessing steps (like phase one in the simplex method).

By using the fastest known deterministic sink-finding algorithm for general USO, we obtain the currently fastest deterministic algorithm for convex QP (and also for LP, see [3]) in the RAM model, for the central case $n = 2m$:

Theorem 1.1. *Any convex QP with $n = 2m$ variables can be solved in time*

$$O\left(1.606^n\right) = O\left(2.58^m\right).$$

Proof. The *Fibonacci Seesaw* algorithm by Szabó and Welzl [6] finds the sink of a USO of the n-cube in time $O\left(1.606^n\right)$. In the following of the section we will prove a reduction of convex QP to finding the sink of a USO. □

1.2 Unique sink orientations

Definition 1.1. *An orientation of the vertex-edge graph of the n-dimensional cube is called* unique sink orientation *(USO) if every subgraph induced by a non-empty cube face has a unique sink.*

In particular, the cube has a unique global sink. The concept of unique sink orientations has first been considered by Stickney and Watson in 1978 [5] as a model to solve *linear complementarity problems*, but has since then proved to be useful for other problems, in particular the *smallest enclosing ball* problem in geometry, and *linear programming*. Also, several results about the combinatorial properties of these objects followed [2, 4, 6].

Szabó and Welzl found a deterministic algorithm—the Fibonacci Seesaw—that finds the global sink of any n-cube USO by looking at less than 1.61^n vertices (more precisely, at the orientations of the incident edges) [6]. This is the *vertex evaluation* model.

1.3 The Karush–Kuhn–Tucker conditions

Let us fix some notation first.

For $x \in \mathbb{R}^n$ and $J \subseteq [n] := \{1, \ldots, n\}$, x_J is the $|J|$-dimensional vector obtained from x by collecting all coordinates with subscript in J.

For $A \in \mathbb{R}^{m \times n}$, $A_J \in \mathbb{R}^{m \times |J|}$ collects the *columns* of A with subscript in J, and in a notation which we will use less frequently, $A^J \in \mathbb{R}^{|J| \times n}$ collects the *rows* of A with subscript in J, for $J \subseteq [m]$.

With $\mathbf{0}$ being the zero vector of the appropriate dimension, we also use

$$\mathbb{R}^J := \{x \in \mathbb{R}^n \mid x_{[n]\setminus J} = \mathbf{0}\}.$$

Now let $f : \mathbb{R}^n \to \mathbb{R}$ be a differentiable convex function with continuous partial derivatives. For $I \subseteq J \subseteq [n]$ we consider the *convex programming* problem

$$\begin{array}{lll} \mathrm{CP}(I,J) & \min & f(x) \\ & \text{s.t.} & Ax = b \\ & & x \in \mathbb{R}^J \\ & & x_{J\setminus I} \geq 0. \end{array} \tag{1.2}$$

where $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$. The following is a specialisation of a very general theorem to our scenario.

Theorem 1.2. *$x^* \in \mathbb{R}^n$ is an optimal solution to problem $\mathrm{CP}(I,J)$ if and only if*

(i) $Ax^* = b,\ x^* \in \mathbb{R}^J,\ x^*_{J\setminus I} \geq 0$, and

(ii) there exists $\lambda \in \mathbb{R}^m$ such that for all $j \in J$,

$$\nabla f(x^*)_j - \lambda^T A_j \geq 0,$$

with equality if $j \in I$ or $x^*_j > 0$.

Here, $\nabla f(x^*)$ is the gradient of f at x^* which by convention is an n-dimensional row vector.

The entries of λ are called *Karush–Kuhn–Tucker* (KKT) *multipliers*. A short proof of this theorem can be found in [3].

1.4 Simple convex programming

This short section recapitulates the important reduction of *strictly convex programming* to unique sink orientations from [3]. Let us fix a *strictly* convex function f and consider for $I \subseteq J \subseteq [n]$ the problem

$$\begin{array}{lll} \mathrm{SCP}(I,J) & \min & f(x) \\ & s.t. & x \in \mathbb{R}^J, \\ & & x_{J\setminus I} \geq 0. \end{array} \tag{1.3}$$

This is just the program $\mathrm{CP}(I,J)$ from the previous section, restricted to the strictly convex case, and with $A \in \mathbb{R}^{0\times n}$. We refer to this set-up as *simple convex programming* (SCP).

Since f is strictly convex and SCP(I,J) is feasible, the program SCP(I,J) has a unique solution $x^*(I,J)$, for all pairs $I \subseteq J$. Applying Theorem 1.2, we see that $x^* = x^*(I,J)$ if and only if we have *primal feasibility*

$$\begin{aligned} x^*_{[n]\setminus J} &= \mathbf{0}, & (1.4)\\ x^*_{J\setminus I} &\geq \mathbf{0}, & (1.5) \end{aligned}$$

along with *dual feasibility*

$$\begin{aligned} \nabla f(x^*)_I &= \mathbf{0}, & (1.6)\\ \nabla f(x^*)_{J\setminus I} &\geq \mathbf{0}, & (1.7) \end{aligned}$$

and *complementarity*

$$\nabla f(x^*)_j x^*_j = 0, \quad j \in J \setminus I. \tag{1.8}$$

Let us focus on the case $I = J$ for a moment. Conditions (1.5), (1.7) and (1.8) are vacuous, so we get that $x^* = x^*(J,J)$ if and only if

$$\begin{aligned} x^*_{[n]\setminus J} &= \mathbf{0},\\ \nabla f(x^*)_J &= \mathbf{0}. \end{aligned} \tag{1.9}$$

Towards the USO. Identifying the n-cube vertices with the sets $J \subseteq [n]$, we will derive the edge orientations from the vectors $x^*(J,J)$. We still need one preparatory

Lemma 1.1. *For $J \subseteq [n]$, $j \in J$ and $I := J \setminus \{j\}$, the following two statements are equivalent.*

(i) $x^*(J,J)_j > 0$.

(ii) $\nabla f(x^*(I,I))_j < 0$.

Proof. If $x^*(J,J)_j > 0$, then $x^*(J,J)$ is feasible and therefore optimal for the more restricted problem SCP(I,J). On the other hand, $x^*(J,J)_j > 0$ shows that $x^*(J,J) \neq x^*(I,I)$. This means, we have $x^*(I,J) \neq x^*(I,I)$, the only possible reason being that (1.7) fails for $x^* = x^*(I,I)$. This shows $\nabla f(x^*(I,I))_j < 0$.

Conversely, $\nabla f(x^*(I,I))_j < 0$ implies $x^*(I,J) \neq x^*(I,I)$, so $x^*(I,J)_j > 0$. Complementarity yields $\nabla f(x^*(I,J))_j = 0$, so $x^*(I,J)$ is also optimal for the less restricted problem SCP(J,J) by (1.9). This yields $x^*(J,J)_j = x^*(I,J)_j > 0$. □

Here is the main result of this section.

Theorem 1.3. *For $J \subseteq [n], j \in J$ and $I := J \setminus \{j\}$, the edge orientations*

$$I \to J :\Leftrightarrow x^*(J,J)_j > 0 \quad (\Leftrightarrow \nabla f(x^*(I,I))_j < 0)$$

define a USO of the n-cube.

Proof. We have to show that every non-empty cube face has a unique sink. In our interpretation of cube vertices as subsets $J \subseteq [n]$, the faces can be identified with *set intervals* of the form

$$[I,J] := \{F \subseteq [n] \mid I \subseteq F \subseteq J\}.$$

We claim that

$$S := I \cup \{j \in J \mid x^*(I,J)_j > 0\} \tag{1.10}$$

is the desired sink of the face $[I,J], I \subseteq J$. First observe that by this definition of S, $x^* = x^*(I,J)$ satisfies

$$\begin{aligned} x^*_{[n]\setminus S} &= \mathbf{0}, & (1.11)\\ \nabla f(x^*)_S &= \mathbf{0}, & (1.12) \end{aligned}$$

by (1.6) and complementarity (1.8). It follows that $x^*(I,J) = x^*(S,S)$, by (1.9). Therefore,

$$\begin{aligned} x^*(S,S)_j &> 0, \quad j \in S \setminus I, & (1.13)\\ \nabla f(x^*(S,S))_j &\geq 0, \quad j \in J \setminus S, & (1.14) \end{aligned}$$

by (1.7). According to the definition of the orientation, S is a sink in $[I,J]$.

Conversely, if S is any sink in $[I,J]$, then the two previous inequalities hold, so $x^*(S,S)$ is feasible for SCP(I,J) since (1.11) and (1.13) imply (1.4) and (1.5), and it is dual feasible since (1.12) and (1.14) imply (1.6) and (1.7). Complementarity (1.8) follows from (1.11) and (1.12). Thus, $x^*(S,S) = x^*(I,J)$, where (1.13) forces S to coincide with the set defined in (1.10). □

1.5 QP-induced USO

Given the convex quadratic program (QP) with n variables, we define for any $\varepsilon > 0$ a quadratic function f_ε by

$$\begin{aligned} f_\varepsilon(x) &:= x^T(A^TA + \varepsilon Q + \varepsilon^2 I)x - 2b^TAx - 2\varepsilon c^Tx \\ &= \|Ax - b\|^2 + 2\varepsilon(\frac{1}{2}x^TQx - c^Tx) + \varepsilon^2\|x\|^2 - b^Tb \end{aligned} \quad (1.15)$$

Here, I is the identity matrix of the appropriate dimension (n in this case). Since Q is positive semi-definite, is follows that $A^TA + \varepsilon Q + \varepsilon^2 I$ is positive definite for all $\varepsilon > 0$, so $f_\varepsilon(x)$ is a strictly convex function.

Let us denote by $\mathrm{SCP}_\varepsilon(I, J)$ the program (1.3) with function $f = f_\varepsilon$. We are interested in the behavior for $\varepsilon \to 0$. We expect that in the limit, the program lexicographically minimises the triple

$$\left(\|Ax - b\|^2\,,\ \frac{1}{2}x^TQx - 2c^Tx\,,\ \|x\|^2\right).$$

In the feasible and bounded case, the solution $x^*_\varepsilon(\emptyset, [n])$ of $\mathrm{SCP}_\varepsilon(\emptyset, [n])$ should therefore converge to the optimal QP solution of minimum norm. In the following we will render this claim more precisely.

In order to understand the USO induced by f_ε, we have to know the values $x^*_\varepsilon(J, J)$. From (1.9) it follows that $x^*_\varepsilon(J, J) \in \mathbb{R}^J$ is obtained as the unique solution of the linear equation system

$$\frac{\nabla f_\varepsilon(x)^T_J}{2} = (A^T_JA_J + \varepsilon Q^J_J + \varepsilon^2 I)x_J - A^T_Jb - \varepsilon c_J = \mathbf{0} \quad (1.16)$$

with $x \in \mathbb{R}^J$.

Where by writing A^T_J, we mean $(A_J)^T$. Now as also stated in [3], we see that these induced USO "converge" in the following sense:

Lemma 1.2. *Let $\xrightarrow{\varepsilon}$ be the USO of the n-cube induced by f_ε according to Theorem 1.3. Then there exists a USO $\to$ such that $\xrightarrow{\varepsilon} = \to$ for sufficiently small ε.*

This "limiting" USO is called the USO *induced* by the QP.

Proof. Using Cramer's rule to compute the solution $x^*_\varepsilon(J,J)_J$ of the system (1.16), we see that the entries of all $x^*_\varepsilon(J,J)_j$ are rational functions in ε. By Theorem 1.3, $\xrightarrow{\varepsilon}$ is determined by the signs of finitely many of these rational functions.

Now, for any nonzero rational function $r(\varepsilon)$, there is an open interval of the form $(0,\delta)$ in which neither its numerator nor its denominator has any zeros. In this interval, the sign of $r(\varepsilon)$ is fixed. The lemma follows. □

This also provides a way of computing edge orientations in the limiting USO $\to$: simply compute the rational function "responsible" for the orientation in question and find the terms with the smallest ε-power in both the numerator and the denominator. These determine the sign of the rational function for $\varepsilon \to 0$.

Nevertheless, since the limiting USO does not depend on ε, there must be a way of avoiding computations involving ε. Our approach is to develop $x^*_\varepsilon(J,J)$ into a power series, and this will also be crucial for understanding the global sink of $\to$ in the next section.

A zoo of unconstrained programs. It will turn out that the coefficients of the power series expansion are solutions to unconstrained strictly convex quadratic programs (where unconstrained means that there are no inequalities). All "animals" in the following list are unique optimal solutions of their defining programs.

Definition 1.2. *For $J \subseteq [n]$, set*

$$\begin{aligned}
\bar{b}(J) := \quad & \operatorname{argmin} \quad (b-y)^T(b-y) \\
& \text{s.t.} \qquad\quad Ax \;=\; y && x \in \mathbb{R}^J,
\end{aligned}$$

$$\begin{aligned}
b(J) := \quad & \operatorname{argmin} \quad (b-y)^T(b-y) \\
& \text{s.t.} \qquad\quad A_J^T y \;=\; \mathbf{0},
\end{aligned}$$

$$\begin{aligned}
\bar{c}(J) := \quad & \operatorname{argmin} \quad (c-x)^T(c-x) \\
& \text{s.t.} \qquad \begin{pmatrix} A_J \\ Q_J \end{pmatrix}^T z \;=\; x_J && z \in \mathbb{R}^{2m},
\end{aligned}$$

$$\begin{aligned}
g_{-1}(J) := c(J) := \quad & \operatorname{argmin} \quad (c-x)^T(c-x) \\
& \text{s.t.} \qquad \begin{pmatrix} A \\ Q \end{pmatrix} x \;=\; \mathbf{0} && x \in \mathbb{R}^J,
\end{aligned}$$

$$\begin{aligned}
g_0(J) := x(J) := \quad & \operatorname{argmin} \quad x^T x \\
& \text{s.t.} \qquad \begin{pmatrix} A_J^T A \\ Q^J \end{pmatrix} x \;=\; \begin{pmatrix} A_J^T b \\ \bar{c}(J)_J + A_J^T A\lambda \end{pmatrix} && x, \lambda \in \mathbb{R}^J,
\end{aligned}$$

$$\begin{aligned}
g_1(J) := t(J) := \quad & \operatorname{argmin} \quad x^T x \\
& \text{s.t.} \qquad \begin{pmatrix} A_J^T A \\ Q^J \end{pmatrix} x \;=\; \begin{pmatrix} -Q^J x(J) + \bar{c}(J)_J \\ -x(J)_J + A_J^T A\lambda \end{pmatrix} && x, \lambda \in \mathbb{R}^J,
\end{aligned}$$

and for $i \geqslant 2$, we recursively define

$$\begin{aligned}
g_i(J) \qquad := \quad & \operatorname{argmin} \quad x^T x \\
& \text{s.t.} \quad \begin{pmatrix} A_J^T A \\ Q^J \end{pmatrix} x \;=\; \begin{pmatrix} -Q^J g_{i-1}(J) - g_{i-2}(J)_J \\ -g_{i-1}(J)_J + A_J^T A\lambda \end{pmatrix} && x, \lambda \in \mathbb{R}^J.
\end{aligned}$$

If not indicated otherwise, x ranges over $\mathbb{R}^n$ and y over $\mathbb{R}^m$.

Note that the first four programs are easily seen to have feasible solutions and, since the objective functions are strictly convex, the optimal solutions are unique.

To get an intuition what these values are, let us consider for $\gamma \in \mathbb{R}^n$ and $\beta \in \mathbb{R}^m$ the (unconstrained, and therefore quite boring) quadratic program

$$\begin{aligned}
QP(J) \qquad & \min \quad \tfrac{1}{2} x^T Q x - \gamma^T x \\
& \text{s.t.} \quad Ax \;=\; \beta, \quad x \in \mathbb{R}^J,
\end{aligned} \tag{1.17}$$

along with its dual

$$QP^{\triangle}(J) \qquad \begin{array}{ll} \max & -\frac{1}{2}x^TQx - y^T\beta \\ \text{s.t.} & Q_J^Tx + A_J^Ty \;=\; \gamma_J, \quad x \in \mathbb{R}^J, \end{array} \tag{1.18}$$

The vector $\beta = \bar{b}(J)$ is the vector closest to b such that $QP(J)$ is feasible. Dually, $\gamma = \bar{c}(J)$ is the vector closest to c such that $QP^{\triangle}(J)$ is feasible. $b(J)$ is the projection of b onto the kernel of A_J^T, while $c(J)$ is the projection of c onto the kernel of $\binom{A_J}{Q_J}$.

First, we would like to prove feasibility of all programs of the "zoo". To do so, we are required to bring A^TA and Q into a nicer form than they have in general. We will also use this special form later in this section to prove that our defined "animals" are indeed the coefficients of the desired power series of $x^*_\varepsilon(J,J)$.

A "simultaneously diagonal form" for $A_J^TA_J$ and Q_J^J. For any A, the matrix $A_J^TA_J$ is symmetric. We choose coordinates such that $A_J^TA_J$ is diagonal, i.e. let T be an orthogonal transformation matrix such that $T\,A_J^TA_J\,T^T = diag(a_1,\ldots,a_l,0,\ldots,0)$ with all a_i being nonzero, $T^TT = I$.

Furthermore, we would like to have Q_J^J "as diagonal as possible" at the same time, but we have to be careful about which of the coordinates we are still allowed to change, without destroying the nice form of A. What can be done is the following: For $T\,Q_J^J\,T^T = \begin{pmatrix} Q_1 & Q_a \\ Q_a^T & Q_b \end{pmatrix}$, $Q_1 \in \mathbb{R}^{l\times l}$, let U be such that $U\,Q_b\,U^T = diag(q_{l+1},\ldots,q_m,0,\ldots,0)$ with the q_i being nonzero; $U^TU = I$. This is possible because Q, and thus also Q_b, is symmetric.

We now define a new transformation $P := \begin{pmatrix} I_{l\times l} & 0 \\ 0 & U \end{pmatrix} T$, and observe that this P still diagonalises $A_J^TA_J$ in exactly the same way, and that $P^TP = I$. (geometrically we have just chosen another basis for the kernel of $A_J^TA_J$).

By our choice of P we have:

$$P\,Q_J^J\,P^T = \begin{pmatrix} Q_1 & Q_2 & Q_3 \\ Q_2^T & \begin{matrix} q_{l+1} & & \\ & \ddots & \\ & & q_m \end{matrix} & 0 \\ Q_3^T & 0 & \begin{matrix} 0 & & \\ & \ddots & \\ & & 0 \end{matrix} \end{pmatrix} \tag{1.19}$$

Now since this matrix is still symmetric and positive semi-definite, it further follows that $Q_3 = 0$. This is by the small Lemma 1.3 following here:

Lemma 1.3. *For Q a symmetric, positive semi-definite matrix*

$$Q = \begin{pmatrix} R & S \\ S^T & 0 \end{pmatrix} \quad \Rightarrow \quad Q = \begin{pmatrix} R & 0 \\ 0 & 0 \end{pmatrix}$$

Proof. Assume $R \in \mathbb{R}^{v \times v}$, $S \in \mathbb{R}^{v \times w}$. Let $i \in [v]$, $j \in [w]$. We make use of Q being positive semi-definite: Define $x : x_i = 1, x_{v+j} = -\frac{R_{ii}}{2S_{ij}} + 1$ and $x_k = 0$ otherwise, to obtain $x^T Q x = 2S_{ij} \geqslant 0$.
Set $z : z_i = 1, z_{v+j} = -\frac{R_{ii}}{2S_{ij}} - 1$ and $z_k = 0$ otherwise, to obtain $z^T Q z = -2S_{ij} \geqslant 0$. Both inequalities together imply $S_{ij} = 0$. The claim follows. □

We finally have the following two equalities. This is what we will call the *"simultaneous diagonal form"*:

$$P\, A_J^T A_J\, P^T = \begin{pmatrix} a_1 & & & & \\ & \ddots & & 0 & \\ & & a_l & & \\ & 0 & & \overset{0}{\ddots}_{0} & \end{pmatrix} =: A' \tag{1.20}$$

$$P\, Q_J^J\, P^T = \begin{pmatrix} Q_1 & & Q_2 & & 0 \\ & q_{l+1} & & & \\ Q_2^T & & \ddots & & 0 \\ & & & q_m & \\ 0 & & 0 & & \overset{0}{\ddots}_{0} \end{pmatrix} =: Q' \tag{1.21}$$

We will now make use of this special form to prove the following

Lemma 1.4. *All the defining programs for the $g_i(J)$, $i \geqslant -1$, are feasible, and thus well-defined.*

Proof. To prove feasibility of the problems for the $g_i(J)$, we look at the constraints in the new coordinates we proposed in the paragraph above.

First we would like to prove feasibility for the defining problem for $x(J)$. In the new coordinates, $x'(J) = P\, x(J)$ and $\lambda' = P\, \lambda$, the constraints for $x(J)$ are of the following form:

$$\begin{aligned} A' x'(J) &= P A_J^T b \\ Q' x'(J) &= P \bar{c}(J)_J + A' \lambda' \end{aligned} \tag{1.22}$$

Let $b' \in \mathbb{R}^{|J|}$ such that $PA_J^T b = A'b'$ [2] and let $P\bar{c}(J)_J = A'\mu' + Q'\nu'$ [3]. Now the constraints are equivalent to the following equations:

$$\begin{aligned} A'x'(J) &= A'b' \\ Q'x'(J) - A'\lambda' &= Q'\nu' + A'\mu' \end{aligned} \tag{1.23}$$

To see feasibility we essentially do a top-down insertion. By the form of A', the upper equation will uniquely define $x'(J)_k$ in the coordinates $1 \leqslant k \leqslant l$. Next we satisfy the rows $l < k \leqslant m$ of the lower equation by choosing suitable $x'(J)_k$, $l < k \leqslant m$. This is possible as Q' has "diagonal form" in these coordinates, and the q_k are non-zero for $l < k \leqslant m$.

Finally it remains to get the equality for the rows $1 \leqslant k \leqslant l$ of the lower equation, but this is now easy by just choosing an appropriate λ (remember $a_k \neq 0$). Note that this choice of λ leaves rows $l < k \leqslant n$ untouched since A' is zero in these rows.

This proves feasibility for the defining problem for $x'(J)$, and thus also for $x(J)$.

For the higher $g_i(J)$ we follow the same reasoning, with the difference that here, feasibility of the upper equation for $g_i(J)$ directly follows from the lower equation of the preceding program for $g_{i-1}(J)$.

In the lower equation we analogously use that, from the KKT conditions for the preceding defining program, $g_{i-1}(J) = A_J^T\mu + Q_J^J\nu$ for some μ and ν.

Observe that, if all the defining programs for the $g_i(J)$ are feasible, the $g_i(J)$ are indeed well-defined due to strict convexity. □

Note that for the case of *linear programming* (where $Q = \mathbf{0}$), we do not need this complicated "diagonal" form reasoning at all, since for LP, every lower equation follows automatically from the KKT conditions for the preceding defining problem.

The following technical lemma will explain the meaning of some of the "zoo animals" a bit more precisely. We will make use of these properties at different positions later in the section.

[2] this is possible since $A_J^T A_J x = A_J^T b$ always has a solution for x.

[3] Check that the KKT conditions for the defining program for $c(J)$ imply $\bar{c}(J) = A_J^T\mu + Q_J^J\nu$. Thus we can write $P\bar{c}(J)_J = A'\mu' + Q'\nu'$.

Lemma 1.5. *For all $J \subseteq [n]$, the following holds:*

(i) $b = \bar{b}(J) + b(J)$ *and* $c = \bar{c}(J) + c(J)$.

(ii) $\bar{b}(J)^T\, b(J) = \bar{c}(J)^T\, c(J) = 0$.

(iii) If $b(J) \neq \mathbf{0}$, *then* $b^T b(J) > 0$, *and if* $c(J) \neq \mathbf{0}$, *then* $c^T c(J) > 0$.

(iv) $x(J)$ *has the following alternative definition:*

$$\begin{array}{rcll} x(J) & = & \text{argmin} & x^T x \\ & & \text{s.t.} & \begin{pmatrix} A \\ Q^J \end{pmatrix} x \;=\; \begin{pmatrix} \bar{b}(J) \\ \bar{c}(J)_J + A_J^T A\lambda \end{pmatrix} \qquad x, \lambda \in \mathbb{R}^J. \end{array}$$

Proof. (i) We only give the argument for b here, the one for c is similar. The Karush–Kuhn–Tucker conditions for the program defining $\bar{b}(J)$ restricted to (x_J, y) (Theorem 1.2) show that

$$\begin{aligned} A_J x^* &= \bar{b}(J), \\ (\mathbf{0}^T, 2(\bar{b}(J) - b)) &= \lambda^T (A_J, -I), \end{aligned} \tag{1.24}$$

for some $x^* \in \mathbb{R}^J, \lambda \in \mathbb{R}^m$. Set $y^* = \lambda/2$. It follows that $y^* = b - \bar{b}(J)$ and $A_J^T y^* = \mathbf{0}$. Moreover, with $\mu := -2x^*$, we have $2(y^* - b)^T = -2\bar{b}(J) = \mu^T A_J^T$. This means that y^* and μ satisfy the KKT conditions for the program defining $b(J)$. Then $b(J) = y^* = b - \bar{b}(J)$ follows.

Geometrically, $\bar{b}(J)$ is the projection of b onto the column space of A_J, while $b(J)$ is the projection of b onto the orthogonal complement of the column space.

(ii) The KKT conditions for the defining program for $b(J)$, together with (i), imply $(b - b(J))_J^T = \bar{b}(J)_J^T = \lambda A_J$, so by definition of $b(J)$ we get $\bar{b}(J)_J^T\, b(J)_J = \lambda A_J b(J)_J = 0$, thus $\bar{b}(J)^T\, b(J) = 0$.

The argument for c is the same, but is still given here for completeness: The KKT conditions for $c(J)$, together with (i), imply $\bar{c}(J)_J^T = \lambda \binom{A_J}{Q_J}$, so by definition of $c(J)$ we get $\bar{c}(J)_J^T\, c(J)_J = \lambda \binom{A_J}{Q_J} c(J)_J = 0$, thus $\bar{c}(J)^T\, c(J) = 0$.

(iii) By (i) we have $b^T b(J) = (\bar{b}(J)^T + b(J)^T) b(J)$. And by part (ii) of this lemma, this equals to $b(J)^T b(J)$ which is positive if and only if $b(J) \neq 0$. The argument for c is the same.

Here is an alternative proof without using (i) and (ii): $b(J) \neq \mathbf{0}$, optimality of $b(J)$ under a strictly convex function yields

$$(b - b(J))^T(b - b(J)) < (b - \mathbf{0})^T(b - \mathbf{0}) = b^T b.$$

The inequality $2b^T b(J) > b(J)^T b(J) \geq 0$ follows.

(iv) In proving $A_J^T y^* = \mathbf{0}$ in (i), we have shown that $A_J^T \bar{b}(J) = A_J^T b$. Since any feasible linear system $Mx = q$ is equivalent to $M^T Mx = M^T q$,[4] we know that the system $A_J x_J = \bar{b}(J)$ can be replaced by $A_J^T A_J x_J = A_J^T \bar{b}(J) = A_J^T b$. The claim follows. □

The power series expansion. The following theorem proves that the $g_i(J)$ we defined above are indeed the coefficients of the power series expansion of $x_\varepsilon^*(J, J)$.

Theorem 1.4 (Convergence Theorem)**.** *Let $J \subseteq [n]$. Then for any $k \geqslant 0$,*

$$x_\varepsilon^*(J, J) = \sum_{i=-1}^{k} \varepsilon^i g_i(J) + O(\varepsilon^{k+1}),$$

where the big-O notation refers to the asymptotic behavior for $\varepsilon \to 0$.

Proof. We define remainder terms $r_\varepsilon^{(k)}(J) \in \mathbb{R}^J$ by:

$$r_\varepsilon^{(k)}(J) := x^*(J, J) - \sum_{i=-1}^{k} g_i(J)\varepsilon^i \tag{1.25}$$

In order to prove the theorem, we have to show that $r_\varepsilon^{(k)}(J) \in O(\varepsilon^{k+1})$ for any $k \geqslant 0$.

[4] We need to show that $M^T Mx = M^T q$ implies $Mx = q$. Take any x' such that $Mx' = q$. Then we get $M^T Mx' = M^T q$ and $M^T M(x' - x) = \mathbf{0}$. Also, $(x' - x)^T M^T M(x' - x) = \|M(x' - x)\|^2 = 0$ and $M(x' - x) = \mathbf{0}$ follows; hence $Mx = Mx' = q$.

We plug in the definition of $x^*(J,J)$ (see equation 1.16) into (1.25) to obtain the following equation for $r_\varepsilon^{(k)}(J) \in \mathbb{R}^J$:

$$\begin{aligned}(A_J^T A + \varepsilon Q^J + \varepsilon^2 I)\, r_\varepsilon^{(k)}(J) &= (A_J^T A + \varepsilon Q^J + \varepsilon^2 I) \left(x^*(J,J) - \sum_{i=-1}^{k} g_i(J)\varepsilon^i \right) \\ &= A_J^T b + \varepsilon c \\ &\quad -(A_J^T A + \varepsilon Q^J + \varepsilon^2 I) \left(\sum_{i=-1}^{k} g_i(J)\varepsilon^i \right)\end{aligned}$$

Now to evaluate the right side, we group all terms with the same power in ε. We see that all terms except the last two cancel out, as they exactly have the form of the constraints from the defining programs of the $g_i(J)$, $i \geqslant -1$:

$$\begin{aligned}
& \left(-A_J^T A c(J)\right) \varepsilon^{-1} \\
& \left(-A_J^T A x(J) \quad -Q^J c(J) \quad +A_J^T b\right) \varepsilon^0 \\
& \left(-A_J^T A t(J) \quad -Q^J x(J) \quad +c_J - c(J)_J\right) \varepsilon^1 \\
= & \quad \vdots \qquad\qquad \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\
& \left(-A_J^T A g_k(J) \quad -Q^J g_{k-1}(J) \quad -g_{k-2}(J)_J\right) \varepsilon^k \\
& \left(\qquad\qquad -Q^J g_k(J) \quad -g_{k-1}(J)_J\right) \varepsilon^{k+1} \\
& \left(\qquad\qquad\qquad\qquad -g_k(J)_J\right) \varepsilon^{k+2} \\
= & \left(-Q^J g_k(J) - g_{k-1}(J)_J\right) \varepsilon^{k+1} - g_k(J)_J \varepsilon^{k+2} \\
= & \varepsilon^{k+1} \left(A_J^T A g_{k+1}(J) - \varepsilon g_k(J)_J\right) \qquad (1.26)
\end{aligned}$$

Where to obtain the last equality we again used the definition of $g_{k+1}(J)$. Note that in doing this calculation we assumed $k \geqslant 1$, but you may easily check that the resulting identity also holds for the case $k = 0$.

We now choose coordinates such that $A_J^T A_J$ is diagonal, and Q_J^J is "as diagonal as possible", as we described in the corresponding paragraph above (remember $A' = P\, A_J^T A_J\, P^T$ and $Q' = P\, Q_J^J\, P^T$ from equations 1.20 and 1.21).

In the new coordinates, $r_\varepsilon^{\prime(k)} := P\, r_\varepsilon^{(k)}(J)_J$ and $g_k' := P\, g_k(J)_J$, the equation (1.26) we obtained for $r_\varepsilon^{(k)}(J)$ is of the following form:

$$(A' + \varepsilon Q' + \varepsilon^2 I)\, \frac{r_\varepsilon^{\prime(k)}}{\varepsilon^{k+1}} = A' g_{k+1}' - \varepsilon g_k' \qquad (1.27)$$

Observe that $A'g'_{k+1} \in \mathbb{R}^{[l]}$, and by the definition of $g_{k+2}(J)$ we also know that $g'_k = A'g'_{k+2} + Q'g'_{k+1}$, thus $g'_k \in \mathbb{R}^{[m]}$. This means the defining equations for $\frac{r_\varepsilon^{'(k)}}{\varepsilon^{k+1}}$ all have the desired form for the following Lemma 1.6 which finally gives us $\frac{r_\varepsilon^{'(k)}}{\varepsilon^{k+1}} \in O(1)$ and thus $r_\varepsilon^{(k)}(J) \in O(\varepsilon^{k+1})$. □

Note: a similar argument works to extend the theorem to the case $k = -1$.

Lemma 1.6. *$v \in \mathbb{R}^{[l]}$ and $w \in \mathbb{R}^{[m]}$. Let $r \in \mathbb{R}^n$ be the solution of the following matrix equation:*

$$\left(\begin{pmatrix} a_1 & & & & \\ & \ddots & & 0 & \\ & & a_l & & \\ & 0 & & \ddots & 0 \end{pmatrix} + \varepsilon \begin{pmatrix} Q_1 & Q_2 & 0 \\ & q_{l+1} & & \\ Q_2^T & & \ddots & 0 \\ & & & q_m \\ 0 & 0 & & \ddots 0 \end{pmatrix} + \varepsilon^2 I \right) r \;=\; v + \varepsilon w \;.$$

Then $r \in O(1)$.

Proof. Suppose the claim is wrong, thus there is an i with $r_i \notin O(1)$. Then $r_i \in \Omega(\varepsilon^{-1})$ by Lemma 1.7. Now let i be s.t. $r_i \in \Omega(\varepsilon^k)$ with $k \leqslant -1$ minimal.

We distinguish 3 cases:

Case 1: $1 \leqslant i \leqslant l$: the corresponding line in the matrix equation is:

$$(a_i + \varepsilon^2) r_i + \varepsilon (Q_1\, Q_2\, 0)_i\, r = v_i + \varepsilon w_i \tag{1.28}$$

Then $r_i \in \Omega(\varepsilon^k)$ implies existence of j s.t. $(\varepsilon (Q_1\, Q_2\, 0)\, r)_j \in \Omega(\varepsilon^k)$ and thus $r_j \in \Omega(\varepsilon^{k-1})$, which contradicts the choice of i.

Case 2: $l + 1 \leqslant i \leqslant m$: the corresponding line in the matrix equation is:

$$(q_i + \varepsilon) r_i + (Q_2^T\, 0\, 0)_{i-l}\, r = w_i \tag{1.29}$$

Then $r_i \in \Omega(\varepsilon^k)$ implies existence of j s.t. $((Q_2^T\, 0\, 0) r)_j \in \Omega(\varepsilon^k)$ and thus we have $j \leqslant l$ with $r_j \in \Omega(\varepsilon^k)$, we now proceed exactly as in case 1 to obtain a contradiction.

Case 3: $m + 1 \leqslant i \leqslant n$: the corresponding line in the matrix equation is:

$$r_i = 0 \tag{1.30}$$

Which directly contradicts our assumption. □

Lemma 1.7. *Let f be a rational function in ε, let $k \in \mathbb{N}, k \leqslant 0$. Then*

$$f \notin O(\varepsilon^k) \;\Rightarrow\; f \in \Omega(\varepsilon^{k-1})$$

with O and Ω referring to $\varepsilon \to 0$.

Proof. We factorise both numerator and denominator of f into their unique irreducible representation (they are polynomials over $\mathbb{R}$), and look at the resulting unique monomial in ε:
$f(\varepsilon) = \varepsilon^i\, g(\varepsilon),\ i \in \mathbb{N},\ \ \text{with } g(\varepsilon) \to z \neq 0 \text{ as } \varepsilon \to 0.$
So $f \notin O(\varepsilon^k) \;\Rightarrow\; f \in \Omega(\varepsilon^{k-1})$. □

The convergence theorem shows that we can read off the edge orientation $J \setminus \{j\} \to J$ in the LP-induced USO from the first nonzero coefficient in the power series expansion of $x^*_\varepsilon(J, J)$. The following corollary is taken from [3]:

Corollary 1.8. *Let $J \subseteq [n], j \in J$, and set $I := J \setminus \{j\}$. Furthermore, define*

$$i(J, j) := \min\{i \geq -1 \mid g_i(J)_j \neq 0\}.$$

Then $i(J, j) = \infty$ or $i(J, j) \leq 2|J| - 1$, and the LP-induced USO $\to$ derived in Lemma 1.2 induces the edge orientation

$$I \to J \quad \Leftrightarrow \quad g_{i(J,j)}(J)_j > 0, \tag{1.31}$$

where we set $g_\infty(J)_j := 0$.

Since our power series expansion also induces an expansion of $\nabla f(x^*_\varepsilon(J, J))$ (responsible for the "upward" edges at J for small ε), we can compute the orientations of *all* edges incident to a given vertex J in the LP-induced USO by solving at most $2|J| + 2$ unconstrained quadratic programs, and hopefully much less in most cases; this is the vertex evaluation oracle. By the Karush–Kuhn–Tucker conditions, this is easy and reduces to solving linear equation systems.

Proof. We have

$$I \xrightarrow{\varepsilon} J \Leftrightarrow x^*_\varepsilon(J, J)_j > 0,$$

see the definition of the orientation in Theorem 1.3. Then, according to the previous theorem, the first nonzero value $g_i(J)_j$ determines the sign of

$x^*_\varepsilon(J,J)_j$ for sufficiently small ε, and this is the sign that defines the orientation $I \to J$ in the limiting USO.

For the bound on $i(J,j)$ in the finite case, recall that $x^*_\varepsilon(J,J)_j$ is a rational function, and if it is nonzero, the numerator contains a monomial ε^i with $i \leq 2|J|-1$ (this is again Cramer's rule, applied to the system (1.16). It follows that

$$|x^*_\varepsilon(J,J)_j| = \Omega\left(\varepsilon^{2|J|-1}\right)$$

for $\varepsilon \to 0$, and the previous theorem implies that $i(J,j) \leq 2|J|-1$.

On the other hand, $i(J,j) = \infty$ implies $x^*_\varepsilon(J,J)_j = 0$, so (1.31) gives the right orientation also in this case. □

1.6 QP-induced USO: The sink

Definition 1.3. *For all $J \subseteq [n]$ we define*

$$y(J) := At(J) \tag{1.32}$$

Note that by feasibility of $t(J)$, this implies $A^T_J y(J) + Q^J x(J) = \bar{c}(J)_J$.

Let $S \subseteq [n]$ be the sink of the QP-induced USO $\to$. From Theorem 1.4 we know that

$$x^*_\varepsilon(S,S) = \frac{c(S)}{\varepsilon} + x(S) + \varepsilon t(S) + O(\varepsilon^2), \tag{1.33}$$

which implies

$$\begin{aligned} \nabla f(x^*_\varepsilon(S,S))^T/2 &= A^T(\bar{b}(S) - b) \\ &+ \varepsilon(A^T y(S) + Qx(S) - \bar{c}(S)) \\ &+ O(\varepsilon^2), \end{aligned} \tag{1.34}$$

using (1.16), $Ac(S) = \mathbf{0}, Qc(S) = \mathbf{0}, Ax(S) = \bar{b}(S)$ and $At(S) = y(S)$, see definitions, and $c = c(S) + \bar{c}(S)$.

For sufficiently small ε, S is also the sink in $\xrightarrow{\varepsilon}$, so $x^*_\varepsilon(S,S) = x^*_\varepsilon(\emptyset,[n])$. Using the optimality criteria (1.5), (1.7) and (1.8), we deduce

$$x^*_\varepsilon(S,S) \geqslant \mathbf{0}, \tag{1.35}$$

$$\nabla f(x^*_\varepsilon(S,S)) \geqslant \mathbf{0}, \tag{1.36}$$

$$\nabla f(x^*_\varepsilon(S,S))_j \, x^*_\varepsilon(S,S)_j = 0, \quad j \in [n]. \tag{1.37}$$

This implies the following

Theorem 1.5. *Consider the quadratic program*

$$\begin{array}{ll} \min & \frac{1}{2}x^TQx - \bar{c}(S)^Tx \\ \text{s.t.} & Ax = \bar{b}(S) \\ & x \geqslant 0, \end{array} \tag{1.38}$$

along with its dual

$$\begin{array}{ll} \max & -\frac{1}{2}x^TQx - y^T\,\bar{b}(S) \\ \text{s.t.} & Qx + A^Ty \geqslant \bar{c}(S). \end{array} \tag{1.39}$$

Then, for sufficiently small $\varepsilon > 0$, the following statements hold:

(i) $x(S) + c(S)/\varepsilon$ is optimal for (3.20).

(ii) $(x(S), y(S) - b(S)/\varepsilon)$ is optimal for (3.21).

Proof. Putting together (1.33) and (1.35) shows that $x(S) + c(S)/\varepsilon \geqslant \mathbf{0}$ for sufficiently small $\varepsilon > 0$, and feasibility for (3.20) follows from the definitions of $c(S), x(S)$. On the other hand, combining (1.34) and (1.36) implies that either $-A^Tb(S) \geqslant \mathbf{0}$ and $A^Ty(S) + Qx(S) \geqslant \bar{c}(S)$, or $-A^Tb(S) > \mathbf{0}$ (remember $b(S) = b - \bar{b}(S)$). In both cases $(x(S), y(S) - b(S)/\varepsilon)$ is feasible for (3.21), for sufficiently small ε.

To prove optimality we compare the two objectives and prove that they are equal:

$$\begin{array}{rcl} \frac{1}{2}\left(x(S) + \frac{c(S)}{\varepsilon}\right)^T Q \left(x(S) + \frac{c(S)}{\varepsilon}\right) & & -\bar{c}(S)^T\left(x(S) + \frac{c(S)}{\varepsilon}\right) \\ = & \frac{1}{2}x(S)^TQx(S) & -\bar{c}(S)^Tx(S) \\ = & \frac{1}{2}x(S)^TQx(S) & -\bar{c}(S)_S^Tx(S)_S \\ = & \frac{1}{2}x(S)^TQx(S) & -(A_S^Ty(S) + Q^Sx(S))^Tx(S)_S \\ = & -\frac{1}{2}x(S)^TQx(S) & -y(S)^TAx(S) \\ = & -\frac{1}{2}x(S)^TQx(S) & -y(S)^T\bar{b}(S) \\ = & -\frac{1}{2}x(S)^TQx(S) & -\left(y(S) - \dfrac{b(S)}{\varepsilon}\right)^T\bar{b}(S) \end{array} \tag{1.40}$$

where we used $Qc(S) = 0$, $\bar{c}(S)^Tc(S) = 0$[5] , $b(S)^T\bar{b}(S) = 0$ and $A_S^Ty(S) + Q^Sx(S) = \bar{c}(S)_S$, $x(S) \in \mathbb{R}^S$.
So optimality of (i) and (ii) follows from weak duality. □

[5]From technical Lemma 1.5 (ii).

Note that this calculation also proves *strong duality* for feasible and bounded QP and LP, by using *weak duality*.

We have shown that the sink S of the QP-induced USO gives us a primal-dual pair of optimal solutions to a modified QP (3.20). Here is what we can deduce about the *original* QP, our primary object of interest.

In the following, an *unbounded ray* for a QP is a half-line whose tail (everything except some initial segment) is feasible, and on which the objective function is unbounded.

Theorem 1.6. *Let S be the sink of the USO induced by the QP*

$$(QP) \quad \begin{array}{ll} \min & \frac{1}{2}x^TQx - c^Tx \\ \text{s.t.} & Ax = b \\ & x \geqslant 0. \end{array} \tag{1.41}$$

(i) If $\bar{b}(S) \neq b$, the QP (1.41) is infeasible. Equivalently, the QP

$$\begin{array}{ll} \min & \frac{1}{2}x^TQx - \bar{c}(S)^Tx \\ \text{s.t.} & Ax = b \\ & x \geqslant 0 \end{array}$$

is infeasible, and this is witnessed by the fact that

$$\{\left(x(S),\, y(S) - \frac{b(S)}{\varepsilon}\right) \mid \varepsilon > 0\}$$

is an unbounded ray of the dual problem

$$\begin{array}{ll} \max & -\frac{1}{2}x^TQx - b^Ty \\ \text{s.t.} & Qx + A^Ty \geqslant \bar{c}(S). \end{array}$$

(ii) If $\bar{b}(S) = b$ and $\bar{c}(S) \neq c$, the QP (1.41) is feasible but unbounded, and this is witnessed by the fact that

$$\{x(S) + \frac{c(S)}{\varepsilon} \mid \varepsilon > 0\}$$

is an unbounded ray of (1.41).

(iii) If $\bar{b}(S) = b$ and $\bar{c}(S) = c$, then $x(S)$ and $(x(S), y(S))$ is a pair of primal and dual optimal solutions to the QP (1.41).

Proof. By weak duality, the existence of a dual unbounded ray implies infeasibility of the primal problem, so in order to show (i) and (ii), it remains to prove that the given rays are indeed unbounded. But this follows from $c^T c(S) > 0$ and $b^T b(S) > 0$, see technical lemma 1.5(iii). Property (iii) is a corollary of the previous theorem, under $b(S) = b - \bar{b}(S) = \mathbf{0}$ and $c(S) = c - \bar{c}(S) = \mathbf{0}$. □

We remark that the quadratic programs (3.20) and (3.21) can also be defined without reference to the sink S. The following lemma proves this claim, taking into account that $A^T(\bar{b}(S) - b) \geqslant \mathbf{0}$ and $c(S) \geqslant \mathbf{0}$:

Lemma 1.9. *With S the sink of the QP-induced USO, we have*

$$\begin{array}{rcll} \bar{b}(S) & = & \text{argmin} & (b-y)^T(b-y) \\ & & \text{s.t.} & Ax = y \\ & & & x \geqslant \mathbf{0} \end{array}$$

and

$$\begin{array}{rcll} \bar{c}(S) & = & \text{argmin} & (c-x)^T(c-x) \\ & & \text{s.t.} & \binom{A}{Q}^T y \geqslant x. \end{array}$$

Proof. We see that obviously both newly defined programs are feasible and thus well-defined. Lets denote the values of the new programs by $\tilde{b}(S)$ and $\tilde{c}(S)$ and write the conditions as equalities with all variables on the right hand side (we introduce a new 'slack' variable z):

$$\begin{array}{rcll} \tilde{b}(S) & = & \text{argmin} & (b-y)^T(b-y) \\ & & \text{s.t.} & (A, -I)\begin{pmatrix} x \\ y \end{pmatrix} = \mathbf{0}, \\ & & & x \geqslant \mathbf{0}, \end{array}$$

$$\begin{array}{rcll} \tilde{c}(S) & = & \text{argmin} & (c-x)^T(c-x) \\ & & \text{s.t.} & \left(-I, \binom{A}{Q}^T, -I\right)\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \mathbf{0}, \\ & & & z \geqslant \mathbf{0}. \end{array}$$

Consider the KKT conditions for the first new defining program:

$$\mathbf{0} \geqslant A^T\lambda \tag{1.42}$$

$$-2(b - \tilde{b}(S)) = -I\lambda \tag{1.43}$$

we see that these conditions are satisfied for $\tilde{b}(S) := \bar{b}(S)$, because of our additional inequality $A^T(\bar{b}(S) - b) \geqslant \mathbf{0}$ that holds at the sink S. Thus the optimal solution to the second program is indeed $\bar{b}(S)$.

For the second new defining program, we proceed similarly: The KKT conditions for $\tilde{c}(S)$,

$$\begin{aligned} -2(c - \tilde{c}(S)) &= -I\lambda && (1.44) \\ \mathbf{0} &= \begin{pmatrix} A \\ Q \end{pmatrix}\lambda && (1.45) \\ \mathbf{0} &\geqslant -I\lambda && (1.46) \end{aligned}$$

are also satisfied for $\tilde{c}(S) := \bar{c}(S)$, because of our additional inequality $c - \bar{c}(S) = c(J) \geqslant \mathbf{0}$ that holds at the sink S, and since $\binom{A}{Q} c(S) = \mathbf{0}$ by definition. So the optimal solution to the new program is indeed $\bar{c}(S)$. □

This lemma tells us the real meaning of the strange looking values $\bar{b}(S)$ and $\bar{c}(S)$, and also motivates the modification of our QP as in (3.20) and (3.21): $\bar{b}(S)$ is the closest replacement for b which makes the original QP feasible, while $\bar{c}(S)$ is the closest replacement for c which makes the dual problem feasible. In this sense, the USO approach solves the feasible and bounded QP "closest" to the original one.

2 Perturbing Linear Programming

In this section we consider the unique sink orientations approach to linear programming

$$\begin{array}{lll} (LP) & \max & c^T x \\ & \text{s.t.} & Ax = b \\ & & x \geq 0, \end{array} \tag{2.1}$$

with $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. The reduction of LP to a unique sink orientation of the n-cube is described in [3], and of course also in the previous section, if we just set the quadratic part Q to be the zero-matrix.

Here we show that perturbing the objective function vector c to

$$c' = c + A^T p, \quad p \in \mathbb{R}^m \tag{2.2}$$

can be very helpful to speed up the calculation of the edge orientations of a LP-induced USO, which realizes an idea proposed in [3].

The motivation of altering the LP in this way lies in the easy connection between the modified LP and the unchanged one: For any feasible point x, the new objective value is ${c'}^T x = c^T x + p^T Ax = c^T x + p^T b$, so it's only shifted by a fixed constant, independent of x. Thus any optimal solution to the new problem will automatically be an optimal solution to the old problem and vice versa.

We show that with p chosen randomly according to a suitable distribution over $\mathbb{R}^m$, ties that may occur due to the power series expansion are broken very early. We will show that already the third coefficient of the power series expansion, $t(J)$, can be made non-zero in every coordinate, with probability 1.

As we will use it frequently in this section, we recall the Karush–Kuhn–Tucker conditions (Theorem 1.2) from the previous section for a convex function f in the simple case without any non-negativity constraints:

Theorem 2.1 (KKT Conditions)**.** *$x^* \in \mathbb{R}^n$ is an optimal solution to the problem*

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & Ax = b \end{array}$$

if and only if
$Ax^* = b$ *and there exists* $\lambda \in \mathbb{R}^m$ *such that* $\nabla f(x^*)^T = A^T\lambda$.

Now what is the effect of the perturbation on $x^*(J, J)$ and its power series? We denote the new coefficients arising from the new objective function vector by $c'(J)$, $x'(J)$, $y'(J)$ and $t'(J)$. You might want to recheck the definitions of these coefficients in [3], or directly in the previous section (set $Q = 0$). For completeness we repeat the definition here again:

Definition 2.1. *For* $J \subseteq [n]$, *set*

$$\begin{array}{rcllll} \bar{b}(J) & := & \text{argmin} & (b-y)^T(b-y) & \bar{c}(J) & := & \text{argmin} & (c-x)^T(c-x) \\ & & \text{s.t.} & Ax = y & & & \text{s.t.} & A_J^T y = x_J \\ & & & x \in \mathbb{R}^J & & & & \end{array}$$

$$\begin{array}{rcllll} b(J) & := & \text{argmin} & (b-y)^T(b-y) & c(J) & := & \text{argmin} & (c-x)^T(c-x) \\ & & \text{s.t.} & A_J^T y = \mathbf{0} & & & \text{s.t.} & Ax = \mathbf{0} \\ & & & & & & & x \in \mathbb{R}^J \end{array}$$

$$\begin{array}{rcllll} x(J) & := & \text{argmin} & x^T x & y(J) & := & \text{argmin} & y^T y \\ & & \text{s.t.} & Ax = \bar{b}(J) & & & \text{s.t.} & A_J^T y = \bar{c}(J)_J \\ & & & x \in \mathbb{R}^J & & & & \end{array}$$

$$\begin{array}{rcll} t(J) & := & \text{argmin} & x^T x \\ & & \text{s.t.} & Ax = y(J) \\ & & & x \in \mathbb{R}^J \end{array}$$

If not indicated otherwise, all x *range over* $\mathbb{R}^n$ *and all* y *over* $\mathbb{R}^m$.

2.1 The effect of the perturbation on the coefficients of the power series

Lemma 2.1. *For any* $J \subseteq [n]$ *it holds that*

$$\begin{array}{rrcl} (i) & c'(J) & = & c(J) \\ (ii) & \bar{c}\,'(J) & = & \bar{c}(J) + A^T p \\ (iii) & x'(J) & = & x(J) \end{array}$$

Proof. (i) The KKT Theorem applied to the defining program for $c'(J)$ gives

$$\begin{aligned} & 2\,(c'(J)_J - c'_J) &=& A_J^T\lambda \qquad\qquad \text{for some } \lambda \\ \Leftrightarrow \quad & 2\,(c'(J)_J - c_J) &=& A_J^T(\lambda + 2p). \end{aligned}$$

But this means that $c'(J)$ satisfies the KKT conditions for the original defining program for $c(J)$, and since the solutions are unique we must have that $c'(J) = c(J)$.
(ii) To prove the second equation we use $c' = c'(J) + \bar{c}\,'(J)$ by our preceding technical Lemma, and plug in the first result.
(iii) The third equation is clear since the definition of $x'(J)$ does not depend on c. □

In the following we describe the effect of the perturbation to $y(J)$ and $t(J)$. For the proofs of the two results we need the following small facts from linear algebra:

Lemma 2.2. *Let M be any matrix. Then*

(i) $Ker(M^TM) = Ker(M)$

(ii) $Im(M^TM) = Im(M^T)$

(iii) $Ker(M^T) \cap Im(M) = \{0\}$.

Proof. (i) Obviously $Mx = \mathbf{0} \;\Rightarrow\; M^TMx = \mathbf{0}$. On the other hand $M^TMx = \mathbf{0}$ implies $0 = x^TM^TMx = \|Mx\|^2$, so $Mx = \mathbf{0}$.

(ii) $x = M^TM\lambda$ implies $x = M^T\lambda'$. For the other direction, suppose $x = M^T\lambda$ and consider the unique value $argmin\ z^Tz \;\; s.t.\ M^Tz = M^T\lambda$. The KKT Theorem 2.1 for optimality of z implies $z = M\mu$ for some μ, thus $x = M^T\lambda = M^TM\mu \;\in Im(M^TM)$.

(iii) Let $x \in Ker(M^T) \cap Im(M)$. Then $x = M\lambda$ for some λ, and $M^Tx = M^TM\lambda = \mathbf{0}$. Multiplying this by λ^T yields $0 = \lambda^TM^TM\lambda = \|M\lambda\|^2 = \|x\|^2$, so $x = 0$. □

Lemma 2.3. *For any $J \subseteq [n]$,*

$$y'(J) = y(J) + \bar{p}(J)$$

with $\bar{p}(J)$ being the projection of p onto the image of A_J, i.e.

$$\begin{aligned} \bar{p}(J) \;\; = \;\; & \text{argmin}\;\; (p-y)^T(p-y) \\ & Ax = y\,,\;\; x \in \mathbb{R}^J \end{aligned} \tag{2.3}$$

Proof. We see that the defining program for $\bar{p}(J)$ is obviously feasible, and $\bar{p}(J)$ is thus well-defined. Next we show that $A_J^T p = A_J^T \bar{p}(J)$. The KKT conditions for the defining program for $\bar{p}(J)$, restricted to (x_J, y), state: $\binom{0}{-2(p-\bar{p}(J))} = \binom{A_J^T}{-I}\lambda$. This implies $\frac{\lambda}{2} = p - \bar{p}(J)$, $A_J^T(p - \bar{p}(J)) = 0$ and thus $A_J^T p = A_J^T \bar{p}(J)$.
(Note that this fact is not really a surprise since the definition of $\bar{p}(J)$ is structurally identical to the one of $\bar{b}(J)$ which we used earlier, as in the proof of Lemma 1.5.)

We now make use of $\bar{c}\,'(J) = \bar{c}(J) + A^T p$ from the previous Lemma 2.1. From the definition of $y'(J)$, $y(J)$ we have

$$\begin{aligned} A_J^T\, y(J) &= \bar{c}(J)_J \\ A_J^T\, y'(J) &= \bar{c}(J)_J + A_J^T p\,. \end{aligned}$$

Subtracting these two equations, and taking into account that $A_J^T p = A_J^T \bar{p}(J)$, gives

$$y'(J) - y(J) - \bar{p}(J) \;\in\; Ker(A_J^T)\,. \tag{2.4}$$

Now by definition $\bar{p}(J) \in Im(A_J)$, and by the KKT conditions for the defining programs for $y'(J)$, $y(J)$ we also know that $y'(J), y(J) \in Im(A_J)$, so

$$y'(J) - y(J) - \bar{p}(J) \;\in\; Im(A_J)\,. \tag{2.5}$$

From $Ker(A_J^T) \cap Im(A_J) = \{0\}$ we get $y'(J) = y(J) + \bar{p}(J)$. □

The following lemma characterises the effect of the perturbation on t(J):

Lemma 2.4. *For any $J \subseteq [n]$,*

$$t'(J) = t(J) + \bar{\bar{p}}(J)$$

with

$$\begin{aligned} \bar{\bar{p}}(J) \;=\; & \text{argmin } x^T x \\ & Ax = \bar{p}(J)\,,\; x \in \mathbb{R}^J \end{aligned} \tag{2.6}$$

or in other words, $\bar{\bar{p}}(J)$ being the shortest of all optimal vectors x appearing in the defining program for $\bar{p}(J)$.

Proof. Feasibility of the defining program for $\bar{\bar{p}}(J)$ follows from feasibility of $\bar{p}(J)$, i.e. because $\bar{p}(J) = Ax$ for some $x \in \mathbb{R}^J$. So $\bar{\bar{p}}(J)$ is well-defined.

Now by plugging the previous Lemma into the definitions of $t'(J)$, $t(J)$ we get

$$\begin{aligned} A_J\, t(J)_J &= y(J) \\ A_J\, t'(J)_J &= y(J) + \bar{p}(J) \;=\; y(J) + A_J\, \bar{\bar{p}}(J)_J\,, \end{aligned}$$

the last equality holding by definition of $\bar{\bar{p}}(J)$. Subtracting these two equations gives

$$t'(J)_J - t(J)_J - \bar{\bar{p}}(J)_J \;\in\; Ker(A_J)\,. \tag{2.7}$$

Now the KKT conditions for the defining programs of $t(J)$, $t'(J)$ and $\bar{\bar{p}}(J)$ are all of the same form; they state $t(J)_J$, $t'(J)_J$ and $\bar{\bar{p}}(J)_J \in Im(A_J^T)$, so

$$t'(J)_J - t(J)_J - \bar{\bar{p}}(J)_J \;\in\; Im(A_J^T)\,. \tag{2.8}$$

So from $Ker(A_J) \cap Im(A_J^T) = \{0\}$ we get $t'(J) = t(J) + \bar{\bar{p}}(J)$. □

2.2 Choosing the right perturbation

Now as we know the effect of the perturbation on $t(J)$, the question is whether this influence is powerful enough to make $t(J)$ non-zero everywhere, and thus being able to get the edge orientations of the USO with only the first 3 terms of the power series expansion. To answer this question, we define the following sets:

Definition 2.2. *For $J \subseteq [n]$, $i \in J$*

$$P_{i,J} := \{p \in \mathbb{R}^n \;\mid\; \bar{\bar{p}}(J)_i + t(J)_i = 0\}\,. \tag{2.9}$$

In order to achieve our goal, we would like to find vectors p avoiding these sets for all $J \subseteq [n]$, $i \in J$. In the following I will show that the sets $P_{i,J}$ are very small compared to $\mathbb{R}^m$, and thus that its possible to avoid all of them with very high probability.

My apologies that we will abuse notation a bit by writing $\bar{q}(J)$, $\bar{\bar{q}}(J)$ and $\bar{s}(J)$, $\bar{\bar{s}}(J)$ for the values of our custom designed problems (see (2.3) and (2.6)) if the variable is q or s instead of p.

Lemma 2.5. *For any $J \subseteq [n]$, $\bar{\bar{p}}(J)$ is a linear function in p, for $p \in \mathbb{R}^m$.*

Proof. Let $\mu, \nu \in \mathbb{R}$. Set $s := \mu\, p + \nu\, q$. We have to show that $\bar{\bar{s}}(J) = \mu\, \bar{\bar{p}}(J) + \nu\, \bar{\bar{q}}(J)$.

To do so, we first prove $\bar{s}(J) = \mu\, \bar{p}(J) + \nu\, \bar{q}(J)$, which is intuitively clear since the function has the meaning of a projection onto a smaller space. For a formal proof, we see that by definition $\mu\, \bar{p}(J) + \nu\, \bar{q}(J) \in Im(A_J)$, to be precise

$$\mu\, \bar{p}(J) + \nu\, \bar{q}(J) = \mu\, A\bar{\bar{p}}(J) + \nu\, A\bar{\bar{q}}(J) = A(\mu\, \bar{\bar{p}}(J) + \nu\, \bar{\bar{q}}(J)). \tag{2.10}$$

This tells us that $\mu\, \bar{p}(J) + \nu\, \bar{q}(J)$ is feasible for the defining program of $\bar{s}(J)$. Next, by the KKT conditions for the defining programs for $\bar{p}(J)$ and $\bar{q}(J)$ (see beginning of the proof of Lemma 2.3), we have $A_J^T p = A_J^T \bar{p}(J)$ and $A_J^T q = A_J^T \bar{q}(J)$, so

$$A_J^T\, s = A_J^T(\mu\, p + \nu\, q) = A_J^T(\mu\, \bar{p}(J) + \nu\, \bar{q}(J)), \tag{2.11}$$

which implies the KKT conditions for $\bar{s}(J)$ stating that $\mu\, \bar{p}(J) + \nu\, \bar{q}(J)$ is indeed optimal for the defining program of $\bar{s}(J)$.

Now we are prepared to prove $\bar{\bar{s}}(J) = \mu\, \bar{\bar{p}}(J) + \nu\, \bar{\bar{q}}(J)$. From what we have just shown, we see that $\mu\, \bar{\bar{p}}(J) + \nu\, \bar{\bar{q}}(J)$ is feasible for the defining program for $\bar{\bar{s}}(J)$, by equation (2.10). To see that it is optimal, we again have to verify the KKT conditions, i.e. we have to show that $(\mu\, \bar{\bar{p}}(J) + \nu\, \bar{\bar{q}}(J))_J = A_J^T \lambda$ for some vector λ. But this is obvious by linearity since $\bar{\bar{p}}(J)$ and $\bar{\bar{q}}(J)$ both live in $Im(A_J^T)$ because of their KKT conditions. □

We are now ready to prove the main theorem of this section:

Theorem 2.2. *Let the perturbation vector p be chosen uniformly at random from $[0,1]^m$, and suppose that our matrix A does not contain any zero-columns.*
Then, with probability 1, the vector $t'(J)$ is non-zero in all coordinates, for all $J \subseteq [n], i \in J$.

Proof. By the definition of our spaces $P_{i,J}$ we know that $t'(J)$ does have the desired properties whenever

$$p \notin \bigcup_{J \subseteq [n], i \in J} P_{i,J}. \tag{2.12}$$

To realise the proof of this theorem, it will be shown that $\bigcup_{J\subseteq[n],i\in J} P_{i,J}$ is a (Lebesgue) null set[6] in $\mathbb{R}^m$, and therefore also a null set in $[0,1]^m$. The proof will then be finished since the probability of hitting this unwanted set of measure zero is of course zero.

We fix $J \subseteq [n], i \in J$ and want to show that $P_{i,J}$ is a null set. To do so we use a fact from measure theory that every non-degenerate hyperplane has zero measure in $\mathbb{R}^m$.

From the previous Lemma 2.5 we know that $P_{i,J}$ is the solution set to the linear equation $\bar{\bar{p}}(J)_i = -t(J)_i$, and thus $P_{i,J}$ is a —probably degenerate— hyperplane in $\mathbb{R}^m$. It remains to prove that $P_{i,J}$ is not the degenerate hyperplane $\mathbb{R}^m = \{x \mid 0x_1 + \cdots + 0x_m = 0\}$. To do this, we distinguish two cases:

Case $t(J)_i \neq 0$:
Then $P_{i,J}$ is not the degenerate hyperplane $\mathbb{R}^m$ by definition. (Or alternatively its also easy to see that $\mathbf{0} \notin P_{i,J}$).

Case $t(J)_i = 0$:
Suppose $P_{i,J}$ is the entire space $\mathbb{R}^m$. Then $\bar{\bar{p}}(J)_i = 0$ for all $p \in \mathbb{R}^m$. In particular

$$\bar{\bar{p}}(J)_i = 0 \quad \forall\ p = A_J\lambda,\ \lambda \in \mathbb{R}^{|J|} \tag{2.13}$$

In this case, we have $\bar{p}(J) = p = A_J\lambda$ by definition of $\bar{p}(J)$, and so by inserting the defining program for $\bar{\bar{p}}(J)$ into the equation (2.13) we get:

$$\begin{pmatrix} \text{argmin } x^T x \\ A_J x = A_J \lambda \end{pmatrix}_i = 0 \quad \forall\ \lambda. \tag{2.14}$$

But since A has no zero-columns, this is a contradiction to the following helping Lemma 2.6.

Finally taking the union over a finite number (at most $n\,2^n$) of null sets in $\mathbb{R}^m$, we again get a null set. This finishes the proof. □

[6] A *Lebesgue null set* N is a subset of $\mathbb{R}^m$ with the property that for all $\epsilon > 0$, N possesses a covering by a sequence of m-cubes with total volume $\leqslant \epsilon$.

Lemma 2.6. *Let M be a matrix which does not contain any zero-columns. For any vector λ we define*

$$\begin{aligned} q(\lambda) &= \text{argmin } x^T x \\ &\quad Mx = M\lambda\,, \end{aligned} \tag{2.15}$$

then there is a λ such that $q(\lambda)_i \neq 0$.

Proof. Set $\lambda := M^T M_i$. By the KKT conditions for $q(\lambda)$ we have $q(\lambda) = M^T\mu$ for some vector μ. Now suppose $q(\lambda)_i = M_i^T\mu = 0$. We get the following:

$$\begin{aligned} & Mq(\lambda) &&= M\lambda \\ \Leftrightarrow \quad & MM^T\mu &&= MM^T M_i \\ \Leftrightarrow \quad & MM^T(\mu - M_i) &&= 0 \\ \Leftrightarrow \quad & M^T(\mu - M_i) &&= 0. \end{aligned} \tag{2.16}$$

The last equivalence holding because of our linear algebra Lemma 2.2 (i). Looking at the i-th coordinate we get $0 = M_i^T(\mu - M_i) = -M_i^T M_i$, and thus $M_i = \mathbf{0}$ which contradicts our assumption that M had no zero-colums. □

Note: The given proof for our main theorem only works with the slight restriction that A has *no zero-columns*. We note that this is not a severe restriction since any variable x_i of the LP corresponding to a zero-column A_i can easily be "removed" from the LP: If $c_i = 0$, the variable x_i has no influence at all in the LP, and can therefore be ignored. In the other case, if $c_i \neq 0$, the LP is obviously unbounded as $x_i \to \pm\infty$.

It is also clear that perturbation does *not* work if A contains a zero-column, since if $A_i = \mathbf{0}$, then the all $(g_k(J))_i$, $k \geqslant 0$ will *always* be zero. This is because the variable x_i will then have no influence at all in any of the defining programs for $g_k(J)$, $k \geqslant 0$.

How does this special situation translate to the induced USO? In the USO corresponding to this LP, all edges in direction i will by definition always be oriented "downwards". So the observation that variable x_i can be ignored precisely translates to the USO in the sense that here, dimension i can be ignored because already from the beginning we know that the global sink of the USO is located in the face located "below" dimension i, i.e. the face corresponding to the interval $[\,\emptyset,\, [n] \setminus \{i\}\,]$.

2.3 Conclusion

We proved that using a random perturbation p, the orientation of any edge $J \setminus \{j\} \to J$ of the USO can be decided by only looking at the first 3 coefficients of the power series expansion of $x^*(J, J)$, using a suitable perturbation. This is a linear speed-up compared to the theoretical need of $2|J| + 2$ coefficients without perturbation.

3 Various Results

3.1 'Regularisation' of linear programming

In the previous section we applied a perturbation to the linear program to change its combinatorial properties so that the corresponding USO could be decided easier. Here we will again show that a perturbation of the linear program can be useful, but with just in the opposite direction. We will find a perturbation that retains the combinatorial properties of the linear program, but makes the program itself hopefully easier to solve.

We prove that for every linear program, there exists a *regular* linear program with the same combinatorial properties, in the sense that it induces the same USO.

Let A, b and c be the coefficients of the linear program

$$\begin{aligned} (LP) \quad \max \quad & c^T x \\ \text{s.t.} \quad & Ax = b \,, \\ & x \geqslant 0 \,, \end{aligned} \tag{3.1}$$

and let $\varepsilon > 0$. We define

$$\begin{aligned} A' &:= chol(A^T A + \varepsilon^2 I) \\ b' &:= A'^{\,T^{-1}}(A^T b + \varepsilon c) \\ c' &:= \mathbf{0} \,, \end{aligned} \tag{3.2}$$

with $chol(M)$ denoting the unique matrix obtained by the *Cholesky decomposition* of a symmetric, positive definite matrix M. Note that since the matrix $A^T A + \varepsilon^2 I$ is positive definite, it's in particular regular, which also implies that A' is regular since $A'^{\,T} A' = A^T A + \varepsilon^2 I$. So $A'^{\,T}$ is invertible and unique, and thus all three objects are well-defined.

Lemma 3.1. *Let LP' denote the linear program defined by A', b' and c'. Then*

(i) For ε small enough, the regular *linear program LP' induces the same USO as the original LP.*

(ii) The solution to LP'—if feasible—can be calculated without doing the Cholesky decomposition. It is given by

$$x = (A^T A + \varepsilon^2 I)^{-1}(A^T b + \varepsilon c) \tag{3.3}$$

Proof. (i) The USO corresponding to the LP was defined by the strictly convex quadratic function $f_\varepsilon(x) = x^T(A^TA+\varepsilon^2 I)x - 2b^TAx - 2\varepsilon c^Tx$ (remember Section 1.5). We now consider this function for our new LP, and calculate

$$\begin{aligned} f'_0(x) &= x^T({A'}^T A')x - 2{b'}^T A'x \\ &= x^T(A^TA+\varepsilon^2 I)x - 2({A'}^T b')^T x \\ &= x^T(A^TA+\varepsilon^2 I)x - 2(A^Tb+\varepsilon c)^T x \\ &= f_\varepsilon(x)\ . \end{aligned} \tag{3.4}$$

This equation holds for all ε and all x. Now let ε be small enough so that the USO induced by f_ε is indeed the USO from the LP (see Lemma 1.2). Now since f'_ε is also stricly convex for the case $\varepsilon = 0$, it's clear that the USO induced by LP' is the one defined by f'_0 which equals f_ε, so the two USO are indeed the same.

(ii) As we know that A' is regular, we may write the solution to the equation $A'x = b'$ as follows:

$$\begin{aligned} x &= A'^{-1}b' = A'^{-1}{A'}^{T^{-1}}(A^Tb+\varepsilon c) \\ &= ({A'}^TA')^{-1}(A^Tb+\varepsilon c) = (A^TA+\varepsilon^2 I)^{-1}(A^Tb+\varepsilon c)\ . \end{aligned} \tag{3.5}$$

□

The above result, together with the fact that a regular LP is trivial to solve, is a double edged sword. On one hand—if the modified LP' is feasible—we very easily get a solution that is closely related to the optimal solution of the original LP, in the sense that it corresponds to the global sink of the same USO. This sink would then also directly give us the optimal solution to the original LP. An approach to finding the sink would be to start with the coordinates where the optimal solution $x'(S)$ of the regularised LP (which is the only feasible point) is strictly positive (as it is clear that they have to be part of the sink). It's however not known yet how fast the sink can be constructed from such a unique optimal solution $x'(S)$.

On the other hand—if the modified program is not feasible—we can't say much about the original LP. The problem is that for infeasible LP, the meaning of the sink of the corresponding USO is not yet understood very well. Also, this case can occur often, even if the original LP is feasible there is no guarantee that there exists an $\varepsilon > 0$ such that the modified LP' is still feasible.

We note that there is still some work to do until this concept could probably be exploited algorithmically. But independent of that, this at least sheds some new light on the original method described in Section 1, giving an explanation on the meaning of the perturbed function f_ε inducing the USO. This also leads us to the following section:

3.2 On the relation between LP and symmetric PLCP

Linear complementarity problems (LCP) have a rich combinatorial structure and a variety of results and algorithms for LCP are known [1]. In the following, for $q \in \mathbb{R}^n, M \in \mathbb{R}^{n\times n}$, we will denote by LCP$(q, M)$ the problem of finding a vector $x \in \mathbb{R}^n$ such that

$$\begin{aligned} x &\geqslant \mathbf{0} \\ q + Mx &\geqslant \mathbf{0} \\ x^T(q + Mx) &= 0 \ . \end{aligned}$$

In this section we will examine the connection between our USO approach to linear programming, and linear complementarity problems. When defining our cube orientation, we in fact solved the unconstrained strictly convex program

$$\begin{array}{lll} \mathrm{SCP}_\varepsilon(I, J) & \min & f_\varepsilon(x) \\ & s.t. & x \in \mathbb{R}^J, \\ & & x_{J\setminus I} \geq 0, \end{array} \tag{3.6}$$

with

$$f_\varepsilon(x) \ = \ x^T(A^TA + \varepsilon Q + \varepsilon^2 I)x - 2b^TAx - 2\varepsilon c^Tx \ .$$

Now we observe that for $I = \emptyset, J = [n]$ (this means for the global minimum of the function), our optimality conditions (1.4) through (1.8) exactly define a *linear complementarity problem* (LCP) with matrix $M = A^TA + \varepsilon Q + \varepsilon^2 I$, and $q = -A^Tb - \varepsilon c$.

In our case M is symmetric, and positive definite, thus it is a symmetric P-matrix[7]. Stickney and Watson [5] used unique sink orientations to solve any P-matrix linear complementarity problems (PLCP). And in fact for the above mentioned LCP, their orientation exactly coincides with the orientation we have defined in Theorem 1.3.

We thus deduce that for any convex QP we have a symmetric PLCP that induces the same USO, if we just choose ε small enough.

More interestingly, this relation also works in the other direction: Let M, q be the coefficients of any symmetric PLCP. We use the fact that a

[7] M is a P-matrix $\Leftrightarrow$ all principal minors of M are strictly positive.

symmetric matrix is positive definite if and only if it is also a P-matrix [1]. This means we can compute the Cholesky-decomposition $chol(M)$. We define a linear program (3.2) as follows:

$$\begin{aligned} A &:= chol(M) \\ b &:= A^{T^{-1}} q \\ c &:= \mathbf{0}\ , \end{aligned} \tag{3.7}$$

(remember that M is a P-matrix and thus invertible).

Observe that again, this LP induces exactly the same USO as the original symmetric PLCP. As also described in the preceding section, since $M = A^T A$ is strictly positive definite, we may directly set $\varepsilon := 0$ to obtain the orientation. So for any symmetric PLCP we have a LP inducing the same USO. It does however not automatically follow that this LP is also feasible.

This thinking is also motivated by the fact that despite general PLCP is not known to be polynomial time solvable, symmetric PLCP as well as LP indeed are. In Section 4 we will state some experimental results about possible differences between USO coming from symmetric PLCP compared to such from general PLCP, and ask if such differences could be exploited algorithmically.

Another interesting motivation is that for positive semi-definite LCP, which is solvable in polynomial time, a regularisation method with many similarities to the one stated in Section 3.1 is known. From [1], chapter 5.6:

Theorem 3.1. *Let M be a P_0[8]-matrix. Let $\{\varepsilon_\nu\}$ be a decreasing sequence of positive scalars with $\varepsilon_\nu \to 0$. For each ν, let x^ν be the unique solution of the LCP $(q, M + \varepsilon_\nu I)$.*

> *If M is positive semi-definite and the $LCP(q, M)$ is solvable, then the sequence $\{x_\nu\}$ converges to the least l_2-norm solution of LCP(q,M).*

It would be interesting to further investigate common properties of the corresponding algorithms for LP and symmetric PLCP, especially in the USO context. The interesting additional fact that the theorem shows convergence towards the *shortest* solution allows us to gently pass over to the next section:

[8] M is a P_0-matrix $\Leftrightarrow$ all principal minors of M are nonnegative.

3.3 The USO solution is indeed the shortest one

In this short section we prove that the optimal solution for a linear program obtained by the USO approach is indeed the shortest optimal solution. (Remember the intuitive argument at the beginning of Section 1.5.) This allows us to precisely state what our "canonical" solution is that we obtain by our USO approach for any LP. ('Shortest' referring to the l_2-norm in $\mathbb{R}^n$.)

Theorem 3.2. *If the LP is feasible and bounded, then*

i) $x(S)$ is the shortest optimal solution to the LP, and

ii) $y(S)$ is the shortest optimal solution to the dual LP.

Proof. Remember that from Theorem 1.6, we know that $x(S)$ and $y(S)$ is a pair of primal and dual optimal solutions to the LP. So $x(S)$ is the global shortest optimal solution to the LP if and only if it is the optimal solution to the following program

$$\begin{array}{ll} \min & x^T x \\ \text{s.t.} & Ax = b \\ & c^T x = c^T x(S) \\ & x \geqslant 0 \, . \end{array} \tag{3.8}$$

We immediately see that $x(S)$ is feasible for this program, but to be optimal it would also need to satisfy the following KKT conditions:

$$2x(S)_S \;\geqslant\; (A_S^T \,,\; c_S)\lambda \tag{3.9}$$

$$\mathbf{0} = 2x(S)_{[n]\setminus S} \;\geqslant\; (A_{[n]\setminus S}{}^T,\; c_{[n]\setminus S})\lambda \tag{3.10}$$

for some $\lambda \in \mathbb{R}^{m+1}$, with equality holding in (3.9) for all $i \in S$ with $x(S)_i > 0$.

We start our search for a candidate for λ by remembering the following important property of the sink of our orientation, which we already mentioned at the beginning of Section 1.6, equation (1.36): If S is the sink, we have that for any ε small enough,

$$\nabla f_\varepsilon(x_\varepsilon^*(S,S)) \geqslant \mathbf{0} \, . \tag{3.11}$$

Looking at the power series expansion of ∇f_ε, and computing one more term than we already did in (1.34), we have

$$\begin{aligned} \nabla f_\varepsilon(x^*_\varepsilon(S,S))^T/2 &= A^T(\bar{b}(S)-b) \\ &+ \varepsilon(A^T y(S) - \bar{c}(S)) \\ &+ \varepsilon^2(A^T A g_2(S) - x(S)) \\ &+ O(\varepsilon^3)\,. \end{aligned} \tag{3.12}$$

Dividing the equation by ε^2, and taking into account that $\bar{b}(S) = b$ and $\bar{c}(S) = c$ as the LP is feasible and bounded, we get

$$\begin{aligned} \frac{\nabla f_\varepsilon(x^*_\varepsilon(S,S))^T}{2\varepsilon^2} &= \frac{1}{\varepsilon}(A^T y(S) - c) \\ &+ A^T A g_2(S) - x(S) \\ &+ O(\varepsilon)\,. \end{aligned} \tag{3.13}$$

We plug this into (3.11) and only look at the coordinates outside S (remember $x(S) \in \mathbb{R}^S$), to finally obtain

$$\begin{aligned} \mathbf{0} &\geqslant -\frac{1}{\varepsilon}(A^T_{[n]\setminus S} y(S) - c_{[n]\setminus S}) - A^T_{[n]\setminus S} A g_2(S) \\ &= ({A_{[n]\setminus S}}^T,\ c_{[n]\setminus S}) \left(-\tfrac{1}{\varepsilon}\tbinom{y(S)}{-1} - \tbinom{Ag_2(S)}{0}\right) \end{aligned} \tag{3.14}$$

But that is exactly the KKT condition (3.10) we want to prove for $x(S)$. So lets try $\dfrac{\lambda}{2} := -\dfrac{1}{\varepsilon}\begin{pmatrix} y(S) \\ -1 \end{pmatrix} - \begin{pmatrix} Ag_2(S) \\ 0 \end{pmatrix}$, and check if this promising candidate could probably also satisfy the remaining condition (3.9). We calculate

$$\begin{aligned} (A^T_S,\ c_S)\frac{\lambda}{2} &= (A^T_S,\ c_S) \left(-\tfrac{1}{\varepsilon}\tbinom{y(S)}{-1} - \tbinom{Ag_2(S)}{0}\right) \\ &= -A^T_S A g_2(S) \ = \ x(S)_S \end{aligned} \tag{3.15}$$

by using $A^T_S y(S) = c_S$ from the definition of $y(S)$, and the last equality holding by definition of $g_2(S)$. This indeed proves (3.9).

This finishes the first part of the proof. $x(S)$ is in fact the shortest optimal solution to the LP, because we have found a KKT multiplier λ satisfying all optimality conditions for $x(S)$ to be optimal to (3.8).

For $y(S)$, a very similar argument works, which is even a bit easier. We give the proof in a slightly shorter form than the above reasoning for $x(S)$. To prove that $y(S)$ is optimal to

$$\begin{array}{ll} \min & y^T y \\ \text{s.t.} & A^T y \geqslant c \\ & b^T y = b^T y(S) \ , \end{array} \tag{3.16}$$

we have to show the KKT conditions. If we write the above inequality constraint as $(A^T, -I)\binom{y}{z} = c,\ z \geqslant \mathbf{0}$, these are

$$2y(S) = (A\,,\ b)\lambda \tag{3.17}$$

$$\mathbf{0} \geqslant (-I\,,\ \mathbf{0})\lambda \tag{3.18}$$

for some $\lambda \in \mathbb{R}^{n+1}$. This time we use that at the sink, $x^*_\varepsilon(S, S) \geqslant \mathbf{0}$, which implies $\frac{x(S)}{\varepsilon} + t(S) \geqslant \mathbf{0}$. We set

$$\frac{\lambda}{2} := \frac{1}{\varepsilon}\begin{pmatrix} x(S) \\ -1 \end{pmatrix} + \begin{pmatrix} t(S) \\ 0 \end{pmatrix}, \tag{3.19}$$

and verify that indeed, $\lambda_{[n]} \geqslant \mathbf{0}$, and $y(S) = (A\,,\ b)\lambda$. The last holding because $Ax(S) = b$ and $At(S) = y(S)$ by definition of $x(S)$ and $t(S)$. So $y(S)$ is the shortest optimal solution to the dual LP. □

Besides the existing intuitive motivation at the beginning of Section 1.5, we also ran an experimental test to keep up the motivation during the burdensome search for the right λ ;-): Two million random LPs for the 3-dimensional case were calculated and examined by the implementation (see Section 4), and for all of these, the resulting solution from the USO was indeed the shortest one - which now, in view of Theorem 3.2, does of course not sound surprising anymore.

Furthermore, the theorem now tells us precisely what kind of "canonical" solution we obtain by our USO approach, for any LP (not necessarily feasible and bounded):

Corollary 3.2. *For any LP,*

i) $x(S)$ is the shortest optimal solution to the LP

$$\begin{array}{ll} \max & \bar{c}(S)^T x \\ \text{s.t.} & Ax = \bar{b}(S) \\ & x \geqslant 0, \end{array} \tag{3.20}$$

ii) $y(S)$ is the shortest optimal solution to the dual LP

$$\begin{array}{ll} \min & y^T \bar{b}(S) \\ \text{s.t.} & A^T y \geqslant \bar{c}(S). \end{array} \tag{3.21}$$

So if we also remember the meaning of $\bar{b}(S)$ and $\bar{c}(S)$ from Lemma 1.9, we can finally say that the USO induced solutions $x(S)$ and $y(S)$ are in fact the following: They are the shortest optimal primal and dual solutions to the feasible and bounded LP "closest" to the original one. This nice result now justifies that we called our solution a *canonical* solution.

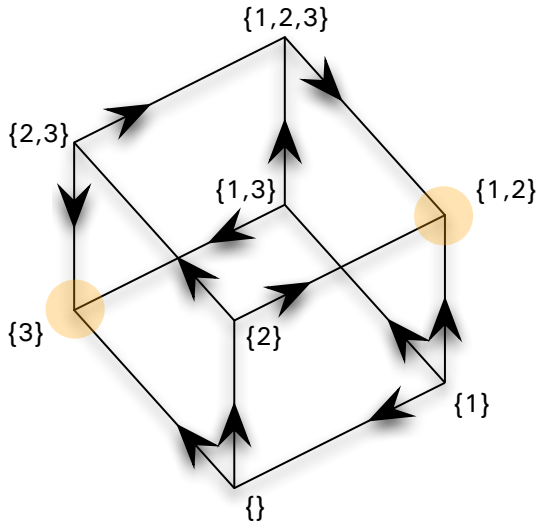
3.4 Cutting off the power series

As an answer to a question proposed in [3], this paragraph will examine the impact of cutting off the power series after the second or the third term. It would be interesting to know if the orientation obtained by just looking at the first two or three coefficients of the power series is also a unique sink orientation. Unfortunately the answer to this question is no. It is thus not possible to speed up the generation of the orientations that easily, which again motivates the need of a correct perturbation method as described in Section 2.

To see that cutting off does not work, we consider the LP

$$A = \begin{pmatrix} 3 & -22 & 0 \\ 0 & 23 & 14 \\ 0 & 0 & 0 \end{pmatrix}, \; b = \begin{pmatrix} 0 \\ 20 \\ 23 \end{pmatrix}, \; c = \begin{pmatrix} 7 \\ -22 \\ 10 \end{pmatrix}. \tag{3.22}$$

If we look only at the first two terms $c(J)$ and $x(J)$ of the power series, we obtain the following orientation of the 3-cube:



This orientation is not a USO, since obviously it has two global sinks ($S' = \{1,2\}$ and $S'' = \{3\}$).

Seeing the previous example you might think that as the LP (3.22) is not feasible, the cut-off-method could probably still work in the feasible and bounded case. But this is not true either: The feasible and bounded LP

$$A = \begin{pmatrix} 18 & 4 & -21 \\ -7 & -21 & 31 \end{pmatrix},\ b = \begin{pmatrix} 21 \\ -31 \end{pmatrix},\ c = \begin{pmatrix} -18 \\ -4 \\ 21 \end{pmatrix} \tag{3.23}$$

does—when looking only at the first two terms—also induce an orientation which does not satisfy the unique sink property. This LP induces the outmaps [3 2 1 0 4 7 7 4][9] on the 3-cube, which again is not a USO.

Even further, our implementation shows that even looking at the first three coefficients in the power series, $c(J)$, $x(J)$ and $t(J)$, does not necessarily result in a USO. For example the feasible and bounded LP

$$A = \begin{pmatrix} 22 & 65 & 57 \\ -27 & 28 & 12 \end{pmatrix},\ b = \begin{pmatrix} 171 \\ 36 \end{pmatrix},\ c = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \tag{3.24}$$

would then induce the outmaps [7 6 5 4 0 3 3 0], which is not a USO either, because it hat two sinks.

[9]We did not define this special notation for USO by their outmaps yet. If you are curious please refer to the beginning of the following Section 4.

4 Implementation

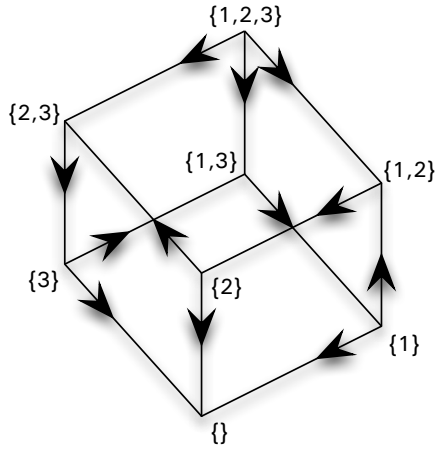
The described new algorithm for linear programming was implemented in Mathematica to improve the understanding of the method in practice. The main calculation method, to evaluate the edge-orientations at a cube vertex J, can be written quite shortly as follows:

```
A = {{4, -4, -2}, {0, 4, 2}, {-1, -4, 5}};
b = {0, 0, 0};
c = {1, -2, -1};

J = {1, 2, 3};

{m, n} = Dimensions[A]; j = Length[J];
AJ = A[[All, J]]; AJt = Transpose[ AJ];
MM = Table [0, {2 j}, {2 j}];
MM[[Range[j], Range[j]]] = AJt.AJ;
MM[[j + Range[j], Range[j]]] = IdentityMatrix[j];
MM[[j + Range[j], j + Range[j]]] = -AJt.AJ;

decided = Table[-2, {j}];
orientations = Table[-1, {j}];
g  = Take[LinearSolve[MM, PadLeft[c[[J]], 2 j] ], {1, j}] ;

k = -1; pp = -AJt.b; p = -(c[[J]] - g);
While[Min[decided] < -1 && k < 2 m,
  orientations = Table[If[decided[[i]] < -1 && g[[i]] != 0,
     If[g[[i]] > 0, 1, -1], orientations[[i]]], {i, j}];
  decided = Table[If[decided[[i]] < -1 && g[[i]] != 0, k, decided[[i]]], {i, j}];
  g  = Take[LinearSolve[MM, PadRight[-pp, 2 j] ], {1, j}] ;
  k = k + 1; pp = p; p = g;
 ];

Print["final orientations: ", orientations];
Print["obtained with coefficients number: ", decided];
```

Furthermore, we will use a convenient representation format for unique sink orientations, as proposed in [4]: Every USO of the n-cube can be seen as a permutation on the ground set $\{0, \ldots, 2^n - 1\}$.

To see this, we identify every cube vertex $J \subseteq [n]$ with a binary vector $(a_0, \ldots, a_{n-1})$, $a_i = \begin{cases} 1 & i \in J, \\ 0 & \text{otherwise.} \end{cases}$ Also, the *outmap* of a vertex, i.e. the set of the directions where the vertex has an outgoing edge, can of course be identified with a vector $o \in \{0,1\}^n$. That is $o_i = 1 \;\Leftrightarrow\; J \to J \oplus \{i\}$.

Furthermore we identify every binary vector $v \in \{0,1\}^n$ with the unique number $\sum_{i=0}^{n-1} v_i 2^i$. Knowing that the outmaps of a unique sink orientations form a bijection[10], we can thus represent the USO (i.e. the outmaps at all vertices) by a permutation of $\{0, \ldots, 2^n - 1\}$, as in the following example:



This USO of the 3-cube can be encoded by [0 3 6 1 5 4 2 7], with for example the number 6 meaning that at the vertex $\{2\}$ (corresponding to $010 \leftrightarrow 2$), the outmap is $\{2,3\}$ (corresponding to $011 \leftrightarrow 6$). As this data format may seem a bit wired to you upon first reading, we also include unambiguous pictures for most of the USO occurring in this section.

When dealing with orientations of the 3-cube, we will always think of the cube vertices J labeled as in the figure above.

4.1 Number of coefficients needed from the power series

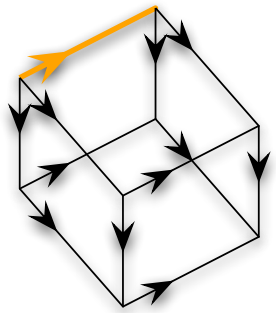
We have already shown in Section 3.4 that cutting of the power series expansion after the second or third term does not necessarily lead to a USO. In this short section we will experimentally examine how many coefficients are needed to decide the orientation of an edge.

The implementation clearly shows that some LPs do in fact give rise to power series where we need more than just the first two coefficients: The LP

[10]This follows from the USO axioms, and is also s sufficient condition for being a USO.

$$A = \begin{pmatrix} 4 & -4 & -2 \\ 0 & 2 & 4 \\ -1 & -4 & 5 \end{pmatrix},\ b = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix},\ c = \begin{pmatrix} 1 \\ -2 \\ -1 \end{pmatrix} \tag{4.1}$$

induces a power series expansion at the vertex $J = \{1, 2, 3\}$ with $c(J)_1 = x(J)_1 = t(J)_1 = g_2(J)_1 = 0$, and $g_3(J)_1 \neq 0$. This means that the edge $\{2, 3\} \to \{1, 2, 3\}$ is decided not earlier than with the *fifth* coefficient $g_3(J)$.



The USO [1 0 3 2 5 4 7 6], induced by the above LP.

This example is in tight connection with Section 3.4: As we proved that cutting off the power series does not work, these LPs must indeed induce power series that have properties as given in this example. In addition, this example shows that perturbing the objective function as described in Section 2 is not only effective from theoretical point of view but can really be useful in practice.

4.2 Which USO are LP-induced?

All USO coming from P-matrix linear complementarity problems (PLCP) according to Stickney and Watson [5] (and thus also all USO coming from our reduction from LP or QP as noted at the end of Section 1.4) have recently been shown to satisfy an interesting combinatorial property, the *Holt-Klee condition* [2]. This means that any k-dimensional face has k vertex-disjoint paths from its unique source to its unique sink.[11]

There are 744 USO of the 3-cube, and 672 of them satisfy the Holt-Klee condition.

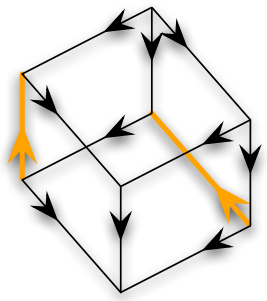
[11] The existence of unique sources follows from the USO axioms.

Using the implementation, ten million random LP instances for the case $n = 3$ were considered to get an idea on which of the USO of the 3-cube are induced by an LP. We found that the vast majority, at least 608 of the 672 Holt-Klee USO of the 3-cube are in fact induced by an LP.

Interestingly, still not all 3-USO satisfying the Holt-Klee condition could be generated. Note that the experimental result of 608 USO obtained was the same even when only 50'000 instead of 10'000'000 random LPs were used, and it was also the same with A, b, c chosen as random *sparse* matrices instead of A, b, c chosen just uniformly at random (both in some fixed integer range).

This suggests that LP induced USO satisfy an even stronger condition than the Holt-Klee conditions, distinguishing them from arbitrary USO. The additional condition must then come from the fact that we're solving a *symmetric* PLCP (which is positive definite), in contrast to a general PLCP (which we know can induce all H.K. 3-USO). It's a very interesting further question what this further condition could be, and if it could possibly be exploited algorithmically, eventually leading to a faster algorithm for LP.

The claim that the condition should be stronger is supported by the experiment that the following Holt-Klee USO

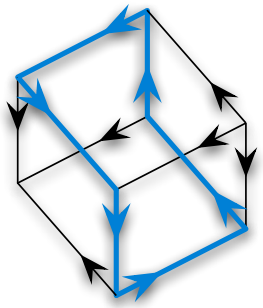


[0 5 2 3 6 1 4 7]

could not by generated by 10 million random LP, and it could also not be generated by *one billion* randomly chosen symmetric PLCP instances. We also note that despite that, the USO [6 5 4 7 0 1 3 2] that is obtained by just rotating the above figure "by one edge", can be generated by a LP, for

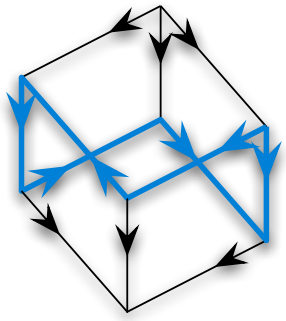
example with $A = \begin{pmatrix} 35 & -8 & -10 \\ 0 & -30 & -17 \\ -40 & -4 & 0 \end{pmatrix}$, $b = \begin{pmatrix} -51 \\ 15 \\ -20 \end{pmatrix}$, $c = \begin{pmatrix} -21 \\ 43 \\ 39 \end{pmatrix}$. This leads us to the following paragraph:

Isomorphic USO The experiment suggests that the isomorphism of USO (rotations and mirroring of the cube) is not "compliant" for LP-induced USO, i.e. we found some USO we think are not induced by LP, despite some isomorphic USO (just a rotation of the same USO) being LP-induced. (For example we could very easily find an LP inducing the cyclic USO



[5 4 2 7 0 3 6 1]

by random search, but we were unable to experimentally find an LP inducing the USO



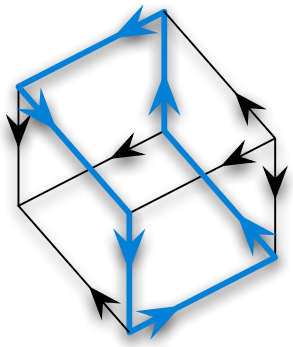
[0 3 6 1 5 4 2 7] ,

which is isomorphic to the one stated previously.)

LP-induced USO may have cycles The implementation clearly showed that some LPs do in fact induce cyclic USO: For example

$$A = \begin{pmatrix} 8 & 1 & 8 \\ -3 & 3 & -4 \\ 15 & -9 & 21 \end{pmatrix}, \; b = \begin{pmatrix} -1 \\ 1 \\ 1 \end{pmatrix}, \; c = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \tag{4.2}$$

results in the cyclic USO



[5 4 2 7 0 3 6 1] .

Note This is not a big surprise as it was already known that the LCP with matrix

$$M = \begin{pmatrix} 5 & -10 & 2 \\ -10 & 41 & -6 \\ 2 & -6 & 1 \end{pmatrix} \tag{4.3}$$

together with right side $q = (1, -7, 1)^T$ results in a cyclic USO. But then by doing a Cholesky decomposition of M we find A and b of an LP that induces the same USO. This is possible since M is a symmetric P-matrix, and thus already positive definite. This means we may just set $\varepsilon = 0$ to get the limiting USO (as we already described more precisely in Section 3.2). The implementation verified that the A, b obtained that way do indeed induce a cyclic USO. But as this matrix A looks a bit complicated, we stated an easier LP above that was found just by random search.

References

[1] R. W. Cottle, J. Pang, and R. E. Stone. *The Linear Complementarity Problem*. Academic Press, 1992.

[2] B. Gärtner, W. D. Morris, and L. Rüst. Unique sink orientations of grids. In *Proc. 11th Conference on Integer Programming and Combinatorial Optimization (IPCO)*, volume 3509 of *Lecture Notes in Computer Science*, pages 210–224, 2005.

[3] B. Gärtner and I. Schurr. Linear Programming and Unique Sink Orientations. In *Proc. 17th Annual Symposium on Discrete Algorithms (SODA)*, pages 749–757, 2006.

[4] I. Schurr. Unique Sink Orientations of Cubes. disertation, ETH Zürich, *http://uso.io5.de*, 2004.

[5] A. Stickney and L. Watson. Digraph models of bard-type algorithms for the linear complementary problem. *Mathematics of Operations Research*, 3:322–333, 1978.

[6] T. Szabó and E. Welzl. Unique sink orientations of cubes. In *Proc.* 42^{nd} *IEEE Symp. on Foundations of Comput. Sci.*, pages 547–555, 2000.

A Duality of quadratic programs

Definition A.1. *For a quadratic program*

$$\begin{array}{ll} \min & \frac{1}{2}x^T Q x - c^T x \\ \text{s.t.} & Ax = b \\ & x \geqslant 0\,, \end{array} \tag{A.1}$$

the dual program is defined as

$$\begin{array}{ll} \max & -\frac{1}{2}x^T Q x - b^T y \\ \text{s.t.} & Qx + A^T y \geqslant c\,. \end{array} \tag{A.2}$$

We remark that for $Q = \mathbf{0}$, this directly corresponds to the concept of duality for linear programs.

Using the above definition, we can also calculate the dual program of an unconstrained quadratic program (having no inequality constraints):

$$\begin{array}{lll} \min & \frac{1}{2}x^T Q x - c^T x \\ \text{s.t.} & Ax = b \end{array} \tag{A.3}$$

has the dual program

$$\begin{array}{lll} \max & -\frac{1}{2}x^T Q x - b^T y \\ \text{s.t.} & Qx + A^T y = c\,. \end{array} \tag{A.4}$$

This can be seen by writing the variable x as $x = u - v$, $u \geqslant 0$, $v \geqslant 0$, and using the matrix $Q' := \left(\begin{smallmatrix} Q & -Q \\ -Q & Q \end{smallmatrix}\right)$, and $c' := \left(\begin{smallmatrix} c \\ -c \end{smallmatrix}\right)$ in the above definition.

B Source code

Mathematica code for calculating a complete USO from any linear program:

```
A = {{8, 1, 8}, {-3, 3, -4}, {15, -9, 21}};
b = {-1, 1, 1};
c = {1, 2, 3};

{m, n} = Dimensions[A];
outmap = Table[0, {2^n}];

For[vertex = 1, vertex < 2^n, vertex++,

 J = Reverse[Pick[Reverse[Range[n]], PadLeft[IntegerDigits[vertex, 2], n], 1]];
 j = Length[J];
 AJ = A[[All, J]]; AJt = Transpose[AJ];
 MM = Table[0, {2 j}, {2 j}];
 MM[[Range[j], Range[j]]] = AJt.AJ;
 MM[[j + Range[j], Range[j]]] = IdentityMatrix[j];
 MM[[j + Range[j], j + Range[j]]] = -AJt.AJ;

 decided = Table[-2, {j}];
 orientations = Table[1, {j}];
 g  = Take[LinearSolve[MM, PadLeft[c[[J]], 2 j]], {1, j}];

 k = -1; pp = -AJt.b; p = -(c[[J]] - g);
 While[Min[decided] < -1 && k < 2 m,
  orientations = Table[If[decided[[i]] < -1 && g[[i]] != 0,
     If[g[[i]] > 0, 0, 1], orientations[[i]]], {i, j}];
  decided = Table[If[decided[[i]] < -1 && g[[i]] != 0, k, decided[[i]]], {i, j}];
  g  = Take[LinearSolve[MM, PadRight[-pp, 2 j]], {1, j}];
  k++; pp = p; p = g;
 ];

 globorient = Table[0, {n}]; globorient[[J]] = orientations;
 outmap[[vertex + 1]] += FromDigits[Reverse[globorient], 2];

 incoming = Pick[J, orientations, 0];
 While[Length[incoming] > 0,
  nbrvtx = vertex - 2^(incoming[[1]] - 1);
  outmap[[nbrvtx + 1]] += 2^(incoming[[1]] - 1);
  incoming = Rest[incoming];
 ]

]
outmap
```